Вопросы преобразования и совместимости

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconCompatibilityIssuesC"}

База данных Microsoft Access 97 не может быть открыта с помощью предыдущих версий Microsoft Access. Однако предусмотрены связь и импорт таблиц Microsoft Access 97 (версия 8.0) в базы данных Microsoft Access 95 (версия 7.0). Существует возможность экспорта, а также вырезания, копирования и восстановления данных из таблицы версии 8.0 в таблицу версии 7.0.

База данных, созданная в предыдущей версии Microsoft Access, может быть подключена в Microsoft Access 97, однако при этом невозможно добавление новых объектов в базу или изменение структуры уже существующих.

Если попытка преобразования базы данных в формат Microsoft Access 97 оказалась неудачной, то рекомендуется создать новую базу данных Microsoft Access 97 и импортировать в нее все объекты из предыдущей версии базы.

Примечание. Если необходимо открыть совместный доступ к базе данных с пользователями, работающими с предыдущей версией Microsoft Access, то следует открыть базу предыдущей версии в режиме подключения, а не в режиме преобразования. Для совместного доступа к базе пользователей предыдущей версии Microsoft Access **не следует** преобразовывать базу данных в формат Microsoft Access 97.

Преобразование базы данных

Преобразование защищенной базы данных из предыдущей версии Microsoft Access

Преобразование библиотек и программ-надстроек

Новые возможности для разработчиков Microsoft Access

Измененные имена свойств

Измененные и удаленные параметры в Microsoft Access 97

Преобразование программ Access Basic в программы Visual Basic

Преобразование программ, содержащих вызовы библиотек динамической компоновки (DLL)

Макрокоманды и методы объекта DoCmd

Преобразование сочетаний клавиш, используемых в макрокоманде «КомандыКлавиатуры» и инструкции SendKeys

Новый формат имен встроенных констант

Новые правила наименования объектов и области определения объектов

Совместимость библиотек объектов доступа к данным (DAO)

Пример преобразования программы объекта доступа к данным (DAO)

Преобразование библиотек и программ-надстроек

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconConvertingLibrariesAndAddInsC"}

Для использования в приложениях Microsoft Access 97 программ-надстроек или библиотечных баз данных, созданных в предыдущих версиях Microsoft Access, необходимо преобразовать их в формат версии 8.0. Для гарантий корректной работы могут также потребоваться некоторые преобразования объектов, макросов и процедур в библиотечных базах данных и в программах-надстройках.

Более подробное описание см. в главе 12 «Использование библиотечных баз данных и библиотек динамической компоновки» книги *Разработка приложений для Microsoft Access 97* или в разделе Определение ссылок на библиотеки типов.

Ссылки на библиотечные базы данных и их загрузка

Для использования библиотеки в Microsoft Access 97 необходимо создать ссылку на библиотечную базу данных в каждом из приложений, в котором она будет использоваться. Для установки ссылки выберите строку **Ссылки** в меню **Сервис** в режиме конструктора. Адресуемая база данных должна быть в формате Microsoft Access 97.

Библиотечная база данных должна содержать только программы Visual Basic, которые могут быть вызваны из любого приложения, имеющего установленную ссылку на эту библиотеку. В версиях 1.*x* и 2.0 Microsoft Access библиотечная база данных загружалась при запуске, для чего требовалось создать элемент в разделе «Libraries» файла инициализации (.ini). Теперь большая часть сведений, которые сохранялись в файле .ini, записывается в реестр Windows. При этом для использования библиотечной базы данных нет необходимости создавать запись в реестре Windows.

Циклические ссылки в библиотеках

В версиях 1.*x* и 2.0 Microsoft Access допускались циклические ссылки в библиотеках. Однако в более поздних версиях они запрещены. Другими словами, если в библиотеке «A» создана ссылка на библиотеку «B», то нельзя создать в библиотеке «B» ссылку на библиотеку «A».

Меню программ-надстроек

В версиях 1.*x* и 2.0 Microsoft Access программы-надстройки автоматически включались в подменю **Программы-надстройки**, которое содержалось в меню **Файл**. В последующих версиях Microsoft Access подменю **Надстройки** входит в меню **Сервис**. В Microsoft Access 97 ссылки на программы-надстройки могут быть сделаны и любой панели инструментов или меню.

Новые правила наименования объектов и области определения объектов

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acconScopingAndObjectNamingCompatibilityC;daproOwner;vaconAvoidingNamingConflicts;vaconUnderstandingScope"}

Правила Visual Basic, действующие для областей определения переменных и процедур, влияют на имена, выбранные для объектов, модулей и процедур.

Совпадающие имена модулей и других объектов

Имена модулей не должны начинаться со слов «Form_» или «Report_». Такие имена могут вызвать конфликт с существующими программами, написанными для форм и отчетов.

Если в версиях 1.*x* или 2.0 Microsoft Access был создан модуль, имя которого противоречит этому правилу, то при преобразовании приложения возникнет ошибка. Например, модуль, имеющий в базе данных Microsoft Access версии 1.x или 2.0 имя «Form_Заказы», приведет к ошибке при попытке его преобразования, при этом будет запрошено новое имя.

Совпадающие имена модулей и процедур

Допускается совпадение имени процедуры с именем модуля. Однако для вызова такой процедуры из выражений в любых частях приложения необходимо указывать полную ссылку на процедуру, включающую как имя процедуры, так и имя модуля, например:

```
IsLoaded.IsLoaded("Заказы")
```

Совпадающие имена процедур и элементов управления

Если в форме вызывается процедура, имя которой совпадает с именем элемента управления в этой форме, то при вызове такой процедуры требуется указать имя модуля, в котором она сохраняется. Например, если в форме имеется кнопка «ПечатьСчета», то для вызова из формы или модуля формы процедуры с именем «ПечатьСчета», которая сохраняется в стандартном модуле «ВспомогательныеФункции», требуется полная ссылка.

```
ВспомогательныеФункции.Печатьсчета
```

Совпадающие имена элементов управления

Не допускается создание элемента управления, имя которого отличается от имени существующего элемента управления только пробелом или специальным символом. Например, если имеется элемент управления [Дата_доставки], то нельзя создать элемент управления с именем [Дата доставки] или [Дата+доставки].

Совпадающие имена модулей и библиотек типов

Не допускается сохранение модуля под именем, совпадающим с именем библиотеки типов. Попытка сохранить модуль под именем «DAO», «Access» или «VBA» приведет к возникновению ошибки, в сообщении о которой будет указан конфликт с существующим модулем, проектом или библиотекой объектов. Аналогично, если создана ссылка на другую библиотеку типов, например, на библиотеку типов Microsoft Excel, то невозможно сохранить модуль под именем «Excel».

Совпадающие имена полей и методов

Если имя поля таблицы совпадает с именем метода объектов доступа к данным (DAO), действующим на объект **Recordset**, то не допускается ссылка на соответствующее поле в наборе записей с помощью оператора **.** (точка). Необходимо использовать в конструкции оператор **!** (восклицательный знак), иначе возникнет ошибка Microsoft Access. В следующем примере демонстрируется ссылка на поле с именем «AddNew» в наборе записей, открытом по

таблице «Контакты».

```
Dim dbs As Database, rst As Recordset
Set dbs = CurrentDb
Set rst = dbs.OpenRecordset("Контакты")
Debug.Print rst!AddNew
```

Совпадающие имена модулей и функций Visual Basic

Если имя, под которым сохранен модуль, совпадает с именем встроенной функции Visual Basic, то при попытке вызвать эту функцию возникнет ошибка Microsoft Access. Например, если сохранить модуль под именем «MsgBox», то попытка выполнить процедуру, содержащую функцию **MsgBox**, приведет к сообщению об ошибке «Ожидается имя переменной или процедуры, а не модуля».

Совпадающие имена модулей и объектов

Если база данных, созданная в предыдущей версии Microsoft Access включает модуль, имя которого совпадает с именем объекта Microsoft Access или объекта доступа к данным, то при преобразовании базы данных к формату Microsoft Access 97 могут возникнуть ошибки компиляции. Например, к ошибке компиляции могут привести модули с именами «Form» или «Database». Во избежание ошибок такие модули необходимо переименовать.

Наименование полей, используемых в выражениях или присоединенных к элементам управления в формах или отчетах

При создании в таблице поля, которое будет присоединено к элементу управления в отчете или использовано в выражении, определяющем свойство **Данные (ControlSource)** элемента управления или отчета, нельзя присваивать полю имя, совпадающее с именем метода объекта **Application**. Для просмотра списка методов объекта **Application** выберите в меню **Вид** в режиме конструктора модуля команду **Просмотр объектов**. Выберите **Access** в списке **Проет/Библиотека**, выберите **Application** в списке **Classes** просмотрите список методов объекта **Application** в поле **Компонент <имя>**.

При создании в таблице поля, которое будет присоединено к элементу управления в форме или отчете, не присваивайте полям следующие имена: «AddRef», «GetIDsOfNames», «GetTypeInfo», «GetTypeInfoCount», «Invoke», «QueryInterface» и «Release».

Совпадающие имена идентификаторов и ключевых слов Visual Basic

Версия Visual Basic, используемая в Microsoft Access 97, содержит некоторые новые ключевые слова, которые не могут совпадать с идентификаторами. Эти ключевые слова следующие: **AddressOf**, **Assert**, **Decimal**, **DefDec**, **Enum**, **Event**, **Friend**, **Implements**, **RaiseEvent** и **WithEvents**. При преобразовании базы данных, разработанной в предыдущей версии Microsoft Access, идентификаторы, которые совпадают с перечисленными ключевыми словами, вызовут ошибки компиляции. Для избежания подобной ситуации следует переименовать идентификаторы.

Совпадающие имена проектов и объектов Microsoft Access

Именем проекта является строка, совпадающая с именем приложения Microsoft Access. В предыдущих версиях Microsoft Access в качестве имени проекта использовалось имя базы данных. В Microsoft Access 97 имя проекта задается пользователем в свойстве **Имя проекта (ProjectName)**. Имя базы данных является значением этого свойства по умолчанию. При преобразовании базы данных, имя которой совпадает с именем класса объектов, например, «application», «form» или «report», Microsoft Access добавляет к имени базы данных символ подчеркивания, чтобы создать имя проекта, не конфликтующее с именами существующих объектов.

Макрокоманды и методы объекта DoCmd

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconMacroActionsAndMethodsC"}

Для выполнения макрокоманд из программ в Microsoft Access 97 используется объект **DoCmd** и его методы. Этот объект заменил инструкцию **DoCmd**, которая применялась в версиях 1.*x* и 2.0 Microsoft Access для выполнения макрокоманд.

При преобразовании базы данных любые инструкции **DoCmd** и проводимые ими действия в программе Access Basic изменяются на методы объекта **DoCmd** путем замены пробела оператором **.** (точка).

Работа некоторых макрокоманд в Microsoft Access 97 отличается от их работы в версиях 1.*x*, 2.0 или 7.0. Отличия представлены ниже.

Базы данных, созданные в Microsoft Access 95

*Макрокоманда **КомандаМеню (DoMenuItem**)*

Макрокоманда **КомандаМеню (DoMenuItem)** больше не используется в Microsoft Access 97. Вместо нее определена макрокоманда **ВыполнитьКоманду (RunCommand)**.

Если база данных, созданная в предыдущей версии Microsoft Access, объявлена доступной, то макрокоманда **КомандаМеню (DoMenuItem)** будет работать.

Если осуществляется преобразование базы данных, созданной в предыдущей версии Microsoft Access, то во всех макросах макрокоманды **КомандаМеню (DoMenuItem)** заменяются макрокомандами **ВыполнитьКоманду (RunCommand)** при первом сохранении макросов после преобразования. Инструкции вызова метода **DoMenuItem** в процедурах Visual Basic не изменяются.

Базы данных, созданные в Microsoft Access версии 1.*x* или 2.0

*Макрокоманда **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)***

Microsoft Access 95 не поддерживает импорт электронных таблиц Microsoft Excel версии 2.0 и электронных таблиц Lotus 1-2-3 версии 1.0. Если преобразуемая база данных содержит макрос, который обеспечивал эти действия с помощью команды **ПреобразоватьЭлектроннуюТаблицу** в Microsoft Access версии 1.x или 2.0, то после преобразования базы данных в аргументе «Тип электронной таблицы» будет указан тип Microsoft Excel версии 3.0 (если ранее был указан Microsoft Excel версии 2.0). Если ранее был указан формат Lotus 1-2-3 версии 1.0, то преобразование приведет к ошибке.

Для разрешения этой проблемы перед импортированием в Microsoft Access преобразуйте электронные таблицы к формату более поздней версии Microsoft Excel или Lotus 1-2-3.

*Макрокоманды **ПреобразоватьТекст (TransferText**) и **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet**)*

В Microsoft Access 97 не допускается использование инструкции SQL для указания данных, экспортируемых с помощью макрокоманд **ПреобразоватьТекст (TransferText)** или **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)**. Вместо использования инструкции SQL необходимо предварительно создать запрос и указать имя этого запроса в аргументе «Таблица».

*Значения **Null** в операциях сравнения*

Если в версиях Microsoft Access 1.x и 2.0 с помощью оператора сравнивались два выражения, одно из которых имело значение **Null**, то Access Basic возвращал в операции сравнения значение **True** или **False**, в зависимости от используемого оператора сравнения. В Microsoft Access 95 и Microsoft Access 97 Visual Basic возвращает значение **Null**, если один из операндов

имеет значение **Null**. Для проверки результатов сравнения на значения **Null** следует использовать функцию **IsNull**.

Преобразование программ Access Basic в программы Visual Basic

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconConvertingAccessBasicCodeToVisualBasicC"}

В Microsoft Access 97 используется язык программирования Visual Basic для приложений (VBA) вместо использовавшегося в версиях 1.*x* и 2.0 языка Access Basic. Во многих отношениях язык Visual Basic идентичен Access Basic, и Microsoft Access автоматически выполняет большинство необходимых изменений в программах при преобразовании базы данных.

Однако пользователь должен иметь представление об изменениях, вносимых в программы в процессе преобразования, а в некоторых случаях должен самостоятельно внести в программы некоторые дополнительные изменения, чтобы обеспечить успешное выполнение приложения в Microsoft Access 97.

Примечание. Параметр командной строки /**runtime** не является доступным в автономной версии Microsoft Access 97 или в версии Microsoft Access 97 в профессиональном издании Office. Однако параметр командной строки /**runtime** является доступным в версии Microsoft Access 97 в издании Office для разработчиков.

Вызов интерфейса Windows API

Если в существующих программах Access Basic версий 1.*x* и 2.0 вызывается интерфейс прикладного программирования Windows (API), то при преобразовании базы данных может возникнуть необходимость изменения соответствующих инструкций. Версии 1.*x* и 2.0 Microsoft Access являлись 16-разрядными приложениями и выполнялись в 16-разрядных версиях Windows. Microsoft Access 97 является 32-разрядным приложением и выполняется в 32-разрядных версиях Windows − Windows 95 и Windows NT.

Необходимо проверить инструкции **Declare** и обеспечить ссылки на правильные 32-разрядные библиотеки динамической компоновки (DLL). Например, 32-разрядные версии Windows включают библиотеку User32.dll; в предыдущих версиях Windows эта библиотека имела имя User.dll. Более подробное описание см. в разделе Преобразование программ с вызовами библиотек DLL, а также в Web в публикации «Porting your 16-bit Microsoft Office-Based Solutions to 32-Bit Microsoft Office» в разделе «Technical Information» на узле Microsoft Access Developer по адресу http://www.microsoft.com/accessdev/.

Изменились имена некоторых функций, вызываемых из интерфейса Windows API. Кроме того, функции в интерфейсе API 32-разрядной версии Windows являются чувствительными к регистру. Полное описание можно найти в документации комплекта разработчика Microsoft Win32 Software Development Kit.

Некоторые свойства и необязательные аргументы имеют определенные типы

Некоторые свойства, которые возвращали значения **Variant** в предыдущих версиях Microsoft Access, в новой версии возвращают строковые значения. Программы, использующие эти свойства, будут выполняться быстрее. Однако при преобразовании базы данных из предыдущей версии Microsoft Access к формату Microsoft Access 97 необходимо проверить, что новые значения свойств не приводят к возникновению ошибок.

Кроме того, некоторые необязательные аргументы, имевшие тип **Variant** в предыдущих версиях Microsoft Access, в новой версии имеют определенные типы. Затрагиваются следующие методы, свойства и функции: метод **GoToPage** объекта **Form**, методы **OpenCurrentDatabase**, **Echo**, **Quit**, **CreateForm**, **CreateReport**, **CreateControl** и **CreateReportControl** объекта **Application**, свойство **Column** поля со списком или списка, а также статистические функции по подмножеству. Если программа содержит эти компоненты, необходимо проверить, что преобразование базы данных не приводит к возникновению ошибок.

Для проверки необходимо откомпилировать все программы в базе данных. После этого проверьте, не возникают ли ошибки при выполнени программ. Наиболее вероятно

возникновение ошибок несогласования типов или недопустимого использованиия значений Null.

Кроме того, следует проверить возможность возникновения логических ошибок при преобразовании базы данных. Например, в свойстве **Источник записей (RecordSource)** элемента управления вместо возвращаемого ранее значения **Variant** теперь возвращается строковое значение. Строка не может содержать значение **Null**, являющееся одним из допустимых значений типа **Variant**. Программа в преобразованной базе данных, проверяющая, имеет ли свойство **RecordSource** значение **Null**, не будет выполняться правильно, поскольку теперь свойство **RecordSource** не может иметь значение **Null**. Ниже приводится пример процедуры, в которой использовалось значение **Null** свойства **RecordSource** поля.

```
Private Sub Form_Load()
   If IsNull(Me.RecordSource) Then
      Me.Recordsource = "Сотрудники"
   End If
End Sub
```

В Microsoft Access 97 функция **IsNull** никогда не будет возвращать значение **True** для свойства **RecordSource**. Если в предыдущем примере значение свойства **RecordSource** не было задано, то его значением является пустая строка. В этом случае функция **IsNull** возвращает значение **False**, и процедура никогда не войдет в конструкцию **If...Then**, в которой определяется свойство **RecordSource**.

Для исправления этой ошибк следует с помощью функции **Len** проверить длину строки, возвращаемой свойством **RecordSource**, и определить, не является ли строка пустой. Для предыдущего примера ошибку исправит следующая инструкция:

```
If Len(Me.RecordSource) = 0 Then
   ' Далее следует остальная часть процедуры.
```

Изменения в модулях

В версиях 1.*x* и 1.0 Microsoft Access модули, содержащие процедуры общего назначения, не являющиеся специфическими для конкретной формы или отчета, назывались глобальными модулями. В Microsoft Access 97 такие модули называются стандартными модулями.

Microsoft Access 97 включает модули класса, которые могут служить шаблонами для объектов, определяемых пользователем. Любые общие процедуры из модуля класса становятся методами и свойствами специального объекта при создании нового экземпляра класса. Более подробное описание модулей класса см. в разделе Программы с модулями класса.

Функция CurrentDb и конструкция DBEngine(0)(0)

Для возвращения объектной переменной типа **Database**, представляющей текущую базу данных, следует использовать функцию **CurrentDb** вместо конструкции `DBEngine(0)(0)`. Функция **CurrentDb** создает новый элемент текущей базы данных, тогда как конструкция `DBEngine(0)(0)` представляет ссылку на открытую копию текущей базы данных. Использование конструкции `DBEngine(0)(0)` не дает пользователю возможности использовать несколько переменных типа **Database**, представляющих текущую базу данных.

Синтаксис `DBEngine(0)(0)` по-прежнему поддерживается, поэтому Microsoft Access не изменяет эту конструкцию в процессе преобразования базы данных. Однако пользователям рекомендуется выполнить это изменение самостоятельно, чтобы избежать возможных конфликтов при работе в сети.

Функция CurDir

Из-за изменения характера взаимодействия приложения со средой, в Windows 95 функция **CurDir** выполняется не так, как в версиях 1.x или 2.0. Поскольку каждое приложение имеет

собственный текущий каталог, выбор текущего каталога в Windows 95 двойным нажатием кнопки мыши на значке не влияет на текущий каталог Microsoft Access. В Microsoft Access 95 функция **CurDir** всегда возвращает текущий путь.

Присваивание значения объекта OLE переменной

При работе в программе с объектами OLE или другими данными в двоичном формате для сохранения двоичных данных должен использоваться байтовый массив. В версиях 1.*x* и 2.0 Microsoft Access для программной обработки объектов OLE или других двоичных данных с размером меньше 64K требовалось присвоить значение строковой переменной. Строковым переменным присваивались также значения, возвращаемые методом **GetChunk**. Однако в Visual Basic определен тип данных **Byte** и функции, работающие с байтовыми данными, такие как **LeftB** и **RightB**. В Microsoft Access 97 пользователь должен сохранять двоичные данные в байтовых массивах, а не в строковых переменных, и использовать для обработки этих данных байтовые функции.

Описание каналов DDE с типами Variant или Long

При открытии канала DDE с помощью функции **DDEInitiate** необходимо описать переменную, в которой сохраняется номер канала, с типом данных **Long** или **Variant** с подтипом **Long**. В версиях 1.*x* и 2.0 Microsoft Access номер канала сохранялся в переменной типа **Integer** поэтому любые инструкции описания в программе, в которых для сохранения номера канала создаются переменные типа **Integer**, должны быть изменены.

Ошибки преобразования для строк, содержащих символ процентов (%)

Не допускается присваивание строкового значения, представляющего число с символом процентов, переменной или полю, имеющим числовой тип данных:

```
Dim intX As Double
   intX = "10"          ' Эта инструкция выполняется.
   intX = "10%"         ' Эта инструкция возвращает ошибку.
```

Свойство hWnd

При использовании в программе свойства **hWnd** для передачи дескриптора окна формы или отчета в подпрограмму Windows, необходимо прямо передать в подпрограмму значение. Не следует присваивать значение этого свойства переменной, например:

```
If Not IsZoomed(Screen.ActiveForm.hWnd) Then
   DoCmd.Maximize
EndIf
```

В версиях 1.*x* и 2.0 Microsoft Access свойство **hWnd** формы или отчета имело значение типа **Integer**. В Microsoft Access 97 значение свойства **hWnd** принадлежит к типу **Long**, что делает необходимым внесение изменений в программы.

Свойства объектов доступа к данным и свойства объектов Access Microsoft Access

Не допускается использование в программах объектных переменных для ссылок на объект-свойство **Category**. Свойство **Category** более не поддерживается для объектов **Form**, **Report** и **Control**.

Свойство Parent

В Microsoft Access 97 при ссылках на свойство **Parent** элемента управления в программах или в выражениях формы или отчета обычно возвращается объект **Form** или **Report**, содержащий данный элемент управления. Например, если «КодТипа» является именем поля в форме «Типы», то конструкция `Forms!Тип!КодТипа.Parent` возвращает ссылку на форму «Типы». Из этого правила имеются два исключения: для присоединенных надписей свойство **Parent**

возвращает элемент управления, к которому присоединена надпись; а для элементов управления, содержащихся в группе параметров, возвращается группа.

Вызов программ мастеров Microsoft Access

Если в приложениях версий 1.x или 2.0, были созданы программы, вызывающие процедуры мастеров Microsoft Access, то после преобразования приложения необходимо установить в приложении ссылку на базу данных мастеров, в которой содержатся вызываемые процедуры. Для получения дополнительных сведений об установке ссылок см. раздел Определение ссылок на библиотеки типов

В Microsoft Access версии 2.0 не существовало различий между мастерами и библиотеками, поэтому содержащиеся в них общие программы были всегда доступны из текущей базы данных. В Microsoft Access 97 мастера и другие программы-надстройки больше не рассматриваются как библиотеки. Кроме того, поскольку мастера могут существенно изменяться от одной версии Microsoft Access к другой, то при переходе к новой версии Microsoft Access может возникнуть необходимость переработки некоторых программ в соответствие с изменениями мастеров.

Для программ, которые более не входят в набор мастеров Microsoft Access, таких как автоматический набор номера (AutoDialer), соответствующие функциональные характеристики поддерживаются системной базой данных Utility.mda, устанавливаемой вместе с Microsoft Access. Ссылка на эту базу данных создается автоматически при преобразовании базы данных к версии Microsoft Access 97.

Установка ссылки на базу данных Microsoft Access

В Microsoft Access 97 нельзя установить ссылку на базу данных, созданную в предыдущей версии Microsoft Access. Для создания ссылки необходимо предварительно преобразовать базу данных в формат Microsoft Access 97 (версия 8.0).

Обработка ошибок программирования объектов

В версиях 1.*x* и 2.0 Microsoft Access компоненты ActiveX, которые поддерживают программирование объектов (ранее использовался термин «серверы механизма управления OLE»), возвращали только одну ошибку. Visual Basic позволяет компоненту ActiveX возвращать сведения об ошибке, описывающие конкретную ошибку. Если в программах существующей базы данных была предусмотрена обработка единственной ошибки программирования объектов, то теперь появляется возможность включить в эти программы обработку конкретных ошибок.

Ошибки Microsoft Access и функция Error

В Microsoft Access 97 более не допускается использование свойств объекта **Err** для получения информации об ошибке.

Для получения номера ошибки Microsoft Access и ее описания используются свойства **Number** и **Description** объекта **Err**, как показано в следующем примере:

```
Debug.Print Err.Number, Err.Description
```

Информация по ошибкам Microsoft Access, Visual Basic и объекта доступа к данным может быть получена с помощью метода **AccessError**, причем независимо от того, действительно ли ошибка имела место.

Ссылки на объекты доступа к данным

Если при ссылках на объект **Field** объекта **Recordset** в приложениях, созданных в версиях 1.*x* или 2.0 Microsoft Access, использовался оператор **.** (точка), то необходимо заменить этот оператор на **!** (восклицательный знак). Другой возможностью, позволяющей по-прежнему использовать синтаксис с оператором **.** (точка), является установка ссылки на библиотеку

«Microsoft DAO 2.5/3.0 Compatibility Library» в диалоговом окне **Ссылки**. Это окно открывается командой **Ссылки** в меню **Сервис** в режиме конструктора.

Преобразованные элементы ActiveX

Если приложение содержит элементы ActiveX (ранее называемые элементы управления OLE или специальные элементы управления), созданные в версии 2.0, необходимо вставить ключевое слово **ByVal** перед аргументами, которые передаются в процедуры обработки событий, возникающих в элементах ActiveX, например:

```
Sub ChangeMonth_Click(ByVal intCurrentYear As Integer)
```

Для того чтобы определить, следует ли передавать аргумент по значению, выберите команду **Компилировать все модули** в меню **Запуск** окна конструктора модуля. Если будет получено следующее сообщение об ошибке: «Описание процедуры обработки события не соответствует описанию события с тем же именем», - то необходима вставка ключевого слова **ByVal** перед аргументами.

В Microsoft Access 97 усовершенствована проверка типов аргументов, поэтому в новые процедуры обработки событий, создаваемые для элементов ActiveX, ключевое слово **ByVal** при необходимости будет вставляться автоматически.

Вызов процедур в модулях форм и отчетов

В версиях 1.*x* и 2.0 Microsoft Access вызов процедур, описанных в форме или отчете, был возможен только из модуля этой формы или отчета. В Microsoft Access 97 допускается вызов общих процедур, описанных в модуле формы или отчета, из любой процедуры в текущей базе данных. При вызове процедуры необходимо указать соответствующее имя класса модуля формы или отчета. Например, процедура «ВыводСообщения», описанная в модуле формы «Заказы», вызывается следующим образом:

```
Form_Заказы.ВыводСообщения
```

Рекомендуется по возможности помещать процедуры, которые будут вызываться извне, в стандартный модуль, а не в модуль формы или отчета.

Ссылки на формы и элементы управления в окне отладки

При проверке и отладке программы необходимо указывать в окне отладки полные ссылки на используемые объекты в окне отладки, за исключением случая, когда прервано выполнение модуля формы или отчета. Это означает, что на панели проверки окна отладки для ссылки на элемент управления «КодТипа» формы «Типы» в режиме формы необходимо использовать конструкцию `Forms!Типы!КодТипа`, а не ограничиваться именем `КодТипа`, даже если форма «Типы» является текущей формой.

Кроме того, в окне отладки не допускается использование ключевого слова **Me** для ссылки на объект в форме или отчете, находящихся в режиме конструктора, за исключением случая, когда было прервано выполнение программы из модуля формы или отчета.

Преобразование форм и отчетов, содержащих Элементы ActiveX

При преобразовании базы данных Microsoft Access 2.0 для работы с Microsoft Access 97 (версия 8.0), элементы ActiveX в формах или отчетах могут оказаться не преобразованными автоматически. Microsoft Access 2.0 поддерживает 16-разрядные элементы ActiveX, тогда как версия 8.0 поддерживает только 32-разрядные элементы ActiveX.

При попытке преобразовать базу данных, в которой форма или отчет содержат 16-разрядную версию элемента ActiveX, а в системе отсутствует 32-разрядная версия, Microsoft Access выведет сообщение об ошибке. Необходимо приобрести 32-разрядную версию каждого элемента ActiveX, который требуется обновить, и зарегистрировать ее в системном реестре. После успешного преобразования необходимо сохранить форму или отчет в преобразованной

базе данных, а затем закрыть и вновь открыть базу данных.

Объявление базы данных, созданной в предыдущей версии Microsoft Access, доступной

При объявлении доступной большой базы данных, созданной в версии 1.*x* или 2.0 Microsoft Access, но не преобразованной к формату Microsoft Access 97, может потребоваться увеличение максимального размера временного буфера по сравнению со стандартными размерами. Для изменения данной настройки откройте реестр Windows в редакторе реестра и перейдите к записи \HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Jet\3.0\Engines\Jet 2.x. Создайте новую вложенную запись ISAM и введите в нее новое значение типа DWORD с именем «MaxBufferSize». Задайте для него новое значение 1024 в десятичной системе. Для получения дополнительных сведений о редактировании реестра Windows см. документацию Windows.

Значения времени в условиях отбора в запросе

После преобразования базы данных версии 1.*x* или 2.0 Microsoft Access в формат Microsoft Access 97 запросы, содержащие условия отбора, в которых используются значения времени из полей типа даты/времени, могут возвращать результаты, отличающиеся от тех, которые возвращались в предыдущих версиях. Это же может случиться при связывании таблиц из баз данных, созданных в версиях 1.*x* и 2.0, с базой данных Microsoft Access 97. Такие проблемы возникают только со значениями времени из полей даты/времени.

Функции Visual Basic, которые не поддерживаются в выражениях

Следующие шесть функций Visual Basic более не поддерживаются в выражениях вне определяемых пользователем процедур **Sub** или **Function**:

EOF	**Loc**
FileAttr	**LOF**
FreeFile	**Seek**

При необходимости использовать эти функции в выражениях, не входящих в процедуру, необходимо создать определяемую пользователем функцию, которая вызывает одну из этих функций, и включить определяемую пользователем функцию в выражение.

Номера строк в процедурах Visual Basic

Не допускается использование в процедурах Visual Basic номеров строк, превышающих 65529. Если преобразуемое приложение Microsoft Access версии 1.*x* или 2.0 содержит номера строк, превышающие 65529, необходимо изменить эти номера на допустимые.

Кнопки «Предыдущая процедура» и «Следующая процедура»

Кнопки **Предыдущая процедура** и **Следующая процедура**, которые выводились на панели инструментов **Модуля** в версиях 1.*x* и 2.0, отсутствуют в Microsoft Access 97. Если преобразуется база данных версии 1.*x* или 2.0, содержащая специальную панель инструментов, на которой выводится одна из этих кнопок, то сообщение об ошибке выведено не будет, но нажатие этих кнопок на панели инструментов не будет иметь последствий.

Ошибка «Недостаточно памяти» при преобразовании больших баз данных

В языке Visual Basic для приложений количество модулей в базе данных ограничивается 1,082. Каждая форма и отчет содержат по одному модулю. Чтобы разрешить эту проблему следует уменьшить количество объектов в базе или разбить приложение на несколько баз данных. Если модуль содержит программу большого объема, то рекомендуется использовать библиотечные базы данных для хранения программы.

Форматирование значений Null и пустых строк

В версии 2.0 Microsoft Access предусмотрено использование функции **Format**, которая возвращает одно значение для пустых строк и другое для значений **Null**. Свойство **Format** также применяется для форматирования полей в таблице в режиме таблицы, элементе управления, форме или отчете. Например в программе следующее выражение может быть передано функции **Format** для получения соответствующего значения:

```
Dim var As Variant, strX As String
' Назначить некоторое значение strX и передать в функцию Format.
var = Format(strX, "@;ZLS;Null")
```

В Microsoft Access 97 необходимо отдельно проверять случай значения **Null** и возвращать подходящее значение, основанное на результате. Например функция **IIf** может использоваться в выражении с функцией **Format**:

```
var = IIf(IsNull(strX),"Null", Format(strX, "@;ZLS"))
```

Эти изменения касаются только случая использования функции **Format** для форматирования строк в зависимости от того, являются ли они пустыми строками или значение **Null**. Все остальные выражения для форматирования с помощью функции **Format** работают также как и в предыдущих версиях.

При преобразовании базы данных из версии 2.0 Microsoft Access в Microsoft Access 97 потребуется изменить программы или параметры свойств для использования данных методов. Свойство **Format** не может быть использовано в режиме таблицы для установления различий между значения **Null** и пустыми строками.

Совместимость библиотек объектов доступа к данным (DAO)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconDAOObjectLibraryCompatibilityC"}

В Microsoft Access 97 включены новые объекты доступа к данным (DAO), методы и свойства, заменяющие определенные в версиях 1.*x*, 2.0 и 7.0. Хотя программы Microsoft Access 97 являются совместимыми с программами объектов доступа к данным предыдущих версий, возможно, что в будущих версиях Microsoft Access не будут поддерживаться некоторые старые объекты, методы и свойства. В следующих разделах обсуждаются вопросы совместимости Microsoft Access 97 с предыдущими версиями, а также даются советы по созданию новых и изменению существующих приложений таким образом, чтобы обеспечить совместимость с будущими версиями Microsoft Access.

Совместимость с предыдущими версиями

Для того чтобы продолжать использование в приложении старых версий объектов доступа к данными и их свойств и методов, необходимо предварительно создать ссылку на библиотеку «Microsoft DAO 2.5/3.5 Compatibility Library». Эта ссылка устанавливается в диалоговом окне **Ссылки**, которое открывается командой **Ссылки** в меню **Сервис** в режиме конструктора. Библиотека «Microsoft DAO 2.5/3.5 Compatibility Library» обеспечивает полную совместимость с версиями 1.*x*, 2.0 и 7.0 Microsoft Access. Ссылка на данную версию библиотеки добавляется в приложение предыдущей версии во время его преобразования в формат Microsoft Access 97.

В библиотеку объектов «Microsoft DAO 3.5 Object Library», ссылка на которую устанавливается по умолчанию при создании новой базы данных, старые объекты, методы и свойства не включаются. Для того чтобы исключить использование старых методов, следует во всех новых приложениях, создаваемых в Microsoft Access 97, использовать только методы из библиотеки объектов «Microsoft DAO 3.5 Object Library». Кроме того, если приложение использует только библиотеку объектов «Microsoft DAO 3.5 Object Library», нет необходимости передавать пользователям библиотеку «Microsoft DAO 2.5/3.5 Compatibility Library» при передаче им разработанного приложения.

Совет. Для того чтобы проверить, что приложение использует только те объекты, методы и свойства, которые содержатся в библиотеке объектов "Microsoft DAO 3.5 Object Library, следует снять в окне **Ссылки** флажок **Microsoft DAO 2.5/3.5 Compatibility Library** и убедиться, что установлен флажок **Microsoft DAO 3.5 Object Library**, а затем снова откомпилировать приложение командой **Компилировать все модули** из меню **Запуск** в режиме конструктора модуля. Если приложение компилируется без ошибок, ссылка на библиотеку «Microsoft DAO 2.5/3.5 Compatibility Library» становится излишней, и имеется гарантия, что приложение будет работать со следующей версией объектов доступа к данным.

Подготовка для работы с будущим версиями

В следующей таблице перечислены объекты, методы и свойства, не включенные в библиотеку объектов «Microsoft DAO 3.5 Object Library», а также характеристики, которые были добавлены для их замены. Рекомендуется использовать элементы из правого столбца для модификации программ, написанных в предыдущих версиях Microsoft Access. Это обеспечит возможность преобразования к будущим версиям Microsoft Access, в которых элементы из первого столбца поддерживаться не будут.

Пример изменения программы см. в разделе Пример преобразования программы объекта доступа к данным (DAO).

Не поддерживается в DAO 3.5	Рекомендуемые элементы DAO 3.5
FreeLocks	Метод **Idle** объекта **DBEngine** (не требуется для баз данных Microsoft Access 97)
SetDefaultWorkspace	Свойства **DefaultUser**/**DefaultPassword**

	объекта **DBEngine**
SetDataAccessOption	Свойство **IniPath** объекта **DBEngine**
Database.**BeginTrans**	*Workspace*.**BeginTrans**
Database.**CommitTrans**	*Workspace*.**CommitTrans**
Database.**CreateDynaset**	*Database*.**OpenRecordset** типа **dbOpenDynaset**
Database.**CreateSnapshot**	*Database*.**OpenRecordset** типа **dbOpenSnapshot**
Database.**DeleteQueryDef**	Метод **Delete** семейства **QueryDefs**
Database.**ExecuteSQL**	Метод *Database*.**Execute** и свойство *Database*.**RecordsAffected**
Database.**ListTables**	Семейство *Database*.**TableDefs**
Database.**OpenQueryDef**	Семейство *Database*.**QueryDefs**
Database.**OpenTable**	*Database*.**OpenRecordset** типа **dbOpenTable**
Database.**Rollback**	*Workspace*.**Rollback**
Метод **ListFields** объектов **Table**, **Dynaset** и **Snapshot**	Семейство *Recordset*.**Fields**
Table.**ListIndexes**	Семейство *TableDef*.**Indexes**
QueryDef.**CreateDynaset**	*QueryDef*.**OpenRecordset**
QueryDef.**CreateSnapshot**	*QueryDef*.**OpenRecordset**
QueryDef.**ListParameters**	Семейство *QueryDef*.**Parameters**
Объект **Dynaset**	Динамический объект **Recordset**
Объект **Snapshot**	Статический объект **Recordset**
Объект **Table**	Табличный объект **Recordset**
Метод **CreateDynaset** объектов **Dynaset** и **QueryDef**	*Recordset*.**OpenRecordset** с параметром **dbOpenDynaset**
Метод **CreateSnapshot** объектов **Dynaset** и **QueryDef**	*Recordset*.**OpenRecordset** с параметром **dbOpenSnapshot**

Новый формат имен встроенных констант

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconNewStyleForIntrinsicConstantsC"}

В Microsoft Access 97 все встроенные константы содержатся в библиотеках типов и доступны для просмотра в окне просмотра объектов. В Microsoft Access включены библиотеки типов Microsoft Access, объектов доступа к данным (DAO) и Visual Basic. Каждая из этих библиотек типов содержит встроенные константы.

В отличие от предыдущих версий, имена встроенных констант в Microsoft Access 97 записываются с использованием символов как нижнего, так и верхнего регистра, а различные компоненты имен сливаются, а не соединяются символом подчеркивания. Например, имя константы A_NORMAL из предыдущих версий теперь записывается как **acNormal**.

Хотя встроенные константы из предыдущих версий Microsoft Access не будут автоматически преобразованы к новому формату, их использование не приведет к ошибкам. Однако при создании новых программ рекомендуется применять новый формат.

Преобразование сочетаний клавиш, используемых в макрокоманде «КомандыКлавиатуры» и инструкции SendKeys

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconConvertingSendKeysKeyCombinationsC;vastmSendKeys"}

При преобразовании базы данных версии 1.*x* или 2.0 Microsoft Access 97 может потребоваться внесение изменений в окна диалога или меню для декодирования инструкции **SendKeys** или макрокоманды **КомандыКлавиатуры (SendKeys)**. Например, подменю **Программы настройки** переведено из меню **Файл** в меню **Сервис**. Подменю **Импорт**, **Экспорт** и **Присоединить таблицу**, входившие в предыдущих версиях в меню **Файл**, преобразованы в подменю **Внешние данные** и **Сохранить как/Экспорт**. Поскольку подобные изменения могут иметь место и в будущих версиях Microsoft Access, рекомендуется по возможности не использовать инструкцию **SendKeys** или соответствующую макрокоманду для выбора параметров в окнах диалога или выбора команд в меню.

Перечислим некоторые способы, позволяющие обойтись без инструкции **SendKeys** или макрокоманды КомандыКлавиатуры

- До того как использовать макрокоманду **КомандыКлавиатуры** или инструкцию **SendKeys** для выполнения команды меню, проверьте, существует ли эквивалентная макрокоманда или метод Visual Basic. Большинство из часто выполняемых команд меню могут быть выполнены с помощью отдельной макрокоманды или метода. Например, вместо передачи кода клавиши `"{F4}",` может быть использован метод **Dropdown** поля со списком.
- Если для команды меню не определены эквивалентная макрокоманда или метод, вызывайте эту команду с помощью макрокоманды **ВыполнитьКоманду (RunCommand),** а не с помощью инструкции **SendKeys.**
- Старайтесь не использовать инструкцию **SendKeys** или макрокоманду **КомандыКлавиатуры (SendKeys)** для выбора общих параметров в диалоговом окне **Параметры**. В новых версиях Microsoft Access вероятно появление новых или измененных параметров, поэтому программы, содержащие инструкцию **SendKeys** или макрокоманду **КомандыКлавиатуры (SendKeys)**, могут потребовать переработки. Вместо них рекомендуется пользоваться методами **GetOption** и **SetOption**.
- При использовании инструкции **SendKeys** для выполнения определенных действий рекомендуется вводить константы, содержащие значения, передаваемые в инструкции **SendKeys**. Описание нажатий клавиш с помощью констант облегчит изменение программы в будущем.
- В Microsoft Access версии 2.0 существует возможность заключать аргумент «Клавиши» макрокоманды **КомандыКлавиатуры (SendKeys)**, хотя это не является обязательным. Использование кавычек в последующих версиях Microsoft Access може привести к ошибке. Для включения кавычек в строку аргумента «Клавиши» следует водить их парами, например, **Павел (""Паша"") Иванов**. К ошибке может также привести неверный синтаксис аргумента «Клавиши», ошибочно набранные символы или другие значения, являющиеся неподжходящими для окна, в которое посылаются нажатия клавиш.

Преобразование таблиц, форм и отчетов Microsoft Access

{ewc HLP95EN.DLL,DYNALINK,"пїSпїS. пїSпїSпїSпїSпїS":"acconConvertingMicrosoftAccessTablesFormsReportsC"}

Некоторые изменения, внесенные в Microsoft Access 97, могут отразиться на работе приложений версий 1.*x* или 2.0. Такие изменения перечислены в следующих разделах.

Индексы и связи

Таблица Microsoft Access может содержать до 32 индексов. В очень сложных таблицах, участвующих во многих связях, этот предел может быть превышен. Преобразование базы данных в этом случае становится невозможным. База данных Microsoft Jet версии 3.5 создает индексы для таблиц на обеих сторонах связи. Если преобразование базы данных выполнить не удается, удалите некоторые связи и повторите попытку преобразования базы данных.

Свойство поля со списком «Ограничиться списком» (LimitToList)

В Microsoft Access 97 поле со списком, у которого свойство **ОграничитьсяСписком (LimitToList)** имеет значение **True** (-1), принимает значения **Null** вне зависимости от того, содержит ли список значение **Null**. В версии 2.0, поле со списком, у которого свойство **ОграничитьсясСписком (LimitToList)** имеет значение **True**, принимает значения **Null** только в том случае, когда список содержит значение **Null**. Для того чтобы запретить пользователям ввод значений **Null**, следует задать для свойства **ОбязательноеПоле (Required)** поля в таблице значение **Да**.

Меню и активизация объектов ActiveX по месту

Для того чтобы предоставить пользователям дополнительные функциональные возможности при активизации объектов ActiveX по месту, некоторые команды меню переведены в те меню, которые не заменяются при активизации компонента ActiveX.

В преобразованном приложении макросы, в которых макрокоманда **КомандаМеню (DoMenuItem)** используется для выполнения команд меню версии 2.0 при активизации компонента, не будут затронуты этими изменениями. Вместо команд версии 2.0 автоматически подставляются соответствующие команды Microsoft Access 97.

Ссылка на элемент управления в форме, открытой только для чтения

В Microsoft Access 97 не допускается использование выражений для ссылок на значение элемента управления в форме, открытой только для чтения, которая присоединена к пустому источнику записей. В предыдущих версиях в таком случае возвращалось значение **Null**. Перед тем как ссылаться на элемент управления в форме, открытой только для чтения, необходимо проверить, что источник записей формы не пуст.

Ввод значений в поля даты

При вводе значения **3/3** в поле даты в форме или в таблице Microsoft Access 97 автоматически добавит текущий год. Однако, попытка ввода в то же поле значения **3/3/** приведет к ошибке. Для преобразования даты к подходящему формату необходимо опускать последний разделитель.

Создание кнопок с помощью мастера

Если в версиях 2.0 или 7.0 Microsoft Access с помощью мастера по созданию кнопок были созданы программы кнопок, вызывающие другие приложения, то эти кнопки необходимо удалить и создать заново с помощью мастера по созданию кнопок Microsoft Access 97.

Модули класса отчета и формы

В предыдущих версиях Microsoft Access объекты **Form** и **Report** имели связанные модули

класса, даже если в них не было кода. В Microsoft Access 97 возможна установка значения **False** для свойства отчета или формы **HasModule**. Если свойство **HasModule** принимает значение **False**, то форма или отчет занимают меньше места на диске и загружаются быстрее, так как они не имеют более связанного модуля класса.

Ширина полей в отчете, преобразованном из версии 2.0

При распечатке или предварительном просмотре отчета Microsoft Access 97, который был преобразован из Microsoft Access 2.0, могут возникнуть некоторые трудности, если некоторые поля имеют нулевой размер. При преобразовании отчета Microsoft Access 2.0 поля устанавливаются не в 0, а по минимальному допустимому значению поля для стандартного принтера. Это предотвращает попадание данных в области, недоступные для печати.

Для решения этой проблемы следует уменьшить ширину столбца, расстояние между столбцами или количество столбцов в отчете до таких значений, чтобы суммарная ширина столбцов плюс ширина поля по умолчанию не превышала размера бумаги.

Трудности при использовании свойства Format для отражения различий между значениями Null и пустыми строками

В версиях 1.*x* и 2.0 предусмотрено использование свойства **Format** элемента управления для вывода различных значений для значений **Null** и пустых строк (""). В Microsoft Access 97 для различия между значениями **Null** и пустыми строками в элементе управления свойству элемента управления **Данные (ControlSource)** присваивается выражение, которое выявляет случай значения **Null**. Например, для отображения строк «Null» или «ZLS» в элементе управления следует присвоить его свойству **Данные (ControlSource)** следующее выражение:

```
=IIf(IsNull([MyControl]), "Null", Format([MyControl], "@;ZLS"))
```

Надпись

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlLabelControlC"}

Элемент управления в форме или отчете, предназначенный для вывода описательного текста, например, названия, заголовка, или краткой инструкции. Надписи имеют следующие характеристики.

- В надписях невозможно вывести значения полей или выражений.
- Надписи всегда являются свободными.
- Надписи не изменяются, когда пользователь переходит от записи к записи.

Элемент: **Инструмент:**

Дополнительные сведения

Надписи могут присоединяться к другим элементам управления. Например, при создании поля возникает присоединенная к нему надпись, в которой выводится подпись этого поля. Содержимое этой надписи выводится как заголовок столбца в форме в режиме таблицы.

Если надпись создается с помощью кнопки **Надпись**, то она является самостоятельной и не присоединяется ни какому другому элементу управления. Такие надписи используются для вывода общей информации, например, заголовков форм и отчетов или другого описательного текста. Надписи, не присоединенные к элементам управления, в режиме таблицы не выводятся.

Поле

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acctlTextBoxControlC"}

Элемент управления-поле в форме или отчете предназначается для отображения данных из таблицы, запроса или инструкции SQL. Такие поля называются присоединенными элементами управления, поскольку они присоединены к данным в полях таблиц или запросов. Кроме того, существуют также свободные поля. Например, свободные поля создаются для отображения результатов расчетов или для приема данных, вводящихся пользователем. При сохранении базы данных содержимое свободного поля не сохраняется.

Элемент:

Инструмент:

Флажок

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlCheckBoxControlC"}

Флажок в форме или отчете является самостоятельным элементом управления, который предназначен для отображения логического значения из базовой таблицы, запроса или инструкции SQL.

Элемент: **Инструмент:**

Дополнительные сведения

Если пользователь устанавливает или снимает флажок, присоединенный к некоторому логическому полю, то в базовой таблице отображается значение, соответствующее настройке, заданной для этого поля в свойстве **Формат поля (Format)** (Да/Нет, **True**/**False** (Истина/Ложь) или Вкл/Выкл).

Флажки в составе группы переключателей применяются для выбора параметров.

Свободный флажок может использоваться также в специальном диалоговом окне для приема действий пользователя.

Выключатель

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlToggleButtonControlC"}

Выключатель в форме является самостоятельным элементом управления, который предназначен для отображения логического значения из базовой таблицы, запроса или инструкции SQL.

Элемент:

Инструмент:

Дополнительные сведения

Если пользователь нажимает выключатель, присоединенный к некоторому логическому полю, то в базовой таблице отображается значение, соответствующее настройке, заданной для этого поля в свойстве **Формат поля (Format)** (Да/Нет, **True**/**False** (Истина/Ложь) или Вкл/Выкл).

Выключатели удобно использовать в составе группы переключателей, вместе с другими элементами группы.

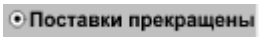
Выключатель может использоваться также в специальном диалоговом окне для приема действий пользователя.

Переключатель

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlOptionButtonControlC"}

Переключатель в форме или отчете является самостоятельным элементом управления, который предназначен для отображения логического значения из базовой таблицы, запроса или инструкции SQL.

Элемент:



Инструмент:

Дополнительные сведения

Если пользователь выбирает или отключает переключатель, присоединенный к некоторому логическому полю, то в базовой таблице отображается значение, соответствующее настройке, заданной для этого поля в свойстве **Формат поля (Format)** (Да/Нет, **True**/**False** (Истина/Ложь) или Вкл/Выкл).

Переключатели в составе группы применяются для выбора параметров.

Свободный переключатель может использоваться также в специальном диалоговом окне для приема действий пользователя.

Группа переключателей

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlOptionGroupControlC"}

Элемент управления в форме или отчете, представляющий группу параметров, предназначенных для выбора значений. Объединение параметров в группу делает выбор их значений (обычно с помощью мыши) простым и наглядным. Параметры выбираются из группы только по одному.

Группа состоит из рамки группы и набора размещенных внутри нее флажков, выключателей или переключателей.

Элемент:

Инструмент:

Дополнительные сведения

Если группа является присоединенной к полю, то только рамка группы присоединена к этому полю, а не содержащиеся в ней флажки, выключатели или переключатели. Вместо того чтобы задавать значение свойства **Данные (ControlSource)** для каждого входящего в группу элемента управления, следует определить свойство **Значение параметра (OptionValue)** каждого флажка, выключателя или переключателя с помощью числа, которое имеет смысл в качестве значения поля, к которому присоединена рамка группы. Когда пользователь выбирает в группе параметр, Microsoft Access присваивает его свойству **Значение параметра (OptionValue)** значение поля, к которому присоединена группа.

Примечание. Свойству **Значение параметра** может быть присвоено только число, поскольку значениями параметров группы являются числа, а не текст. Microsoft Access сохраняет значение выбранного параметра в базовой таблице. В предыдущем примере для вывода в таблице «Заказы» названия грузоотправителя вместо его номера следует создать отдельную таблицу «Доставка», в которой сохраняются названия грузоотправителей, а затем сделать поле «Доставка» в таблице «Заказы» подстановочным полем, в котором выводятся данные из таблицы «Доставка».

Допускается также связывание группы с выражением или создание свободной группы. Свободная группа может использоваться в специальном диалоговом окне для приема действий пользователя и выполнения действий, зависящих от выбора пользователя.

Кнопка

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlCommandButtonControlC"}

Элемент управления в форме или отчете, который используется для запуска макрокоманды или набора макрокоманд. Например, можно создать кнопку, открывающую другую форму. Для того чтобы указать действия, выполняемые при нажатии кнопки, следует создать макрос или процедуру обработки события и связать этот макрос или процедуру со свойством **Нажатие кнопки (OnClick)** этой кнопки.

Элемент: **Инструмент:**

Дополнительные сведения

Пользователь имеет возможность указать с помощью свойства **Подпись (Caption)** текст подписи на кнопке или с помощью свойства **Рисунок (Picture)** поместить на кнопку рисунок.

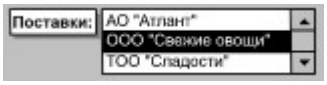
Совет. Мастер по созданию кнопок позволяет создать более 30 различных типов кнопок. С помощью мастера кнопок создается кнопка и определенная для нее процедура обработки события.

Список

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlListBoxControlC"}

Элемент управления, предоставляющий список значений для выбора. Во многих случаях удобно выбирать значение из списка, а не вводить его в поле самостоятельно. Кроме того, при выборе значения из списка гарантируется, что значение вводится без ошибок.

Элемент:



Инструмент:

Список образуется из данных, распределенных по строкам. Строки образуются из одного или нескольких столбцов, над которыми могут быть помещены заголовки.

Дополнительные сведения

Если список, содержащий несколько столбцов, является присоединенным, то Microsoft Access сохраняет значения только из одного указанного столбца.

Значение, выбранное в свободном списке, может быть использовано в другом элементе управления. Например, допускается использование свободного списка для отбора значений в другом списке или в специальном диалоговом окне. Кроме того, значение, выбранное в свободном списке, может быть использовано в условии отбора записей.

Если в форме недостаточно места для вывода списка, или если необходим не только выбор значений из списка, но и ввод новых значений, то вместо элемента управления-списка следует использовать поле со списком.

Линия

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlLineControlC"}

Элемент управления, который изображается в форме или отчете как горизонтальная, вертикальная или наклонная прямая линия.

Элемент:

Инструмент: 

Дополнительные сведения

Для изменения толщины линии используется кнопка **Толщина линии/границы**  . Кнопка **Цвет линии/границы**  позволяет изменить цвет линии или сделать линию прозрачной. Тип линии (точечная, пунктирная и т.д.) определяется значением свойства элемента управления **Тип границы (BorderStyle)**.

Прямоугольник

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlRectangleControlC"}

Элемент управления, который изображается в форме или отчете как прямоугольник.

Элемент: **Инструмент:**

Дополнительные сведения

Прямоугольник и размещенные внутри него элементы управления можно перемещать как единое целое. Для этого следует протащить указатель по диагонали через весь прямоугольник. Будут выделены сам прямоугольник и все содержащиеся в нем элементы управления, после чего все вместе можно переместить в новое положение.

Присоединенная рамка объекта

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acctlBoundObjectFrameControlC"}

Присоединенная рамка объекта предназначена для отображения рисунка, диаграммы или любого другого объекта OLE, сохраняемого в таблице базы данных Microsoft Access. Например, присоединенная рамка объекта позволяет вывести в форме или отчете фотографии сотрудников, сохраненные в таблице Microsoft Access.

Элемент:

Photo

Инструмент:

Дополнительные сведения

Данный элемент управления позволяет создавать и изменять объекты в форме или отчете с помощью сервера OLE.

Присоединенная рамка объекта присоединена к полю в базовой таблице.

Поле в базовой таблице, с которым связана присоединенная рамка объекта, должно иметь тип данных «Поле объекта OLE».

Для различных записей в присоединенной рамке объекта выводятся разные объекты. Присоединенная рамка объекта позволяет выводить как связанные, так и внедренные объекты. Для вывода объектов, не сохраненных в базовой таблице, следует использовать свободную рамку объекта или элемент управления-рисунок..

Диаграмма

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlChartControlC"}

Элемент управления-диаграмма предназначен для создания в форме или отчете внедренной диаграммы, в которой выводятся данные из Microsoft Access. Допускается изменение диаграммы непосредственно в форме или отчете с помощью приложения Microsoft Graph.

Элемент:

Дополнительные сведения

Когда пользователь помещает элемент управления-диаграмму в форму или отчет, вызывается мастер по созданию диаграмм, упрощающий выполнение этой задачи.

Примечание. Для того чтобы разместить в форме или отчете Microsoft Access внедренную или связанную диаграмму, в которой представлены данные из другого приложения, следует использовать свободную рамку объекта или присоединенную рамку объекта.

Поле со списком

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlComboBoxControlC"}

Поле со списком является составным элементом управления, объединяющим элемент управления поле и элемент управления список. Данный элемент управления позволяет вводить значение с клавиатуры или выбирать значение из предварительно созданного списка.

Элемент:

Инструмент:

Дополнительные сведения

В режиме формы список поля со списком не выводится до тех пор, пока не будет нажата кнопка раскрытия списка.

Если мастера элементов управления включены, а затем нажата кнопка **Поле со списком**, то элемент управления «Поле со списком» создается с помощью мастера. Для включения или отключения мастеров по созданию элементов управления следует нажать кнопку **Мастера элементов**  на панели элементов.

Возможность вводить в поле значения, отсутствующие в списке, определяется значением свойства **Ограничиться списком (LimitToList)**.

Список может состоять из одного или нескольких столбцов; при желании вверху этих столбцов могут быть размещены заголовки.

Конец страницы

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlPageBreakControlC"}

Элемент управления-конец страницы указывает начало нового экрана или начало печатной страницы в форме или отчете.

Элемент: **Инструмент:**

Дополнительные сведения

В режиме формы элемент управления конец страницы является активным только в том случае, если свойство формы **Режим по умолчанию (DefaultView)** имеет значение «Простая форма». Конец страницы не влияет на отображение формы в режиме таблицы.

В режиме формы для перехода на предыдущий или следующий конец страницы следует нажать клавишу PAGE UP или PAGE DOWN соответственно.

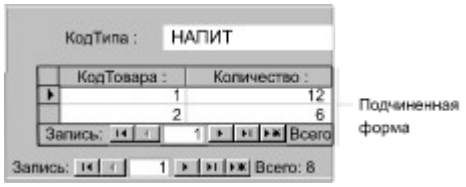
На одном уровне с концом страницы не должны находиться другие элементы управления; в противном случае данные, выводящиеся в таком элементе управления, могут оказаться напечатанными на разных страницах.

Подчиненная форма/отчет

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlEmbeddedFormRptControlC"}

Элемент управления подчиненная форма/отчет предназначен для внедрения формы в форму или отчета в отчет.

Элемент:



Инструмент:

Форма вместе с содержащейся в ней подчиненной формой (составная форма), может использоваться для представления отношения «один-ко-многим», например, между одним типом товаров и относящимися к нему товарами. В этом случае в главной форме может выводиться код, название и описание типа товара, а в подчиненной форме сведения о товарах, относящихся к указанному типу.

Совет. Вместо создания главной формы с последующим добавлением в нее подчиненной формы удобно одновременно создать составную форму с помощью мастера. Кроме того, имеется способ создания подчиненной формы или подчиненного отчета с помощью мыши. Для этого достаточно перетащить существующую форму или существующий отчет из окна базы данных в главную форму или главный отчет.

Свободная рамка объекта

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlUnboundObjectFrameControlC"}

Элемент управления свободная рамка объекта предназначен для изображения рисунка, диаграммы или любого другого объекта OLE, не сохраненного в таблице базы данных Microsoft Access. Например в свободную рамку объекта можно поместить диаграмму, созданную и сохраненную в приложении Microsoft Graph.

Элемент: **Инструмент:**

Дополнительные сведения

Данный элемент управления позволяет создавать и изменять объекты в форме или отчете Microsoft Access с помощью приложения, в котором был создан объект.

Для вывода объектов, сохраненных в базе данных Microsoft Access, следует использовать присоединенную рамку объекта.

В свободной рамке объекта в каждой записи выводится один и тот же объект.

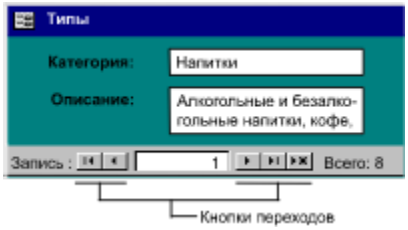
Свободная рамка объекта позволяет выводить как связанные, так и внедренные объекты.

Совет. Неприсоединенные рисунки могут быть выведены в формах или отчетах в свободной рамке объекта или в элементе управления-рисунке. Преимуществом свободной рамки объекта является возможность изменять объект непосредственно из формы или отчета. Преимуществом элемента управления-рисунка является более быстрое отображение рисунка.

Форма

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acconFormC"}

Форма является объектом базы данных Microsoft Access, в котором размещаются элементы управления, предназначенные для выполнения действий, а также для ввода данных в поле, их отображения или изменения.



Раздел формы

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconFormSectionC"}

Раздел формы является частью формы, например, заголовок, примечание или область данных.

Задаваемые пользователем значения свойств раздела формы являются атрибутами формы и определяют оформление и характеристики данного раздела. Например, свойство **Расширение (CanGrow)** позволяет указать, будет ли раздел при печати увеличен в размере по вертикали таким образом, чтобы вместить все содержащиеся в разделе данные. Значения свойств разделов задаются в режиме конструктора формы.

Отчет

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconReportC"}

Отчет является объектом базы данных Microsoft Access, предназначенным для представления данных, отбираемых и форматируемых в соответствие с заданными пользователем спецификациями. С помощью отчетов печатаются сводки продаж, телефонные справочники или почтовые наклейки.

Раздел отчета

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconReportSectionC"}

Раздел отчета является частью отчета, например, заголовок, примечание или область данных.

Задаваемые пользователем значения свойств раздела отчета являются атрибутами отчета и определяют оформление и характеристики данного раздела. Например, свойство **Расширение (CanGrow)** позволяет указать, будет ли раздел при печати увеличен в размере по вертикали таким образом, чтобы вместить все содержащиеся в разделе данные. Значения свойств разделов задаются в режиме конструктора отчета.

Индекс

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconIndexC"}

Создание индексов для таблиц Microsoft Access ускоряет операции поиска и сортировки. Для того чтобы создать индексы для полей, следует в режиме конструктора таблицы открыть окно индексов или задать для свойства поля таблицы **Индексированное поле (Indexed)** значение «Да».

Примечание. Для ключевых полей таблиц индексы создаются автоматически. Не допускается создание индексов для полей с типами данных «Поле MEMO» или «Поле объекта OLE».

Рисунок

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlImageControlC"}

Элемент управления-рисунок используется для вывода рисунков в форме или отчете. Например, в отчете «Счет» в таком элементе управления выводится эмблема фирмы.

Элемент: **Инструмент:**

Дополнительные сведения

Неприсоединенные рисунки могут быть выведены в элементе управления-рисунке или в свободной рамке объекта. Преимуществом элемента управления-рисунка является более быстрое отображение рисунка. Преимуществом свободной рамки объекта является возможность изменять объект непосредственно из формы или отчета.

Присоединенный элемент управления

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlBoundControlC"}

Присоединенным элементом управления называется элемент управления в форме или отчете, который получает свое содержимое из поля базовой таблицы, запроса или инструкции SQL. (Значением свойства **Данные (ControlSource)** такого элемента управления является имя поля таблицы, запрос или инструкция SQL). Например, поле, в котором выводится фамилия сотрудника, является присоединенным к полю «Фамилия» в таблице «Сотрудники».

Окно «Сортировка и группировка»

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconSortingGroupWindowC"}

Окно **Сортировка и группировка** используется для указания порядка сортировки данных, а также для определения уровней группировки в отчетах.

Для того чтобы открыть окно **Сортировка и группировка**, нажмите кнопку **Сортировка и группировка**  на панели инструментов.

Таблица

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconTableC"}

Таблицы являются основными структурными элементами системы управления реляционными базами данных. В Microsoft Access таблица является объектом, в котором данные сохраняются в формате записей (строк) и полей (столбцов). В отдельную таблицу обычно помещают однотипные данные, например, сведения о сотрудниках или заказах.

Построитель палитры

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconColorBuilderC"}

Построитель палитры является средством Microsoft Access, позволяющим пользователю выбирать цвета на палитре или создавать собственные цвета, которые затем могут быть сохранены и использованы в формах или отчетах.

Для вызова данного построителя нажмите кнопку **Построить**  рядом с ячейкой соответствующего свойства в окне свойств.

Поле запроса

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconQueryFieldsC"}

В поле запроса выводятся данные из таблицы, связанной с этим запросом. По умолчанию поле запроса наследует все свойства поля базовой таблицы или базового запроса. Например, если в режиме конструктора таблицы для поля «ДатаРазмещения» в свойстве **Формат поля (Format)** задан «Средний формат даты», то поле «ДатаРазмещения» в запросе также получит средний формат даты. Поскольку свойства базовых полей наследуются по умолчанию, эти свойства не выводятся в окне свойств запроса.

Если в режиме конструктора базовой таблицы значение свойства поля будет изменено, то поле запроса автоматически получит измененное значение свойства. Однако, если пользователь изменяет значение свойства поля в запросе, то это значение имеет приоритет над значением свойства поля базовой таблицы. После явного изменения значения свойства поля в запросе изменения свойства базового поля не отражаются в запросе.

Значения свойств полей запроса задаются в окне свойств полей, которое открывается в режиме конструктора запроса.

Для того чтобы в режиме конструктора запроса открыть окно свойств для поля запроса, следует нажать кнопку **Свойства**  на панели инструментов окна конструктора запросов.

Поле таблицы

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconTableFieldsC"}

Поля таблицы содержат данные, являющиеся компонентами записи. Пользователь имеет возможность определять формат отображения данных, указывать значения по умолчанию и ускорять операции поиска и сортировки, задавая значения свойств полей в области «Свойства поля» в режиме конструктора таблицы.

В Microsoft Access свойства полей используются при просмотре или изменении данных пользователем. Например, заданные пользователем значения свойств **Формат поля (Format)**, **Маска ввода (InputMask)** и **Подпись (Caption)** определяют вид базы данных таблицы и запроса. Элементы управления в новых формах и отчетах, присоединенные к полям таблицы, наследуют эти свойства полей базовой таблицы по умолчанию. Другие свойства позволяют определить условия на значения полей или задать обязательный ввод данных в поле. Microsoft Access будет проверять выполнение этих условий при каждом добавлении или изменении данных в таблице.

Для того чтобы открыть таблицу в режиме конструктора, следует перейти в окно базы данных, выбрать вкладку **Таблицы**, выделить нужную таблицу и нажать кнопку **Конструктор**.

Запрос на изменение

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlActionQueriesC"}

В запросах на изменение выполняется изменение или копирование данных. В число запросов на изменение входят запросы на добавление, удаление и обновление записей, а также на создание таблицы. В отличие от запросов на изменение, в запросах на выборку данные не копируются и не изменяются; в этих запросах проверяются условия отбора и возвращается набор записей, удовлетворяющих этим условиям.

Запрос на:	**Описание**
Добавление записей	Добавляет набор записей, отобранных в запросе, в конец существующей таблицы.
Удаление записей	Удаляет набор записей, удовлетворяющих указанным условиям отбора.
Создание таблицы	Создает новую таблицу на основе набора записей, отобранных в запросе.
Обновление записей	Изменяет набор записей в соответствии с указанными условиями.

Перекрестный запрос

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlCrosstabQueriesC"}

В перекрестном запросе выполняются статистические расчеты на основе данных из всех строк и столбцов. В таких запросах вычисляется сумма, среднее, число значений или другие итоговые значения по записям, после чего выполняется группировка результатов по двум параметрам, один из которых определяет заголовки строк таблицы, а другой – заголовки столбцов. Примером перекрестного запроса является запрос «Квартальные обороты по товарам» из учебной базы данных «Борей».

Заказы по кварталам : перекрестный

Марка	Код клиента	Кв1
Alice Mutton	Antonio Moreno	
Alice Mutton	Berglunds snabi	312 00
Alice Mutton	Bólido Comidas	

Запрос с параметрами

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlParameterQueriesC"}

Запрос с параметрами представляет собой запрос на выборку с расширенными возможностями задания условий. В таких запросах пользователь имеет возможность при каждом запуске запроса вводить параметры, определяющие условия отбора в запросе. Строго говоря, запрос с параметрами не является новым типом запроса; это, скорее, запрос определенного типа с расширенными функциональными характеристиками.

Запрос на выборку

{ewc HLP95EN.DLL,DYNALINK,"пїSпїS. пїSпїSпїSпїSпїS":"acctlSelectQueriesC"}

В запросе на выборку в соответствии с заданными условиями отбираются данные, сохраненные в базе данных пользователя. Результирующий набор записей возвращается без изменения данных. Только после вывода результирующего набора записей на экран становится возможным просмотр и, в некоторых случаях, изменение данных в базовых таблицах. В отличие от этого, при выполнении запроса на изменение данные пользователя могут изменяться.

Специфические запросы SQL

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlSQLSpecificQueriesC"}

Специфическими запросами SQL называются запросы, для создания которых необходимо составить инструкцию SQL в режиме SQL. К запросам SQL относятся запросы на объединение, запросы к серверу и управляющие запросы.

Тип запроса SQL	Описание
На объединение	Специфический запрос SQL на выборку, в котором в одном поле объединяются данные, отбираемые из соответствующих полей двух или нескольких таблиц или запросов. Например, в запросе на объединение таблиц «Клиенты» и «Поставщики» создается статический набор записей, в который включаются все записи, отобранные в таблицах «Клиенты» и «Поставщики».
Запрос к серверу	Специфический запрос SQL, в котором команды передаются на сервер базы данных SQL (например, Microsoft SQL Server). Запросы к серверу позволяют работать с таблицами, сохраняемыми непосредственно на сервере, без связывания этих таблиц с собственной базой данных Microsoft Access.
Управляющий запрос	Специфические запросы SQL, в которых создаются или удаляются индексы, а также создаются, изменяются или удаляются таблицы.

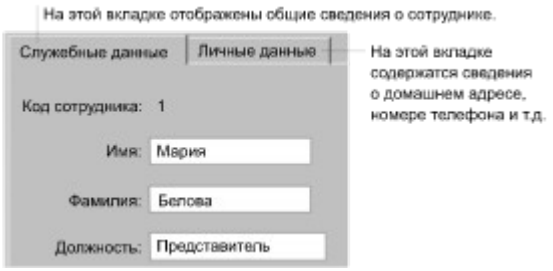
Элемент управления «Набор вкладок»

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acctlTabControlC"}

Элемент управления «Набор вкладок» содержит страницы, на которые помещаются другие элементы управления, например, поля или переключатели. При выборе пользователем одной из вкладок активизируется соответствующая ей страница.

Элементы управления «Набор вкладок» используются для построения простых форм и диалоговых окон, содержащих несколько различных вкладок. При этом на каждой странице набора вкладок однотипные параметры или данные можно объединить в группы. Например, с помощью элемента управления «Набор вкладок» в форме «Сотрудники» можно отделить общую информацию от персональных сведений о сотрудниках.

Элемент:



Инструмент:

Элемент ActiveX

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlActiveXControlC"}

Кроме встроенных элементов управления, которые выводятся на панели элементов, Microsoft Access поддерживает элементы ActiveX (ранее называемые специальными элементами управления или элементами управления OLE). Подобно встроенному элементу управления, элемент ActiveX помещается в форму для расширения возможностей взаимодействия пользователя с приложением. Для элементов ActiveX возникают события, и они могут входить в состав других элементов управления. Имена файлов элементов ActiveX имеют расширение .ocx. Примером элемента ActiveX может служить элемент управления-календарь.

Элемент:

Инструмент:

Объект Page

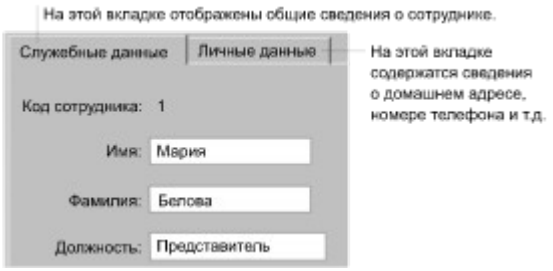
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acctlPageC"}

Объект **Page** соответствует отдельной странице элемента управления «Набор вкладок»

Объект **Page** может содержать один или несколько элементов управления, таких как поля или переключатели. Элементы управления объекта **Page** принадлежат к семейству **Controls** этого объекта. Для выполнения операций с некоторым элементом управления объекта **Page** необходимо создать ссылку на этот элемент управления как компонент семейства **Controls** объекта **Page**.

Элементы управления «Набор вкладок» используются для построения простых форм и диалоговых окон, содержащих несколько различных вкладок. При этом на каждой странице набора вкладок однотипные параметры или данные можно объединить в группы. Например, с помощью элемента управления «Набор вкладок» в форме «Сотрудники» можно отделить общую информацию от персональных сведений о сотрудниках.

Элемент:



Инструмент:

Элементы языка Microsoft Access, применимые к элементам ActiveX

{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїS. nїSnїSnїSnїSnїS":"acccAccessLanguageElementsC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acccAccessLanguageElementsX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acccAccessLanguageElementsA"}

Наряду со свойствами, методами и событиями, относящимися только к элементу ActiveX, ряд общих свойств, методов и событий Microsoft Access также являются применимыми к таким элементам. Для получения дополнительных сведений об этих элементах языка см. соответствующие разделы справки Microsoft Access.

Свойства

Следующие свойства Microsoft Access применимы к элементам ActiveX и доступны в окне свойств:

Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Класс (Class)
Данные (ControlSource) (только для элементов ActiveX, поддерживающих присоединение)
Всплывающая подсказка (ControlTipText)
Режим вывода (DisplayWhen)
Доступ (Enabled)
Идентификатор справки (HelpContextID)
Высота (Height)
От левого края (Left)
Блокировка (Locked) (только для элементов ActiveX, поддерживающих присоединение)
Имя (Name)
Объект (Object)
Класс OLE (OLEClass)
Вход (OnEnter)
Выход (OnExit)
Получение фокуса (OnGotFocus)
Потеря фокуса (OnLostFocus)
При обновлении (OnUpdated)
Оформление (SpecialEffect)
Индекс перехода по Tab (TabIndex)
Переход по Tab (TabStop)
Дополнительные сведения (Tag)
От верхнего края (Top)
Команда (Verb)
Вывод на экран (Visible)
Ширина (Width)

Следующие свойства Microsoft Access применимы к элементам ActiveX и доступны только в макросах или из Visual Basic:

ControlType
EventProcPrefix
InSelection
ObjectVerbsCount
Parent
Section

События

К элементам ActiveX применяются следующие события Microsoft Access:

Вход (Enter)
Выход (Exit)
Получение фокуса (GotFocus)
Потеря фокуса (LostFocus)
При обновлении (Updated)

Методы

К элементам ActiveX применяются следующие методы Microsoft Access:

SetFocus
SizeToFit

Специальное диалоговое окно свойств элемента ActiveX

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acccViewCustomControlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acccViewCustomControlPropertiesX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acccViewCustomControlPropertiesA"}

Значения свойств элемента ActiveX , могут быть заданы в специальном диалоговом окне свойств. Определение свойств элемента ActiveX в этом специальном диалоговом окне является альтернативой указанию их значений в окне свойств Microsoft Access в режиме конструктора.

Два способа определения свойств

Необходимость использования специального диалогового окна свойств объясняется тем, что не все приложения, допускающие вставку элементов ActiveX, имеют окно свойств, аналогичное окну свойств Microsoft Access. Специальное диалоговое окно свойств обеспечивает интерфейс, через который задаются значения основных свойств специального элемента управления, вне зависимости от интерфейса используемого приложения.

Для некоторых свойств элемента ActiveX у пользователя имеются две возможности определить это свойство:

- в окне свойств Microsoft Access.
- в специальном диалоговом окне свойств элемента ActiveX.

В некоторых случаях специальное диалоговое окно свойств является единственным средством определения значения свойства в режиме конструктора. Обычно такая ситуация возникает, когда интерфейс, требуемый для определения значения свойства, является несовместимым с окном свойств Microsoft Access. Например, свойство **GridFont** календаря имеет несколько аргументов, тогда как окно свойств Microsoft Access позволяет определить только один аргумент.

Поиск специального диалогового окна свойств

Не все элементы ActiveX имеют специальное диалоговое окно свойств. Для того чтобы узнать, существует ли специальное окно свойств, следует найти свойство **Специальное (Custom)** в окне свойств Microsoft Access этого элемента. Наличие свойства **Специальное** в окне свойств свидетельствует о существовании специального диалогового окна свойств.

Работа в специальном диалоговом окне свойств

Для того чтобы открыть специальное диалоговое окно свойств, выберите в окне свойств Microsoft Access свойство **Специальное (Custom)** и нажмите кнопку построителя ... справа от ячейки этого свойства. Специальное окно свойств обычно выводится как диалоговое окно, содержащее несколько вкладок. Пользователь должен выбрать вкладку, содержащую нужные элементы.

После выбора параметров на одной вкладке обычно имеется возможность немедленно применить их нажатием кнопки **Применить**. После этого можно выбрать другую вкладку и определить другие свойства. Для того чтобы подтвердить все изменения настроек, заданные в специальном диалоговом окне свойств, нажмите кнопку **OK**. Для возвращения в окно свойств Microsoft Access без изменения значений какого-либо из свойств следует нажать кнопку **Отмена**.

Специальное диалоговое окно свойств открывается также командой **Свойства** из подменю **Объект** элемента ActiveX (например, **Объект-Календарь**) в меню **Правка** или при выборе этой же команды в контекстном меню элемента ActiveX. Кроме того, в окне свойств Microsoft Access специального элемента управления для некоторых свойств календаря, таких как **GridFontColor** выводится кнопка построителя. При нажатии кнопки построителя на экран выводится соответствующая вкладка специального диалогового окна свойств (например, **Цвета**).

Окно «О программе» элемента ActiveX

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acccViewCustomControlAboutBoxC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acccViewCustomControlAboutBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acccViewCustomControlAboutBoxA"}

Для того чтобы открыть для элемента ActiveX диалоговое окно **О программе**, содержащее сведения о номере версии и авторских правах, выберите в окне свойств Microsoft Access ячейку свойства **О программе (About)** и нажмите кнопку построителя  справа от ячейки свойства.

Свойства, методы и события элемента ActiveX

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acccUsingCustomControlLanguageElementsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acccUsingCustomControlLanguageElementsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acccUsingCustomControlLanguageElementsA"}

Данный раздел содержит сведения об использовании элементов языка элемента ActiveX, а также краткое описание новых характеристик элементов ActiveX в Microsoft Access 97.

Общие сведения

Для элементов ActiveX в Microsoft Access 97 больше не является обязательным синтаксис с использованием свойства **Object** при ссылках на свойства и методы в выражениях. Например, следующие две программные строки теперь являются полностью эквивалентными:

```
ActiveXCtl0.Object.MonthLength = 0
```

```
ActiveXCtl0.MonthLength = 0
```

Единственным исключением из этого правила является случай, когда элемент языка элемента ActiveX совпадает с элементом языка Microsoft Access, который также является применимым к элементу управления. В этом случае использование синтаксиса **.Object** является обязательным.

Переменные элемента ActiveX могут быть определены как переменные типа **Control**; но не как переменные особого вида, такого как **Calendar**.

Свойства

Свойства элемента ActiveX теперь выводятся в окно свойств Microsoft Access. Значения большинства этих свойств могут быть заданы как в специальном диалоговом окне свойств элемента ActiveX, так и в окне свойств Microsoft Access.

Примечание. Свойства элемента ActiveX не включаются в семейство **Properties**. Единственным способом доступа к значениям этих свойств является использование синтаксиса, описанного в предыдущем разделе.

События

В ответ на события Microsoft Access, такие как событие **Выход (Enter)**, которые являются применимыми к элементам ActiveX, пользователь имеет возможность запустить макрос или процедуру обработки события. Для этого следует указать в качестве значения свойства события имя макроса или выбрать элемент «[Процедура обработки событий]». Однако для событий, являющихся специфическими для элементов ActiveX, таких как событие календаря **NewMonth**, необходимо перейти в окно модуля формы в режиме конструктора, выбрать имя элемента ActiveX в поле со списком **Объект**, а затем выбрать имя события в поле со списком **Процедура**. Это приведет к созданию процедуры обработки события, которая будет выполняться при возникновении события элемента ActiveX. Допускается также копирование созданной процедуры для ее использования в качестве шаблона при самостоятельном создании других процедур обработки события.

Если существует событие элемента ActiveX, с именем, совпадающим с именем события Microsoft Access, которое также может возникать для элемента ActiveX, то Microsoft Access добавляет слово «Object» к имени события специального элемента управления в раскрывающемся списке **Процедура**. Это позволяет запускать разные процедуры обработки таких событий.

Сравнение типов данных

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acdatComparisonDataTypesC;daproFieldSize;daproType;dasqlJetDataTypes;vagrpDataType":1}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acdatComparisonDataTypesX":1} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïSnïS":"acdatComparisonDataTypesS"}

В ядре базы данных Microsoft Jet распознается несколько перекрывающихся наборов типов данных. В Microsoft Access указание типов данных выполняется в четырех контекстах – в режиме конструктора таблицы,в диалоговом окне **Параметры запроса**, в конструкциях Visual Basic и в режиме SQL запроса.

В следующей таблице сравниваются пять наборов типов данных, распознаваемых в этих контекстах. В первом столбце перечисляются значения свойства **Type** (типы данных), доступные в режиме конструктора таблицы, и пять значений свойства **Размер поля (FieldSize)**, доступных для поля с типом «Числовой». Во втором столбце перечислены соответствующие типы данных параметров запроса, доступные в режиме конструктора для запросов с параметрами в диалоговом окне **Параметры запроса**. В третьем столбце перечисляются соответствующие типы данных Visual Basic. В четвертом перечислены типы данных объекта доступа к данным (DAO) **Field**, а в пятом – соответствующие типы данных SQL ядра базы Jet, определенные в ядре Jet, и допустимые синонимы к ним.

Поля таблиц	**Параметры запроса**	**Visual Basic**	**Константы свойства Type объекта Field**	**Язык SQL ядра базы данных Microsoft Jet и синонимы**
Не поддерживается	Двоичный	*Не поддерживается*	*Не поддерживается*	BINARY (см. Примечания) (синоним: VARBINARY)
Логический	Логический	**Boolean**	**dbBoolean**	BOOLEAN (синонимы: BIT, LOGICAL, LOGICAL1, YESNO)
Числовой (**Размер поля** = Байт)	Байт	**Byte**	**dbByte**	BYTE (синоним: INTEGER1)
Счетчик (**Размер поля** = Длинное целое)	Длинное целое	**Long**	**dbLong**	COUNTER (синоним: AUTOINCREMENT)
Денежный	Денежный	**Currency**	**dbCurrency**	CURRENCY (синоним: MONEY)
Дата/время	Дата/время	**Date**	**dbDate**	DATETIME (синонимы: DATE, TIME, TIMESTAMP)
Числовой (**Размер поля** = С плавающей точкой (8 байт))	С плавающей точкой (8 байт)	**Double**	**dbDouble**	DOUBLE (синонимы: FLOAT, FLOAT8, IEEEDOUBLE, NUMBER, NUMERIC)
Числовой или Счетчик (**Размер поля** = Код репликации)	Код репликации	*Не поддерживается*	**dbGUID**	GUID
Числовой	Длинное	**Long**	**dbLong**	LONG (см. Примечания)

(**Размер поля** = Длинное целое)	целое			(синонимы: INT, INTEGER, INTEGER4)
Поле объекта OLE	Поле объекта OLE	**String**	**dbLongBinary**	LONGBINARY (синонимы: GENERAL, OLEOBJECT)
Поле MEMO	Поле MEMO	**String**	**dbMemo**	LONGTEXT (синонимы: LONGCHAR, MEMO, NOTE)
Числовой (**Размер поля** = С плавающей точкой (4 байт))	С плавающей точкой (4 байт)	**Single**	**dbSingle**	SINGLE (синонимы: FLOAT4, IEEESINGLE, REAL)
Числовой (**Размер поля** = Целое)	Целое	**Integer**	**dbInteger**	SHORT (см. Примечания) (синонимы: INTEGER2, SMALLINT)
Текстовый	Текстовый	**String**	**dbText**	TEXT (синонимы: ALPHANUMERIC, CHAR, CHARACTER, STRING, VARCHAR)
Гиперссылка	Поле MEMO	**String**	**dbMemo**	LONGTEXT (синонимы: LONGCHAR, MEMO, NOTE)
Не поддерживается	Значение	**Variant**	*Не поддерживается*	VALUE (см. Примечания)

Примечания

- Собственно в Microsoft Access тип данных BINARY не используется. Он поддерживается только для запросов по связанным таблицам из других СУБД, в которых поддерживается тип данных BINARY.
- Тип данных INTEGER в языке SQL ядра базы данных Jet не совпадает с типом **Integer** Visual Basic или с типом «Целое» для поля таблицы и параметра запроса. В SQL тип INTEGER соответствует типам **Long** в Visual Basic и «Длинное целое» для поля таблицы и параметра запроса.
- Зарезервированное слово VALUE не представляет тип данных, определенный в ядре базы данных Jet. Однако в Microsoft Access или в запросах SQL слово VALUE рассматривается как допустимый синоним для типа данных Visual Basic **Variant**.
- При определении типа данных для объектов доступа к данным (DAO) (Data Access Objects) в программе Visual Basic необходимо задать значение свойства **Type** объекта доступа к данным.

Примеры преобразования программ для объектов доступа к данным (DAO)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"achowConvertingJetCodeC"}

В Microsoft Access включена версия 3.5 ядра базы данных Microsoft Jet, в которой определен ряд новых объектов, методов и свойств, улучшающих характеристики существующих приложений. В следующих примерах демонстрируется, как следует преобразовать программные конструкции из баз данных Microsoft Access, созданных с предыдущими версиями ядра Jet, чтобы воспользоваться преимуществами новых характеристик.

Создание указателя текущей базы данных

Версия	Пример
1.*x*	`Dim dbs As Database` `Set dbs = CurrentDb`
2.*x*	`Dim dbs As Database` `Set dbs = DBEngine.Workspaces(0).Databases(0)`
или	`Dim dbs As Database` `Set dbs = DBEngine(0)(0)`
7.0 и выше	`Dim dbs As Database` `Set dbs = CurrentDb`

Примечание. В Microsoft Access текущая база данных является используемой по умолчанию базой данных из семейства **Databases**, поэтому по-прежнему разрешается использовать синтаксис `DBEngine(0)(0)` для получения объектной переменной типа **Database**, указывающей на текущую базу данных. Однако предпочтительнее использовать для этого функцию **CurrentDb**, позволяющую создать несколько объектных переменных типа **Database**, указывающих на текущую базу данных.

Открытие другой базы данных и создание указателя

Версия	Пример
1.*x*	`Set dbs = OpenDatabase("Contacts.mdb")`
2.*x* и выше	`Set dbs = DBEngine.Workspaces(0).OpenDatabase("Contacts.mdb")`
или	`Set dbs = Workspaces(0).OpenDatabase("Contacts.mdb")`
или	`Set dbs = OpenDatabase("Contacts.mdb")`

Открытие табличного объекта Recordset

Версия	Пример
1.*x*	`Dim tbl As Table` `Set tbl = dbs.OpenTable("Заказы")`
2.*x*	`Dim rst As Recordset` `Set rst = dbs.OpenRecordset("Заказы", DB_OPEN_TABLE)`
или	`Dim rst As Recordset` `Set rst = dbs!Заказы.OpenRecordset`
7.0 и выше	`Dim rst As Recordset` `Set rst = dbs.OpenRecordset("Заказы", dbOpenTable)`
или	`Dim rst As Recordset` `Set rst = dbs!Заказы.OpenRecordset`

Открытие табличного объекта Recordset для монопольного доступа

Версия	Пример
1.*x*	`Dim tbl As Table` `Set tbl = dbs.OpenTable("Заказы", DB_DENY_READ)`
2.*x*	`Dim rst As Recordset` `Set rst = dbs.OpenRecordset("Заказы", DB_OPEN_TABLE, DB_DENY_READ)`
или	`Dim rst As Recordset` `Set rst = dbs!Заказы.OpenRecordset(DB_OPEN_TABLE, DB_DENY_READ)`
7.0 и выше	`Dim rst As Recordset` `Set rst = dbs.OpenRecordset("Заказы", dbOpenTable, dbDenyRead)`
или	`Dim rst As Recordset` `Set rst = dbs!Заказы.OpenRecordset(dbOpenTable, dbDenyRead)`

Открытие динамического объекта Recordset

Версия	Пример
1.*x*	`Dim dyn As Dynaset` `Set dyn = dbs.CreateDynaset("Заказы")`
2.*x*	`Dim rst As Recordset` `Set rst = dbs.OpenRecordset("Заказы", DB_OPEN_DYNASET)`
или	`Set rst = dbs!Заказы.OpenRecordset(dbOpenDynaset)`
7.0 и выше	`Dim rst As Recordset` `Set rst = dbs.OpenRecordset("Заказы", dbOpenDynaset)`
или	`Set rst = dbs!Заказы.OpenRecordset(dbOpenDynaset)`

Открытие статического объекта Recordset

Версия	Пример
1.*x*	`Dim snp As Snapshot` `Set snp = dbs.CreateSnapshot("Заказы")`
2.*x*	`Dim rst As Recordset` `Set rst = dbs.OpenRecordset("Заказы", DB_OPEN_SNAPSHOT)`
или	`Set rst = dbs!Заказы.OpenRecordset(DB_OPEN_SNAPSHOT)`
7.0 и выше	`Dim rst As Recordset` `Set rst = dbs.OpenRecordset("Заказы", dbOpenSnapshot)`
или	`Set rst = dbs!Заказы.OpenRecordset(dbOpenSnapshot)`

Вывод перечня полей

Версия	Пример

1.*x*	`' Предполагается, что открыт объект Table с именем tbl.`
	`Dim snp As Snapshot`
	`Set snp = tbl.ListFields()`
	`While Not snp.EOF`
	`  Debug.Print snp!Name, snp!Type`
	`  snp.MoveNext`
	`Wend`
	`snp.Close`

2.x

```
' Предполагается, что открыт объект TableDef с именем tdf.
Dim intI As Integer, fld As Field
For intI = 0 To tdf.Fields.Count - 1
   Set fld = tdf.Fields(intI)
   Debug.Print fld.Name, fld.Type
Next intI
```

7.0 и выше

```
' Предполагается, что открыт объект TableDef с именем tdf.
Dim fld As Field
For Each fld in tdf.Fields
   Debug.Print fld.Name, fld.Type
Next fld
```

Вывод перечня таблиц в базе данных

Версия	Пример

1.x

```
' Предполагается, что открыт объект Database с именем dbs.
Dim snp As Snapshot
Set snp = dbs.ListTables()
While Not snp.EOF
   Debug.Print snp!Name
   snp.MoveNext
Wend snp.Close
```

2.x

```
' Предполагается, что открыт объект Database с именем dbs.
Dim intJ As Integer, tdf As TableDef
For intJ = 0 To dbs.TableDefs.Count - 1
   Set tdf = dbs.TableDefs(intJ)
   Debug.Print tdf.Name
Next intJ
```

7.0 и выше

```
' Предполагается, что открыт объект Database с именем dbs.
Dim tdf As TableDef
For Each tdf in dbs.TableDefs
   Debug.Print tdf.Name
Next tdf
```

Вывод перечня таблиц в базе данных

Версия	Пример

1.x

```
' Предполагается, что открыт объект Table с именем tbl.
Dim snp As Snapshot
Set snp = tbl.ListIndexes()
While Not snp.EOF
   Debug.Print snp!Name
   snp.MoveNext
Wend
snp.Close
```

2.x

```
' Предполагается, что открыт объект TableDef с именем tdf.
Dim intI As Integer, idx As Index
For intI = 0 To tdf.Indexes.Count - 1
   Set idx = tdf.Indexes(intI)
   Debug.Print idx.Name
Next intI
```

7.0 и выше

```
' Предполагается, что открыт объект TableDef с именем tdf.
Dim idx As Index
For Each idx in tdf.Indexes
```

```
   Debug.Print idx.Name
Next idx
```

Вывод перечня параметров запроса

Версия	Пример

1.*x*	`' Предполагается, что открыт объект QueryDef с именем qdf.`
	`Dim snp As Snapshot`
	`Set snp = qdf.ListParameters()`
	`While Not snp.EOF`
	`  Debug.Print snp!Name`
	`  snp.MoveNext`
	`Wend`
	`snp.Close`

2.*x*	`' Предполагается, что открыт объект QueryDef с именем qdf.`
	`Dim intI As Integer, prm As Parameter`
	`For intI = 0 to qdf.Parameters.Count - 1`
	`  Set prm = qdf.Parameters(i)`
	`  Debug.Print prm.Name`
	`Next intI`

7.0 и выше	`' Предполагается, что открыт объект QueryDef с именем qdf.`
	`Dim prm As Parameter`
	`For Each prm in qdf.Parameters`
	`  Debug.Print prm.Name`
	`Next prm`

Динамическое указание параметров запроса

Версия	Пример

1.*x*	`' Предполагается, что открыт объект Database с именем dbs.`
	`Dim qdf As QueryDef, dyn As Dynaset`
	`Set qdf = dbs.OpenQueryDef("ЗапросЗаказы")`
	`qdf.ParameterName = "[Начальная дата]"`
	`Set dyn = qdf.CreateDynaset()`

2.*x*	`' Предполагается, что открыт объект QueryDef с именем qdf.`
	`Dim intI As Integer, prm As Parameter, rst As Recordset`
	`For intI = 0 To qdf.Parameters.Count - 1`
	`  Set prm = qdf.Parameters(intI)`
	`  prm.Value = InputBox("Значение для " & prm.Name)`
	`Next intI`
	`Set rst = qdf.OpenRecordset`

7.0 и выше	`' Предполагается, что открыт объект QueryDef с именем qdf.`
	`Dim prm As Parameter, rst As Recordset`
	`For Each prm in qdf.Parameters`
	`  prm.Value = InputBox("Значение для " & prm.Name)`
	`Next prm`
	`Set rst = qdf.OpenRecordset`

Выполнение запроса к серверу

Версия	Пример

1.*x*	`' Предполагается, что открыт объект Database с именем dbs.`
	`Dim lngRows As Long`
	`' Выполняет сохраненную процедуру на сервере.`

```
lngRows = dbs.ExecuteSQL("EXECUTE SP_StoredProc")
```

или

```
Dim snp As Snapshot
Set snp = dbs.CreateSnapshot("EXECUTE SP_StoredProc", DB_SQLPassThrough)
```

2.*x* и выше

```
' Предполагается, что открыт объект Database с именем dbs.
Dim qdf As QueryDef, rst As Recordset
Set qdf = dbs.CreateQueryDef("ODBCQuery")
qdf.Connect = "ODBC;" & "DSN=MyServer;" _
   & "UID=sa;" & "PWD=hithere;" & "DATABASE=pubs"
qdf.SQL = "EXECUTE SP_StoredProc"
qdf.ReturnsRecords = True
.
.
.
Set rst = qdf.OpenRecordset
```

Создание временного запроса

Версия	Пример

1.*x*

```
' Предполагается, что открыт объект Database с именем dbs.
Dim qdf As QueryDef, strTemp As String
strTemp = "TMP:" & User() & Time()
qdf = dbs.CreateQueryDef(strTemp)
qdf.SQL = "UPDATE Table1 SET Table1.Field1 = Field1 * 1.1;"
.
.
.
qdf.Execute
qdf.Close
.
.
.
dbs.DeleteQueryDef(strTemp)
```

2.*x* и выше

```
qdf = dbs.CreateQueryDef("")
qdf.Execute
qdf.Close
```

Новые элементы языка

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconNewLanguageElementsC "}

Microsoft Access включает в себя целый набор новых расширенных объектов, методов, свойств, функций, инструкций, типов данных и событий, которые позволяют создавать мощные приложения баз данных с помощью языка Visual Basic.

Новые объекты

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconNewAccessObjectsC;daconMSJetDatabaseEngine25"}

В следующий список включены новые объекты Microsoft Access 95 и Microsoft Access 97. Объекты, появившиеся в Microsoft Access 97 отмечены звездочкой (*).

Объекты Microsoft Access

Объект **Application**
Семейство **Controls**
Объект **DoCmd**
Семейство **ItemsSelected**
Объект **Module** *
Семейство **Modules** *
Объект **Page***
Семейство **Pages***
Объект **Reference***
Семейство **References***
Объекты SummaryInfo и UserDefined **Document**

Объекты Visual Basic

Объект **Collection**
Объект **Debug**
Объект **Err**

Объекты доступа к данным (DAO)

Объект **Connection**
Семейство **Errors**

Объекты Microsoft Office

Для просмотра списка новых объектов из библиотеки объектов Microsoft Office 8.0 выберите «Справочник по Microsoft Office Visual Basic» в оглавлении справочной системы.

Для использования объектов Microsoft Office необходимо установить ссылку на библиотеку объектов Microsoft Office 8.0 в открытом окне модуля. Для этого выполните команду **Ссылки** из меню **Сервис** и установите флажок, соответствующий этой библиотеке.

Новые методы

{ewc HLP95EN.DLL, DYNALINK, "nïSnïS. nïSnïSnïSnïSnïS":"acconNewMethodsC "}

В следующий список включены новые методы Microsoft Access 95 и Microsoft Access 97. Методы, появившиеся в Microsoft Access 97 отмечены звездочкой (*).

A–H

AccessError
Add (семейство **Pages**)*
AddFromFile (объект **Module**)*
AddFromFile (семейство **References**)*
AddFromGUID*
AddFromString*
AddToFavorites*
ApplyFilter
Beep
BuildCriteria
CancelEvent
CloseCurrentDatabase
CopyObject
CreateEventProc*
DefaultControl*
DefaultWorkspaceClone
DeleteLines*
DeleteObject
DoMenuItem
Dropdown
Echo
Find*
FindNext
FindRecord
Follow*
FollowHyperlink*
GoToControl
GoToPage
GoToRecord
Hourglass

I–Q

InsertLines*
InsertText
Item
Maximize
Minimize
MoveSize
NewCurrentDatabase
OpenCurrentDatabase
OpenForm
OpenModule
OpenQuery
OpenReport
OpenTable
OutputTo
PrintOut
Quit

R–Z

RefreshDatabaseWindow*
RefreshTitleBar
Remove
Remove (семейство **Pages**)*
Remove (семейство **References**)*
Rename
Repaint
ReplaceLine*
Requery
Restore
Run
RunSQL
Save
SelectObject
SendObject
SetMenuItem
SetWarnings
ShowAllRecords
ShowToolbar
SizeToFit
TransferDatabase
TransferSpreadsheet

RunCommand*	**TransferText**
RunMacro	**Undo**

Новые методы объектов доступа к данным (DAO)

Cancel	**NewPassword**
CancelUpdate	**NextRecordset**
CopyQueryDef	**OpenConnection**
GetRows	**SetOption**
MakeReplica	**Synchronize**

Методы Microsoft Office

Для просмотра списка новых методов из библиотеки объектов Microsoft Office 8.0 выберите «Справочник по Microsoft Office Visual Basic» в содержании справочной системы.

Для использования методов Microsoft Office необходимо установить ссылку на библиотеку объектов Microsoft Office 8.0 в открытом окне модуля. Для этого выполните команду **Ссылки** из меню **Сервис** и установите флажок, соответствующий этой библиотеке.

Новые свойства

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconNewPropertiesC "}

В следующий список включены новые свойства Microsoft Access 95 и Microsoft Access 97. Методы, появившиеся в Microsoft Access 97 отмечены звездочкой (*).

A–G

ActiveDatasheet
Разрешить добавление (AllowAdditions)
Разрешить автозамену (AllowAutoCorrect)
AllowBreakIntoCode
AllowBuiltInToolbars
AllowBypassKey
Разрешить удаление (AllowDeletions)
Разрешить изменение (AllowEdits)
AllowFullMenus
AllowShortcutMenus
AllowSpecialKeys
AllowToolbarChanges
AppIcon
Application
AppTitle
Тип границы (BorderStyle)
BuiltIn
Кнопка закрытия (CloseButton)
CodeContextObject
Collection
Всплывающая подсказка (ControlTipText)
ControlType
CountOfDeclarationLines*
CountOfLines*
CurrentRecord
Цикл табуляции (Cycle)
Database properties
Ввод данных (DataEntry)
DatasheetBackColor*
DatasheetCellsEffect
DatasheetForeColor*
DatasheetGridlinesBehavior
DatasheetGridlinesColor
DataType
DBEngine
Описание (Description)
Dirty
Тип элемента управления (DisplayControl)
Разделители записей (DividingLines)
FailOnError*
Фильтр (Filter)
Применение автофильтра (FilterLookup)
Фильтр включен (FilterOn)
Формат поля (Format)
FullPath
Неразрывная группа (GrpKeepTogether)
GUID

H–P

HasContinued
HasData
HasModule*
hWndAccessApp
Hyperlink*
HyperlinkAddress*
HyperlinkSubAddress*
ImageHeight
MousePointer*
MultiRow*
Несвязное выделение (MultiSelect)
NewRecord
Новые значения (NewValues)
Свойства объекта
Порядок сортировки (OrderBy)
Сортировка включена

ImageWidth
InSelection
InsideHeight
InsideWidth
IsVisible*
Перехват нажатия клавиш (KeyPreview)
Kind*
Lines*
ListIndex
Major*
MaxRecords
MDE*
MenuBar
Кнопки размеров окна (MinMaxButtons)
Minor*
(OrderByOn)
PageIndex*
Parent
Рисунок (Picture)
Выравнивание рисунка (PictureAlignment)
PictureData
Страницы с рисунком (PicturePages)
Масштабы рисунка (PictureSizeMode)
Мозаичный рисунок (PictureTiling)
Тип рисунка (PictureType)
ProcBodyLine*
ProcCountLines*
ProcOfLine*
ProcStartLine*
ProjectName*

R–Z

Тип набора записей (RecordsetType)
Повторение раздела (RepeatSection)
ReplicationConflictFunction
Selected
SelHeight
SelLeft
SelTop
SelWidth
ShortcutMenu
Контекстное меню (ShortcutMenuBar)
Оформление (SpecialEffect)
StartupForm
StartupMenuBar
StartupShortcutMenuBar
StartupShowDBWindow
StartupShowStatusBar
Style*
TabFixedHeight*
TabFixedWidth*
Toolbar*
Тройное состояние (TripleState)
UserControl
UseTransaction*
Вывод на экран (Visible)
Visible (Application object)
Кнопка контекстной справки (WhatsThisButton)
WillContinue

Новые свойства объектов доступа к данным (DAO)

AbsolutePosition
AllPermissions
ConflictTable
Connection*
DefaultCursorDriver*
DefaultType*
DefaultUser
HelpContext
Файл Справки (HelpFile)
Inherit
IniPath
KeepLocal
MaxRecords*
RecordsAffected

DesignMasterID
Direction*
DistinctCount
EditMode
Размер поля (FieldSize)*
Replicable
ReplicaID
StillExecuting*
Transactions*

Свойства Microsoft Office

Для просмотра списка новых свойств в библиотеке объектов Microsoft Office 8.0 выберите «Справочник по Microsoft Office Visual Basic» в содержании справочной системы.

Для использования свойств Microsoft Office необходимо установить ссылку на библиотеку объектов Microsoft Office 8.0 в открытом окне модуля. Для этого выполните команду **Ссылки** из меню **Сервис** и установите флажок, соответствующий этой библиотеке.

Новые функции

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconNewFunctionsC "}

Ниже перечислены новые функции Microsoft Access 95 и Microsoft Access 97. Функции, введенные в Microsoft Access 97, отмечены звездочкой (*).

Array
CBool
CByte
CDate
ChrB
CVErr
FileDateTime
FileLen
GetAllSettings
GetAttr
GetSetting
GUIDFromString
HyperlinkPart*
InputB
IsArray
IsError
IsMissing
IsObject
LeftB
LoadPicture
MidB
Nz
RightB
StringFromGUID
TypeName

Новые инструкции

{ewc HLP95EN.DLL, DYNALINK, "пїSпїS. пїSпїSпїSпїSпїS":"acconNewStatementsC "}

Ниже перечислены новые инструкции, которые поддерживает Visual Basic для использования в Microsoft Access. Все они были первоначально введены в Microsoft Access 95.

DeleteSetting
FileCopy
For Each...Next
Option Private
Private
Property Get
Property Let
Property Set
Public
SaveSetting
SetAttr
With

Новые типы данных

{ewc HLP95EN.DLL, DYNALINK, "пїSпїS. пїSпїSпїSпїSпїS":"acconNewDataTypesC "}

Ниже перечислены новые типы данных, которые поддерживает Visual Basic для использования в Microsoft Access. Все они были первоначально введены в Microsoft Access 95.

Boolean

Byte

Date

Новые события

{ewc HLP95EN.DLL, DYNALINK, "пїSпїS. пїSпїSпїSпїSпїS":"acconNewFormReportEventsC "}

Ниже перечислены новые события форм и отчетов Microsoft Access. Все они были первоначально введены в Microsoft Access 95.

Применение фильтра (ApplyFilter)

Фильтр (Filter)

NoData

Page

В Microsoft Access 97 вошли также следующие новые события.

Initialize

ItemAdded

ItemRemoved

Terminate

Семейство Controls

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"accolControlsC;vakeyMe;vastmForEach"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"accolControlsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"accolControlsP"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"accolControlsM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"accolControlsE"}

Семейство **Controls** содержит все элементы управления формы, отчета, или раздела, содержащиеся или присоединенные к другому элементу управления . Семейство **Controls** является компонентом объекта **Form**, **Report**, **Section** или **Control.** Следующий рисунок демонстрирует взаимосвязи между семейством **Controls** и объектами **Form** и **Report**.

Дополнительные сведения

Пользователь имеет возможность перебирать отдельные компоненты семейства **Controls**, подсчитывать их число и задавать значения их свойств. Например, можно перебрать все компоненты семейства **Controls** конкретной формы и задать для каждого из них значение свойства **Высота (Height)**.

Совет Для перечисления компонентов семейства обычно используют конструкцию **For Each...Next**.

Быстрее других выполнятся неявные ссылки на семейство **Controls**, демонстрируемые в следующих примерах, в которых приводятся ссылки на элемент управления «НовыеДанные» в форме «Заказы». Синтаксис `Me!НовыеДанные` обеспечивает саму быструю ссылку на элемент управления.

```
Me!НовыеДанные ' Или Forms!Заказы!НовыеДанные.

Me![Новые данные] ' Прямые скобки для имен с пробелами.

Me("НовыеДанные") ' Выполняется несколько медленнее.
```

Допускаются также ссылки на отдельные элементы управления с помощью явных ссылок на семейство **Controls**.

```
Me.Controls!НовыеДанные ' Или Forms!Заказы.Controls!НовыеДанные.

Me.Controls![Новые данные]

Me.Controls("НовыеДанные")
```

Кроме того, возможны ссылки на элемент управления по его индексу в семействе. Нумерация компонентов семейства **Controls** начинается с нуля.

```
Me(0) ' Ссылка на первый элемент семейства.

Me.Controls(0)
```

Примечание Ключевое слово **Me** может представлять форму или отчет только при ссылках на форму или отчет в программе, находящейся в модуле формы или в модуле отчета. При ссылках из стандартного модуля, а также из модуля другой формы или отчета, необходимо указать для формы или отчета полную ссылку.

Чтобы обратиться к элементу управления раздела формы или отчета, следует использовать свойство **Section,** возвращающее ссылку на объект **Section**, а затем сослаться на семейство **Controls** объекта **Section**.

Два типа объектов **Control**, элементы управления набор вкладок и группа, содержат семейства **Controls**, в которые может входить несколько элементов управления. Семейство **Controls**, принадлежащее группе, содержит все элементы управления из этой группы. Такими элементами могут являться переключатели, флажки, выключатели и надписи.

Элемент управления набор вкладок содержит семейство **Pages**, которое является особой разновидностью семейства **Controls**. Семейству **Pages** принадлежат объекты **Page,** которые также являются элементами управления. Константой свойства **ControlType** для элемента управления **Page** является константа **acPage**. В свою очередь объект **Page** содержит собственное семейство **Controls**, в которое входят все элементы управления на отдельной вкладке.

Остальные объекты **Control** содержат семейство **Controls**, которому могут принадлежать связанные подписи. К таким элементам относятся поле, группа, переключатель, выключатель, флажок, поле со списком, список, кнопка, присоединенная рамка объекта и не присоединенная рамка объекта.

Использование констант в Microsoft Access

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"accstConstantsC;damscDataAccessConstants;vaconUnderstandingScope;vafctVarType;vamscTypeLibConstants"}

Общие сведения

Константа представляет числовое или строковое значение, которое не изменяется при выполнении программы. Константы упрощают чтение программы Visual Basic и облегчают работу с программой. Кроме того, использование встроенных констант гарантирует, что программа будет работать даже при изменении представляемых константами значений в будущих версиях Microsoft Access.

В Microsoft Access поддерживаются следующие три типа констант.

- Символьные константы, которые создаются пользователем с помощью инструкции **Const** и используются в модулях.
- Встроенные константы, определенные в Microsoft Access или в адресуемой библиотеке.
- Системные константы: **True**, **False** и **Null.**

Символьные константы

Очень часто в программах приходится многократно использовать одни и те же значения, причем смысл используемых в программе числовых значений ясен не сразу. В подобных случаях применение символьных или определяемых пользователем констант со значащими именами вместо чисел или строк облегчает работу с программой и делает ее текст более понятным.

После описания константы с помощью инструкции **Const** не разрешается изменять эту константу или присваивать ей новое значение. Не допускается также создание пользователем константы, имя которой совпадает с именем встроенной константы.

Ниже приведены примеры применения инструкции **Const** для описания числовых и строковых констант:

```
Const ПИ = 3.14159265        ' Число ПИ.
Const ПИ2 = ПИ * 2          ' Описание одной константы с помощью другой.
Const ВЕРСИЯ = "Версия 7.0"  ' Описывает строковую константу.
```

Встроенные константы

Помимо констант, которые пользователь описывает с помощью инструкции **Const**, в Microsoft Access автоматически описывается ряд встроенных констант и обеспечивается доступ к константам языка Visual Basic для приложений (Visual Basic for Applications, VBA) и к константам объектов доступа к данным (Data Access Objects, DAO). Кроме того, пользователь имеет возможность использовать константы из других адресуемых библиотек объектов. Подробнее о добавлении ссылок на библиотеки см. в разделе Определение ссылок на библиотеки типов.

Любая встроенная константа может быть использована в макросе или в программе Visual Basic. Эти константы всегда доступны. Конкретные встроенные константы, используемые с определенной функцией, методом или свойством, описываются в разделах справки для данной функции, метода или свойства.

Примечание. Списки встроенных констант из любых доступных библиотек объектов выводятся в окне **Просмотр объектов**.

Первые две буквы в имени встроенной константы являются префиксом, указывающим библиотеку объектов, в которой определена эта константа. Константы из библиотеки Microsoft Access имеют префикс «ac», константы из библиотеки объектов доступа к данным (DAO) имеют

префикс «db», а константы из библиотеки VBA имеют префикс «vb». Например:

- **acForm**
- **dbAppendOnly**
- **vbCurrency**

Примечание. Поскольку конкретные значения, представляемые встроенными константами, могут быть изменены в будущих версиях Microsoft Access, следует всегда использовать в программах сами именованные константы вместо их значений. Пользователь, однако, всегда имеет возможность вывести значение константы. Для этого следует выбрать константу в окне просмотра объектов или ввести инструкцию **?***имяКонстанты* в окне отладки.

Встроенные константы могут использоваться везде, где допускается использование символьных или определяемых пользователем констант, в том числе и в выражениях. В следующем примере встроенная константа **vbCurrency** используется для проверки, относится ли переменная `varNum` типа **Variant** к подтипу **Currency** (денежный), для которого функция **VarType** возвращает значение 6.

```
Dim varNum As Variant
If VarType(varNum) = vbCurrency Then
   Debug.Print "varNum содержит значение денежного типа."
Else
   Debug.Print "varNum не содержит значение денежного типа."
End If
```

Встроенные константы подразделяются на несколько категорий. Для вывода списка встроенных констант, принадлежащих к определенной категории, выберите один из перечисленных ниже разделов:

- Константы макрокоманд
- Константы доступа к данным
- Константы процедур обработки событий
- Константы кодов клавиатуры
- Другие константы
- Константы метода **RunCommand** (являются константами макрокоманд)
- Константы защиты данных
- Константы Visual Basic
- Константы функции **VarType**

Системные константы

Системные константы **True**, **False** и **Null** могут быть использованы в любых компонентах Microsoft Access. Например, допускается использование константы **True** для определения условий в макросе. Следующее выражение окажется истинным, если свойство **Вывод на экран (Visible)** формы «Сотрудники» имеет значение **True** («Да»):

```
Forms!Сотрудники.Visible = True
```

Константа **Null** может использоваться в любых компонентах Microsoft Access. Например, допускается применение константы **Null** для определения свойства **Значение по умолчанию (DefaultValue)** элемента управления в форме с помощью следующего выражения:

```
=Null
```

Константы защиты данных

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"accstSecurityConstantsC;daproPermissions"}

Разрешения на доступ к объектам базы данных можно определить с помощью программы Visual Basic или путем присвоения свойству **Permissions** значения, равного одной из следующих констант:

acSecFrmRptExecute
acSecFrmRptReadDef
acSecFrmRptWriteDef
acSecMacExecute
acSecMacReadDef
acSecMacWriteDef
acSecModReadDef
acSecModWriteDef
dbSecCreate
dbSecDBAdmin
dbSecDBCreate
dbSecDBExclusive
dbSecDBOpen
dbSecDelete
dbSecDeleteData
dbSecFullAccess
dbSecInsertData
dbSecNoAccess
dbSecReadSec
dbSecReadDef
dbSecReplaceData
dbSecRetrieveData
dbSecWriteDef
dbSecWriteOwner
dbSecWriteSec

Примечание. Константы, имена которых начинаются с «acSec», используются для определения разрешений доступа к объектам базы данных только в программах Visual Basic.

Константы процедур обработки событий

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"accstEventProcConstantsC"}

Константы процедуры обработки события «После подтверждения Del» (AfterDelConfirm)

Следующие константы используются в процедурах обработки события **После подтверждения Del (AfterDelConfirm)**.

- **acDeleteCancel**
- **acDeleteOK**
- **acDeleteUserCancel**

Константы процедуры обработки события «До подтверждения Del» (BeforeDelConfirm)

Следующие константы используются в процедурах обработки события **До подтверждения Del (BeforeDelConfirm)**.

- **acDataErrContinue**
- **acDataErrDisplay**

Константы, определяющие маску кнопки

Следующие константы, определяющие маску кнопки, используются в процедурах обработки событий **Кнопка вниз (MouseDown)**, **Кнопка вверх (MouseUp)** и **Перемещение указателя (MouseMove)**.

- **acLeftButton**
- **acMiddleButton**
- **acRightButton**

Константы, определяющие маску клавиши

Следующие константы, определяющие маску клавиши, используются в процедурах обработки событий **Клавиша вниз (KeyDown)**, **Клавиша вверх (KeyUp)**, **Кнопка вниз (MouseDown)**, **Кнопка вверх (MouseUp)** и **Перемещение указателя (MouseMove)**.

- **acAltMask**
- **acCtrlMask**
- **acShiftMask**

Константы процедуры обработки события «Отсутствие в списке» (NotInList)

Следующие константы используются в процедурах обработки события **Отсутствие в списке (NotInList)**.

- **acDataErrAdded**
- **acDataErrContinue**
- **acDataErrDisplay**

Константы процедуры обработки события «Ошибка» (Error)

Следующие константы используются в процедурах обработки события **Ошибка (Error)**.

- **acDataErrContinue**
- **acDataErrDisplay**

Константы процедуры обработки события «При обновлении» (Updated)

Следующие константы используются в процедурах обработки события **При обновлении (Updated).**

- **acOLEChanged**
- **acOLEClosed**
- **acOLERenamed**
- **acOLESaved**

Константы процедуры обработки события «Применение фильтра» (ApplyFilter)

Следующие константы используются в процедурах обработки события **Применение фильтра» (ApplyFilter)**.

- **acApplyFilter**
- **acCloseFilterWindow**
- **acShowAllRecords**

Константы процедуры обработки события «Фильтрация» (Filter)

Следующие константы используются в процедурах обработки события **Фильтрация (Filter)**.

- **acFilterAdvanced**
- **acFilterByForm**

Константы макрокоманд

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"accstMacroActConstantsC"}

Следующие константы используются при вызове методов объекта **DoCmd** в программах Visual Basic. Вызов метода объекта **DoCmd** с одной из этих констант аналогичен выполнению соответствующей макрокоманды в макросе. Следует отметить что метод **DoMenuItem** вышел из употребления. Вместо него в Microsoft Access 97 используется метод **RunCommand** и его константы. Константы, отмеченные в следующем списке звездочкой (*), в настоящее время заменены другими константами, но они все еще могут использоваться с соответствующими методами.

Константа объекта DoCmd	Методы
acActiveDataObject	**GoToRecord**
acAdd	**OpenQuery**, **OpenTable**
acAll	**FindRecord**
acAnywhere	**FindRecord**
acCopy	**DoMenuItem**
acCurrent	**FindRecord**
acCut	**DoMenuItem**
acDataForm	**GoToRecord**
acDataQuery	**GoToRecord**
acDataTable	**GoToRecord**
acDefault	**Close**, **CopyObject**, **DeleteObject**, **Rename**, **RepaintObject**, **Save**
acDelete	**DoMenuItem**
acDesign	**OpenForm**
acDialog	**OpenForm**
acDown	**FindRecord**
acDraft	**PrintOut**
acEdit	**OpenQuery**, **OpenTable**
acEditMenu	**DoMenuItem**
acEntire	**FindRecord**
acExit*	**Quit** (объект **DoCmd**), **Quit** (объект **Application**)
acExport	**TransferDatabase**, **TransferSpreadsheet**
acExportDelim	**TransferText**
acExportFixed	**TransferText**
acExportHTML	**TransferText**
acExportMerge	**TransferText**
acFile	**DoMenuItem**
acFirst	**GoToRecord**
acForm	**Close**, **CopyObject**, **DeleteObject**, **Rename**, **RepaintObject**, **Save**, **SelectObject**, **TransferDatabase**
acFormAdd	**OpenForm**
acFormatActiveXServer	**OutputTo**
acFormatHTML	**OutputTo**, **SendObject**
acFormatIIS	**OutputTo**

acFormatRTF	**OutputTo**, **SendObject**
acFormatTXT	**OutputTo**, **SendObject**
acFormatXLS	**OutputTo**, **SendObject**
acFormBar	**DoMenuItem**
acFormDS	**OpenForm**
acFormEdit	**OpenForm**
acFormPropertySettings	**OpenForm**
acFormReadOnly	**OpenForm**
acGoTo	**GoToRecord**
acHidden	**OpenForm**
acHigh	**PrintOut**
acIcon	**OpenForm**
acImport	**TransferDatabase**, **TransferSpreadsheet**
acImportDelim	**TransferText**
acImportFixed	**TransferText**
acImportHTML	**TransferText**
acLast	**GoToRecord**
acLink	**TransferDatabase**, **TransferSpreadsheet**
acLinkDelim	**TransferText**
acLinkFixed	**TransferText**
acLinkHTML	**TransferText**
acLow	**PrintOut**
acMacro	**Close**, **CopyObject**, **DeleteObject**, **Rename**, **RepaintObject**, **Save**, **SelectObject**, **TransferDatabase**
acMedium	**PrintOut**
acMenuCheck	**SetMenuItem**
acMenuGray	**SetMenuItem**
acMenuUncheck	**SetMenuItem**
acMenuUngray	**SetMenuItem**
acMenuVer1X	**DoMenuItem**
acMenuVer20	**DoMenuItem**
acMenuVer70	**DoMenuItem**
acModule	**Close**, **CopyObject**, **DeleteObject**, **Rename**, **RepaintObject**, **Save**, **SelectObject**, **TransferDatabase**
acNew	**DoMenuItem**
acNewRec	**GoToRecord**
acNext	**GoToRecord**
acNormal	**OpenForm**
acObject	**DoMenuItem**
acObjectUpdate	**DoMenuItem**
acObjectVerb	**DoMenuItem**
acOutputForm	**OutputTo**
acOutputModule	**OutputTo**
acOutputQuery	**OutputTo**

acOutputReport	**OutputTo**
acOutputTable	**OutputTo**
acPages	**PrintOut**
acPaste	**DoMenuItem**
acPreview	**OpenForm**
acPrevious	**GoToRecord**
acPrintAll	**PrintOut**
acPrompt*	**Close**, **Quit** (объект **DoCmd**), **Quit** (объект **Application**)
acQuery	**Close**, **CopyObject**, **DeleteObject**, **Rename**, **RepaintObject**, **Save**, **SelectObject**, **TransferDatabase**
acQuitPrompt	**Quit** (объект **DoCmd**), **Quit** (объект **Application**)
acQuitSaveAll	**Quit** (объект **DoCmd**), **Quit** (объект **Application**)
acQuitSaveNone	**Quit** (объект **DoCmd**), **Quit** (объект **Application**)
acReadOnly	**OpenQuery**, **OpenTable**
acRecordsMenu	**DoMenuItem**
acRefresh	**DoMenuItem**
acReport	**Close**, **CopyObject**, **DeleteObject**, **Rename**, **RepaintObject**, **Save**, **SelectObject**, **TransferDatabase**
acSaveForm	**DoMenuItem**
acSaveFormAs	**DoMenuItem**
acSave*	**Quit** (объект **DoCmd**), **Quit** (объект **Application**)
acSaveNo	**Close**
acSavePrompt	**Close**
acSaveRecord	**DoMenuItem**
acSaveYes	**Close**
acSearchAll	**FindRecord**
acSelectAllRecords	**DoMenuItem**
acSelection	**PrintOut**
acSelectRecord	**DoMenuItem**
acSendForm	**SendObject**
acSendModule	**SendObject**
acSendNoObject	**SendObject**
acSendQuery	**SendObject**
acSendReport	**SendObject**
acSendTable	**SendObject**
acSpreadsheetTypeExcel3	**TransferSpreadsheet**
acSpreadsheetTypeExcel4	**TransferSpreadsheet**
acSpreadsheetTypeExcel5	**TransferSpreadsheet**
acSpreadsheetTypeExcel7	**TransferSpreadsheet**
acSpreadsheetTypeExcel97	**TransferSpreadsheet**

acSpreadsheetTypeLotusWK1	**TransferSpreadsheet**
acSpreadsheetTypeLotusWK3	**TransferSpreadsheet**
acSpreadsheetTypeLotusWK4	**TransferSpreadsheet**
acSpreadsheetTypeLotusWJ2 — Только в версиях для Японии	**TransferSpreadsheet**
acStart	**FindRecord**
acTable	**Close**, **CopyObject**, **DeleteObject**, **Rename**, **RepaintObject**, **Save**, **SelectObject**, **TransferDatabase**
acToolbarNo	**ShowToolbar**
acToolbarWhereApprop	**ShowToolbar**
acToolbarYes	**ShowToolbar**
acUndo	**DoMenuItem**
acUp	**FindRecord**
acViewDesign	**OpenQuery**, **OpenReport**, **OpenTable**
acViewNormal	**OpenQuery**, **OpenReport**, **OpenTable**
acViewPreview	**OpenQuery**, **OpenReport**, **OpenTable**
acWindowNormal	**OpenForm**

Константы различного назначения

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"accstMiscConstantsC"}

В следующих перечнях приводятся встроенные константы различного назначения, определенные в Microsoft Access.

Константы списка и поля со списком

Следующие константы применяются при заполнении списка или поля со списком с помощью функции.. Функция, с которой используются эти константы, определяет значение свойства **Тип источника строк (RowSourceType)** с помощью специального формата функций.

AcLBClose
acLBEnd
acLBGetColumnCount
acLBGetColumnWidth
acLBGetFormat
acLBGetRowCount
acLBGetValue
acLBInitialize
acLBOpen

Константы функции SysCmd

Следующие константы используются с функцией **SysCmd**.

acForm
acMacro
acModule
acObjStateDirty
acObjStateNew
acObjStateOpen
acQuery
acReport
acSysCmdAccessDir
acSysCmdAccessVer
acSysCmdClearHelpTopic
acSysCmdClearStatus
acSysCmdGetObjectState
acSysCmdGetWorkgroupFile
acSysCmdIniFile
acSysCmdInitMeter
acSysCmdProfile
acSysCmdRemoveMeter
acSysCmdRuntime
acSysCmdSetStatus
acSysCmdUpdateMeter
acTable

Константы свойства ControlType и функций CreateControl и CreateReportControl

Следующие константы используются в свойстве **ControlType**, а также с функциями **CreateControl** и **CreateReportControl** для определения типа элемента управления.

AcBoundObjectFrame
acCheckBox
acComboBox
acCommandButton
acCustomControl
acImage
acLabel
acLine
acListBox
acObjectFrame
acOptionButton
acOptionGroup
acPage
acPageBreak
acRectangle
acSubform
acTabCtl
acTextBox
acToggleButton

Константы свойства Section и функций CreateControl и CreateReportControl

Следующие константы используются в свойстве **Section**, а также с функциями **CreateControl** и

CreateReportControl для определения раздела формы или отчета.

AcDetail	**acGroupLevel2Header**
acFooter	**acHeader**
acGroupLevel1Footer	**acPageFooter**
acGroupLevel1Header	**acPageHeader**
acGroupLevel2Footer	

Константы OLE

В следующей таблице приводятся константы OLE и элементы языка, с которыми используются эти константы.

Константа OLE	Элемент языка
acOLEActivate	**Action** (свойство)
acOLEActivateDoubleClick	**Автоматический запуск (AutoActivate)** (свойство)
acOLEActivateGetFocus	**Автоматический запуск (AutoActivate)** (свойство)
acOLEActivateManual	**Автоматический запуск (AutoActivate)** (свойство)
acOLEChanged	При обновлении (Updated) (событие)
acOLEClose	**Action** (свойство)
acOLEClosed	При обновлении (Updated) (событие)
acOLECopy	**Action** (свойство)
acOLECreateEmbed	**Action** (свойство)
acOLECreateFromFile (same as **acOLECreateLink**)	**Action** (свойство)
acOLECreateLink	**Action** (свойство)
acOLECreateNew (same as **acOLECreateEmbed**)	**Action** (свойство)
acOLEDelete	**Action** (свойство)
acOLEDisplayContent	**Тип вывода (DisplayType)** (свойство)
acOLEDisplayIcon	**Тип вывода (DisplayType)** (свойство)
acOLEEither	**Допустимый тип OLE (OLETypeAllowed)** (свойство)
acOLEEmbedded	**Тип OLE (OLEType)** (свойство), **Допустимый тип OLE (OLETypeAllowed)** (свойство)
acOLEFetchVerbs	**Action** (свойство)
acOLEInsertObjDlg	**Action** (свойство)
acOLELinked	**Тип OLE (OLEType)** (свойство), **Допустимый тип OLE (OLETypeAllowed)** (свойство)
acOLENone	**Тип OLE (OLEType)** (свойство)
acOLEPaste	**Action** (свойство)
acOLEPasteSpecialDlg	**Action** (свойство)
acOLERenamed	При обновлении (Updated) (событие)
acOLESaved	При обновлении (Updated) (событие)
acOLESizeAutoSize (not used)	**Установка размеров (SizeMode)** (свойство)
acOLESizeClip	**Установка размеров (SizeMode)** (свойство)
acOLESizeStretch	**Установка размеров (SizeMode)** (свойство)

acOLESizeZoom	**Установка размеров (SizeMode)** (свойство)
acOLEUpdate	**Action** (свойство)
acOLEUpdateAutomatic	**Параметры обновления (UpdateOptions)** (свойство)
acOLEUpdateFrozen (not used)	**Параметры обновления (UpdateOptions)** (свойство)
acOLEUpdateManual	**Параметры обновления (UpdateOptions)** (свойство)
acOLEVerbHide	**Команда (Verb)** (свойство)
acOLEVerbInPlaceActivate	**Команда (Verb)** (свойство)
acOLEVerbInPlaceUIActivate	**Команда (Verb)** (свойство)
acOLEVerbOpen	**Команда (Verb)** (свойство)
acOLEVerbPrimary	**Команда (Verb)** (свойство)
acOLEVerbShow	**Команда (Verb)** (свойство)

Константы свойства «Оформление» (SpecialEffect)

В следующей таблице перечислены константы, с помощью которых проверяется или задается значение свойства **Оформление (SpecialEffect)**. Константы **acEffectNormal**, **acEffectRaised** и **acEffectSunken** используются также в свойстве **DatasheetCellsEffect**.

acEffectChisel	**acEffectRaised**
acEffectEtched	**acEffectShadow**
acEffectNormal	**acEffectSunken**

Константы сетки

В следующей таблице перечислены константы, с помощью которых проверяют или задают внешний вид сетки в режиме таблицы. Эти константы могут использоваться в свойстве **DatasheetGridlinesBehavior**.

acGridlinesBoth	**acGridlinesNone**
acGridlinesBothV2	**acGridlinesVert**
acGridlinesHoriz	

Константы типа модуля

Ниже приводятся константы, с помощью которых определяют, является ли текущий модуль стандартным модулем или модулем класса. Эти константы могут использоваться в свойстве **Type**.

acClassModule
acStandardModule

Константы функции HyperlinkPart

Следующие константы используются с функцией **HyperlinkPart**.

acDisplayedValue	**acAddress**
acDisplayName	**acSubAddress**

Константы процедур

Следующие константы используются в свойствах **ProcStartLine**, **ProcBodyLine**, **ProcOfLine** и **ProcCountLines**.

vbext_pk_Get **vbext_pk_Proc**
vbext_pk_Let **vbext_pk_Set**

Константы свойства Kind

Следующие константы используются в свойстве **Kind** объекта **Reference**.

Project
TypeLib

Константы кодов клавиатуры

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"accstKeyCodeConstantsC"}

Следующие константы представляют коды клавиатуры. Эти константы используются в Microsoft Access с процедурами обработки событий **Клавиша вниз (KeyDown)** и **Клавиша вверх (KeyUp)**.

vbKey0
vbKey1
vbKey2
vbKey3
vbKey4
vbKey5
vbKey6
vbKey7
vbKey8
vbKey9
vbKeyA
vbKeyAdd
vbKeyB
vbKeyBack
vbKeyC
vbKeyCancel
vbKeyCapital
vbKeyClear
vbKeyControl
vbKeyD
vbKeyDecimal
vbKeyDelete
vbKeyDivide
vbKeyDown
vbKeyE
vbKeyEnd
vbKeyEscape
vbKeyExecute
vbKeyF
vbKeyF1
vbKeyF2
vbKeyF3
vbKeyF4
vbKeyF5
vbKeyF6
vbKeyF7
vbKeyF8
vbKeyF9
vbKeyF10
vbKeyF11
vbKeyF12
vbKeyF13
vbKeyF14
vbKeyF15
vbKeyF16
vbKeyG
vbKeyH
vbKeyHelp
vbKeyHome
vbKeyI
vbKeyInsert
vbKeyJ
vbKeyK
vbKeyL
vbKeyLButton
vbKeyLeft
vbKeyM
vbKeyMButton
vbKeyMenu
vbKeyMultiply
vbKeyN
vbKeyNumLock
vbKeyNumPad0
vbKeyNumPad1
vbKeyNumPad2
vbKeyNumPad3
vbKeyNumPad4
vbKeyNumPad5
vbKeyNumPad6
vbKeyNumPad7
vbKeyNumPad8
vbKeyNumPad9
vbKeyO
vbKeyP
vbKeyPageDown
vbKeyPageUp
vbKeyPause
vbKeyPrint
vbKeyQ
vbKeyR
vbKeyRButton
vbKeyReturn
vbKeyRight
vbKeyS
vbKeySelect
vbKeySeparator
vbKeyShift
vbKeySnapshot
vbKeySpace
vbKeySubtract
vbKeyT
vbKeyTab
vbKeyU
vbKeyUp
vbKeyV
vbKeyW
vbKeyX
vbKeyY
vbKeyZ

Константы метода RunCommand

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"accstRunMethodConstantsC"}

Следующие константы используются при вызове метода **RunCommand** в программах Visual Basic. Вызов метода объекта **RunCommand** с одной из этих констант аналогичен передаче значения макрокоманде **RunCommand** в макросе.

acCmdAboutMicrosoftAccess
acCmdAddWatch
acCmdAdvancedFilterSort
acCmdAlignBottom
acCmdAlignLeft
acCmdAlignRight
acCmdAlignToGrid
acCmdAlignTop
acCmdAlignToShortest
acCmdAlignToTallest
acCmdAnalyzePerformance
acCmdAnalyzeTable
acCmdAnswerWizard
acCmdApplyDefault
acCmdApplyFilterSort
acCmdAppMaximize
acCmdAppMinimize
acCmdAppMove
acCmdAppRestore
acCmdAppSize
acCmdArrangeIconsAuto
acCmdArrangeIconsByCreated
acCmdArrangeIconsByModified
acCmdArrangeIconsByName
acCmdArrangeIconsByType
acCmdAutoCorrect
acCmdAutoDial
acCmdAutoFormat
acCmdBookmarksClearAll
acCmdBookmarksNext
acCmdBookmarksPrevious
acCmdBookmarksToggle
acCmdBringToFront
acCmdCallStack
acCmdChangeToCheckBox
acCmdChangeToComboBox
acCmdChangeToImage
acCmdChangeToLabel
acCmdChangeToListBox
acCmdHideColumns
acCmdHideTable
acCmdHorizontalSpacingDecrease
acCmdHorizontalSpacingIncrease
acCmdHorizontalSpacingMakeEqual
acCmdHyperlinkDisplayText
acCmdImport
acCmdIndent
acCmdIndexes
acCmdInsertActiveXControl
acCmdInsertChart
acCmdInsertFile
acCmdInsertFileIntoModule
acCmdInsertHyperlink
acCmdInsertLookupColumn
acCmdInsertLookupField
acCmdInsertObject
acCmdInsertPage
acCmdInsertPicture
acCmdInsertProcedure
acCmdInsertQueryColumn
acCmdInsertRows
acCmdInsertTableColumn
acCmdInvokeBuilder
acCmdJoinProperties
acCmdLastPosition
acCmdLayoutPreview
acCmdLineUpIcons
acCmdLinkTables
acCmdListConstants
acCmdLoadFromQuery
acCmdMacroConditions
acCmdMacroNames
acCmdMakeMDEFile
acCmdMicrosoftAccessHelpTopics
acCmdMicrosoftOnTheWeb
acCmdMinimize
acCmdMoreWindows
acCmdNewDatabase
acCmdRefreshPage
acCmdRegisterActiveXControls
acCmdRelationships
acCmdRemoveFilterSort
acCmdRemoveTable
acCmdRename
acCmdRenameColumn
acCmdRepairDatabase
acCmdReplace
acCmdReportHdrFtr
acCmdReset
acCmdResolveConflicts
acCmdRowHeight
acCmdRun
acCmdRunMacro
acCmdRunOpenMacro
acCmdSave
acCmdSaveAllModules
acCmdSaveAs
acCmdSaveAsASP
acCmdSaveAsHTML
acCmdSaveAsIDC
acCmdSaveAsQuery
acCmdSaveAsReport
acCmdSaveLayout
acCmdSaveModuleAsText
acCmdSaveRecord
acCmdSelectAll
acCmdSelectAllRecords
acCmdSelectForm
acCmdSelectRecord
acCmdSelectReport
acCmdSend
acCmdSendToBack
acCmdSetControlDefaults
acCmdSetDatabasePassword
acCmdSetNextStatement
acCmdShowAllRelationships
acCmdShowDirectRelationships

acCmdChangeToOptionButton
acCmdChangeToTextBox
acCmdChangeToToggleButton
acCmdClearAll
acCmdClearAllBreakpoints
acCmdClearGrid
acCmdClearItemDefaults
acCmdClose
acCmdCloseWindow
acCmdColumnWidth
acCmdCompactDatabase
acCmdCompileAllModules
acCmdCompileAndSaveAllModules
acCmdCompileLoadedModules
acCmdCompleteWord
acCmdControlWizardsToggle
acCmdConvertDatabase
acCmdConvertMacrosToVisualBasic
acCmdCopy
acCmdCopyHyperlink
acCmdCreateMenuFromMacro
acCmdCreateRelationship
acCmdCreateReplica
acCmdCreateShortcut
acCmdCreateShortcutMenuFromMacro
acCmdCreateToolbarFromMacro
acCmdCut
acCmdDatabaseProperties
acCmdDataEntry
acCmdDatasheetView
acCmdDateAndTime
acCmdDebugWindow
acCmdDelete
acCmdDeletePage
acCmdDeleteQueryColumn
acCmdDeleteRecord
acCmdDeleteRows
acCmdDeleteTab
acCmdDeleteTableColumn
acCmdDeleteWatch
acCmdDesignView
acCmdDocMaximize
acCmdDocMove
acCmdDocRestore
acCmdDocSize
acCmdNewObjectAutoForm
acCmdNewObjectAutoReport
acCmdNewObjectClassModule
acCmdNewObjectForm
acCmdNewObjectMacro
acCmdNewObjectModule
acCmdNewObjectQuery
acCmdNewObjectReport
acCmdNewObjectTable
acCmdObjBrwFindWholeWordOnly
acCmdObjBrwGroupMembers
acCmdObjBrwHelp
acCmdObjBrwShowHiddenMembers
acCmdObjBrwViewDefinition
acCmdObjectBrowser
acCmdOLEDDELinks
acCmdOLEObjectConvert
acCmdOLEObjectDefaultVerb
acCmdOpenDatabase
acCmdOpenHyperlink
acCmdOpenNewHyperlink
acCmdOpenSearchPage
acCmdOpenStartPage
acCmdOpenTable
acCmdOpenURL
acCmdOptions
acCmdOutdent
acCmdOutputToExcel
acCmdOutputToRTF
acCmdOutputToText
acCmdPageHdrFtr
acCmdPageNumber
acCmdPageSetup
acCmdParameterInfo
acCmdPaste
acCmdPasteAppend
acCmdPasteSpecial
acCmdPreviewEightPages
acCmdPreviewFourPages
acCmdPreviewOnePage
acCmdPreviewTwelvePages
acCmdPreviewTwoPages
acCmdPrimaryKey
acCmdPrint
acCmdPrintPreview
acCmdShowMembers
acCmdShowNextStatement
acCmdShowOnlyWebToolbar
acCmdShowTable
acCmdSingleStep
acCmdSizeToFit
acCmdSizeToFitForm
acCmdSizeToGrid
acCmdSizeToNarrowest
acCmdSizeToWidest
acCmdSnapToGrid
acCmdSortAscending
acCmdSortDescending
acCmdSortingAndGrouping
acCmdSpelling
acCmdSQLView
acCmdStartupProperties
acCmdStepOut
acCmdStepOver
acCmdStepToCursor
acCmdStopLoadingPage
acCmdSubformDatasheet
acCmdSynchronizeNow
acCmdTabControlPageOrder
acCmdTableNames
acCmdTabOrder
acCmdTestValidationRules
acCmdTileHorizontally
acCmdTileVertically
acCmdToggleBreakpoint
acCmdToggleFilter
acCmdToolbarControlProperties
acCmdToolbarsCustomize
acCmdTransparentBackground
acCmdTransparentBorder
acCmdUndo
acCmdUnfreezeAllColumns
acCmdUnhideColumns
acCmdUserAndGroupAccounts
acCmdUserAndGroupPermission
acCmdUserLevelSecurityWizard
acCmdVerticalSpacingDecrease
acCmdVerticalSpacingIncrease
acCmdVerticalSpacingMakeEqua
acCmdViewCode

acCmdDocumenter
acCmdDuplicate
acCmdEditHyperlink
acCmdEditingAllowed
acCmdEditRelationship
acCmdEditWatch
acCmdEncryptDecryptDatabase
acCmdEnd
acCmdExit
acCmdFavoritesAddTo
acCmdFavoritesOpen
acCmdFieldList
acCmdFilterByForm
acCmdFilterBySelection
acCmdFilterExcludingSelection
acCmdFind
acCmdFindNextWordUnderCursor
acCmdFindPrevious
acCmdFindPrevWordUnderCursor
acCmdFitToWindow
acCmdFont
acCmdFormatCells
acCmdFormHdrFtr
acCmdFormView
acCmdFreezeColumn
acCmdGoBack
acCmdGoContinue
acCmdGoForward
acCmdProcedureDefinition
acCmdProperties
acCmdPublish
acCmdPublishDefaults
acCmdQueryParameters
acCmdQueryTotals
acCmdQueryTypeAppend
acCmdQueryTypeCrosstab
acCmdQueryTypeDelete
acCmdQueryTypeMakeTable
acCmdQueryTypeSelect
acCmdQueryTypeSQLDataDefinition
acCmdQueryTypeSQLPassThrough
acCmdQueryTypeSQLUnion
acCmdQueryTypeUpdate
acCmdQuickInfo
acCmdQuickPrint
acCmdQuickWatch
acCmdRecordsGoToFirst
acCmdRecordsGoToLast
acCmdRecordsGoToNew
acCmdRecordsGoToNext
acCmdRecordsGoToPrevious
acCmdRecoverDesignMaster
acCmdRedo
acCmdReferences
acCmdRefresh
acCmdViewDetails
acCmdViewForms
acCmdViewGrid
acCmdViewLargeIcons
acCmdViewList
acCmdViewMacros
acCmdViewModules
acCmdViewQueries
acCmdViewReports
acCmdViewRuler
acCmdViewSmallIcons
acCmdViewTables
acCmdViewToolbox
acCmdWindowArrangeIcons
acCmdWindowCascade
acCmdWindowHide
acCmdWindowSplit
acCmdWindowUnhide
acCmdWordMailMerge
acCmdZoom10
acCmdZoom25
acCmdZoom50
acCmdZoom75
acCmdZoom100
acCmdZoom150
acCmdZoom200
acCmdZoomBox

Макрокоманда «ДобавитьМеню» (AddMenu)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactAddMenuC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactAddMenuX":1}

Макрокоманда **ДобавитьМеню (AddMenu)** используется для выполнения следующих действий:

- Создание специальной строки меню для формы или отчета. Специальная строка меню заменяет встроенную строку меню формы или отчета.
- Создание специального контекстного меню для формы, элемента управления или отчета. Специальное контекстное меню заменяет встроенное контекстное меню формы, элемента управления или отчета.
- Создание общей строки меню. Общая строка меню заменяет встроенное главное меню во всех окнах Microsoft Access, за исключением тех, в которые пользователь добавил специальную строку меню формы или отчета.
- Создание общего контекстного меню. Общее контекстное меню заменяет встроенное контекстное меню для всех полей таблиц или запросов в режиме таблицы, форм в режиме формы, режиме таблицы и режиме предварительного просмотра, а также отчетов в режиме предварительного просмотра, за исключением тех, в которые пользователь добавил специальное контекстное меню формы, отчета или элемента управления.

Примечание. Хотя в Microsoft Access 97 допускается создание всех вышеперечисленных видов меню с помощью макросов, содержащих макрокоманду **ДобавитьМеню (AddMenu)** (и все существующие специальные меню, созданные в них будут работать), для создания новых меню или настройки встроенных рекомендуется использовать диалоговое окно **Настройка панелей инструментов**, вызываемое путем выбора в меню **Вид** команды **Панели инструментов** и нажатия кнопки **Настройка**.

Значения

Макрокоманда **ДобавитьМеню (AddMenu)** использует следующие аргументы.

Аргумент	Описание
Название меню	Имя ниспадающего меню, которое добавляется в специальную строку меню или в общую строку меню. Название меню следует ввести в поле **Название меню** в разделе **Аргументы макрокоманды** в окне макросов. Данный аргумент является обязательным для специальных и общих меню. Для специальных и общих контекстных меню этот аргумент игнорируется. Для создания клавиши доступа, позволяющей использовать клавиатуру для выбора меню, следует перед буквой, которую необходимо сделать клавишей доступа, ввести амперсанд (&). В строке меню в имени меню эта буква будет подчеркнута.
Имя макроса	Имя группы макросов, содержащей макросы создания команд меню (обязательный аргумент). Если макрос, содержащий макрокоманду **ДобавитьМеню (AddMenu)**, запущен из библиотечной базы данных, Microsoft Access будет искать группу макросов с указанным именем только в текущей базе данных.
Текст строки состояния	Текст, который будет выводиться в строке состояния при выборе данного меню. Для специальных и общих контекстных меню этот аргумент игнорируется.

Дополнительные сведения

Для создания специальной строки меню, специального контекстного меню, общей строки меню или общего контекстного меню необходимо:

1. Создать макрос создания меню, содержащий макрокоманду **ДобавитьМеню (AddMenu)** для каждого ниспадающего меню, которое следует включить в специальную строку меню или в общую строку меню. Для специального контекстного меню или общего контекстного меню макрос создания меню должен содержать только одну макрокоманду **ДобавитьМеню (AddMenu)**.
2. Определить команды для каждого ниспадающего меню, создав группу макросов для каждого меню. При выборе любой команды меню будут выполняться макрокоманды, входящие в соответствующий ей макрос из этой группы макросов. Для специального контекстного меню или общего контекстного меню следует создать только одну группу макросов, каждый макрос в которой содержит набор макрокоманд, соответствующих данной команде контекстного меню.
3. Связать макрос создания меню с соответствующим объектом Microsoft Access:
   - Для специальной строки меню укажите имя макроса создания меню в свойстве **Строка меню (MenuBar)** формы или отчета.
   - Для специального контекстного меню укажите имя макроса создания меню в свойстве **Контекстное меню (ShortcutMenuBar)** формы, элемента управления в форме или отчета.
   - Для общей строки меню выберите в меню **Сервис** команду **Параметры запуска** и введите имя макроса создания меню в поле **Строка меню** в диалоговом окне **Параметры запуска**.
   - Для общего контекстного меню выберите в меню **Сервис** команду **Параметры запуска** и введите имя макроса создания меню в поле **Контекстное меню** в диалоговом окне **Параметры запуска**.

Для создания каждого пункта в строке меню требуется отдельная макрокоманда **ДобавитьМеню**.

Специальные и общие строки меню и контекстные меню заменяют встроенные меню объектов, с которыми они связываются. Для включения в специальное меню конкретных команд стандартного меню Microsoft Access необходимо с помощью макрокоманды **ВыполнитьКоманду (RunCommand)** включить соответствующие команды в нужные группы макросов.

Для запуска макроса с помощью команды меню следует использовать макрокоманду **ЗапускМакроса (RunMacro)** в макросе, соответствующем этой команде.

Примечание. Макрокоманды **ДобавитьМеню (AddMenu)** можно использовать только в макросе создания меню, указанном в значении свойств **Строка меню (MenuBar)** или **Контекстное меню (ShortcutMenuBar)** формы, элемента управления в форме или отчета, а также в макросе, указанном в полях **Строка меню** или **Контекстное меню** в диалоговом окне **Параметры запуска**. Макрос создания меню не может содержать другие макрокоманды кроме макрокоманды **ДобавитьМеню (AddMenu)**.

Если для формы, отчета или базы данных задан макрос строки меню, Microsoft Access запускает его всякий раз когда открывается форма, отчет или база данных. При внесении изменений в макрос строки меню или в группу макросов, определяющих команды ниспадающих меню, в то время как форма, отчет или база данных открыты, чтобы увидеть изменения в специальной строке меню и его подменю, следует закрыть их и затем открыть снова.

При создании групп макросов, содержащих команды создания специального меню, могут быть полезными следующие сведения:

- В группе макросов имя в столбце **Имя макроса** для каждой макрокоманды будет являться именем команды. Текст столбца **Примечание** той же строки при выборе команды будет

отображен в строке состояния.

- Для создания строки-разделителя между двумя командами меню следует в столбце **Имя макроса** между подходящими командами ввести дефис (-).
- Для создания клавиши доступа, позволяющей использовать клавиатуру для выбора меню, следует перед буквой, которую необходимо сделать клавишей доступа, ввести амперсанд (&). В имени меню эта буква будет подчеркнута.

Для создания подменю в специальном меню или специальном контекстном меню следует включить макрокоманду **ДобавитьМеню (AddMenu)** в группу макросов, указанную в аргументе «Имя макроса». Макрокоманда **ДобавитьМеню**, включенная в такую группу макросов, создает подменю, имя которого и соответствующий текст строки состояния задаются аргументами «Название меню» и «Текст строки состояния». Команды подменю определяются группой макросов, имя которой задается в аргументе «Имя макроса» этой макрокоманды. Аргументы «Имя меню» и «Текст строки состояния» такой макрокоманды **ДобавитьМеню** игнорируются, поскольку эта макрокоманда создает подменю, а не команду меню первого уровня. Для создания многоуровневого меню следует использовать макрокоманды **ДобавитьМеню** на каждом уровне меню.

Условия анализируются только в макросе создания меню первого уровня. Другими словами, условие в макросе создания меню можно использовать для определения набора команд в строке меню, однако, нельзя использовать для определения набора команд в ниспадающем меню или подменю любого уровня. Допускается также применение условий, определяющих вывод или скрытие специального контекстного меню или общего контекстного меню.

Макрокоманда **ДобавитьМеню (AddMenu)** недоступна в Visual Basic. Однако пользователь имеет возможность назначить в программе Visual Basic специальную строку меню или специальное контекстное меню для формы, элемента управления в форме или отчета с помощью свойств **Строка меню (MenuBar)** и **Контекстное меню (ShortcutMenuBar)**. Общая строка меню задается с помощью свойства **StartupMenuBar** (программный эквивалент введения значения в поле **Строка меню**). Кроме того, создание общей строки меню возможно с помощью свойства **MenuBar** объекта **Application** (Приложение). Аналогично этому, свойство **StartupShortcutMenuBar** является программным эквивалентом поля **Контекстное меню**, а для создания общего контекстного меню можно использовать свойство **ShortcutMenuBar** объекта **Application**.

Макрокоманда «ПрименитьФильтр» (ApplyFilter)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactApplyFilterC;dasqlWhere":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactApplyFilterX":1}

Макрокоманда **ПрименитьФильтр (ApplyFilter)** применяет фильтр, запрос или предложение WHERE (SQL) к таблице, форме или отчету для отбора и сортировки записей в таблице, а также в базовой таблице или базовом запросе формы или отчета. Для отчетов эту макрокоманду можно использовать только в макросе, указанном в свойстве события отчета **Открытие (OnOpen).**

Значения

Макрокоманда **ПрименитьФильтр (ApplyFilter)** использует следующие аргументы.

Аргумент	Описание
Имя фильтра	Имя фильтра или запроса, который следует использовать для отбора и сортировки записей таблицы, формы или отчета. Введите имя существующего запроса или фильтра, сохраненного в виде запроса в поле **Имя фильтра** в разделе **Аргументы макрокоманды** в окне макросов.
Условие отбора	Правильно составленное предложение SQL WHERE (без слова WHERE) или выражение, определяющее отбор записей таблицы, формы или отчета. **Примечание.** Левая часть выражения, указанного в аргументе «Условие отбора», обычно содержит имя поля из базовой таблицы или базового запроса формы или отчета. Правая часть этого выражения содержит условия отбора, которые следует применить к этому полю для отбора и сортировки записей (например, имя элемента управления из другой формы, задающего значение, которое должны содержать отобранные записи первой формы). Имя элемента управления следует указывать полностью в формате: **Forms!***имяФормы***!***имяЭлемента* Максимальная длина выражения в аргументе «Условие отбора» равняется 256 символам. Для ввода более сложной инструкции SQL с предложением WHERE (длиной до 32 768 символов) следует использовать метод **ApplyFilter** объекта **DoCmd** в программе Visual Basic.

Примечание. Необходимо определить один или оба аргумента. Используйте аргумент «Имя фильтра», если нужный фильтр уже определен и сохранен. Аргумент «Условие отбора» позволяет непосредственно указать условие отбора. Если заданы оба аргумента, предложение WHERE применяется к записям, отобранным в фильтре.

Дополнительные сведения

Фильтр или запрос может быть применен к форме в режиме формы или в режиме таблицы.

Фильтр или предложение WHERE становятся значением свойства **Фильтр (Filter)** формы или отчета.

Для таблиц и форм вызов данной макрокоманды эквивалентен выбору команды **Применить фильтр** в меню **Записи** или нажатию кнопки **Применить фильтр**  на панели инструментов. Отличие состоит в том, что команда меню или кнопка применяет к таблице или форме последний созданный фильтр, в то время как макрокоманда применяет указанный фильтр или запрос.

Если после запуска макрокоманды **ПрименитьФильтр (ApplyFilter)** выбрать в меню **Записи** команду **Фильтр** и подкоманду **Расширенный фильтр**, то в окне расширенного фильтра выводится условие отбора, заданное в макрокоманде.

Для снятия фильтра и отображения всех записей в таблице или форме следует выполнить макрокоманду **ПоказатьВсеЗаписи (ShowAllRecords),** выбрать в меню **Записи** команду **Удалить фильтр** или нажать кнопку **Удалить фильтр**  на панели инструментов.

При сохранении таблицы или формы Microsoft Access сохраняет текущий фильтр, определенный для данного объекта. Однако этот фильтр не применяется автоматически при следующем открытии объекта (хотя будет автоматически применен последний порядок сортировки, заданный для объекта перед сохранением). Для автоматического применения фильтра при открытии формы следует указать в свойстве события **Открытие (OnOpen)** макрос, содержащий макрокоманду **ПрименитьФильтр (ApplyFilter)**, или процедуру обработки события, в которой вызывается метод **ApplyFilter** объекта **DoCmd**. Применить фильтр позволяют также макрокоманды **ОткрытьФорму (OpenForm)** и **ОткрытьОтчет (OpenReport)** или соответствующие им методы. Для автоматического применения фильтра при открытии таблицы следует открыть таблицу с помощью макроса, содержащего макрокоманду **ОткрытьТаблицу (OpenTable),** сразу за которой вызывается макрокоманда **ПрименитьФильтр (ApplyFilter)**.

Макрокоманда «Сигнал» (Beep)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactBeepC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactBeepX":1}

Макрокоманда **Сигнал (Beep)** подает звуковой сигнал через динамик компьютера.

Значения

Макрокоманда **Сигнал (Beep)** не требует аргументов.

Дополнительные сведения

Макрокоманду **Сигнал (Beep)** используют для подачи звукового сигнала в следующих случаях:

- На экране произошли изменения, требующие обязательного внимания пользователя.
- В элемент управления введено недопустимое значение (например, числовое значение вводится в текстовое поле.)
- Выполняемый макрос достиг определенной точки или завершился.

Частота и длительность звукового сигнала определяются конструктивными особенностями используемого оборудования.

Для вызова макрокоманды **Сигнал (Beep)** в программе Visual Basic следует использовать метод **Beep** объекта **DoCmd.**

Макрокоманда «ОтменитьСобытие» (CancelEvent)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactCancelEventC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactCancelEventX":1}

Макрокоманда **ОтменитьСобытие (CancelEvent)** отменяет событие, вызвавшее запуск макроса, содержащего эту макрокоманду. Имя макроса задается в значении свойства события, например, **До обновления (BeforeUpdate)**, **Открытие (OnOpen)**, **Выгрузка (OnUnload)** или **Печать (OnPrint)**.

Значения

Макрокоманда **ОтменитьСобытие (CancelEvent)** не требует аргументов.

Дополнительные сведения

В форме макрокоманду **ОтменитьСобытие (CancelEvent)** обычно используют в макросе, связанном со свойством **До обновления (BeforeUpdate)** и выполняющем проверку условия на значение. После завершения ввода данных в элемент управления или запись Microsoft Access запускает данный макрос перед занесением этих данных в базу данных. Если введенные данные противоречат указанным в макросе условиям, макрокоманда **ОтменитьСобытие (CancelEvent)** отменяет процесс обновления.

Довольно часто эта макрокоманда используется вместе с макрокомандой **Сообщение (MsgBox),** которая открывает диалоговое окно с сообщением о попытке ввода недопустимых данных и сведениями о требуемых данных.

В следующей таблице перечислены события, которые могут быть отменены с помощью макрокоманды **ОтменитьСобытие (CancelEvent)**:

ПрименитьФильтр (ApplyFilter)	Удаление (Delete)	Кнопка вниз (MouseDown)
До подтверждения Del (BeforeDelConfirm)	Выход (Exit)	Отсутствие данных (NoData)
До вставки (BeforeInsert)	Фильтрация (Filter)	Открытие (Open)
До обновления (BeforeUpdate)	Форматирование (Format)	Печать (Print)
Двойное нажатие кнопки (DblClick)	Нажатие клавиши (KeyPress)	Выгрузка (Unload)

Примечание. Макрокоманда **ОтменитьСобытие** позволяет отменить событие **Кнопка вниз (MouseDown)** только при нажатии правой кнопки мыши.

Если в свойстве события элемента управления **Двойное нажатие кнопки (OnDblClick)** указан макрос, содержащий макрокоманду **ОтменитьСобытие (CancelEvent)**, этот макрос отменяет двойное нажатие кнопки мыши.

Для событий, допускающих отмену, стандартное действие (т.е. действие, которое Microsoft Access обычно выполняет при возникновении события) выполняется после макроса, связанного с событием. Это дает возможность отменить стандартное действие. Например, если установить указатель мыши в поле на слове, в котором находится курсор, и дважды нажать кнопку мыши, то Microsoft Access обычно выделяет слово. Данная макрокоманда позволяет отменить выделение слова и выполнить какое-либо другое действие в ответ на событие **Двойное нажатие кнопки (DblClick)**, например, открыть форму, содержащую дополнительные сведения. Для событий, не допускающих отмену, стандартное действие выполняется до выполнения макроса.

Примечание. Если в значении свойства формы **Выгрузка (OnUnload)** указан макрос, содержащий макрокоманду **ОтменитьСобытие (CancelEvent)**, то пользователь не сможет закрыть форму. В таком случае необходимо открыть этот макрос и либо исправить условие,

вызывающее макрокоманду **ОтменитьСобытие**, либо вообще удалить макрокоманду **ОтменитьСобытие**. Если форма является модальной, то пользователь не сможет даже открыть этот макрос.

Для вызова макрокоманды **ОтменитьСобытие (CancelEvent)** в программе Visual Basic следует использовать метод **CancelEvent** объекта **DoCmd.**

Макрокоманда «Закрыть» (Close)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactCloseC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactCloseX":1}

Макрокоманда **Закрыть (Close)** закрывает указанное окно Microsoft Access или текущее окно (по умолчанию).

Значения

Макрокоманда **Закрыть (Close)** использует следующие аргументы.

Аргумент	Описание
Тип объекта	Тип объекта, окно которого следует закрыть. Допустимые значения «Таблица», «Запрос», «Форма», «Отчет», «Макрос» и «Модуль» выбираются в раскрывающемся списке в разделе **Аргументы макрокоманды** в окне макроса. Для того чтобы указать текущее окно, оставьте значение этого аргумента пустым.
Имя объекта	Имя объекта, который следует закрыть. В раскрывающемся списке содержатся имена всех объектов текущей базы данных, относящихся к типу, указанному в аргументе «Тип объекта». Если значение аргумента «Тип объекта» оставлено пустым, следует оставить пустым и значение данного аргумента.
Сохранить	Определяет сохранение изменений объекта при его закрытии. Выберите «Да» для автоматического сохранения изменений при закрытии объекта, «Нет» для закрытия объекта без сохранения изменений или «Подсказка» (значение по умолчанию) для вывода диалогового окна с приглашением подтвердить или отменить сохранение объекта.

Дополнительные сведения

Макрокоманда **Закрыть (Close)** применима к любому объекту базы данных, который может быть явно открыт и закрыт пользователем. Вызов этой макрокоманды эквивалентен выделению объекта и выбору команды **Закрыть** в меню **Файл**, команды **Закрыть** в оконном меню объекта или нажатию кнопки **Закрыть** ☒ в окне объекта.

Если для аргумента «Сохранить» выбрано значение «Подсказка», а объект не был сохранен перед выполнением макрокоманды **Закрыть (Close)**, на экране появится диалоговое окно с приглашением сохранить объект. Отключить вывод этого диалогового окна позволяет макрокоманда **УстановитьСообщения (SetWarnings)** со значением «Нет» в аргументе «Включить сообщения».

Для вызова макрокоманды **Закрыть (Close)** в программе Visual Basic следует использовать метод **Close** объекта **DoCmd.**

Макрокоманда «КопироватьОбъект» (CopyObject)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactCopyObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactCopyObjectX":1}

Макрокоманда **КопироватьОбъект (CopyObject)** копирует указанный объект базы данных в другую базу данных Microsoft Access или в ту же базу данных под новым именем. Например, эту макрокоманду используют для копирования или сохранения существующего объекта в другой базе данных или при использовании существующего объекта в качестве прототипа нового объекта.

Значения

Макрокоманда **КопироватьОбъект (CopyObject)** использует следующие аргументы.

Аргумент	Описание
База данных	Полное имя файла базы данных-получателя. Введите это имя в ячейку аргумента «База данных» в разделе **Аргументы макрокоманды** окна макросов. Для того чтобы указать текущую базу данных, оставьте значение этого аргумента пустым. Если вызвать макрос, содержащий макрокоманду **КопироватьОбъект (CopyObject)**, из библиотечной базы данных и оставить данный аргумент пустым, то объект будет скопирован в библиотечную базу данных.
Новое имя	Новое имя объекта. Чтобы сохранить прежнее имя при копировании в другую базу данных, оставьте значение этого аргумента пустым.
Тип объекта	Тип копируемого объекта. Допустимые значения «Таблица», «Запрос», «Форма», «Отчет», «Макрос» и «Модуль» выбираются в раскрывающемся списке. Для того чтобы скопировать объект, выделенный в окне базы данных, оставьте значение этого аргумента пустым.
Имя объекта	Имя объекта, который следует скопировать. Раскрывающийся список содержит имена всех объектов текущей базы данных, которые относятся к типу, указанному в аргументе «Тип объекта». Если значение аргумента «Тип объекта» оставлено пустым, значение этого аргумента также следует оставить пустым. Если макрос, содержащий макрокоманду **КопироватьОбъект (CopyObject)**, запускается из библиотечной базы данных, Microsoft Access проводит поиск объекта с указанным именем сначала в библиотечной базе данных, а затем в текущей базе данных.

Дополнительные сведения

Необходимо определить либо один из аргументов «База данных» и «Новое имя», либо оба этих аргумента.

Если значения аргументов «Тип объекта» и «Имя объекта» оставлены пустыми, будет скопирован объект, выделенный в окне базы данных. Выделить объект в окне базы данных позволяет макрокоманда **ВыделитьОбъект (SelectObject)** со значением «Да» аргумента «В окне базы данных».

Вызов макрокоманды **КопироватьОбъект** эквивалентен выделению объекта в окне базы данных и последующему выбору команд **Копировать** и **Вставить** в меню **Правка**, после чего

на экране появляется диалоговое окно **Вставка**, в которое вводится новое имя объекта. Макрокоманда **КопироватьОбъект** выполняет все эти действия автоматически.

Кроме того, имеется возможность создать копию объекта, выбранного в окне базы данных, или объекта, открытого в окне таблицы, в окне запроса, в окне формы, в окне отчета, в окне макроса или в окне модуля, с помощью команды **Сохранить как/Экспорт** в меню **Файл**. В диалоговом окне **Сохранение объекта** предлагается выбрать сохранение копии объекта в текущей базе данных под новым именем или сохранение объекта в другой базе данных. Если объект уже был сохранен, то при его сохранении в текущей базе данных под новым именем остается исходная версия с прежним именем.

Указанная база данных-получатель должна существовать до вызова макрокоманды **КопироватьОбъект (CopyObject)**; в противном случае на экране появится сообщение об ошибке.

Для вызова макрокоманды **КопироватьОбъект (CopyObject)** в программе Visual Basic следует использовать метод **CopyObject** объекта **DoCmd.**

Макрокоманда «КомандаМеню» (DoMenuItem)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactDoMenuItemC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactDoMenuItemX":1}

Макрокоманда **КомандаМеню (DoMenuItem)** выполняет команду стандартного меню Microsoft Access.

Примечание. В Microsoft Access 97 макрокоманда **КомандаМеню (DoMenuItem)** заменена на **ВыполнитьКоманду (RunCommand)**. Макрокоманда **КомандаМеню (DoMenuItem)** включена в настоящую версию только для совместимости с более ранними версиями. При открытии и сохранении макроса из предыдущих версий Microsoft Access, содержащего макрокоманду **КомандаМеню (DoMenuItem)**, эта макрокоманда и ее аргументы автоматически преобразовываются в эквивалентную макрокоманду **ВыполнитьКоманду (RunCommand)**. В Microsoft Access 97 макрокоманда **КомандаМеню (DoMenuItem)** больше не появляется в списке макрокоманд в окне макроса.

При преобразовании базы данных Microsoft Access предыдущих версий некоторые команды могут оказаться недоступны. Они могут быть переименованы, перемещены в другое меню или удалены из Microsoft Access 97. Для таких команд макрокоманда **КомандаМеню (DoMenuItem)** не может быть преобразована в макрокоманду **ВыполнитьКоманду (RunCommand)**. При открытии макроса для них будет выведена макрокоманда **КомандаМеню (DoMenuItem)** с пустыми полями аргументов. Следует либо изменить макрос и ввести допустимую макрокоманду **ВыполнитьКоманду**, либо вообще удалить ее. Список команд предыдущих версий Microsoft Access, недоступных в Microsoft Access 97 содержится в разделе Аргументы макрокоманды «КомандаМеню» (DoMenuItem), недопустимые в макрокоманде «ВыполнитьКоманду» (RunCommand).

Данную макрокоманду используют для выполнения следующих действий:

- Вызов любой команды меню Microsoft Access из макроса. Выполняемая команда должна соответствовать режиму, который является текущим на момент ее вызова. Например, нельзя выполнить ни одну из команд меню **Записи**, если активный объект находится в режиме конструктора таблицы, формы, запроса, или отчета.
- Добавление команды стандартного меню Microsoft Access в специальное меню формы или отчета или в общее меню. Для каждой добавляемой команды следует ввести макрокоманду **КомандаМеню** в группу макросов, определяющую команды специального или общего меню.
- Добавление команды стандартного меню Microsoft Access в специальное контекстное меню формы, элемента управления формы или отчета или в общее контекстное меню. Для каждой добавляемой команды следует ввести макрокоманду **КомандаМеню** в группу макросов, определяющую команды специального или общего контекстного меню.

Значения

Макрокоманда **КомандаМеню (DoMenuItem)** использует следующие аргументы.

Аргумент	Описание
Строка меню	Имя строки меню, содержащей команду, которую следует выполнить. Имя строки меню является именем одного из режимов, в котором она появляется. Это имя следует ввести в ячейку аргумента «Строка меню» в разделе **Аргументы макрокоманды** окна макроса. Данный аргумент является обязательным.
Название меню	Название меню, содержащего команду, которую следует выполнить. Данный аргумент является обязательным.
Команда	Команда, которую следует выполнить. Данный аргумент является обязательным.
Подкоманда	Подкоманда, которую следует выполнить. Этот аргумент

применим только в том случае, если указанная команда открывает подменю.

Дополнительные сведения

Вызов макрокоманды **КомандаМеню (DoMenuItem)** эквивалентен выбору указанной команды в стандартном меню Microsoft Access. Если команда меню открывает диалоговое окно, это же диалоговое окно появляется при вызове макрокоманды **КомандаМеню (DoMenuItem)**. Для ввода данных в диалоговое окно в макросах используют макрокоманду **КомандыКлавиатуры (SendKeys).**

Для вызова макрокоманды **КомандаМеню (DoMenuItem)** в программе Visual Basic следует использовать метод **DoMenuItem** объекта **DoCmd.**

Макрокоманда «ВыводНаЭкран» (Echo)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactEchoC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactEchoX":1}

Макрокоманда **ВыводНаЭкран (Echo)** определяет режим вывода на экран результатов выполнения текущих операций. Например, эта макрокоманда позволяет вывести на экран или скрыть промежуточные результаты выполнения макроса.

Значения

Макрокоманда **ВыводНаЭкран (Echo)** использует следующие аргументы.

Аргумент	Описание
Включить вывод	Выберите значение «Да» (включает вывод на экран; используется по умолчанию) или «Нет» (отключает вывод) в разделе **Аргументы макрокоманды** окна макроса.
Текст строки состояния	Текст, который должен отображаться в строке состояния при отключении вывода на экран (например, «Ждите...»).

Дополнительные сведения

Во время выполнения макроса на экране часто отображается информация, без которой вполне можно обойтись. Если аргумент «Включить вывод» имеет значение «Нет», макрос выполняется без обновления экрана. По завершении выполнения макроса Microsoft Access автоматически включает вывод на экран и обновляет текущее окно. Значение «Нет» аргумента «Включить вывод» не влияет на результат выполнения макроса.

Макрокоманда **ВыводНаЭкран** не препятствует выводу модальных диалоговых окон (например, сообщений об ошибках) и всплывающих форм (таких как окна свойств), поэтому их можно использовать для сбора недостающих сведений и для вывода важной информации даже при отключенном выводе на экран. Для отключения вывода всех окон сообщений и диалоговых окон, кроме сообщений об ошибках и диалоговых окон, предназначенных для ввода данных, следует использовать макрокоманду **УстановитьСообщения (SetWarnings).**

В одном макросе допускается несколько вызовов макрокоманды **ВыводНаЭкран**. Это позволяет изменять текст в строке состояния во время выполнения макроса.

В режиме отключения вывода можно использовать макрокоманду **ПесочныеЧасы (Hourglass)** для изменения формы указателя на песочные часы (или любой другой значок указателя, заданный для состояния «Система недоступна» в параметрах настройки мыши в Microsoft Windows 95 или для состояния ожидания в Windows NT). Такой значок свидетельствует о выполнении макроса.

Для вызова макрокоманды **ВыводНаЭкран (Echo)** в программе Visual Basic следует использовать метод **Echo** объекта **DoCmd.**

Макрокоманда «СледующаяЗапись» (FindNext)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactFindNextC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactFindNextX":1}

Макрокоманда **СледующаяЗапись (FindNext)** находит следующую запись, удовлетворяющую условиям, указанным в предыдущей макрокоманде **НайтиЗапись (FindRecord)** или в диалоговом окне **Поиск в поле**, открывающемся командой **Найти** из меню **Правка**. Макрокоманда **СледующаяЗапись (FindNext)** позволяет выполнять повторный поиск записей. Например, можно последовательно просмотреть все записи со сведениями о конкретном клиенте.

Значения

Макрокоманда **СледующаяЗапись (FindNext)** не требует аргументов. Эта макрокоманда находит следующую запись, удовлетворяющую условиям, заданным либо в макрокоманде **НайтиЗапись (FindRecord)**, либо непосредственно в диалоговом окне **Поиск в поле**. Аргументы макрокоманды **НайтиЗапись (FindRecord)** соответствуют параметрам поиска, указанным в диалоговом окне **Поиск в поле**.

Для того чтобы задать условия поиска, следует использовать макрокоманду **НайтиЗапись**. Обычно, сначала в макрос вводят макрокоманду **НайтиЗапись**, а затем вызывают макрокоманду **СледующаяЗапись** для последовательного поиска записей, удовлетворяющих тем же условиям. Для выполнения поиска записей только при наличии определенных условий можно ввести условное выражение в столбец «Условие» в строке макрокоманды **СледующаяЗапись**.

Дополнительные сведения

Вызов данной макрокоманды эквивалентен нажатию кнопки **Поиск далее** в диалоговом окне **Поиск в поле**.

Примечание. Макрокоманда **НайтиЗапись (FindRecord)** соответствует команде **Найти** в меню **Правка** окна таблиц, запросов, и форм, но не соответствует команде **Найти** в меню **Правка** окна модуля. Макрокоманды **НайтиЗапись (FindRecord)** и **СледующаяЗапись (FindNext)** нельзя использовать для поиска текста в модулях.

Совет. Если в аргументе «Только в текущем поле» макрокоманды **НайтиЗапись (FindRecord)** задано значение «Да», то перед вызовом макрокоманды **СледующаяЗапись (FindNext)** может потребоваться вызов макрокоманды **КЭлементуУправления (GoToControl)** для перевода фокуса на элемент управления, содержащий искомые данные.

В Microsoft Access 97 и Microsoft Access 95 макрокоманда **СледующаяЗапись (FindNext)** выполняется иначе в отношении места, с которого начинается поиск следующего выполнения условия. В более ранних версиях Microsoft Access поиск всегда начинался со следующей записи в наборе записей. Начиная с Microsoft Access 95, если при выполнении макрокоманды **СледующаяЗапись (FindNext)** выделенный текст совпадает с искомым, поиск текста начинается сразу после выделенного в том же поле и в той же записи. Иначе поиск начинается с начала текущей записи. Это позволяет найти многократное вхождение одного и того же условия поиска в одной записи, в то время как Microsoft Access версии 2.0 позволяет найти только первое вхождение условия поиска в записи.

Однако следует отметить, что при использовании кнопки для запуска макроса, содержащего макрокоманду **СледующаяЗапись (FindNext)**, будет постоянно находиться первое вхождение условия поиска. Это происходит, потому что нажатие кнопки убирает фокус с поля, содержащего искомое значение, и макрокоманда **СледующаяЗапись (FindNext)** выполняет поиск с начала записи. При этом будет повторно найдено первое вхождение условия поиска в той же самой записи. Во избежание этого для запуска макроса следует использовать технологию, не изменяющую фокус, например кнопку специальной панели инструментов или

комбинацию клавиш, определенную в макросе AutoKeys, или в макросе, перед выполнением макрокоманды **СледующаяЗапись (FindNext)**, устанавливать фокус на поле, содержащее условие поиска.

То же самое происходит при запуске с помощью кнопки макроса, содержащего макрокоманду **НайтиЗапись (FindRecord)** со значением «Нет» аргумента «Первое вхождение». Сведения о том, как выполняется поиск с помощью макрокоманды **СледующаяЗапись (FindNext)**, можно найти в статье Q150665 «FindNext Does Not Advance To Next Record To Search» в базе данных Microsoft Knowledge Base. Для подключения к странице поиска базы данных Microsoft Access Knowledge Base нажмите кнопку  . Сведения о доступе к базе данных Knowledge Base или другим техническим ресурсам содержатся в разделе Подключение к службам технической поддержки Microsoft.

Для вызова макрокоманды **СледующаяЗапись (FindNext)** в программе Visual Basic следует использовать метод **FindNext** объекта **DoCmd.**

Макрокоманда «НайтиЗапись» (FindRecord)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactFindRecordC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactFindRecordX":1}

Макрокоманда **НайтиЗапись (FindRecord)** находит данные, удовлетворяющие условиям поиска, указанным с помощью аргументов этой макрокоманды. Данные могут быть в текущей записи, в следующей или предыдущей записи или в самой первой. Допускается поиск записей в текущей таблице в режиме таблицы, в запросе в режиме таблицы, в форме в режиме таблицы или в форме.

Значения

Макрокоманда **НайтиЗапись (FindRecord)** использует следующие аргументы.

Аргумент	Описание
Образец поиска	Искомое значение: текст, число или дата. При определении этого аргумента в ячейке аргумента «Образец поиска» в разделе **Аргументы макрокоманды** окна макроса можно использовать выражение с предшествующим знаком равенства (=). Допускается использование подстановочных знаков. Данный аргумент является обязательным.
Совпадение	Указывает положение искомого значения в поле. Допускается поиск образца в любой части поля («С любой частью поля»); образца, полностью заполняющего поле («Поля целиком», используется по умолчанию), а также образца, расположенного в начале поля («С начала поля»).
С учетом регистра	Указывает, следует ли учитывать регистр символов при выполнении поиска. Если этот аргумент имеет значение «Да», строчные и прописные буквы считаются разными. По умолчанию используется значение «Нет».
Область поиска	Указывает направление поиска относительно текущей записи: «Вниз» - вниз от текущей записи до последней записи; «Вверх» - вверх от текущей записи до первой записи; или «Все» (задается по умолчанию) - вниз от текущей записи до последней записи с продолжением от первой записи до текущей.
С учетом формата поля	Указывает, следует ли учитывать формат данных при выполнении поиска. Если этот аргумент имеет значение «Да», данные анализируются в том виде, в котором они изображаются на экране. Если этот аргумент имеет значение «Нет» (используется по умолчанию), данные анализируются в том виде, в котором они хранятся в базе данных. Этот аргумент позволяет проводить поиск данных в определенном формате. Например, выберите «Да» и укажите в аргументе «Образец поиска» 1 234 для поиска значения в формате в разделителями тысяч. Если этот аргумент имеет значение «Нет», то для поиска этого значения можно ввести просто 1234. При поиске дат выберите «Да», если требуется найти дату в определенном формате (например, 03-мар-1991). Если этот аргумент имеет значение «Нет», дату в аргументе «Образец поиска» следует вводить в кратком формате

	даты, установленном в панели управления Microsoft Windows. Например, если выбран краткий формат даты «м/д/гг», можно ввести 3/9/91, и Microsoft Access найдет в поле даты все значения, соответствующие 9 марта 1991 года, вне зависимости от формата этого поля. **Примечание.** Этот аргумент используется только в том случае, если текущее поле является присоединенным элементом управления, аргумент «Совпадение» имеет значение «Поля целиком», аргумент «Только в текущем поле» значение «Да», а аргумент «С учетом регистра» значение «Нет». Если задать значение «Да» для аргумента «С учетом регистра» или значение «Нет» для аргумента «Только в текущем поле», то необходимо задать значение «Да» для аргумента «С учетом формата поля».
Только в текущем поле	Указывает, следует ли ограничить поиск только текущим полем в каждой записи («Да») или проверять все поля во всех записях («Нет»). Поиск только в текущем поле выполняется быстрее. Значение по умолчанию «Да».
Первое вхождение	Указывает, с какой записи следует начать выполнение поиска: с первой («Да»; используется по умолчанию) или с текущей («Нет»).

Дополнительные сведения

При выполнении в макросе макрокоманды **НайтиЗапись (FindRecord)** осуществляется переход к следующей записи, которая содержит указанные данные, и выделение искомого значения в этой записи (порядок поиска определяется аргументом «Область поиска»).

Вызов макрокоманды **НайтиЗапись (FindRecord)** эквивалентен выбору команды **Найти** в меню **Правка**, а ее аргументы эквивалентны параметрам в диалоговом окне **Поиск в поле**. После указания значений аргументов макрокоманды в окне макроса и выполнения этого макроса в диалоговом окне **Поиск в поле** будут установлены соответствующие значения параметров.

Microsoft Access сохраняет последний набор аргументов макрокоманды **НайтиЗапись (FindRecord)** в течение сеанса работы с базой данных; поэтому нет необходимости вводить одни и те же условия при выполнении повторного поиска. Если оставить какой-либо аргумент пустым, Microsoft Access использует последнее значение этого аргумента, установленное предыдущей макрокомандой **НайтиЗапись (FindRecord)** или в диалоговом окне **Поиск в поле**.

Для поиска записей с помощью макроса рекомендуется использовать макрокоманду **НайтиЗапись (FindRecord)**, а не макрокоманду **ВыполнитьКоманду (RunCommand),** аргументы которой соответствуют команде **Найти**.

Примечание. Макрокоманда **НайтиЗапись (FindRecord)** соответствует команде **Найти** в меню **Правка** окна таблицы, запроса и формы, но не соответствует команде **Найти** в меню **Правка** окна модуля. Макрокоманду **НайтиЗапись (FindRecord)** нельзя использовать для поиска текста в модулях.

В Microsoft Access 97 и Microsoft Access 95 макрокоманда **НайтиЗапись (FindRecord)** выполняется иначе в отношении места, с которого начинается поиск следующего выполнения условия, когда значение аргумента «Первое вхождение» равно «Нет». В более ранних версиях Microsoft Access поиск всегда начинался со следующей (или предыдущей) записи в наборе записей. Начиная с Microsoft Access 95, если при выполнении макрокоманды **НайтиЗапись (FindRecord)** выделенный текст совпадает с искомым, поиск текста начинается сразу после выделенного в том же поле и в той же записи. Иначе поиск начинается с начала текущей

записи. Это позволяет найти многократное вхождение одного и того же условия поиска в одной записи, в то время как Microsoft Access версии 2.0 позволяет найти только первое вхождение условия поиска в записи.

Однако следует отметить, что при использовании кнопки для запуска макроса, содержащего макрокоманду **НайтиЗапись (FindRecord)**, будет постоянно находиться первое вхождение условия поиска. Это происходит, потому что нажатие кнопки убирает фокус с поля, содержащего искомое значение, и макрокоманда **НайтиЗапись (FindRecord)** выполняет поиск с начала записи. При этом будет повторно найдено первое вхождение условия поиска в той же самой записи. Во избежание этого для запуска макроса следует использовать технологию, не изменяющую фокус, например кнопку специальной панели инструментов или комбинацию клавиш, определенную в макросе AutoKeys, или в макросе, перед выполнением макрокоманды **НайтиЗапись (FindRecord)**, устанавливать фокус на поле, содержащее условие поиска.

То же самое происходит при запуске с помощью кнопки макроса, содержащего макрокоманду **СледующаяЗапись (FindNext)**. Сведения о том, как выполняется поиск с помощью макрокоманды **СледующаяЗапись (FindNext)**, можно найти в статье Q150665, «FindNext Does Not Advance To Next Record To Search» в базе данных Microsoft Knowledge Base Для подсоединения к странице поиска базы данных Microsoft Access Knowledge Base нажмите кнопку . Сведения о доступе к Knowledge Base или другим техническим ресурсам содержатся в разделе Подключение к службам технической поддержки Microsoft.

Для вызова макрокоманды **НайтиЗапись (FindRecord)** в программе Visual Basic следует использовать метод **FindRecord** объекта **DoCmd.**

Макрокоманда «КЭлементуУправления» (GoToControl)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactGoToControlC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactGoTOControlX":1}

Макрокоманда **КЭлементуУправления (GoToControl)** переводит фокус на указанное поле или элемент управления в текущей записи открытой формы, формы в режиме таблицы, таблицы в режиме таблицы или динамического набора, полученного при выполнении запроса. Эту макрокоманду используют для передачи фокуса конкретному полю или элементу управления. Затем это поле или элемент управления можно использовать для сравнения или выполнения макрокоманды **НайтиЗапись (FindRecord)**. Кроме того, эту макрокоманду используют для автоматического перемещения по форме согласно определенным условиям. Например, если оператор введет «Да» в поле **Холост**, то поле **Супруг** будет пропущено автоматически, а фокус передан следующему элементу управления.

Значения

Макрокоманда **КЭлементуУправления (GoToControl)** использует следующий аргумент.

Аргумент	Описание
Имя элемента	Имя поля или элемента управления, которому следует передать фокус. Имя поля или элемента управления следует ввести в ячейке аргумента «Имя элемента» в разделе **Аргументы макрокоманды** окна макроса. Данный аргумент является обязательным. **Примечание.** В качестве значения аргумента «Имя элемента» следует вводить только имя поля или элемента управления, а не полную ссылку, такую как Forms!Товары![Товар].

Дополнительные сведения

С помощью макрокоманды **КЭлементуУправления (GoToControl)** нельзя передать фокус элементу управления в скрытой форме.

Совет. Макрокоманда **КЭлементуУправления (GoToControl)** позволяет передать фокус подчиненной форме, которая является одним из элементов управления. Затем с помощью макрокоманды **НаЗапись (GoToRecord)** можно перейти к конкретной записи подчиненной формы. Для передачи фокуса элементу управления, размещенному в подчиненной форме, следует использовать макрокоманду **КЭлементуУправления (GoToControl)** для перевода фокуса в подчиненную форму, а затем снова вызвать эту макрокоманду для передачи фокуса элементу управления.

Для вызова макрокоманды **КЭлементуУправления (GoToControl)** в программе Visual Basic следует использовать метод **GoToControl** объекта **DoCmd.** Кроме того, можно использовать метод **SetFocus** для перевода фокуса на элемент управления в форме или в подчиненной форме, а также для перевода фокуса на поле в открытой таблице, в запросе в режиме таблицы или в форме в режиме таблицы.

Макрокоманда «НаСтраницу» (GoToPage)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactGoToPageC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactGoToPageX":1}

Макрокоманда **НаСтраницу (GoToPage)** передает фокус в активной форме первому элементу управления, размещенному на указанной странице активной формы. Данную макрокоманду используют при создании многостраничной формы с группировкой родственных данных на разных страницах. Например, можно создать форму «Сотрудники», на одной странице которой выводятся личные данные, на другой служебные, а на третьей сведения о выработке. Макрокоманда **НаСтраницу (GoToPage)** позволит перейти на нужную страницу такой формы. Несколько страниц информации могут быть размещены в одной форме с использованием элемента управления набор вкладок.

Значения

Макрокоманда **НаСтраницу (GoToPage)** использует следующие аргументы.

Аргумент	Описание
Номер страницы	Номер страницы, на которую переводится фокус. Введите номер страницы в ячейке аргумента «Номер страницы» в разделе **Аргументы макрокоманды** окна макроса. Если оставить данный аргумент пустым, то фокус останется на текущей странице. Аргументы «От левого края» и «От верхнего края» позволяют вывести на экран нужную часть страницы.
От левого края	Горизонтальное положение верхнего левого угла страницы относительно левого края активного окна. Если определен аргумент «От верхнего края», данный аргумент является обязательным.
От верхнего края	Вертикальное положение верхнего левого угла страницы относительно верхнего края активного окна. Если определен аргумент «От левого края», данный аргумент является обязательным.

Примечание. Значения аргументов «От левого края» и «От верхнего края» измеряются в дюймах или сантиметрах, в зависимости от единиц измерения, установленных в панели управления Microsoft Windows.

Дополнительные сведения

Данную макрокоманду используют для выделения первого (в последовательности перехода) в форме) элемента управления на указанной странице. Для перемещения к определенному элементу управления следует использовать макрокоманду **КЭлементуУправления (GoToControl)**.

Аргументы «От левого края» и «От верхнего края» используются также при работе с формой, размеры страницы которой превышают размеры окна Microsoft Access. Аргумент «Номер страницы» задает перемещение к нужной странице, а аргументы «От левого края» и «От верхнего края» определяют фрагмент страницы, который следует вывести на экран.

С помощью макрокоманды **НаСтраницу (GoToPage)** нельзя перевести фокус на конкретную страницу в скрытой форме.

Для вызова макрокоманды **НаСтраницу (GoToPage)** в программе Visual Basic следует использовать метод **GoToPage** объекта **DoCmd.**

Макрокоманда «НаЗапись» (GoToRecord)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactGotoRecordC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactGotoRecordX":1}

Макрокоманда **НаЗапись (GoToRecord)** делает указанную запись текущей записью открытой таблицы, формы или результирующего набора записей запроса.

Значения

Макрокоманда **НаЗапись (GoToRecord)** использует следующие аргументы.

Аргумент	Описание
Тип объекта	Тип объекта, содержащего запись, которую следует сделать текущей. Допустимые значения «Таблица», «Запрос» и «Форма» выбираются в раскрывающемся списке в разделе **Аргументы макрокоманды** окна макроса. Для того чтобы выбрать активный объект, оставьте данный аргумент пустым.
Имя объекта	Имя объекта, содержащего запись, которую следует сделать текущей. В раскрывающийся списке в ячейке выводятся имена всех объектов текущей базы данных, которые относятся к типу, указанному в аргументе «Тип объекта». Если значение аргумента «Тип объекта» оставлено пустым, оставьте пустым и значение этого аргумента.
Запись	Запись, которую следует сделать текущей. Допустимые значения: «Предыдущая», «Следующая» (используется по умолчанию), «Первая», «Последняя», «Конкретная» и «Новая».
Смещение	Целое число или выражение с предшествующим знаком равенства, имеющее целое значение. Этот аргумент определяет запись, которую следует сделать текущей, одним из следующих способов: • Если аргумент «Запись» имеет значение «Следующая» или «Предыдущая», осуществляется перемещение вперед или назад на число записей, указанное в аргументе «Смещение». • Если аргумент «Запись» имеет значение «Конкретная», осуществляется перемещение на запись, номер которой равняется значению аргумента «Смещение». Номер записи отображается в поле номера записи в нижней части окна. **Примечание.** Если аргумент «Запись» имеет значение «Первая», «Последняя» или «Новая», значение аргумента «Смещение» игнорируется. Слишком большое или отрицательное значение аргумента «Смещение» приводит к возникновению ошибки.

Дополнительные сведения

Если фокусом обладает конкретный элемент управления в записи, то после выполнения данной макрокоманды фокус будет переведен на этот же элемент управления в новой записи.

Для перемещения на новую пустую запись в конце формы или таблицы и ввода новых данных используйте значение «Новая» аргумента «Запись».

Вызов этой макрокоманды эквивалентен выбору команды **Переход** в меню **Правка**. Подменю содержит следующие команды: **первая запись**, **последняя запись**, **следующая запись**, **предыдущая запись** и **новая запись**. Эти команды выполняют те же действия над выделенным объектом, что и аналогичные значения аргумента «Запись». Кроме того, для

перемещения по записям можно воспользоваться кнопками перехода в нижней части окна.

Макрокоманда **НаЗапись (GoToRecord)** позволяет сделать текущей запись в скрытой форме, указанной в аргументах «Тип объекта» и «Имя объекта».

Для вызова макрокоманды **НаЗапись (GoToRecord)** в программе Visual Basic следует использовать метод **GoToRecord** объекта **DoCmd.**

Макрокоманда «ПесочныеЧасы» (Hourglass)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactHourglassC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactHourglassX":1}

Макрокоманда **ПесочныеЧасы (Hourglass)** придает указателю форму песочных часов (или другого указанного значка) на время выполнения макроса. Данная макрокоманда позволяет создать указатель, показывающий, что макрос выполняется. Это особенно удобно, если выполнение макрокоманды или самого макроса занимает достаточно большое время.

Значения

Макрокоманда **ПесочныеЧасы (Hourglass)** использует следующий аргумент.

Аргумент	Описание
Включить	Для вывода значка песочных часов в ячейке аргумента «Включить» в разделе **Аргументы макрокоманды** окна макроса выберите «Да» (используется по умолчанию). Для восстановления обычной формы указателя выберите «Нет».

Дополнительные сведения

Обычно, данная макрокоманда используется при отключении режима отображения с помощью макрокоманды **ВыводНаЭкран (Echo).** В таком режиме экран не обновляется до завершения выполнения макроса.

По завершении выполнения макроса обычная форма указателя восстанавливается автоматически.

Примечания

- В Microsoft Windows 95 данная макрокоманда выводит значок указателя, задающийся для состояния «Система недоступна» на вкладке **Указатели** в диалоговом окне **Свойства: Мышь** в окне настроек Windows (по умолчанию указатель имеет форму песочных часов).
- В Windows NT данный значок задается на панели управления для состояния ожидания (по умолчанию, указатель также имеет форму песочных часов).
- В любом случае пользователь имеет возможность выбрать другую форму указателя.

Для вызова макрокоманды **ПесочныеЧасы (Hourglass)** в программе Visual Basic следует использовать метод **Hourglass** объекта **DoCmd.**

Макрокоманда «Развернуть» (Maximize)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactMaximizeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactMaximizeX":1}

Макрокоманда **Развернуть (Maximize)** увеличивает размеры активного окна до размеров окна Microsoft Access. Используйте эту макрокоманду, если в активном окне требуется отобразить как можно большую часть объекта.

Значения

Макрокоманда **Развернуть (Maximize)** не требует аргументов.

Дополнительные сведения

Вызов этой макрокоманды эквивалентен нажатию кнопки развертывания окна в правом верхнем углу окна, или выбору команды **Развернуть** в оконном меню.

Для восстановления прежних размеров развернутого окна следует использовать макрокоманду **Восстановить (Restore).**

Советы

- Если окно, которое требуется развернуть, не является текущим, следует предварительно выполнить макрокоманду **ВыделитьОбъект (SelectObject).**
- При развертывании окна в Microsoft Access все остальные окна при их открытии или переключении в них также развертываются, за исключением всплывающих форм. Чтобы формы сохраняли свой размер при развертывании других окон следует в их свойстве **Всплывающее окно (Pop Up)** установить значение «Да». Сведения о развертывании Microsoft Access окон можно найти в статье Q147152, «Maximizing One Form Maximizes All Forms (7.0)» в базе данных Microsoft Knowledge Base. Для подсоединения к странице поиска базы данных Microsoft Access Knowledge Base нажмите кнопку  . Сведения о доступе к Knowledge Base или другим техническим ресурсам содержатся в разделе Подключение к службам технической поддержки Microsoft.

Для вызова макрокоманды **Развернуть (Maximize)** в программе Visual Basic следует использовать метод **Maximize** объекта **DoCmd.**

Макрокоманда «Свернуть» (Minimize)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactMinimizeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactMinimizeX":1}

Макрокоманда **Свернуть (Minimize)** свертывает активное окно в маленький значок в нижней части окна Microsoft Access.

Значения

Макрокоманда **Свернуть (Minimize)** не требует аргументов.

Дополнительные сведения

Данная макрокоманда позволяет удалить окно с экрана, оставляя при этом объект открытым. Кроме того, эту макрокоманду используют, если требуется открыть объект, не открывая его окно. Если потребуется вывести объект на экран, вызовите макрокоманду **ВыделитьОбъект (SelectObject)** вместе с макрокомандой **Развернуть (Maximize)** или **Восстановить (Restore).** Макрокоманда **Восстановить (Restore)** восстанавливает размеры окна, которые окно имело до свертывания.

Вызов макрокоманды **Свернуть (Minimize)** эквивалентен нажатию кнопки свертывания окна, расположенной в правом верхнем углу окна, или выбору команды **Свернуть** в оконном меню.

Советы

- Перед тем как приступить к изменению размеров окна, не являющегося текущим, следует сделать это окно текущим с помощью макрокоманды **ВыделитьОбъект (SelectObject)**.
- Чтобы свернуть окно базы данных, используйте макрокоманду **ВыделитьОбъект (SelectObject)** с аргументом «В окне базы данных», имеющим значение «Да», а затем макрокоманду **Свернуть (Minimize)**.
- Для временного удаления активного окна с экрана можно использовать команду **Скрыть** (меню **Окно**). В этом случае окно не свертывается в значок, а просто становится невидимым. Для возврата на экран временно удаленного окна используйте команду **Отобразить** (меню **Окно**). Вызов любой из этих команд из макроса осуществляется с помощью макрокоманды **ВыполнитьКоманду (RunCommand).**
- Кроме того, макрокоманда **ЗадатьЗначение (SetValue)** позволяет указать значение свойства формы **Вывод на экран (Visible),** определяющее вывод на экран или скрытие окна формы.

Для вызова макрокоманды **Свернуть (Minimize)** в программе Visual Basic следует использовать метод **Minimize** объекта **DoCmd.**

Макрокоманда «СдвигРазмер» (MoveSize)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactMoveSizeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactMoveSizeX":1}

Макрокоманда **СдвигРазмер (MoveSize)** изменяет положение или размеры активного окна.

Значения

Макрокоманда **СдвигРазмер (MoveSize)** использует следующие аргументы.

Аргумент	Описание
От левого края	Новое положение верхнего левого угла окна (по горизонтали) относительно левого края содержащего его окна. Значение вводится в ячейку аргумента «От левого края» в разделе **Аргументы макрокоманды** окна макроса.
От верхнего края	Новое положение верхнего левого угла окна (по вертикали) относительно верхнего края содержащего его окна.
Ширина	Новая ширина окна.
Высота	Новая высота окна.

Если значение аргумента будет оставлено пустым, Microsoft Access использует соответствующий параметр текущего окна.

Необходимо определить хотя бы один аргумент.

Примечание. Значения всех четырех аргументов измеряются в дюймах или сантиметрах, в зависимости от единиц измерения, установленных в панели управления Windows.

Дополнительные сведения

Вызов этой макрокоманды эквивалентен выбору команды **Переместить** или **Размер** в оконном меню. Отличие состоит в том, что после выбора команды в меню пользователь может использовать клавиши перемещения курсора для изменения положения или размеров окна, а при использовании макрокоманды **СдвигРазмер (MoveSize)** все размеры должны быть указаны в аргументах. Для перемещения и изменения размеров окон можно также использовать мышь

Эту макрокоманду можно вызывать для любого окна в любом режиме.

Советы

- Для того чтобы изменить положение окна, не изменяя его размеры, введите значения аргументов «От левого края» и «От верхнего края», а аргументы «Ширина» и «Высота» оставьте пустыми.
- Для того чтобы изменить размеры окна, не изменяя его положение, введите значения аргументов «Ширина» и «Высота», а аргументы «От левого края» и «От верхнего края» оставьте пустыми.

Для вызова макрокоманды **СдвигРазмер (MoveSize)** в программе Visual Basic следует использовать метод **MoveSize** объекта **DoCmd.**

Макрокоманда «Сообщение» (MsgBox)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactMsgBoxC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactMsgBoxX":1}

Макрокоманда **Сообщение (MsgBox)** выводит на экран окно, содержащее предупреждение или информационное сообщение. Например, допускается использование макрокоманды **Сообщение** в макросах, выполняющих проверку условий на значение. Если условие на значение нарушается в элементе управления или в записи, на экран выводится сообщения о том, что введены неправильные данные, и инструкции по вводу требуемых данных.

Значения

Макрокоманда **Сообщение (MsgBox)** использует следующие аргументы.

Аргумент	Описание
Сообщение	Текст, выводящийся в окне сообщения. Его необходимо ввести в ячейку аргумента «Сообщение» в разделе **Аргументы макрокоманды** окна макроса. Допускается ввод строкового значения (до 255 символов) или выражения(с предшествующим знаком равенства).
Сигнал	Определяет подачу звукового сигнала при выводе сообщения. Допустимые значения: «Да» (подается один звуковой сигнал; используется по умолчанию) и «Нет».
Тип	Тип окна сообщения. Каждому типу соответствует свой значок. Допустимые значения: «Отсутствует» (используется по умолчанию), «Критическое», «Предупреждающее?», «Предупреждающее!» и «Информационное».
Заголовок	Текст, выводящийся в строке заголовка окна сообщения, например, «Неправильный код клиента». Если значение этого аргумента оставлено пустым, заголовок окна сообщения будет содержать надпись «Microsoft Access».

Дополнительные сведения

Макрокоманда **Сообщение (MsgBox)** создает форматированное сообщение об ошибке, аналогичное встроенным сообщениям об ошибках Microsoft Access. Аргумент «Сообщение» макрокоманды **Сообщение (MsgBox)** позволяет ввести текст, состоящий из трех разделов, которые разделяются символом **@**.

В следующем примере выводится окно с форматированным сообщением. Первый раздел задает текст заголовка сообщения, который отображается полужирным шрифтом. Текст второго раздела выводится обычным шрифтом под заголовком. Текст третьего раздела выводится обычным шрифтом под заголовком **Решение**.

Введите следующее значение в ячейку аргумента «Сообщение»:

Неверная кнопка!@Эта кнопка не работает.@Нажмите другую.

Примечание. Если окно помощника является видимым, сообщение появится в окне сообщений помощника, а не в отдельном окне сообщений.

Microsoft Windows 95 и Windows NT используют несколько отличающиеся значки для разных типов окон сообщений.

Вызов макрокоманды **Сообщение (MsgBox)** в программе Visual Basic невозможен. Вместо этого следует использовать функцию **MsgBox.**

Макрокоманда «ОткрытьФорму» (OpenForm)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactOpenFormC;dasqlWhere":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactOpenFormX":1}

Макрокоманда **ОткрытьФорму (OpenForm)** открывает форму в режиме формы, в режиме конструктора формы, в режиме предварительного просмотра или в режиме таблицы. При этом пользователь имеет возможность выбрать режим ввода данных и режим окна, а также отобрать записи, которые следует показать в форме.

Значения

Макрокоманда **ОткрытьФорму (OpenForm)** использует следующие аргументы.

Аргумент	Описание
Имя формы	Имя формы, которую следует открыть. Раскрывающийся список в ячейке «Имя формы» в разделе **Аргументы макрокоманды** окна макроса содержит имена всех форм текущей базы данных. Данный аргумент является обязательным. При запуске макроса, содержащего макрокоманду **ОткрытьФорму (OpenForm)**, из библиотечной базы данных, Microsoft Access проводит поиск формы с указанным именем сначала в библиотечной базе данных, а затем в текущей базе данных.
Режим	Режим открытия формы. Допустимые значения: «Форма» (используется по умолчанию), «Конструктор», «Просмотр» и «Таблица» выбираются в раскрывающемся списке. **Примечание.** Значение аргумента «Режим» имеет приоритет над значениями свойств формы **Режим по умолчанию (DefaultView)** и **Допустимые режимы (ViewsAllowed).** Например, даже если свойство формы **Допустимые режимы (ViewsAllowed)** имеет значение «Таблица», макрокоманда **ОткрытьФорму** позволяет открыть эту форму в режиме формы.
Имя фильтра	Имя фильтра, который следует использовать для отбора и сортировки записей формы. Введите имя существующего запроса или имя фильтра, сохраненного в виде запроса. При этом запрос должен содержать все поля, представленные в форме, или его свойство **Вывод всех полей (OutputAllFields)** должно иметь значение «Да».
Условие отбора	Правильно составленное предложение WHERE (SQL) (без слова WHERE) или выражение, которое следует использовать для отбора записей из базовой таблицы или базового запроса формы. Если значение аргумента «Имя фильтра» определено, Microsoft Access применит это предложение WHERE к фильтру. Для того чтобы открыть форму и отобрать те ее записи, которые содержат значения, совпадающие со значением элемента управления в другой форме, используйте следующий синтаксис: **[***имяПоля***] = Forms![***имяФормы***]!** **[***имяЭлементаВДругойФорме***]** Здесь аргумент *имяПоля* представляет имя поля в базовой

	таблице или базовом запросе формы, которую следует открыть. Аргумент *имяЭлементаВДругойФорме* представляет имя элемента управления в другой форме, который содержит нужное значение. **Примечание.** Максимальная длина значения аргумента «Условие отбора» равняется 256 символам. Для ввода более сложной инструкции SQL с предложением WHERE (длиной до 32 768 символов) следует вызвать метод **OpenForm** объекта **DoCmd** в программе Visual Basic.
Режим данных	Режим ввода данных в форму. Этот аргумент применим только к формам, открываемым в режиме формы или режиме таблицы. Допустимые значения: «Добавление» (пользователь может вводить новые, но не может изменять существующие записи), «Изменение» (пользователь может вводить новые и изменять существующие записи; используется по умолчанию) и «Только чтение» (пользователю разрешается только просматривать записи). **Примечания** • Значение аргумента «Режим данных» имеет приоритет над значениями свойств формы **Разрешить изменение (AllowEdits)**, **Разрешить удаление (AllowDeletions)**, **Разрешить добавление (AllowAdditions)** и **Ввод данных (DataEntry)**. Например, даже если свойство **Разрешить изменение (AllowEdits)** имеет значение «Нет», макрокоманда **ОткрытьФорму (OpenForm)** позволяет открыть эту форму в режиме изменения данных. • Если этот аргумент оставлен пустым, форма будет открыта в режиме ввода данных, определяемом свойствами формы **Разрешить изменение (AllowEdits)**, **Разрешить удаление (AllowDeletions)**, **Разрешить добавление** (**AllowAdditions)** и **Ввод данных (DataEntry)**.
Режим окна	Режим окна, в котором открывается форма. Допустимые значения: «Обычное» (режим открытия окна формы определяется настройками ее свойств, используется по умолчанию), «Невидимое» (скрытая форма), «Значок» (форма открывается как значок в нижней части экрана) и «Окно диалога» (свойства формы **Модальное окно (Modal)** и **Всплывающее окно (PopUp)** получают значение «Да»).

Дополнительные сведения

Вызов этой макрокоманды эквивалентен нажатию кнопки **Открыть** или **Конструктор** в окне базы данных после выбора формы на вкладке **Формы**.

Форма может быть модальной (это означает, что ее необходимо закрыть или убрать с экрана, чтобы получить возможность работать с другим окнами) или немодальной (пользователь в любой момент может перейти в другое окно). Кроме того, она может быть всплывающей (это означает, что она всегда открывается поверх всех остальных окон Microsoft Access). Значения свойств **Модальное окно (Modal)** и **Всплывающее окно (PopUp)** устанавливаются при разработке формы. Если для аргумента «Режим окна» задано значение «Обычное», режим открытия окна формы определяется этими свойствами. Если аргумент «Режим окна» имеет значение «Окно диалога», для обоих свойств автоматически устанавливается значение «Да». При выводе на экран или восстановлении формы, открытой как скрытая форма или значок, восстанавливается режим окна, указанный в значениях ее свойств.

Если форма открывается с заданным для аргумента «Режим окна» значением «Окно диалога», выполнение макроса приостанавливается до закрытия формы или ее удаления с экрана. Для того чтобы скрыть форму, задайте для ее свойства **Вывод на экран (Visible)** значение «Нет» с помощью макрокоманды **ЗадатьЗначение (SetValue).**

Совет. Можно выбрать форму в окне базы данных и переместить ее с помощью мыши в строку макрокоманды в макросе. При этом в макрос автоматически добавляется макрокоманда **ОткрытьФорму (OpenForm)**, открывающая форму в режиме формы.

При переводе открытой формы в режим конструктора значения аргументов «Режим данных» и «Режим окна» теряются и не восстанавливаются даже при возврате в режим формы или таблицы.

Фильтр или предложение WHERE становятся значением свойства **Фильтр (Filter)** формы или отчета.

Макрокоманда «ОткрытьЗапрос» (OpenQuery)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactOpenQueryC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactOpenQueryX":1}

Макрокоманда **ОткрытьЗапрос (OpenQuery)** открывает запрос на выборку или перекрестный запрос в режиме таблицы, в режиме конструктора или в режиме предварительного просмотра. Эта макрокоманда запускает запрос на изменение. Кроме того, данная макрокоманда позволяет указать для запроса режим ввода данных.

Значения

Макрокоманда **ОткрытьЗапрос (OpenQuery)** использует следующие аргументы.

Аргумент	Описание
Имя запроса	Имя запроса, который следует открыть. В раскрывающемся списке в ячейке «Имя запроса» в разделе **Аргументы макрокоманды** окна макроса содержатся имена всех запросов в текущей базе данных. Данный аргумент является обязательным. Если макрос, содержащий макрокоманду **ОткрытьЗапрос (OpenQuery)**, запущен из библиотечной базы данных, поиск запроса с указанным именем проводится сначала в библиотечной базе данных, а затем в текущей базе данных.
Режим	Режим открытия запроса. Допустимые значения: «Таблица» (используется по умолчанию), «Конструктор» и «Просмотр».
Режим данных	Режим ввода данных в запрос. Этот аргумент применим только к запросам, открываемым в режиме таблицы. Допустимые значения: «Добавление» (пользователь может вводить новые записи, но не может изменять существующие), «Изменение» (пользователь может вводить новые и изменять существующие записи; используется по умолчанию) и «Только чтение» (пользователю разрешается только просматривать записи).

Дополнительные сведения

Если для аргумента «Режим» выбрано значение «Таблица», Microsoft Access выводит на экран динамический набор записей (для запроса на выборку, перекрестного запроса, запроса на объединение или запроса к серверу, в свойстве которого **Возврат записей (ReturnsRecords)** задано значение «Да»). Данная макрокоманда запускает запрос на изменение, управляющий запрос или запрос к серверу, у которого свойство **Возврат записей (ReturnsRecords)** имеет значение «Нет».

Вызов макрокоманды **ОткрытьЗапрос (OpenQuery)** эквивалентен нажатию кнопки **Открыть** или **Конструктор** в окне базы данных после выбора запроса на вкладке **Запросы**. Эта макрокоманда предоставляет пользователю более широкий выбор параметров.

Совет. Можно выбрать запрос в окне базы данных и переместить его с помощью мыши в строку макрокоманды. в макросе. При этом в макрос автоматически добавляется макрокоманда **ОткрытьЗапрос (OpenQuery)**, открывающая запрос в режиме таблицы.

При переводе открытого запроса в режим конструктора значение аргумента «Режим данных» теряется и не восстанавливается даже при возврате в режим таблицы.

Совет. Вывод системных сообщений, которые появляются на экране во время выполнения запроса на изменение и показывают количество записей, подлежащих обработке, можно отключить с помощью макрокоманды **УстановитьСообщения (SetWarnings).**

Для вызова макрокоманды **ОткрытьЗапрос (OpenQuery)** в программе Visual Basic следует использовать метод **OpenQuery** объекта **DoCmd.**

Макрокоманда «ОткрытьОтчет» (OpenReport)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactOpenReportC;dasqlWhere":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactOpenReportX":1}

Макрокоманда **ОткрытьОтчет (OpenReport)** открывает отчет в режиме конструктора, в режиме предварительного просмотра или сразу выводит отчет на печать. При этом допускается отбор записей, которые следует включить в отчет.

Значения

Макрокоманда **ОткрытьОтчет (OpenReport)** использует следующие аргументы.

Аргумент	Описание
Имя отчета	Имя отчета, который следует открыть. В раскрывающемся списке в ячейке «Имя отчета» в разделе **Аргументы макрокоманды** окна макроса содержатся имена всех отчетов в текущей базы данных. Данный аргумент является обязательным. Если макрос, содержащий макрокоманду **ОткрытьОтчет (OpenReport)**, запущен из библиотечной базы данных, поиск отчета с указанным именем проводится сначала в библиотечной базе данных, а затем в текущей базе данных.
Режим	Режим открытия отчета. Допустимые значения: «Печать» (выводит отчет на печать), «Конструктор» или «Просмотр» (используется по умолчанию) выбираются в раскрывающемся списке.
Имя фильтра	Имя фильтра, который следует использовать для отбора записей отчета. Введите имя существующего запроса или фильтра, сохраненного в виде запроса. При этом запрос должен содержать все поля, представленные в отчете, или свойство запроса **Вывод всех полей (OutputAllFields)** должно иметь значение «Да».
Условие отбора	Правильно составленное предложение WHERE (SQL) (без слова WHERE) или выражение, которое следует использовать для отбора записей из базовой таблицы или запроса. Если в значении аргумента «Имя фильтра» указан фильтр, Microsoft Access применит это предложение WHERE к фильтру. Для того чтобы открыть отчет и отобрать те записи, которые содержат значения, совпадающие со значением элемента управления в форме, используют следующий синтаксис: **[***имяПоля***] = Forms![***имяФормы***]![***имяЭлементаВФорме***]** Аргумент *имяПоля* представляет имя поля в базовой таблице или базовом запросе открываемого отчета. Аргумент *имяЭлементаВФорме* представляет имя элемента управления в форме, который содержит нужное значение. **Примечание.** Максимальная длина значения аргумента «Условие отбора» равняется 256 символам. Для ввода более сложной инструкции SQL с предложением WHERE (длиной до 32 768 символов) следует вызвать метод **OpenReport** объекта **DoCmd** в программе Visual Basic.

Дополнительные сведения

Если аргумент «Режим» имеет значение «Печать», Microsoft Access печатает отчет на текущем принтере без открытия диалогового окна **Печать**. Кроме того, можно открыть отчет с помощью макрокоманды **ОткрытьОтчет**, задать соответствующие настройки, а затем вызвать макрокоманду **Печать (PrintOut)** для печати этого отчета. Например, может понадобиться изменить отчет или использовать макрокоманду **Печать** для изменения параметров печати отчета.

Фильтр или предложение WHERE становятся значением свойства **Фильтр (Filter)** отчета.

Вызов макрокоманды **ОткрытьОтчет (OpenReport)** эквивалентен нажатию кнопки **Конструктор** или **Просмотр** или выбору команды **Печать** в меню **Файл** в окне базы данных после выбора отчета на вкладке **Отчеты**.

Советы

- Если требуется напечатать однотипные отчеты для разных наборов данных, используйте фильтр или предложение WHERE для отбора записей, которые следует включить в отчет. Затем измените макрос: примените другой фильтр или измените значение аргумента «Условие отбора».
- Можно выбрать отчет в окне базы данных и переместить его с помощью мыши в строку макрокоманды в макросе. При этом в макрос автоматически добавляется макрокоманда **ОткрытьОтчет (OpenReport)**, открывающая отчет в режиме предварительного просмотра.

Макрокоманда «ОткрытьТаблицу» (OpenTable)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactOpenTableC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactOpenTableX":1}

Макрокоманда **ОткрытьТаблицу (OpenTable)** открывает таблицу в режиме таблицы, в режиме конструктора или в режиме предварительного просмотра. При этом допускается выбор режима ввода данных в таблицу.

Значения

Макрокоманда **ОткрытьТаблицу (OpenTable)** использует следующие аргументы.

Аргумент	Описание
Имя таблицы	Имя таблицы, которую следует открыть. В раскрывающемся списке в ячейке «Имя таблицы» в разделе **Аргументы макрокоманды** окна макроса содержатся имена всех таблиц в текущей базы данных. Данный аргумент является обязательным. Если макрос, содержащий макрокоманду **ОткрытьТаблицу (OpenTable)**, запущен из библиотечной базы данных, поиск таблицы с указанным именем проводится сначала в библиотечной базе данных, а затем в текущей базе данных.
Режим	Режим открытия таблицы. Допустимые значения: «Таблица» (используется по умолчанию), «Конструктор» и «Просмотр» выбираются в раскрывающемся списке.
Режим данных	Режим ввода данных в таблицу. Этот аргумент применим только к таблицам, открываемым в режиме таблицы. Допустимые значения: «Добавление» (пользователь может вводить новые записи, но не может изменять существующие записи), «Изменение» (пользователь может вводить новые и изменять существующие записи; используется по умолчанию) и «Только чтение» (пользователю разрешается только просматривать записи).

Дополнительные сведения

Вызов этой макрокоманды эквивалентен нажатию кнопки **Открыть** или Конструктор в окне базы данных после выбора таблицы на вкладке **Таблицы**.

Совет. Можно выбрать таблицу в окне базы данных и переместить ее с помощью мыши в строку макрокоманды в макросе. При этом в макрос автоматически добавляется макрокоманда **ОткрытьТаблицу (OpenTable)**, открывающая таблицу в режиме таблицы.

При переводе открытой таблицы в режим конструктора значение аргумента «Режим данных» теряется и не восстанавливается даже при возврате в режим таблицы.

Для вызова макрокоманды **ОткрытьТаблицу (OpenTable)** в программе Visual Basic следует использовать метод **OpenTable** объекта **DoCmd.**

Макрокоманда «Печать» (PrintOut)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactPrintC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactPrintX":1}

Макрокоманда **Печать (PrintOut)** печатает активный объект открытой базы данных. Допускается печать объектов в режиме таблицы, отчетов, форм и модулей.

Значения

Макрокоманда **Печать (PrintOut)** использует следующие аргументы.

Аргумент	Описание
Печатать	Определяет часть объекта, которую следует напечатать. Допустимые значения выбираются в раскрывающемся списке в ячейке аргумента «Печатать» в разделе **Аргументы макрокоманды** окна макроса: «Все» (печатается весь объект), «Фрагмент» (печатается выделенная часть объекта) или «Страницы» (печатается диапазон страниц, указанный с помощью аргументов «Со страницы» и «По страницу»). По умолчанию выбирается значение «Все».
Со страницы	Номер первой страницы, которую следует напечатать. Печать начинается с начала указанной страницы. Этот аргумент является обязательным, если аргумент «Печатать» имеет значение «Страницы».
По страницу	Номер последней страницы, которую следует напечатать. Печать заканчивается в конце указанной страницы. Этот аргумент является обязательным, если аргумент «Печатать» имеет значение «Страницы».
Разрешение	Определяет качество печати. Допустимые значения: «Высокое» (используется по умолчанию), «Среднее», «Низкое» и «Черновик». Чем выше качество, тем больше времени занимает печать.
Число копий	Число печатающихся копий (значение по умолчанию 1).
Разобрать копии	Указывает, следует ли печатать с раскладкой по копиям. По умолчанию используется значение «Да». Выбор значения «Нет» может ускорить печать.

Дополнительные сведения

Вызов этой макрокоманды эквивалентен выделению объекта с последующим выбором команды **Печать** в меню **Файл**, однако, данная макрокоманда не открывает диалоговое окно **Печать**.

Совет. Если постоянно приходится использовать одни и те же параметры печати, создайте макрос и включите в него макрокоманду **Печать (PrintOut)** с нужными значениями аргументов.

Аргументы этой макрокоманды эквивалентны параметрам в диалоговом окне **Печать**. Однако, в отличие от макрокоманды **НайтиЗапись (FindRecord)** и диалогового окна **Поиск**, значения аргументов данной макрокоманды и параметры в диалоговом окне **Печать** являются независимыми.

Для вызова макрокоманды **Печать (PrintOut)** в программе Visual Basic следует использовать метод **PrintOut** объекта **DoCmd.**

Макрокоманда «ЗапускЗапросаSQL» (RunSQL)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acactRunSQLC;damthBeginTrans;daproTransactions;dasqlJetSQLReservedWords"} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactRunSQLX":1}

Макрокоманда **ЗапускЗапросаSQL (RunSQL)** запускает запрос на изменение Microsoft Access с помощью соответствующей инструкции SQL. Кроме того, эта макрокоманда позволяет запустить управляющий запрос.

Значения

Макрокоманда **ЗапускЗапросаSQL (RunSQL)** использует следующие аргументы.

Аргумент	Описание
Инструкция SQL	Текст инструкции SQL, определяющей запрос на изменение или управляющий запрос. Максимальная длина инструкции SQL, задаваемой данным аргументом, составляет 256 символов. Данный аргумент является обязательным.
Использовать транзакцию	Для включения запроса на изменение в транзакцию этот аргумент должен иметь значение «Да». Если использование транзакции нежелательно следует задать значение «Нет». Значение по умолчанию для этого аргумента - «Да». В предыдущих версиях Microsoft Access при запуске данной макрокоманды запрос на изменение всегда включался в транзакцию. Задание значения «Нет» может ускорить выполнение запроса.

Дополнительные сведения

Запросы на изменение используются для добавления, удаления и обновления записей, а также для сохранения результирующего набора записей запроса в виде новой таблицы. Управляющие запросы позволяют создавать, изменять и удалять таблицы, а также создавать и удалять индексы. Макрокоманда **ЗапускЗапросаSQL (RunSQL)** делает возможным выполнение этих действий в макросе без необходимости предварительного создания сохраненных запросов.

Если требуется выполнить инструкцию SQL, длина которой превышает 256 символов, следует вызвать метод **RunSQL** объекта **DoCmd** в программе Visual Basic. В программах Visual Basic допускается использование инструкций SQL длиной до 32 768 символов.

Запросы Microsoft Access фактически являются инструкциями SQL, которые создаются при определении запроса в бланке в окне запроса. В следующей таблице перечислены запросы на изменение и управляющие запросы Microsoft Access, а также соответствующие им инструкции SQL.

Тип запроса	Инструкция SQL
Запрос на изменение	
На добавление	**INSERT INTO**
На удаление	**DELETE**
На создание таблицы	**SELECT...INTO**
На обновление	**UPDATE**
Управляющий (запрос SQL)	
На создание таблицы	**CREATE TABLE**
На изменение таблицы	**ALTER TABLE**
На удаление таблицы	**DROP TABLE**

На создание индекса **CREATE INDEX**

На удаление индекса **DROP INDEX**

Включение предложения **IN** в эти инструкции позволяет изменять данные в другой базе данных.

Примечание. Для запуска в макросе запроса на выборку или перекрестного запроса следует с помощью аргумента «Режим» макрокоманды **ОткрытьЗапрос (OpenQuery)** открыть существующий запрос на выборку или перекрестный запрос в режиме таблицы. Эта же макрокоманда позволяет выполнить сохраненные запросы на изменение и запросы SQL.

Совет. Для просмотра эквивалентной инструкции SQL, соответствующей запросу Microsoft Access, выберите в меню **Вид** окна запроса команду **Режим SQL**. Выводящиеся инструкции SQL следует использовать как образцы при создании запросов, предназначенных для выполнения с помощью макрокоманды **ЗапускЗапросаSQL (RunSQL)**. Копирование инструкции SQL в аргумент «Инструкция SQL» макрокоманды **ЗапускЗапросаSQL (RunSQL)** дает тот же результат, что и запуск запроса Microsoft Access из окна запроса.

Макрокоманда «ПреобразоватьБазуДанных» (TransferDatabase)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactTransferDatabaseC;daproConnect"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactTransferDatabaseX":1}

Макрокоманда **ПреобразоватьБазуДанных (TransferDatabase)** используется при импорте или экспорте данных между текущей базой данных Microsoft Access и другой базой данных. Кроме того, допускается связывание таблицы из другой базы данных в текущую базу данных Microsoft Access. Связанные (присоединенные) таблицы обеспечивают доступ к содержимому таблиц других баз данных, причем сами таблицы при этом остаются в других базах данных.

Значения

Макрокоманда **ПреобразоватьБазуДанных (TransferDatabase)** использует следующие аргументы.

Аргумент	Описание
Тип преобразования	Тип выполняемого преобразования «Импорт», «Экспорт» или «Связь» выбирается в ячейке аргумента «Тип преобразования» в бланке **Аргументы макрокоманды** в нижней половине окна макроса. По умолчанию задается значение «Импорт».
Тип базы данных	Тип базы данных, из которой выполняется импорт или связывание или в которую выполняется экспорт. В ячейке «Тип базы данных» допускается выбор Microsoft Access или базы данных другого типа. По умолчанию задается формат Microsoft Access. Набор типов, предлагаемых в аргументе «Тип базы данных», определяется параметрами установки Microsoft Access. Не все драйверы баз данных устанавливаются по умолчанию. Если тип базы данных, требуемый для импорта, экспорта или связывания, отсутствует в списке, следует запустить программу установки Microsoft Access, нажать кнопку **Добавить/Удалить**, выбрать в диалоговом окне параметр **Доступ к данным** и нажать кнопку **Выбор**, после чего в списке **Драйверы баз данных** выбрать нужный тип базы данных и снова нажать кнопку **Выбор**. Некоторые драйверы баз данных доступны только в пакете Office 97 ValuPack (который содержит также все драйверы баз данных, доступные в программе установки Microsoft Access). Более подробно об установке драйверов баз данных из пакета ValuPack см. в разделе Пакет Office 97 ValuPack.
Имя базы данных	Имя базы данных, из которой выполняется импорт или связывание или в которую выполняется экспорт. Необходимо указать полный путь. Данный аргумент является обязательным. Для баз данных, в которых каждая таблица сохраняется в отдельном файле, таких как FoxPro, Paradox или dBASE, следует указать путь к каталогу, содержащему файлы таблиц. Имя файла следует указать в аргументе «Источник» (для импорта или связывания) или «Адресат» (для экспорта). Для баз данных ODBC следует ввести полную строку подключения, требуемую протоколом ODBC. Для того чтобы увидеть пример строки подключения, свяжите внешнюю таблицу с базой данных Microsoft Access с помощью команды **Связь с таблицами** из подменю **Внешние данные** в меню

	Файл. Откройте таблицу в режиме конструктора и откройте окно свойств таблицы. Текст в ячейке свойства **Описание (Description)** представляет строку подключения для данной таблицы. Для получения дополнительных сведений о строках подключения ODBC следует обратиться к файлу справки или к другой документации драйвера ODBC, соответствующего типу базы данных ODBC.
Тип объекта	Тип импортируемого или экспортируемого объекта. Если в аргументе «Тип базы данных» выбран Microsoft Access, в ячейке данного аргумента предлагается выбор типа объекта «Таблица», «Запрос», «Форма», «Отчет», «Макрос» или «Модуль». Если выбран другой типы базы данных или если в аргументе «Тип преобразования» выбрана «Связь», то данный аргумент игнорируется. По умолчанию задается значение «Таблица». При экспорте запроса на выборку в базу данных Microsoft Access в данном аргументе следует выбрать «Таблица» для экспорта результирующего набора записей запроса или «Запрос» для экспорта самого запроса. Если запрос на выборку экспортируется в базу данных другого типа, данный аргумент игнорируется и экспортируется результирующий набор записей.
Источник	Имя таблицы, запроса на выборку или объекта Microsoft Access, которые импортируются, экспортируются или связываются. Для некоторых баз данных других типов, таких как FoxPro, Paradox или dBASE, в этом аргументе задается имя файла. Имя файла следует указывать с расширением (например, .dbf). Данный аргумент является обязательным.
Адресат	Имя импортируемой, экспортируемой или связываемой таблицы, запроса на выборку или объекта Microsoft Access в базе данных получателе. Для некоторых баз данных других типов, таких как FoxPro, Paradox или dBASE, в этом аргументе задается имя файла. Имя файла следует указывать с расширением (например, .dbf). Данный аргумент является обязательным. Если в аргументе «Тип преобразования» указан «Импорт», а в аргументе «Тип объекта» «Таблица», то Microsoft Access создает новую таблицу, в которую помещаются импортируемые данные. Если имя импортируемой таблицы или другого объекта совпадает с существующим именем объекта этого типа, то Microsoft Access добавляет к имени номер. Например, если импортируется таблица «Сотрудники», а в базе данных уже имеется таблица «Сотрудники», то импортируемая таблица получит имя «Сотрудники1». При экспорте в базу данных Microsoft Access или базу данных другого типа любые существующие таблицы или другие объекты автоматически заменяются на новый объект с тем же именем.
Только структура	Определяет импорт или экспорт только структуры таблицы, содержащейся в базе данных, без импорта или экспорта содержимого таблицы. Допустимые значения «Да» или

«Нет». По умолчанию задается значение «Нет».

Дополнительные сведения

Операции импорта или экспорта допускаются между базой данных Microsoft Access и базой данных того же или другого типа. Кроме того, допускается экспорт запросов на выборку Microsoft Access в базы данных других типов. Microsoft Access экспортирует результирующий набор записей запроса в виде таблицы. Между двумя базами данных Microsoft Access поддерживается импорт и экспорт любых объектов базы данных.

В Microsoft Access 97 при импорте связанной таблицы из другой базы данных Microsoft Access эта таблица остается связанной после импорта. Это означает, что импортируется связь, а не сама таблица.

Если для доступа к другой базе данных требуется пароль, то при запуске макроса открывается диалоговое окно, в которое следует ввести пароль.

Примечание. Между базами данных Microsoft Access и FoxPro 3.0 допускается лишь импорт и экспорт данных. Связывание с такими базами данных FoxPro невозможно.

Вызов макрокоманды **ПреобразоватьБазуДанных (TransferDatabase)** аналогичен выбору в меню **Файл** в окне базы данных команды **Сохранить как/Экспорт** или команды **Внешние данные** и одной из команд, **Импорт** или **Связь с таблицами**, в подменю. Эти команды позволяют выбрать тип источника данных, например, базу данных Microsoft Access или другого типа, электронную таблицу или текстовый файл. Если выбран один из типов баз данных, открывается серия диалоговых окон, в которых следует выбрать тип импортируемого или экспортируемого объекта (для баз данных Microsoft Access), имя объекта, а также другие параметры, зависящие от выбранного типа базы данных. Аргументы макрокоманды **ПреобразоватьБазуДанных (TransferDatabase)** соответствуют параметрам этих диалоговых окон.

При необходимости ввести сведения об индексах связанной таблицы FoxPro или dBASE следует предварительно связать таблицу FoxPro или dBASE. Для этого выберите в меню **Файл** команду **Внешние данные** и команду **Связь с таблицами** в подменю, а затем укажите индексы в окнах диалога, открывающихся после этой команды. Microsoft Access сохраняет сведения об индексах в специальном информационном (.inf) файле, расположенном в папке C:\Program Files\Microsoft Office\Office. После этого связь с присоединенной таблицей можно удалить. При следующем вызове макрокоманды **ПреобразоватьБазуДанных (TransferDatabase)** для связывания этой таблицы FoxPro или dBASE будут использованы сведения об индексах, сохраняемые в информационном файле.

Примечание. Операции запроса и установки фильтра в связанной таблице выполняются с учетом регистра.

Для выполнения макрокоманды **ПреобразоватьБазуДанных (TransferDatabase)** в программе Visual Basic следует вызвать метод **TransferDatabase** объекта **DoCmd.**

Макрокоманда «ПреобразоватьЭлектроннуюТаблицу» (TransferSpreadsheet)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactTransferSpreadsheetC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactTransferSpreadsheetX":1}

Макрокоманда **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** используется при импорте или экспорте данных между текущей базой данных Microsoft Access и файлом электронной таблицы. Кроме того, допускается связывание данных из электронной таблицы Microsoft Excel в текущую базу данных Microsoft Access. После связывания электронной таблицы становится возможным просмотр и изменение электронной таблицы из Microsoft Access. При этом пользователь сохраняет все возможности работы с данными в Microsoft Excel. Возможно также связывание в файл электронной таблицы Lotus 1-2-3, но в этом случае в Microsoft Access данные из него будут доступны лишь для чтения.

Значения

Макрокоманда **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** использует следующие аргументы.

Аргумент	Описание
Тип преобразования	Тип выполняемого преобразования «Импорт», «Экспорт» или «Связь» выбирается в ячейке аргумента «Тип преобразования» в бланке **Аргументы макрокоманды** в нижней половине окна макроса. По умолчанию задается значение «Импорт».
Тип электронной таблицы	Тип электронной таблицы, из которой выполняется импорт или в которую выполняется экспорт или связывание. Допускается выбор формата электронной таблицы. По умолчанию задается формат Microsoft Excel 3. **Примечания** • Для импорта, экспорта или связывания данных электронных таблиц Lotus 1-2-3 необходимо установить драйвер Lotus 1-2-3 из пакета Office 97 ValuPack. Более подробно об установке драйверов см. в разделе Пакет Office 97 ValuPack. • Для электронных таблиц Microsoft Excel поддерживаются операции импорта, экспорта или связывания, однако, для электронных таблиц Lotus .WK4 допустимыми являются только операции импорта или связывания. Кроме того, Microsoft Access больше не поддерживает импорт, экспорт или связывание для электронных таблиц Lotus .WKS и Microsoft Excel версии 2.0. Для импорта из или связывания в файлы этих форматов в Microsoft Access, следует преобразовать их в формат более поздней версии Microsoft Excel или Lotus 1-2-3.
Имя таблицы	Имя таблицы Microsoft Access, в которую импортируются или связываются данные, или из которой экспортируются данные. Кроме того, в данном аргументе можно указать запрос на выборку Microsoft Access, из которого экспортируются данные. Данный аргумент является обязательным. Если в аргументе «Тип преобразования» выбран «Импорт», то Microsoft Access добавляет данные из электронной таблицы в указанную в данном аргументе таблицу, если

	таблица с таким именем уже существует. В противном случае создается новая таблица, в которую помещаются данные, импортируемые из электронной таблицы. В Microsoft Access 97 в макрокоманде **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** не допускается использование инструкции SQL для указания экспортируемых данных. Вместо этого следует создать запрос и затем указать его имя в аргументе «Имя таблицы».
Имя файла	Полное имя файла электронной таблицы, из которой выполняется импорт или связывание или в которую выполняется экспорт. Данный аргумент является обязательным. При экспорте данных из Microsoft Access создается новая электронная таблица. Если указанное имя файла совпадает с именем существующей электронной таблицы, Microsoft Access заменяет существующую электронную таблицу на новую, за исключением случаев экспорта в рабочую книгу Microsoft Excel версии 5.0, 7.0 или Excel 97. В последних случаях Microsoft Access копирует экспортируемые данные на первый свободный лист рабочей книги. При импорте из или связывании в электронную таблицу Microsoft Excel версии 5.0, 7.0 или Excel 97 в аргументе «Диапазон» возможно задание конкретного листа рабочей книги.
С именами полей	Определяет, содержатся ли в первой строке электронной таблицы имена полей. Если выбрано значение «Да», то имена из этой строки используются как имена полей таблицы Microsoft Access, в которую импортируются или связываются данные. Если выбрано значение «Нет», то первая строка электронной таблицы рассматривается как обычная строка данных. По умолчанию задается значение «Нет». При экспорте таблицы Microsoft Access или запроса на выборку в электронную таблицу имена полей помещаются в первую строку электронной таблицы независимо от значения этого аргумента.
Диапазон	Диапазон импортируемых или связываемых ячеек. При импорте или связывании всей электронной таблицы оставьте данный аргумент пустым. Допускается указание имени диапазона или адреса в формате A1:E25 (следует заметить, что формат A1..E25 в Microsoft Access 97 больше не поддерживается). При импорте или связывании из рабочей книги Microsoft Excel версии 5.0 или Excel 97 перед именем диапазона, при необходимости, задается имя листа и восклицательный знак; например, Бюджет!A1:C7. **Примечание** При экспорте в электронную таблицу следует оставить этот аргумент пустым. Если в нем будет указан диапазон, операция экспорта завершится неудачей.

Дополнительные сведения

Допускается экспорт запросов на выборку Microsoft Access в электронные таблицы. Microsoft Access экспортирует результирующий набор записей запроса в виде таблицы.

Данные электронной таблицы, добавляемые в существующую таблицу Microsoft Access, должны быть совместимы со структурой таблицы. Каждое поле электронной таблицы должно иметь тот же тип данных, что и соответствующее поле таблицы. Необходимо также совпадение порядка полей (за исключением случая, когда для аргумента «С именами полей» задано значение «Да»; в этом случае имена полей электронной таблицы должны совпадать с именами полей таблицы).

Вызов данной макрокоманды аналогичен выбору в меню **Файл** в окне базы данных команды **Сохранить как/Экспорт** или команды **Внешние данные** и одной из команд, **Импорт** или **Связь с таблицами**, в подменю. Эти команды позволяют выбрать тип источника данных, например, базу данных Microsoft Access или другого типа, электронную таблицу или текстовый файл. Если выбирается электронная таблица, открывается серия диалоговых окон или запускается мастер Microsoft Access, открывающий окна диалога, в которых следует выбрать имя электронной таблицы и другие параметры. Аргументы макрокоманды **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** соответствуют параметрам этих диалоговых окон или мастера.

Примечание. Операции запроса и установки фильтра в связанной электронной таблице выполняются с учетом регистра.

При связывании в электронную таблицу Excel, открытую в режиме изменения, для завершения связи Access будет ждать окончания работы в режиме изменения.

Для выполнения макрокоманды **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** в программе Visual Basic следует вызвать метод **TransferSpreadsheet** объекта **DoCmd.**

Макрокоманда «ПреобразоватьТекст» (TransferText)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactTransferTextC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactTransferTextX":1}

Макрокоманда **ПреобразоватьТекст (TransferText)** используется при импорте или экспорте данных между текущей базой данных Microsoft Access и текстовым файлом. Кроме того, допускается связывание данных из текстового файла в текущую базу данных Microsoft Access. После связывания текстового файла становится возможным просмотр и изменение текста из Microsoft Access. При этом пользователь сохраняет все возможности работы с текстовыми данными в текстовом редакторе. Возможен также импорт из, экспорт в или связывание в таблицу или список, находящиеся в файле HTML (*.html).

Примечание При связывании в текстовый файл или в файл HTML, данные в Microsoft Access доступны только для чтения.

Значения

Макрокоманда **ПреобразоватьТекст (TransferText)** использует следующие аргументы.

Аргумент	Описание
Тип преобразования	Тип выполняемого преобразования. Поддерживаются операции импорта, экспорта или связывания для текстовых файлов с разделителями, текстовых файлов с фиксированной длиной записей или файлов HTML. Поддерживается также экспорт в файл данных составного документа Microsoft Word для Windows, который потом может быть использован для создания почтовых наклеек и других составных документов средствами Word для Windows. В аргументе «Тип преобразования» в разделе **Аргументы макрокоманды** окна макроса предлагается выбор: «Импорт (разделители)» (используется по умолчанию), «Импорт (фиксированный)», «Импорт (HTML)», «Экспорт (разделители)», «Экспорт (фиксированный)», «Экспорт (HTML)», «Экспорт слияний Word для Windows», «Связь (разделители)», «Связь (фиксированный)», «Связь (HTML)».
Название спецификации	Название спецификации, определяющей параметры импорта, экспорта или связывания текстового файла (обязательный аргумент для текстовых файлов с фиксированной длиной записей). При создании спецификации для конкретного типа текстовых файлов, например, для текстового файла, в котором разделителем полей служит символ табуляции и используется формат дат МДГ, следует выбрать в меню **Файл** команду **Сохранить как/Экспорт** или команды **Внешние данные** и подкоманду **Импорт** или **Связь с таблицами**. При выборе одной из этих команд с последующим указанием типа текстового файла запускается мастер по импорту, мастер по экспорту или мастер по связыванию текстовых

	файлов. Пользователь имеет возможность нажать кнопку **Дополнительно** в диалоговом окне мастера и определить, а затем сохранить спецификацию. После этого имя сохраненной спецификации всегда может быть использовано в данном аргументе макрокоманды при импорте или экспорте текстовых файлов того же типа. Допускается выполнение операций импорта, экспорта или связывания текстового файла с разделителями без указания названия спецификации в данном аргументе. В этом случае используются стандартные настройки из окна диалога мастера. Кроме того, Microsoft Access использует стандартный формат для файлов данных составных документов, поэтому для файлов этого типа нет необходимости указывать значение этого аргумента при экспорте. Также допускается использование спецификации импорта/экспорта с файлами HTML. Однако в этом случае будет использована лишь часть спецификации, определяющая форматирование типа данных.
Имя таблицы	Имя таблицы Microsoft Access, в которую следует импортировать или связать данные из текстового файла или из которой следует экспортировать данные в текстовый файл. Допускается также указание имени запроса на выборку Microsoft Access, из которого экспортируются данные. Данный аргумент является обязательным. Если для аргумента «Тип преобразования» выбрано значение «Импорт (разделители)», «Импорт (фиксированный)» или «Импорт (HTML)» и указано имя существующей таблицы, то данные, импортируемые из текстового файла, будут добавлены в конец этой таблицы. В противном случае данные из текстового файла будут помещены в новую таблицу. В Microsoft Access 97 в макрокоманде **ПреобразоватьТекст (TransferText)** не допускается использование инструкции SQL для указания экспортируемых данных. Вместо этого следует создать запрос и затем указать его имя в аргументе «Имя таблицы».
Имя файла	Полное имя текстового файла, из которого следует импортировать или связать данные или в который следует экспортировать данные. Этот аргумент является обязательным. При экспорте данных из Microsoft Access создается новый текстовый файл. Если в качестве аргумента «Имя файла» указано имя существующего текстового файла, существующий текстовый файл заменяется. При необходимости импортировать или связать конкретную таблицу или список файла HTML

	следует использовать аргумент «Имя таблицы HTML».
С именами полей	Указывает, содержит ли первая строка текстового файла имена полей. Если этот аргумент имеет значение «Да», то при импорте данных из текстового файла содержимое первой строки будет использовано в качестве имен полей таблицы Microsoft Access. Если этот аргумент имеет значение «Нет», то первая строка будет обрабатываться как обычная строка данных. По умолчанию задается значение «Нет». Значение этого аргумента игнорируется для файла данных Word для Windows, поскольку первая строка такого файла обязана содержать имена полей. Если данный аргумент имеет значение «Да», то при экспорте таблицы или запроса на выборку Microsoft Access в текстовый файл имена полей автоматически вставляются в первую строку текстового файла. Если данный аргумент имеет значение «Да», то для разделения имен полей в первой строке импортируемого или связываемого текстового файла с фиксированной шириной полей необходимо использовать разделитель, указанный в спецификации импорта/экспорта. При экспорте в текстовый файл с фиксированной шириной полей Microsoft Access вставит имена полей в первую строку текстового файла, используя этот разделитель.
Имя таблицы HTML	Имя импортируемой или связываемой таблицы или списка файла HTML. Этот аргумент принимается во внимание, только если «Тип преобразования» равен «Импорт (HTML)» или «Связь (HTML)». Если этот аргумент не заполнен, импортируется или связывается первая таблица или список файла HTML. Имя таблицы или списка из файла HTML задается в команде <CAPTION>, если такая существует. Иначе имя определяется в команде <TITLE>. Если имена нескольких таблиц или списков совпадают, Microsoft Access добавляет в конец каждого имени число; например, «Сотрудники1» и «Сотрудники2».

Дополнительные сведения

Допускается экспорт запросов на выборку Microsoft Access в текстовые файлы. Microsoft Access экспортирует результирующий набор записей запроса в виде таблицы.

Текстовые данные, добавляемые в существующую таблицу Microsoft Access, должны быть совместимы со структурой таблицы. Каждое поле текстового файла должно иметь тот же тип данных, что и соответствующее поле таблицы. Необходимо также совпадение порядка полей (за исключением случая, когда для аргумента «С именами полей» задано значение «Да»; в этом случае имена полей текстового файла должны совпадать с именами полей таблицы).

Вызов данной макрокоманды аналогичен выбору в меню **Файл** в окне базы данных команды **Сохранить как/Экспорт** или команды **Внешние данные** и одной из команд, **Импорт** или **Связь с таблицами**, в подменю. Эти команды позволяют выбрать тип источника данных, например, базу данных Microsoft Access или другого типа, электронную таблицу или текстовый файл. Если выбирается текстовый файл с разделителями, с фиксированной шириной полей или файл HTML, мастер выводит приглашение указать имя текстового файла и выбрать необходимые параметры. Аргументы макрокоманды **ПреобразоватьТекст (TransferText)** соответствуют параметрам этих диалоговых окон или мастера.

Совет. Спецификация импорта/экспорта содержит сведения, необходимые Microsoft Access для импорта и экспорта текстового файла. После сохранения этих сведений в спецификации их не надо будет вводить при каждой операции импорта, связывания или экспорте данных для текстового файла с такой же структурой. Предположим, что каждый понедельник с большого компьютера поступают сведения о продажах в виде текстового файла. В таком случае достаточно один раз создать и сохранить спецификацию, а затем использовать ее еженедельно.

Примечание. Операции запроса и установки фильтра в связанном текстовом файле выполняются с учетом регистра.

Для выполнения макрокоманды **ПреобразоватьТекст (TransferText)** в программе Visual Basic следует вызвать метод **TransferText** объекта **DoCmd.**

Макрокоманда «Выход» (Quit)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactQuitC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactQuitX":1}

Макрокоманду **Выход (Quit)** используют для выхода из Microsoft Access. При этом допускается выбор параметров сохранения объектов базы данных.

Значения

Макрокоманда **Выход (Quit)** использует следующий аргумент.

Аргумент	Описание
Параметры	Указывает, что должно происходить с не сохраненными объектами при завершении работы с Microsoft Access. Допустимые значения: «Подтверждение» (для каждого не сохраненного объекта выводится окно диалога, в котором следует подтвердить сохранение), «Сохранить все» (все не сохраненные объекты сохранятся автоматически; используется по умолчанию) и «Выход» (выход из Microsoft Access без сохранения объектов) в диалоговом окне **Параметры** в разделе **Аргументы макрокоманды** окна макрокоманды.

Дополнительные сведения

Макрокоманды, следующие после макрокоманды **Выход (Quit)**, не выполнятся.

Эту макрокоманду используют для быстрого завершения работы с Microsoft Access, связывая ее с командой специального меню или кнопкой в форме. Предположим, что имеется главная форма, предназначенная для создания специальной рабочей области, в которой выполняется работа с объектами. Эта форма может содержать кнопку «Выход», запускающую макрос, содержащий макрокоманду **Выход (Quit)** с аргументом «Параметры», имеющим значение «Сохранить все».

Вызов этой макрокоманды эквивалентен выбору команды **Выход** в меню **Файл**. При наличии не сохраненных объектов эта команда меню открывает те же окна диалога, что и макрокоманда **Выход (Quit)** с аргументом «Параметры», имеющим значение «Подтверждение».

Макрокоманда **Сохранить (Save)** позволяет сохранить в макросе указанный объект без выхода из Microsoft Access или закрытия объекта.

Для запуска макрокоманды **Выход (Quit)** в программе Visual Basic следует использовать метод **Quit** объекта **DoCmd.**

Макрокоманда «Переименовать» (Rename)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactRenameC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactRenameX":1}

Макрокоманда **Переименовать (Rename)** переименовывает указанный объекта базы данных.

Значения

Макрокоманда **Переименовать (Rename)** использует следующие аргументы.

Аргумент	Описание
Новое имя	Новое имя для объекта базы данных. Введете имя объекта в поле **Новое имя** раздела **Аргументы макрокоманды** в окне макрокоманды. (обязательный аргумент).
Тип объекта	Тип объекта, подлежащего переименованию. Выберите «Таблица», «Запрос», «Форма», «Отчет», «Макрос» или «Модуль». При переименовании выбранного объекта окне базы данных этот аргумент вводить не следует.
Старое имя	Имя объекта, подлежащего переименованию. В поле **Старое имя** выводятся все объекты базы данных, имеющие тип, заданный аргументом «Тип объекта». Если аргумент «Тип объекта» не задан, то следует оставить незаполненным и данный аргумент. **Примечание.** Если макрос, содержащий макрокоманду **Переименовать (Rename)**, запускается из библиотечной базы данных, поиск объекта с этим именем проводится сначала в библиотечной базе данных, а затем в текущей базе данных.

Дополнительные сведения

Новое имя объекта базы данных должно удовлетворять стандартным соглашениям об именах объектов Microsoft Access.

Нельзя переименовать открытый объект.

Если значения аргументов «Тип объекта» и «Старое имя» оставлены пустыми, переименовывается объект, выделенный в окне базы данных. Для выделения объекта в окне базы данных используют макрокоманду **Выделить Объект (SelectObject)** с аргументом «В окне базы данных», имеющим значение «Да».

Кроме того, можно переименовать объект в окне базы данных. Для этого следует выбрать объект, повторно нажать кнопку мыши и ввести новое имя. Макрокоманда **Переименовать (Rename)** позволяет обойтись без выделения объекта в окне базы данных и ввести новое имя без остановки макроса.

Данная макрокоманда отличается от макрокоманды **КопироватьОбъект (CopyObject),** которая создает копию объекта под другим именем.

Для запуска макрокоманды **Переименовать (Rename)** в программе Visual Basic следует использовать метод **Rename** объекта **DoCmd.**

Макрокоманда «ОбновитьОбъект» (RepaintObject)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactRepaintObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactRepaintObjectX":1}

Макрокоманда **ОбновитьОбъект (RepaintObject)** завершает все отложенные операции обновления указанного объекта базы данных или активного объекта базы данных (по умолчанию). При необходимости, выполняется пересчет значений элементов управления в этом объекте.

Значения

Макрокоманда **ОбновитьОбъект (RepaintObject)** использует следующие аргументы.

Аргумент	Описание
Тип объекта	Тип объекта, предназначенного для обновления. Выберите «Таблица», «Запрос», «Форма», «Отчет», «Макрос» или «Модуль» в поле **Тип объекта** в разделе **Аргументы макрокоманды** окна макрокоманды. Чтобы задать активный объект оставьте это поле незаполненным.
Имя объекта	Имя объекта, который следует обновить. В раскрывающемся списке в ячейке содержатся имена всех объектов текущей базы данных, которые относятся к типу, указанному в аргументе «Тип объекта». Если значение аргумента «Тип объекта» оставлено пустым, следует оставить пустым и значение этого аргумента.

Дополнительные сведения

Как правило, Microsoft Access откладывает обновление изображения на экране до тех пор, пока не завершит выполнение других текущих задач. Данная макрокоманда позволяет выполнить обновление указанного объекта немедленно. Эту макрокоманду используют в следующих ситуациях:

- После изменения значений нескольких элементов управления с помощью макрокоманд **ЗадатьЗначение (SetValue).** Эти изменения не обязательно будут отражены на экране немедленно, особенно в том случае, если другие элементы управления (например, вычисляемые элементы управления) зависят от значений измененных элементов управления.
- Для того чтобы гарантировать, что во всех элементах управления в форме отображаются их значения. Например, в элементах управления, содержащих объекты OLE, данные часто не выводятся сразу после открытия формы.

Примечания

- Эта макрокоманда не приводит к выполнению повторного просмотра базы данных, поэтому она не показывает новые, измененные и удаленные записи из базовой таблицы или базового запроса объекта. Для повторного просмотра объекта с учетом изменений источника данных объекта или одного из его элементов управления следует использовать макрокоманду **Обновление (Requery).** Для отображения последней версии записей и снятия любых фильтров следует использовать макрокоманду **ПоказатьВсеЗаписи (ShowAllRecords).**
- Вызов макрокоманды **ОбновитьОбъект (RepaintObject)** не эквивалентен выбору команды **Обновить** в меню **Записи**. Команда меню приводит к отображению всех изменений, внесенных текущим пользователем или другими пользователями в записи, изображаемые в формах и в объектах в режиме таблицы.

Для запуска макрокоманды **ОбновитьОбъект (RepaintObject)** в программе Visual Basic следует использовать метод **RepaintObject** объекта **DoCmd.**

Макрокоманда «Обновление» (Requery)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactRequeryC;damthRequery"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactRequeryX":1}

Макрокоманда **Обновление (Requery)** обновляет данные в указанном элементе управления в активном объекте путем повторного просмотра источника данных этого элемента управления. Если элемент управления не указан, эта макрокоманда задает повторный просмотр источника самого объекта. Данная команда обеспечивает отображение текущих данных в объекте или в элементе управления.

Значения

Макрокоманда **Обновление (Requery)** использует следующий аргумент.

Аргумент	Описание
Имя элемента	Имя элемента, который следует изменить. Введите имя в раскрывающийся список **Имя элемента** в раздел **Аргументы макрокоманды** окна макрокоманды. Здесь требуется только имя элемента управления, а не полный идентификатор. (такой как **Forms!***имяформы***!***имяэлемента*). При обновлении активного объекта следует оставить это поле пустым. Если активный объект является таблицей или результирующим набором записей запроса, то этот аргумент также не должен быть заполнен.

Дополнительные сведения

Макрокоманда **Обновление (Requery)** выполняет одно из следующих действий:

- Повторно выполняет запрос, который является источником данных для элемента управления или объекта.
- Отображает все новые или измененные записи и удаляет удаленные записи из таблицы, которая является источником элемента управления или объекта.

Таблица или запрос может служить источником данных для следующих элементов управления:

- Список или поле со списком.
- Подчиненная форма/отчет.
- Объект OLE, например, диаграмма.
- Элементы управления, которые содержат статистические функции по подмножеству, например, **DSum.**

Если источником данных указанного элемента управления не является ни таблица, ни запрос, данная макрокоманда выполняет пересчет этого элемента управления.

Вызов макрокоманды **Обновление (Requery)** с пустым значением аргумента «Имя элемента» эквивалентен нажатию клавиш SHIFT+F9, когда данный объект обладает фокусом. Если фокус имеет элемент управления-подчиненная форма, данная макрокоманда вызывает повторный просмотр только источника данных подчиненной формы (эквивалентно нажатию клавиш SHIFT+F9).

Примечание. Макрокоманда **Обновление (Requery)** вызывает повторный просмотр источника данных элемента управления или объекта. В отличие от нее макрокоманда **ОбновитьОбъект (RepaintObject)** обновляет изображение элементов управления в указанном объекте, однако, не выполняет повторный просмотр базы данных и не показывает новые записи. Макрокоманда **ПоказатьВсеЗаписи (ShowAllRecords)** не только вызывает повторный просмотр активного объекта; она также удаляет все фильтры, чего не делает макрокоманда **Обновление (Requery)**.

Для повторного просмотра элемента управления, находящегося вне активного объекта, необходимо использовать метод **Requery** в программе Visual Basic, а не макрокоманду **Обновление (Requery)** или соответствующий ей метод **Requery** объекта **DoCmd.** Метод **Requery** в программе Visual Basic выполняется быстрее, чем макрокоманда **Обновление (Requery)** или метод **Requery** объекта **DoCmd.** Кроме того, при использовании макрокоманды **Обновление (Requery)** или метода **Requery** объекта **DoCmd** Microsoft Access закрывает запрос и повторно загружает его из базы данных, а при использовании метода **Requery** Microsoft Access повторно выполняет запрос без закрытия и перезагрузки. Отметим, что метод **Requery** объектов доступа к данным выполняется так же, как метод **Requery** Microsoft Access.

Макрокоманда «Восстановить» (Restore)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactRestoreC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactRestoreX":1}

Макрокоманда **Восстановить (Restore)** восстанавливает прежние размеры развернутого или свернутого окна.

Значения

Макрокоманда **Восстановить (Restore)** не требует аргументов.

Дополнительные сведения

Эта макрокоманда применяется к выделенному объекту. Свернутый объект необходимо сначала выделить с помощью макрокоманды **ВыделитьОбъект (SelectObject),** а затем восстановить с помощью макрокоманды **Восстановить (Restore)**.

Для изменения положения и размеров восстановленного окна используют макрокоманду **СдвигРазмер (MoveSize).**

Вызов макрокоманды **Восстановить (Restore)** эквивалентен нажатию кнопки в правом верхнем углу окна или выбору команды **Восстановить** в оконном меню.

Для запуска макрокоманды **Восстановить (Restore)** в программе Visual Basic следует использовать метод **Restore** объекта **DoCmd.**

Макрокоманда «ЗапускПриложения» (RunApp)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactRunAppC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactRunAppX":1}

Макрокоманда **ЗапускПриложения (RunApp)** запускает из Microsoft Access приложение Windows или MS-DOS (например, Microsoft Excel, Microsoft Word для Windows или Microsoft PowerPoint). Например, эта макрокоманда позволяет выполнить вставку электронной таблицы Microsoft Excel в базу данных Microsoft Access.

Значения

Макрокоманда **ЗапускПриложения (RunApp)** использует следующий аргумент.

Аргумент	Описание
Командная строка	Командная строка применяется для запуска приложения (включает в себя путь к файлу и другие необходимые параметры, такие как аргументы, запускающие приложение в особых режимах). Введите командную строку в поле **Командная строка** раздела **Аргументы макрокоманды** окна макрокоманды. Данный аргумент является обязательным.

Дополнительные сведения

Приложение, указанное с помощью этой макрокоманды, загружается и запускается в основном режиме. Выполнение макроса, содержащего эту макрокоманду, продолжается.

Для обмена данными между другими приложениями и Microsoft Access используют механизм динамического обмена данными (DDE) или буфер. Для передачи нажатий клавиш в другие приложения используют макрокоманду **КомандыКлавиатуры (SendKeys)** (хотя механизм DDE является более эффективным средством передачи данных). Кроме того, совместное использование данных с другими приложениями обеспечивает механизм программирования объектов.

Приложения MS-DOS выполнятся в окне MS-DOS в среде Windows.

В Windows 3.1 вызов этой макрокоманды эквивалентен двойному щелчку на значке, соответствующем данному приложению, или выбору команды **Выполнить** в окне диспетчера программ Windows. В Windows 95 имеется несколько способов запуска приложения, в том числе запуск приложения из окна **Проводник**, с помощью команды **Выполнить** в меню, открывающемся кнопкой **Пуск**, а также с помощью двойного нажатия кнопки мыши на значке приложения на рабочем столе.

Запуск макрокоманды **ЗапускПриложения (RunApp)** в программе Visual Basic невозможен. Вместо этого следует использовать функцию **Shell** Visual Basic.

Макрокоманда «ЗапускПрограммы» (RunCode)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactRunCodeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactRunCodeX":1}

Макрокоманда **ЗапускПрограммы (RunCode)** вызывает процедуру **Function** Visual Basic.

Значения

Макрокоманда **ЗапускПрограммы (RunCode)** использует следующий аргумент.

Аргумент	Описание
Имя функции	Имя процедуры типа **Function** Visual Basic, которая будет вызвана. Все аргументы функции должны быть заключены в скобки. Введите имя функции в поле **Имя функции** в разделе **Аргументы макрокоманды** окна макрокоманды. Данный аргумент является обязательным. **Примечание.** Чтобы вызвать построитель выражений для выбора функции, нажмите кнопку **Построить** ... справа от поля **Имя функции**. Затем выберите нужную функцию в списке построителя выражений.

Дополнительные сведения

Определенные пользователем процедуры **Function** сохранятся в модулях Microsoft Access.

Круглые скобки после имени функции необходимы, даже если процедура **Function** не требует аргументов, например:

```
TestFunction()
```

В отличие от определяемых пользователем функций, используемых для указания значения свойства события, не следует в аргументе «Имя функции» помещать перед именем функции знак равенства (=).

Microsoft Access игнорирует значение, возвращаемое функцией.

Совет. Для выполнения процедуры **Sub** или процедуры обработки события, написанной на языке Visual Basic, следует создать процедуру **Function**, вызывающую процедуру **Sub** или процедуру обработки события. После этого становится возможным вызов этой процедуры **Function** с помощью макрокоманды **ЗапускПрограммы (RunCode)**.

При использовании макрокоманды **ЗапускПрограммы (RunCode)** для вызова функции поиск функции с указанным именем проводится в стандартных модулях базы данных. Однако если эта макрокоманда выполняется в результате выбора команды в меню формы или отчета или в ответ на событие в форме или отчете, Microsoft Access ищет эту функцию сначала в модуле класса формы или отчета, а затем в стандартных модулях. Поиск в модулях классов, которые появляются на вкладке **Модули** окна базы данных, не проводится для функции, имя которой определено в аргументе «Имя функции».

Вызов данной макрокоманды в программе Visual Basic невозможен. Необходимо вызвать процедуру **Function** непосредственно в программе Visual Basic.

Макрокоманда «ЗапускМакроса» (RunMacro)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactRunMacroC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactRunMacroX":1}

Макрокоманда **ЗапускМакроса (RunMacro)** запускает макрос. Указанный макрос может входить в состав группы макросов.

Эту макрокоманду используют для выполнения следующих действий.

- Вызов макроса из другого макроса.
- Вызов макроса при выполнении определенного условия.
- Связывание макроса с командой специального меню.

Значения

Макрокоманда **ЗапускМакроса (RunMacro)** использует следующие аргументы.

Аргумент	Описание
Имя макроса	Имя запускаемого макроса. В поле со списком **Имя макроса** в разделе **Аргументы макрокоманды** окна макрокоманды выводятся все макросы (и группы макросов) текущей базы данных. Если макрос принадлежит группе, то он выводится в соответствующем списке группы под именем *имяГруппыМакроса*.*имяМакроса*. Этот аргумент является обязательным. Если макрос, содержащий макрокоманду **ЗапускМакроса (RunMacro)**, запускается из библиотечной базы данных, то Microsoft Access ищет макрос с указанным именем в библиотечной базе данных и не проводит поиск в текущей базе данных.
Число повторов	Максимальное число повторов выполнения данного макроса. Если значение этого аргумента (и аргумента «Условие повтора») будет оставлено пустым, макрос выполняется только один раз.
Условие повтора	Выражение, имеющее значение **True** (-1) или **False** (0). Выполнение макроса прекращается, если указанное выражение принимает значение **False**. Истинность выражения проверяется при каждом запуске макроса.

Дополнительные сведения

Если в значении аргумента «Имя макроса» указано имя группы макросов, Microsoft Access выполнит первый макрос из этой группы.

Вызов этой макрокоманды эквивалентен запуску макроса с помощью команды **Макрос** в меню **Сервис**. Однако после выбора команды меню макрос выполняется только один раз, в то время как макрокоманда **ЗапускМакроса** позволяет выполнить макрос произвольное число раз.

Совет. Число повторений выполнения макроса определяется с помощью аргументов «Число повторов» и «Условие повтора» следующим образом.

- Если оба аргумента оставлены пустыми, макрос будет выполнен один раз.
- Если значение аргумента «Число повторов» определено, а значение аргумента «Условие повтора» оставлено пустым, то макрос будет выполнен указанное число раз.
- Если значение аргумента «Число повторов» оставлено пустым, а значение аргумента «Условие повтора» определено, то макрос будет выполняться до тех пор, пока указанное условие не примет значение **False**.
- Если определены значения обоих аргументов, то макрос будет выполняться до тех пор, пока

указанное условие не станет ложным, но не более указанного числа раз.

При вызове макрокоманды **ЗапускМакроса (RunMacro)** в макросе Microsoft Access выполняет вызванный макрос, а после завершения выполнения вызванного макроса выполняется следующая макрокоманда вызывающего макроса.

Примечания

- Допускается как вызов макроса, входящего в ту же группу макросов, так и вызов макроса из другой группы макросов.
- Допускается организация вложенных вызовов макросов. Это означает, что можно запустить макрос «A», из которого вызывается макрос «B» и т.д. В любом случае после завершения вызванного макроса происходит возврат в макрос, из которого был сделан вызов, и выполнение следующей макрокоманды в вызывающем макросе.

Совет. Макрокоманда **ЗапускМакроса (RunMacro)** в группе макросов, которая определяет специальные команды для специальной строки меню, используется для запуска макроса из специального меню. Однако, более удобным способом является применение новых возможностей панелей команд для запуска макроса из строки меню, панели инструментов или контекстного меню. В меню **Вид** выберите **Панели инструментов** и нажмите кнопку **Настройка**. Появится диалоговое окно **Настройка панелей инструментов**. На вкладке **Панели инструментов** выберите строку меню, панель инструментов или контекстное меню, из которого будет запускаться макрос. На вкладке **Команды** выберите значение «Все макросы» в поле **Категории**, затем в поле **Команды** выберите и перетащите макрос, который будет запускаться из строки меню, панели инструментов или контекстного меню. При выборе значка этого макроса из строки меню, панели инструментов или контекстного меню он будет запущен. Для создания значка запуска макроса перетащите макрос из окна базы данных в строку меню или панель инструментов.

Для запуска макрокоманды **ЗапускМакроса (RunMacro)** в программе Visual Basic следует использовать метод **RunMacro** объекта **DoCmd.**

Макрокоманда «ВыделитьОбъект» (SelectObject)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactSelectObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactSelectObjectX":1}

Макрокоманда **ВыделитьОбъект (SelectObject)** выделяет указанный объект базы данных.

Значения

Макрокоманда **ВыделитьОбъект (SelectObject)** использует следующие аргументы.

Аргумент	Описание
Тип объекта	Тип объекта, который будет выбран. Выберите «Таблица», «Запрос», «Форма», «Отчет», «Макрос» или «Модуль» в поле со списком **Тип объекта** в разделе **Аргументы макрокоманды** окна макрокоманды. Этот аргумент является обязательным.
Имя объекта	Тип объекта, который будет выбран. В поле со списком **Имя объекта** выводятся все объекты базы данных с типом, определенным аргументом «Тип объекта». Этот аргумент является обязательным, если в поле «В окне базы данных» установлено значение «Да». Если это значение установлено, и аргумент «Тип объекта» остался незаданным, то Microsoft Access выберет вкладку окна базы данных для ввода аргумента «Тип объекта».
В окне базы данных	Определяет, следует ли выделять объект в окне базы данных. Выберите **Да** (для выбора объекта в окне базы данных) или **Нет** (не проводить выбор объекта в окне базы данных). По умолчанию используется значение «Нет» .

Дополнительные сведения

Макрокоманда **ВыделитьОбъект (SelectObject)** применима к любому объекту Microsoft Access, которому можно передать фокус. Эта макрокоманда переводит фокус на указанный объект и, если объект скрыт, выводит его на экран. Для формы макрокоманда **ВыделитьОбъект (SelectObject)** автоматически устанавливает значение «Да» свойства формы **Вывод на экран** (**Visible**) и восстанавливает режим окна формы, указанный в свойствах формы (например, режим модальной или всплывающей формы).

Если объект не открыт в одном из других окон Microsoft Access, его можно выделить в окне базы данных, указав значение «Да» для аргумента «В окне базы данных». Если аргумент «В окне базы данных» имеет значение «Нет», то попытка выделить не открытый объект приводит к ошибке.

Обычно, эту макрокоманду используют для предварительного выделения объекта, над которым следует выполнить какую-либо операцию, например, при восстановлении свернутого объекта с помощью макрокоманды **Восстановить (Restore)** или при развертывании окна, содержащего объект, с помощью макрокоманды **Развернуть (Maximize).**

Для перемещения по выделенной форме используют макрокоманды **КЭлементуУправления (GoToControl), НаЗапись (GoToRecord)** или **НаСтраницу (GoToPage).** Макрокоманду **НаЗапись** используют также для перемещения по объектам в режиме таблицы.

Для запуска макрокоманды **ВыделитьОбъект (SelectObject)** в программе Visual Basic следует использовать метод **SelectObject** объекта **DoCmd.**

Макрокоманда «КомандыКлавиатуры» (SendKeys)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactSendKeysC;vastmSendKeys":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactSendKeysX":1}

Макрокоманда **КомандыКлавиатуры (SendKeys)** посылает нажатия клавиш непосредственно в Microsoft Access или в активное приложение Microsoft Windows.

Значения

Макрокоманда **КомандыКлавиатуры (SendKeys)** использует следующие аргументы.

Аргумент	Описание
Клавиши	Строка нажатий клавиш, которые должны быть обработаны Microsoft Access или другим приложением, длиной до 255 символов. Введите клавиши в поле **Клавиши** в разделе **Аргументы макрокоманды** окна макрокоманды Данный аргумент является обязательным.
Пауза	Указывает, должно ли выполнение макроса останавливаться до тех пор, пока не будут обработаны все нажатия клавиш. Допустимые значения: «Да» и «Нет» (используется по умолчанию).

Дополнительные сведения

Нажатия клавиш, переданные с помощью макрокоманды **КомандыКлавиатуры (SendKeys)**, обрабатываются точно так же, как и при непосредственном вводе с клавиатуры в окно Microsoft Access.

Для описания нажатий клавиш следует использовать тот же синтаксис, что и в инструкции **SendKeys.**

Примечание. Неверный синтаксис, ошибочно набранный текст и другие неподходящие значения клавиш могут привести к возникновению ошибки.

Обычно эту макрокоманду используют для ввода данных в диалоговое окно, особенно в тех случаях, когда прерывание выполнения макроса нежелательно. Некоторые макрокоманды Microsoft Access, например, **Печать (PrintOut)** и **НайтиЗапись (FindRecord)**, автоматически устанавливают нужные параметры в некоторых наиболее часто используемых диалоговых окнах. Макрокоманда **КомандыКлавиатуры (SendKeys)** позволяет выбрать параметры в менее часто используемых диалоговых окнах.

Примечания

- Поскольку вывод окна диалога прерывает выполнение макроса, макрокоманда **КомандыКлавиатуры (SendKeys)** должна находиться в макросе до макрокоманды, открывающей это окно диалога; кроме того, аргумент «Пауза» должен иметь значение «Нет».
- Синхронизация поступления нажатий клавиш в Microsoft Access или в другое приложение может оказаться затруднительной. По возможности, рекомендуется вместо макрокоманды **КомандыКлавиатуры (SendKeys)** использовать другие способы ввода параметров в окна диалога (например, макрокоманду **НайтиЗапись (FindRecord)**).

Для передачи в Microsoft Access или в другое приложение Windows последовательности нажатий клавиш, длина которой превышает 255 символов, можно включить в макрос подряд несколько макрокоманд **КомандыКлавиатуры (SendKeys)**.

При передаче нажатий клавиш с помощью макрокоманды **КомандыКлавиатуры (SendKeys)** возникают события **Клавиша вниз (KeyDown), Клавиша вверх (KeyUp)** и **Нажатие клавиши (KeyPress).** При передаче нажатий клавиш, не входящих в набор символов ANSI (например, функциональных клавиш), событие **Нажатие клавиши (KeyPress)** не возникает.

Вызов данной макрокоманды в программе Visual Basic невозможен. Вместо этого следует использовать инструкцию **SendKeys**.

Макрокоманда «ЗадатьЗначение» (SetValue)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactSetValueC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactSetValueX":1}

Макрокоманда **ЗадатьЗначение (SetValue)** задает значение поля, элемента управления или свойства в форме, в форме в режиме таблицы или в отчете.

Значения

Макрокоманда **ЗадатьЗначение (SetValue)** использует следующие аргументы.

Аргумент	Описание
Элемент	Имя поля, имя элемента управления или название свойства, значение которого следует задать. Введите имя поля, элемента управления или свойства в поле **Элемент** в разделе **Аргументы макрокоманды** окна макрокоманды.Для ссылки на элемент управления в форме или отчете, в которых вызывается макрос, достаточно указать *имяЭлемента*, а для ссылки на другие документы следует использовать полный синтаксис, т.е. **Forms!***имяФормы***!** *имяЭлемента.* Данный аргумент является обязательным.
Выражение	Выражение Microsoft Access, которое используется для установки значения данного элемента. Для ссылок на любые объекты в выражении следует использовать полный синтаксис. Например, для увеличения значения в элементе управления «Зарплата» в форме «Сотрудники» на 10 процентов используется следующее выражение `Forms!Сотрудники!Зарплата*1.1`. Данный аргумент является обязательным. **Примечание.** При определении этого аргумента не следует вводить знак равенства (=) перед выражением. Если знак равенства будет введен, то Microsoft Access вычислит значение выражения, а затем обработает его как выражение. Это может привести к непредсказуемым результатам, если выражение является строковым. Например, если ввести для этого аргумента выражение **="Строка1"**, то сначала будет определено значение этого выражения («Строка1»), а затем полученное значение будет использовано в качестве значения этого аргумента, т.е. будет проведен поиск элемента управления или свойства с именем **Строка1** в форме или отчете, из которого был вызван макрос.

Примечание. Для создания выражения, определяющего значение любого из аргументов данной макрокоманды, с помощью построителя выражений следует нажать кнопку построителя ... справа от соответствующей ячейки.

Дополнительные сведения

Данную макрокоманду используют для задания значения поля или элемента управления в форме (в режиме формы или таблицы) или в отчете. Кроме того, можно задать значение практически любого свойства элемента управления, формы или отчета в лбом режиме. Сведения о том, в каких режимах конкретное свойство допускает определение с помощью макроса, можно найти в описании этого свойства.

Кроме того, можно задать значение поля в базовой таблице формы, даже если в этой форме нет элемента управления, присоединенного к этому пол. В этом случае для аргумента «Элемент» следует использовать синтаксис **Forms!***имяФормы***!***имяПоля.* Для ссылки на поле в

базовой таблице отчета следует использовать синтаксис **Reports!***имяОтчета***!***имяПоля*, однако, в этом случае необходимо, чтобы отчет содержал элемент управления, присоединенный к этому пол, или на это поле должна иметься ссылка в вычисляемом элементе управления в отчете.

При задании значения элемента управления в форме с помощью макрокоманды **ЗадатьЗначение (SetValue)** не выполняется проверка условий на значение, наложенных на значение этого элемента управления на уровне формы. Однако если данный элемент управления является присоединенным, то выполняется проверка условий, наложенных на уровне таблицы, которая содержит базовое поле. Кроме того, макрокоманда **ЗадатьЗначение (SetValue)** активизирует пересчет, однако, пересчет может быть отложен. Для того чтобы активизировать немедленное обновление элемента управления и завершить отложенный пересчет, следует использовать макрокоманду **ОбновитьОбъект (RepaintObject)**. На значение элемента управления, задающееся с помощью макрокоманды **ЗадатьЗначение (SetValue)**, не влияют также определенная для элемента управления маска ввода и значение свойства **Маска ввода** (**InputMask**) поля базовой таблицы.

Для изменения значения элемента управления макрокоманду **ЗадатьЗначение (SetValue)** можно включить в макрос, связанный со свойством **После обновления** (**AfterUpdate)** этого элемента управления. Однако нельзя использовать макрокоманду **ЗадатьЗначение** в макросе, связанном со свойством **До обновления (BeforeUpdate**) для изменения значения этого элемента управления (хотя и можно использовать эту макрокоманду для изменения значений других элементов управления). Кроме того, макрокоманду **ЗадатьЗначение (SetValue)** можно использовать в макросе, связанном со свойствами формы **До обновления (BeforeUpdate)** или **После обновления (AfterUpdate)** для изменения значения любого элемента управления в текущей записи.

Примечание. Макрокоманду **ЗадатьЗначение (SetValue)** нельзя использовать для указания значения следующих элементов управления:

- присоединенные и вычисляемые элементы управления в отчетах.
- вычисляемые элементы управления в формах.

Совет. Макрокоманду **ЗадатьЗначение (SetValue)** можно использовать в режиме формы для удаления формы с экрана или вывода ее на экран. В качестве значения аргумента «Элемент» введите **Forms!***имяФормы***.Visible** и значение «Да» или «Нет» в аргументе «Выражение». Значение «Нет» для свойства **Вывод на экран (Visible)** модальной формы удаляет форму с экрана и делает ее немодальной. Значение «Да» этого свойства выводит форму на экран и снова делает ее модальной.

Изменение данных в элементе управления или добавление новых данных с помощью макрокоманды **ЗадатьЗначение (SetValue)** не приводит к возникновению таких событий как **До обновления (BeforeUpdate)**, **До вставки (BeforeInsert)** или **Изменение (Change)**, которые возникают при выполнении этих действий через интерфейс пользователя. Эти события также не возникают при задании значения элемента управления в программе Visual Basic.

Вызов данной макрокоманды в программе Visual Basic невозможен. Значение следует задавать непосредственно в инструкции Visual Basic.

Макрокоманда «УстановитьСообщения» (SetWarnings)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactSetWarningsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactSetWarningsX":1}

Макрокоманда **УстановитьСообщения (SetWarnings)** включает или отключает вывод системных сообщений.

Значения

Макрокоманда **УстановитьСообщения (SetWarnings)** использует следующий аргумент.

Аргумент	Описание
Включить сообщения	Определяет будут ли выводится системные сообщения. Возможные значения «Да» (сообщения включены) или «Нет» (сообщения не включены, используется по умолчанию) выбираются в разделе **Аргументы макрокоманды** окна макроса в ячейке аргумента «Включить сообщения».

Дополнительные сведения

Эту макрокоманду используют, если требуется запретить прерывание выполнения макроса для открытия модальных окон предупреждений и сообщений. Однако сообщения об ошибках выводятся на экран в любом случае. Кроме того, Microsoft Access выводит на экран все окна диалога, которые требуют от пользователя действий, отличных от простого нажатия кнопки; например, ввода текста или выбора параметра в группе.

Выполнение этой макрокоманды в режиме включенных сообщений вызывает те же действия, что и нажатие клавиши ENTER в ответ на появление окна предупреждения или сообщения. Обычно в таком случае нажимается кнопка **OK** или **Да**.

После завершения макроса Microsoft Access автоматически включает вывод системных сообщений.

Обычно, эту макрокоманду используют вместе с макрокомандой **ВыводНаЭкран (Echo),** которая отключает обновление экрана до завершения выполнения макроса. Макрокоманда **УстановитьСообщения (SetWarnings)** позволяет дополнительно отключить вывод окон предупреждений и сообщений.

Осторожно! Хотя использование макрокоманды **УстановитьСообщения (SetWarnings)** упрощает взаимодействие с макросом, необходима осторожность при отключении системных сообщений, поскольку существуют сообщения, появление которых требует немедленного завершения макроса. Если нет уверенности в том, что выполнение макроса не может привести к нежелательным последствиям, следует избегать использования этой макрокоманды.

Для запуска макрокоманды **УстановитьСообщения (SetWarnings)** в программе Visual Basic следует использовать метод **SetWarnings** объекта **DoCmd.**

Макрокоманда «ПоказатьВсеЗаписи» (ShowAllRecords)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactShowAllRecordsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactShowAllRecordsX":1}

Макрокоманда **ПоказатьВсеЗаписи (ShowAllRecords)** удаляет любой фильтр, примененный к активной таблице, форме или результирующему набору записей запроса, и выводит все записи, содержащиеся в таблице или в наборе записей, отобранном в запросе, или же все записи из базовой таблицы или запроса формы.

Значения

Макрокоманда **ПоказатьВсеЗаписи (ShowAllRecords)** не требует аргументов.

Дополнительные сведения

Эта макрокоманда обеспечивает вывод всех записей (в том числе, новых и измененных) в таблице, результирующем наборе записей запроса или форме. Данная макрокоманда вызывает повторный просмотр источников данных формы или подчиненной формы.

Кроме того, эту макрокоманду используют для удаления фильтра, примененного с помощью макрокоманды **ПрименитьФильтр (ApplyFilter),** команды **Применить фильтр** в меню **Записи** или с помощью макрокоманды **ОткрытьФорму (OpenForm),** в которой фильтр задается в аргументах «Имя фильтра» или «Условие отбора».

Вызов этой макрокоманды эквивалентен выбору команды **Удалить фильтр** в меню **Записи** или нажатию кнопки **Удалить фильтр**  на панели инструментов в режиме формы или в режиме таблицы.

Для запуска макрокоманды **ПоказатьВсеЗаписи (ShowAllRecords)** в программе Visual Basic следует использовать метод **ShowAllRecords** объекта **DoCmd.**

Макрокоманда «ОстановитьВсеМакросы» (StopAllMacros)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactStopAllMacrosC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactStopAllMacrosX":1}

Макрокоманда **ОстановитьВсеМакросы (StopAllMacros)** прекращает выполнение всех выполняемых макросов.

Значения

Макрокоманда **ОстановитьВсеМакросы (StopAllMacros)** не требует аргументов.

Дополнительные сведения

Обычно эта макрокоманда используется для остановки всех макросов при возникновении условия ошибки. В строку макрокоманды можно ввести условное выражение, при значении которого **True** (-1) Microsoft Access останавливает все макросы.

Например, окно сообщений в макросе может выводиться как результат выполнения нескольких макрокоманд, в том числе вызываемых из разных макросов. Нажатие кнопки **Отмена** в этом окне сообщения, запускающей макрокоманду **ОстановитьВсеМакросы (StopAllMacros)**, приведет к прекращению выполнения всех макросов.

Если с помощью макрокоманд **ВыводНаЭкран (Echo)** или **УстановитьСообщения (SetWarnings)** в макросе был отключен режим отображения или вывод системных сообщений, макрокоманда **ОстановитьВсеМакросы (StopAllMacros)** автоматически включает эти режимы.

Данная макрокоманда недоступна в Visual Basic.

Макрокоманда «ОстановитьМакрос» (StopMacro)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactStopMacroC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactStopMacroX":1}

Макрокоманда **ОстановитьМакрос (StopMacro)** прекращает выполнение текущего макроса.

Значения

Макрокоманда **ОстановитьМакрос (StopMacro)** не требует аргументов.

Дополнительные сведения

Обычно эта макрокоманда используется для остановки макроса при выполнении определенного условия. В строку макрокоманды можно ввести условное выражение, при значении которого **True** (-1) Microsoft Access останавливает данный макрос.

Например, можно создать макрос, который открывает форму, предназначенную для отображения заказов, соответствующих дате, введенной пользователем в специальное окно диалога. Для проверки допустимости значения даты, введенной в поле элемента управления «Дата размещения», можно использовать логическое выражение, которое при обнаружении неправильной даты будет задавать вывод на экран сообщения об ошибке с помощью макрокоманды **Сообщение (MsgBox)** с последующим прерыванием выполнения макроса с помощью макрокоманды **ОстановитьМакрос**.

Если с помощью макрокоманд **ВыводНаЭкран (Echo)** или **УстановитьСообщения (SetWarnings)** в макросе был отключен режим отображения или вывод системных сообщений, макрокоманда **ОстановитьМакрос (StopMacro)** автоматически включает эти режимы.

Данная макрокоманда недоступна в Visual Basic.

Макрокоманда «УдалитьОбъект» (DeleteObject)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactDeleteObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactDeleteObjectX":1}

Макрокоманда **УдалитьОбъект (DeleteObject)** позволяет удалить указанный объект базы данных.

Значения

Макрокоманда **УдалитьОбъект (DeleteObject)** использует следующие аргументы.

Аргумент	Описание
Тип объекта	Тип удаляемого объекта. Допустимые значения: «Таблица», «Запрос», «Форма», «Отчет», «Макрос» и «Модуль» выбираются в ячейке **Тип объекта** в бланке **Аргументы макрокоманды** в окне макроса. Для удаления объекта, выделенного в окне базы данных, оставьте эту ячейку пустой.
Имя объекта	Имя удаляемого объекта. Раскрывающийся список в ячейке **Имя объекта** содержит все объекты текущей базы данных, которые относятся к типу, указанному в аргументе «Тип объекта». Если значение аргумента «Тип объекта» оставлено пустым, оставьте пустым и значение этого аргумента. Если макрос, содержащий макрокоманду **УдалитьОбъект**, запускается из библиотечной базы данных, поиск объекта с указанным именем проводится сначала в библиотечной базе данных, а затем в текущей базе данных.

Осторожно! Если значения аргументов «Тип объекта» и «Имя объекта» оставлены пустыми, Microsoft Access удалит объект, выделенный в окне базы данных, без вывода предупреждающего сообщения. Для выделения объекта в окне базы данных можно использовать макрокоманду **ВыделитьОбъект (SelectObject)** с аргументом «В окне базы данных», имеющим значение «Да».

Дополнительные сведения

Макрокоманда **УдалитьОбъект** позволяет удалять временные объекты, созданные при выполнении макроса. Например, с помощью макрокоманды **ОткрытьЗапрос (OpenQuery)** можно выполнить запрос на создание таблицы, в котором создается временная таблица. После закрытия отчета можно использовать макрокоманду **УдалитьОбъект (DeleteObject)** для удаления этой временной таблицы.

Вызов данной макрокоманды эквивалентен выделению объекта в окне базы данных и последующему нажатию клавиши DEL или выбору команды **Удалить** в меню **Правка**.

Для выполнения макрокоманды **УдалитьОбъект (DeleteObject)** в программе Visual Basic следует вызвать метод **DeleteObject** объекта **DoCmd.**

Макрокоманда «ВывестиВФормате» (OutputTo)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactOutputToC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactOutputToX":1}

Макрокоманда **ВывестиВФормате (OutputTo)** выводит данные, содержащиеся в указанном объекте базы данных Microsoft Access (таблице, форме, отчете или модуле) в файл в формате Microsoft Excel 97 (*.xls), в текстовом формате MS-DOS (*.txt) или в формате RTF (*.rtf). Также возможен вывод в файл формата HTML (*.html), в файлы формата Microsoft Internet Information Server (*.htx, *.idc) или в файл в формате страниц Microsoft ActiveX Server (*.asp). Совместно с Microsoft Personal Web Server допускается использование файлов в формате Microsoft Internet Information Server.

Значения

Макрокоманда **ВывестиВФормате (OutputTo)** использует следующие аргументы.

Аргумент	Описание
Тип объекта	Тип объекта, содержащего данные, которые требуется преобразовать. Выберите «Таблица» (для таблицы в режиме таблицы; используется по умолчанию), «Запрос» (для запроса в режиме таблицы), «Форма» (для формы в режиме формы или режиме таблицы), «Отчет» и «Модуль» в ячейке **Тип объекта** в бланке **Аргументы макрокоманды** в окне макроса. Эта макрокоманда неприменима к макросам. Для того чтобы вывести содержимое активного объекта, укажите его тип, однако, оставьте пустым значение аргумента «Имя объекта». Данный аргумент является обязательным.
Имя объекта	Имя объекта, который содержит выводящиеся данные. Раскрывающийся список содержит все объекты текущей базы данных, которые относятся к типу, указанному в аргументе «Тип объекта». Если макрос, содержащий макрокоманду **ВывестиВФормате (OutputTo)**, запускается из библиотечной базы данных, поиск объекта с указанным именем проводится сначала в библиотечной базе данных, а затем в текущей базе данных.
Формат вывода	Формат, в котором следует вывести данные. Выберите в раскрывающемся списке «Формат HTML (*.html)», «Страницы Microsoft ActiveX Server (*.asp)», «Microsoft Excel (*.xls)», «Microsoft IIS (*.htx, *.idc)», «Текст MS-DOS (*.txt)» или «Формат RTF (*.rtf)». Модули допускают вывод только в текстовом формате. Форматы Microsoft Internet Information Server и Microsoft ActiveX Server доступны только для таблиц, запросов и форм. Если этот аргумент оставлен пустым, Microsoft Access выводит приглашение указать формат во время выполнения макрокоманды.
Файл вывода	Полное имя файла, в который следует вывести данные. Можно указать стандартное расширение имени файла (.asp, .htm или .html, .htx, .xls, .txt, или .rtf) для выбранного в аргументе «Формат вывода» формата. При выводе в файлы формата Microsoft Internet Information Server или Microsoft ActiveX Server всегда будут созданы файлы со стандартными расширениями имени .htx, .idc или .asp. Если этот аргумент оставлен пустым, Microsoft Access выводит приглашение указать имя выходного файла во время выполнения макрокоманды.

Автозагрузка	Указывает, следует ли сразу после вызова макрокоманды **ВывестиВФормате (OutputTo)** запустить соответствующее приложение и загрузить в него файл, указанный в аргументе «Файл вывода». Если для данного аргумента выбрано значение «Да», то будет запущено одно из следующих приложений: Microsoft Excel (для файлов с расширением .xls), Блокнот Microsoft Windows (для файлов с расширением .txt), или Microsoft Word (для файлов с расширением .rtf). Для файлов с расширением .html будет запущено определенное по умолчанию средство просмотра Интернета (например Microsoft Internet Explorer). Этот аргумент не используется для файлов Microsoft Internet Information Server и Microsoft ActiveX Server. Если выбрано значение «Нет», эти приложения не запускаются автоматически. По умолчанию используется значение «Нет».
Файл шаблона	Полное имя файла, используемого в качестве шаблона для файла в формате .html, .htx или .asp. Файл шаблона является файлом, содержащим теги HTML.

Дополнительные сведения

Microsoft Access поддерживает преобразование данных в различные форматы для чтения любым приложением, использующим этот формат. Например, отчет Microsoft Access (вместе с элементами форматирования) можно преобразовать в формат RTF, а затем открыть полученный документ в Word для Windows.

При выводе объекта базы данных в формате HTML, создается файл в формате HTML, содержащий данные этого объекта. Для задания файла, используемого в качестве шаблона для файла .html следует использовать аргумент «Файл шаблона».

При выводе объекта в формате Microsoft Internet Information Server создается два файла.

- Файл в формате .idc, содержащий сведения о подключении источника данных ODBC и инструкцию SQL, которую следует выполнить над этим источником. В этом случае выводимый объект Microsoft Access содержит данные, определяемые инструкцией SQL, и текущей базой данных Microsoft Access является источник данных ODBC.
- Файл в формате .htx, задающий способ форматирования данных, возвращаемых указанной в файле .idc инструкцией SQL в виде документа HTML. Для задания файла в формате .html, используемого в качестве шаблона для файла .htx можно воспользоваться аргументом «Файл шаблона».

Microsoft Internet Information Server использует файлы в формате .htx и .idc для создания файла .html, содержащего данные из выводимого объекта Microsoft Access.

При выводе объекта базы данных в формате Microsoft ActiveX Server создается файл в формате .asp, в котором содержатся сведения о способах доступа и форматирования данных объекта. Файл .asp используется Microsoft ActiveX Server для создания файла в формате .html, содержащего данные выводимого объекта Microsoft Access. Для задания файла в формате .html, используемого в качестве шаблона для файла .asp можно воспользоваться аргументом «Файл шаблона».

При выводе содержимого объекта базы данных в одном из форматов, за исключением формата Microsoft ActiveX Server (некоторые правила относятся также и к нему), с помощью макрокоманды **ВывестиВФормате (OutputTo)** действуют следующие правила.

- При выводе данных из таблицы, запроса или формы в режиме таблицы все поля в выходном файле выглядят так же, как и в Microsoft Access, за исключением полей, содержащих объекты OLE. Столбцы, соответствующие полям объектов OLE, включаются в выходной файл, однако, остаются пустыми.

- Значения элементов управления, связанных с логическими полями (выключателей, переключателей или флажков), в выходном файле выводятся как числовые значения: -1 (установлен) или 0 (снят).
- Для полей, связанных с полем гиперссылки, в выходном файле всех форматов, кроме текста MS-DOS (в этом случае все выводится в виде обычного текста), выводятся гиперссылки.
- При выводе данных из формы в режиме формы выходной файл всегда выглядит как форма в режиме таблицы.
- При выводе данных из отчета в выходной файл из всех элементов управления включаются только поля (для файлов с расширением .xls) или поля и подписи (для файлов с расширением .rtf, .txt и .html). Все остальные элементы управления игнорируются. Содержимое колонтитулов и областей заголовков и примечаний в выходной файл не включается. Единственным исключением из этого правила является включение в выходной файл в формате Microsoft Excel поля из области примечаний группы, содержащего выражение, в котором используется функция **Sum.** Никакие другие элементы управления их этих областей (в том числе содержащие другие статистические функции, отличные от **Sum**) в выходной файл не включаются.
- Подчиненные отчеты включаются в выходной файл, а подчиненные формы не включаются.
- При выводе таблицы или формы в формате HTML создается один файл .html. При выводе отчета в формате HTML для каждой страницы отчета создается свой файл .html.

Более подробные сведения о правилах и ограничениях вывода в файлы в формате .html содержатся в разделах Экспорт объекта в режиме таблицы в статический формат HTML и Экспорт отчета в статический формат HTML.

Более подробные сведения о правилах и ограничениях вывода в файлы в формате Microsoft Internet Information Server содержатся в разделах Экспорт объекта в режиме таблицы в динамический формат HTML и Экспорт формы в динамический формат HTML.

Более подробные сведения о правилах и ограничениях вывода в файлы в формате Microsoft ActiveX Server содержатся в разделах Экспорт объекта в режиме таблицы в динамический формат HTML, Экспорт формы в динамический формат HTML и Поддерживаемые и неподдерживаемые элементы управления в формах для динамического формата ASP.

Совет. Если при выводе отчета в выходном файле столбцы и строки не выравниваются должным образом, попробуйте применить следующие приемы.

- Выберите в меню **Формат** команду **Размер** и команду **по размеру данных** в подменю для изменения размеров элементов управления.
- Избегайте перекрытия элементов управления и не размещайте их слишком близко друг к другу.
- Выберите в меню **Формат** команду **Выровнять** и соответствующую подкоманду для выравнивания элементов управления в отчете. Те элементы управления, которые не выравниваются при выводе в файл другого формата, следует разместить в разных строках.

Вызов макрокоманды **ВывестиВФормате (OutputTo)** эквивалентен выбору в меню **Файл** команды **Сохранить как/Экспорт**, выбору параметра **Во внешнем файле или базе данных** и установке флажка **Вывод в формате** в открывающемся диалоговом окне. Аргументы макрокоманды соответствуют другим параметрам в диалоговых окнах, открывающихся после команды **Сохранить как/Экспорт**. Однако команда меню применима только к объекту, выделенному в окне базы данных. Макрокоманда **ВывестиВФормате (OutputTo)** позволяет указать преобразуемый объект.

Примечание. Допускается вывод в файл другого формата выделенного фрагмента данных с помощью команды **Сохранить как/Экспорт**. Однако это невозможно сделать с помощью макрокоманды **ВывестиВФормате (OutputTo)**.

Вывести файл в формате другого приложения и немедленно открыть его в Microsoft Excel или Microsoft Word позволяют также команды **Анализ в MS Excel** и **Публикация в MS Word** из

подменю **Связи с Office** в меню **Сервис**. С помощью команды **Сохранить как HTML** из меню **Файл** можно запустить мастера публикаций в Web для вывода объекта базы данных в файл формата .html, .htx/.idc или .asp.

Для выполнения макрокоманды **ВывестиВФормате (OutputTo)** в программе Visual Basic следует вызвать метод **OutputTo** объекта **DoCmd.**

Макрокоманда «ОтправитьОбъект» (SendObject)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactSendObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactSendObjectX":1}

Макрокоманда **ОтправитьОбъект (SendObject)** включает указанный объект в режиме таблицы, форму, отчет, или модуль Microsoft Access в сообщение электронной почты, делая возможным его просмотр и отправку. Допускается включение объектов в формате Microsoft Excel 97 (*.xls), в текстовом формате MS-DOS (*.txt), в формате RTF (*.rtf) или HTML в сообщения электронной почты Microsoft Exchange, Microsoft Mail, Microsoft Windows для рабочих групп или другого приложения электронной почты, использующего интерфейс MAPI (Microsoft Mail Applications Programming Interface).

Если у вас установлена программа электронной почты, использующая протокол VIM (Vendor Independent Mail) и установлена библиотека динамической компоновки (Mapivim.dll) , преобразующая сообщения MAPI в формат VIM, становится возможной пересылка объектов Microsoft Access в сообщениях электронной почты VIM.

Значения

Макрокоманда **ОтправитьОбъект (SendObject)** использует следующие аргументы.

Аргумент	Описание
Тип объекта	Тип объекта, который требуется включить в почтовое сообщение. Выберите «Таблица» (для таблицы в режиме таблицы), «Запрос» (для запроса в режиме таблицы), «Форма» (для формы в режиме формы или режиме таблицы), «Отчет» и «Модуль» в ячейке **Тип объекта** в бланке **Аргументы макрокоманды** в окне макроса. Не допускается пересылка макросов. Для того чтобы включить активный объект, укажите его тип, однако, оставьте пустым значение аргумента «Имя объекта».
Имя объекта	Имя объекта, включаемого в сообщение электронной почты. В раскрывающемся списке в ячейке **Имя объекта** выводятся имена всех объектов текущей базы данных, которые относятся к типу, указанному в аргументе «Тип объекта». Если значения обоих аргументов «Тип объекта» и «Имя объекта» оставлены пустыми, Microsoft Access передаст сообщение, не содержащее никаких объектов базы данных. Если макрос, содержащий макрокоманду **ОтправитьОбъект (SendObject)** запускается из библиотечной базы данных, поиск объекта с указанным именем проводится сначала в библиотечной базе данных, а затем в текущей базе данных.
Формат вывода	Формат, в который следует преобразовать передаваемый объект. В ячейке **Формат вывода** выберите «Формат HTML (*.html)», «Excel (*.xls)», «Текст MS-DOS (*.txt)» или «Формат RTF (*.rtf)». Передача модулей возможна только в текстовом формате MS-DOS. Если оставить эту ячейку пустой, то во время выполнения макрокоманды выводится приглашение указать формат вывода. **Примечание.** Не допускается включение объектов в форматах «Страницы Microsoft ActiveX Server (*.asp)» или «Microsoft IIS (*.htx, *.idc)», хотя эти элементы также присутствуют в списке.
Кому (To)	Получатели сообщения, имена которых следует включить в строку «To» («Кому») сообщения электронной почты. Если

	этот аргумент оставлен пустым, во время выполнения макрокоманды выводится приглашение указать имена получателей. Для разделения имен получателей в этом и двух следующих аргументах следует использовать точку с запятой (;) или символ разделителя элементов списка, выбранный на вкладке **Числа** в окне **Язык и стандарты** панели управления Windows. Если почтовое приложение не сможет идентифицировать имена получателей, сообщение отправлено не будет.
Копии (Cc)	Получатели сообщения, имена которых следует включить в строку «Cc» («Копии») сообщения электронной почты. Если этот аргумент оставлен пустым, строка «Cc» в сообщении будет пустой.
Скрытые копии (Bcc)	Получатели сообщения, имена которых следует включить в строку «Bcc» («Скрытые копии ») сообщения электронной почты. Если этот аргумент оставлен пустым, строка «Bcc» в сообщении будет пустой.
Тема	Тема сообщения. Этот текст включается в строку «Subject» («Тема») сообщения электронной почты. Если этот аргумент оставлен пустым, строка «Subject» в сообщении будет пустой.
Сообщение	Произвольный текст, который должен быть включен в сообщение помимо объекта базы данных. Этот текст включается в сообщение электронной почты сразу после объекта. Если этот аргумент оставлен пустым, сообщение не будет содержать дополнительный текст. Если значения обоих аргументов «Тип объекта» и «Имя объекта» оставлены пустыми, Microsoft Access передаст только этот текст без объекта базы данных.
Изменение сообщения	Указывает, может ли сообщение быть изменено перед передачей. Если выбрано значение «Да», приложение электронной почты будет запущено автоматически и пользователю будет предоставлена возможность изменить сообщение. Если выбрано значение «Нет», сообщение передается сразу. По умолчанию используется значение «Нет».
Файл шаблона	Полное имя файла, используемого в качестве шаблона для файла в формате .html. Файл шаблона является файлом, содержащим теги HTML.

Дополнительные сведения

Макрокоманда **ОтправитьОбъект** становится доступной только при установленной программе электронной почты, поддерживающей стандартный интерфейс MAPI, а также для программы электронной почты, поддерживающей протокол VIM, с установленной и настроенной библиотекой Mapivim.dll. Для получения дополнительных сведений об установке и настройке средств Microsoft Access, поддерживающих приложения электронной почты VIM, следует обратиться к документации Microsoft Office 97 Resource Kit.

Объект включается в сообщение электронной почты в указанном формате. Если установить указатель на этот объект и дважды нажать кнопку мыши, будет запущено соответствующее приложение: Microsoft Excel (для файлов с расширением .XLS), Microsoft Word для Windows (для файлов с расширением .RTF) или Блокнот Microsoft Windows (для файлов с расширением .TXT). Для файлов с расширением .html запустится определенное по умолчанию средство просмотра Интернет (например Microsoft Internet Explorer).

При включении объекта базы данных в сообщение электронной почты с помощью макрокоманды **ОтправитьОбъект (SendObject)** действуют следующие правила.

- При передаче таблицы, запроса или формы в режиме таблицы все поля в передаваемом объекте выглядят так же, как и в Microsoft Access, за исключением полей, содержащих объекты OLE. Столбцы, соответствующие этим полям, включаются в данный объект, однако, остаются пустыми.
- Значения элементов управления, связанных с логическими полями (выключателей, переключателей и флажков), в передаваемом объекте выводятся как числовые значения: -1 (установлен) или 0 (снят).
- Для полей, связанных с полем гиперссылки, в выходном файле всех форматов, кроме текста MS-DOS (в этом случае все выводится в виде обычного текста), выводятся гиперссылки.
- При передаче формы в режиме формы в объекте выводится форма в режиме таблицы.
- При выводе данных из отчета в выходной файл из всех элементов управления включаются только поля (для файлов с расширением .xls) или поля и подписи (для файлов с расширением .rtf, .txt и .html). Все остальные элементы управления игнорируются. Содержимое колонтитулов и областей заголовков и примечаний в выходной файл не включается. Единственным исключением из этого правила является включение в выходной файл в формате Microsoft Excel поля из области примечаний группы, содержащего выражение, в котором используется функция **Sum.** Никакие другие элементы управления их этих областей (в том числе содержащие другие статистические функции, отличные от **Sum**) в выходной файл не включаются.
- Подчиненные отчеты включаются в выходной файл, а подчиненные формы не включаются.
- При выводе таблицы или формы в формате HTML создается один файл .html. При выводе отчета в формате HTML для каждой страницы отчета создается свой файл .html.

Более подробные сведения о правилах и ограничениях на включение объектов в файлы в формате HTML содержатся в разделах Экспорт объекта в режиме таблицы в статический формат HTML и Экспорт отчета в статический формат HTML.

Совет. Если при выводе отчета в выходном файле столбцы и строки не выравниваются должным образом, попробуйте применить следующие приемы:

- Выберите в меню **Формат** команду **Размер** и команду **по размеру данных** в подменю для изменения размеров элементов управления.
- Избегайте перекрытия элементов управления и не размещайте их слишком близко друг к другу.
- Выберите в меню **Формат** команду **Выровнять** и соответствующую подкоманду для выравнивания элементов управления в отчете. Те элементы управления, которые не выравниваются при выводе в файл другого формата, следует разместить в разных строках.

Вызов макрокоманды **ОтправитьОбъект (SendObject)** эквивалентен выбору в меню **Файл** команды **Отправить по почте**, а аргументы макрокоманды эквивалентны параметрам в открывающихся диалоговых окнах. Команда **Отправить по почте** применима только к активному объекту. Макрокоманда **ОтправитьОбъект (SendObject)** позволяет указать передаваемый объект.

Примечание. С помощью команды **Отправить по почте** допускается передача выделенного фрагмента данных. Однако это невозможно сделать с помощью макрокоманды **ОтправитьОбъект (SendObject)**.

Для выполнения макрокоманды SendObject в программе Visual Basic следует вызвать метод **SendObject** объекта **DoCmd.**

Макрокоманда «ПанельИнструментов» (ShowToolbar)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactShowToolbarC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactShowToolbarX":1}

Макрокоманда **ПанельИнструментов (ShowToolbar)** выводит на экран или убирает с экрана встроенную или специальную панель инструментов. Пользователь имеет возможность указать вывод встроенной панели инструментов во всех окнах Microsoft Access или только в том окне, для которого эта панель инструментов предназначена (например, вывести панель инструментов «Режим формы» только в окне формы в режиме формы).

Примечание. Макрокоманда **ПанельИнструментов (ShowToolbar)** влияет только на панель инструментов и не влияет на строки меню и контекстные меню.

Значения

Макрокоманда **ПанельИнструментов (ShowToolbar)** использует следующие аргументы.

Аргумент	Описание
Название панели	Имя панели инструментов, которую следует вывести на экран или скрыть. В раскрывающемся списке в ячейке «Название панели» в бланке **Аргументы макрокоманды** в окне макроса выводятся имена всех панелей инструментов Microsoft Access, за которыми следуют имена специальных панелей, созданных пользователем в текущей базе данных. Данный аргумент является обязательным. Если флажок «Стандартные панели инструментов» в диалоговом окне **Параметры запуска** (которое открывается командой **Параметры запуска** в меню **Сервис**) снят, то в списке в ячейке «Название панели» выводятся имена только специальных панелей инструментов, определенных пользователем, а данная макрокоманда позволяет выводить или скрывать только специальные панели инструментов. Если макрос, содержащий макрокоманду **ПанельИнструментов (ShowToolbar)**, запускается из библиотечной базы данных, то поиск панели инструментов с указанным именем проводится сначала в библиотечной базе данных, а затем в текущей базе данных.
Отобразить	Указывает, что панель инструментов следует вывести на экран или убрать с экрана. По умолчанию используется значение «Нет» (убрать с экрана). Для встроенных панелей инструментов значение «Да» этого аргумента задает вывод данной панели инструментов во всех активных окнах Microsoft Access, значение «В обычном режиме» указывает вывод встроенной панели только в тех режимах Microsoft Access, в которых она выводится по умолчанию, а значение «Нет» запрещает вывод этой панели во всех окнах Microsoft Access. Для специальной панели инструментов значения «Да» и «В обычном режиме» указывает вывод панели во всех активных окнах Microsoft Access, а значение «Нет» запрещает вывод этой панели во всех окнах Microsoft Access.

Дополнительные сведения

Допускается использование этой макрокоманды в макросе с условными выражениями для

вывода на экран или скрытия одной или нескольких панелей инструментов в зависимости от определенных условий.

Для того чтобы связать определенную панель инструментов с конкретной формой или отчетом, можно указать имя макроса, содержащего макрокоманду **ПанельИнструментов (ShowToolbar)**, выводящую эту панель инструментов на экран, в значении свойства этой формы или отчета **Включение (OnActivate),** а имя макроса, содержащего макрокоманду **ПанельИнструментов (ShowToolbar)**, убирающую эту панель инструментов с экрана, в значении свойства этой формы или отчета **Отключение (OnDeactivate).**

Вызов этой макрокоманды эквивалентен выбору в меню **Вид** команды **Панели инструментов**, выбору подкоманды и установке или снятию флажка рядом с именем панели инструментов. Макрокоманда **ПанельИнструментов (ShowToolbar)** позволяет указать, следует ли выводить встроенную панель инструментов во всех окнах Microsoft Access или только в режиме, для которого она определена.

Невозможно вывести или скрыть встроенные панели инструментов с помощью данной макрокоманды, если для свойства **AllowBuiltInToolbars** в программе Visual Basic задано значение **False** (0), или если в программе Visual Basic с помощью метода **SetOption** задано значение **False** для параметра **Стандартные панели инструментов**.

Для выполнения макрокоманды **ПанельИнструментов (ShowToolbar)** в программе Visual Basic следует вызвать метод **ShowToolbar** объекта **DoCmd.**

Макрокоманда «ОткрытьМодуль» (OpenModule)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactOpenModuleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactOpenModuleX":1}

Макрокоманда **ОткрытьМодуль (OpenModule)** открывает указанную процедуру в указанном модуле Visual Basic. Эта процедура может быть процедурой **Sub**, процедурой **Function** или процедурой обработки события.

Значения

Макрокоманда **ОткрытьМодуль (OpenModule)** использует следующие аргументы.

Аргумент	Описание
Имя модуля	Имя модуля, который следует открыть. Значение этого аргумента следует оставить пустым, если поиск нужной процедуры следует выполнить во всех стандартных модулях в текущей базе данных. Если макрос, содержащий макрокоманду **ОткрытьМодуль (OpenModule)** запускается из библиотечной базы данных, то поиск модуля с указанным именем проводится сначала в библиотечной базе данных, а затем в текущей базе данных.
Имя процедуры	Имя процедуры, которую следует открыть. Если значение этого аргумента оставлено пустым, будет открыт раздел описания.

Примечание. Необходимо указать правильное имя хотя бы в одном из аргументов, «Имя модуля» или «Имя процедуры».

Дополнительные сведения

Эта макрокоманда позволяет запустить процедуру обработки события. Для этого следует указать имя процедуры обработки события в аргументе «Имя процедуры» и имя модуля в аргументе «Имя модуля». Например, чтобы открыть процедуру обработки события **Нажатие кнопки (Click)**, определенную для кнопки **Печать заказа** в форме «Заказы», введите `Form.Заказы` в качестве значения аргумента «Имя модуля» и `ПечатьЗаказа_Click` в качестве значения аргумента «Имя процедуры». Для просмотра процедуры обработки события формы или отчета необходимо, чтобы эта форма или отчет были открыты.

Аналогично, для открытия процедуры в модуле класса необходимо указать имя модуля, хотя модуль класса может и не быть открыт.

Для того чтобы можно было открыть личную процедуру, должен быть открыт содержащий ее модуль.

Вызов данной макрокоманды эквивалентен выбору модуля в окне базы данных с последующим нажатием кнопки **Конструктор**. Кроме того, эта макрокоманда позволяет указать имя процедуры, а также выполнить поиск процедуры в стандартном модуле базы данных.

Совет. Можно выбрать модуль в окне базы данных и переместить его с помощью мыши в строку макрокоманды в макросе. При этом в строку автоматически включается макрокоманда **ОткрытьМодуль (OpenModule)**, открывающая модуль на разделе описаний.

Для выполнения макрокоманды **ОткрытьМодуль (OpenModule)** в программе Visual Basic следует вызвать метод **OpenModule** объекта **DoCmd.**

Макрокоманда «Сохранить» (Save)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactSaveC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acactSaveX":1}

Макрокоманда **Сохранить (Save)** сохраняет либо указанный объект Microsoft Access, либо активный объект, если объект не указан. В некоторых случаях допускается сохранение активного объекта под новым именем (аналогично команде меню **Сохранить как** в меню **Файл**).

Значения

Макрокоманда **Сохранить (Save)** использует следующие аргументы.

Аргумент	Описание
Тип объекта	Тип объекта, который следует сохранить. Выберите значение «Таблица», «Запрос», «Форма», «Отчет», «Макрос» или «Модуль» в поле **Тип объекта** в разделе **Аргументы макрокоманды** в окне макроса. Для сохранения активного объекта оставьте данный аргумент пустым. Если значение данного аргумента задано, необходимо указать имя существующего объекта в аргументе «Имя объекта».
Имя объекта	Имя объекта, который следует сохранить. В раскрывающемся списке в ячейке содержатся имена всех объектов текущей базы данных, которые относятся к типу, указанному в аргументе «Тип объекта». Если значение аргумента «Тип объекта» не задано, то в некоторых случаях можно оставить данный аргумент пустым, чтобы сохранить активный объект, или ввести новое имя для сохранения активного объекта под этим именем. Новое имя должно удовлетворять стандартным соглашениям об именах объектов Microsoft Access.

Дополнительные сведения

Макрокоманда **Сохранить (Save)** применима к любому объекту базы данных, который может быть в явном виде открыт и сохранен пользователем. Эта макрокоманда применима только к открытым объектам. Вызов данной макрокоманды эквивалентен выделению объекта и выбору в меню **Файл** команды **Сохранить** или нажатию кнопки **Сохранить**  на панели инструментов. Выполнение этой макрокоманды при пустом значении аргумента «Тип объекта» и указанном в аргументе «Имя объекта» новом имени эквивалентно выбору в меню **Файл** команды **Сохранить как** с последующим вводом нового имени для активного объекта. Макрокоманда **Сохранить (Save)** позволяет указать сохраняемый объект и выполнить команду **Сохранить как** в макросе.

Примечание. Макрокоманду **Сохранить (Save)** нельзя использовать для сохранения под новым именем следующих объектов:

- формы в режиме формы или в режиме таблицы;
- отчеты в режиме предварительного просмотра;
- модули.

Макрокоманда **Сохранить (Save)** всегда сохраняет указанный объект или активный объект в той базе данных, в которой объект был создан, вне зависимости от того, выполняется она в макросе в текущей базе данных или в библиотечной базе данных.

Если активный объект сохраняется под новым именем, которое совпадает с именем существующего объекта того же типа, выводится окно диалога с приглашением подтвердить,

следует ли перезаписать существующий объект. При отключении вывода предупреждений с помощью вызванной ранее макрокоманды **УстановитьСообщения (SetWarnings)** со значением «Нет» аргумента «Включить сообщения» окно сообщения не выводится и объект перезаписывается автоматически.

Для запуска макрокоманды **Сохранить (Save)** в программе Visual Basic следует использовать метод **Save** объекта **DoCmd.**

Макрокоманда «ЗадатьКомандуМеню» (SetMenuItem)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acactSetMenuItemC; ofproEnabled "} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acactSetMenuItemX":1}

Макрокоманда **ЗадатьКомандуМеню (SetMenuItem)** задает состояние (доступна или недоступна, отмечена или не отмечена) команд специальной строки меню или общей строки меню для активного окна.

Примечание. Макрокоманда **ЗадатьКомандуМеню (SetMenuItem)** работает только в том случае, если специальные и общие строки меню созданы с помощью макросов строк меню. Макрокоманда **ЗадатьКомандуМеню (SetMenuItem)** включена в данную версию Microsoft Access только для совместимости с предыдущими версиями. Она не будет работать с новыми панелями команд. Для включения и отключения проверенных и не проверенных элементов новых строк меню, панелей инструментов и контекстных меню используются свойства **Доступ (Enabled)** и **State** Visual Basic.

Значения

Макрокоманда **ЗадатьКомандуМеню (SetMenuItem)** использует следующие аргументы.

Аргумент	Описание
Индекс меню	Индекс меню, которое содержит команду, подлежащую изменению. Введите значение типа **Integer** (начиная с 0), характеризующее индекс нужного меню в специальной или общей строке меню активного окна. Значение индекса вводится в поле **Индекс меню** в разделе **Аргументы макрокоманды** окна макроса. Индекс связан с позицией меню в специальной или общей строке меню (позиция команды **Добавить меню (AddMenu)** в макросе строки меню, начинается с 0). Строки меню могут различаться при использовании условных выражений в макросе, которые скрывают или выводят элементы в специальном меню. Этот аргумент является обязательным. Если выбрать меню с данным аргументом и оставить незаполненными аргументы «Индекс команды» и «Индекс подкоманды», то возможно разрешение или запрещение самого имени меню. Однако нельзя включить или выключить имя меню (Microsoft Access игнорирует значения «Включить» и «Выключить» в аргументе «Состояние элемента»).
Индекс команды	Индекс команды, для которой задается состояние. Введите значение типа **Integer**, отсчитываемое от 0, определяющее индекс команды в меню, указанном в аргументе «Индекс меню». Индекс определяет позицию команды в группе макросов, с помощью которой создается данное меню в специальной строке меню или общей строке меню. (Порядковый номер соответствующего данному меню макроса в группе макросов, отсчитываемый от 0). Этот индекс может отличаться от реального положения команды в выведенном меню, поскольку некоторые команды могут быть скрыты с помощью условных выражений в группе макросов, определяющих вывод или скрытие команд специального меню.
Индекс подкоманды	Индекс подкоманды, для которой задается состояние. Применим только в случае, если нужная команда имеет подменю. Введите значение типа **Integer**, отсчитываемое от

	0, определяющее индекс подкоманды в подменю, указанном в аргументе «Индекс команды». Индекс определяет позицию подкоманды в группе макросов с помощью которой создается данное подменю для специальной или общей строки меню (позиция макроса подкоманды в группе макросов, отсчитываемая с 0).
Состояние	Состояние, которое задается для команды или подкоманды. Значения «Включен» и «Отключен» определят включение или отключение данного пункта меню, а значения «Помечен» и «Не помечен» вывод или снятие метки (т.е. переключение состояния команды). По умолчанию задается значение «Включен».

Дополнительные сведения

Макрокоманда **ЗадатьКомандуМеню (SetMenuItem)** действует только на специальную строку меню или общую строку меню активного окна. Эта макрокоманда неприменима к встроенным меню Microsoft Access. Вызов макрокоманды **ЗадатьКомандуМеню (SetMenuItem)** в активном окне, не имеющем специальной или общей строки меню, приводит к ошибке при выполнении. Напомним, что общая строка меню заменяет встроенную строку меню во всех окнах Microsoft Access, для которых не определены специальные строки меню форм или отчетов.

Данная макрокоманда позволяет определить состояние команд меню и подкоманд первого уровня, но не команд подменю следующих уровней.

Для запуска макрокоманды **ЗадатьКомандуМеню** в программе Visual Basic следует использовать метод **SetMenuItem** объекта **DoCmd.**

Макрокоманда «ВыполнитьКоманду» (RunCommand)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acactRunCommandC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acactRunCommandX":1}

Макрокоманда **ВыполнитьКоманду (RunCommand)** применяется для выполнения встроенной команды Microsoft Access. Встроенные команды выводятся в строке меню, на панели инструментов или в контекстном меню.

Значения

Макрокоманда **ВыполнитьКоманду (RunCommand)** использует следующий аргумент.

Аргумент	Описание
Команда	Имя выполняемой команды. Допустимые встроенные команды Microsoft Access выводятся в списке **Команда** в алфавитном порядке. Данный аргумент является обязательным.

Дополнительные сведения

В Microsoft Access 97 макрокоманда **ВыполнитьКоманду (RunCommand)** заменяет макрокоманду **КомандаМеню (DoMenuItem)**. Когда пользователь открывает и сохраняет макрос из предыдущей версии Microsoft Access, который содержит макрокоманду **КомандаМеню**, эта макрокоманда вместе с ее аргументом автоматически преобразуется в эквивалентную макрокоманду **ВыполнитьКоманду**. Макрокоманда **КомандаМеню** более не выводится в списке макрокоманд в окне макроса Microsoft Access 97.

Макрокоманда **ВыполнитьКоманду (RunCommand)** используется для выполнения команды Microsoft Access, которая выводится в специальной строке меню, в общей строке меню, в специальном контекстном меню или в глобальном контекстном меню. Однако, как правило, удобнее добавлять команды Microsoft Access в специальные и контекстные меню с помощью диалогового окна **Настройка**, которое открывается при указании пункта **Панели инструментов** в меню **Вид** и выборе команды **Настройка**. На вкладке **Команды** диалогового окна **Настройка** выводятся встроенные команды Microsoft Access, которые размещаются на панелях инструментов и в меню Microsoft Access.

Для того, чтобы разрешить выполнение некоторой команды только при определенных условиях, следует использовать макрокоманду **ВыполнитьКоманду** в макросе с условными выражениями.

Примечание. В меню **Файл** Microsoft Access 97 выводится список баз данных, которые использовались последними по времени. Эти базы данных удобнее открывать путем выбора их имен в списке, чем с помощью команды меню **Открыть базу данных**. Названия этих баз данных не появляются в раскрывающемся списке для аргумента «Команда», и к ним нельзя получить доступ с помощью макрокоманды **ВыполнитьКоманду (RunCommand)** в макросе.

При использовании базы данных из предыдущей версии Microsoft Access некоторые команды могут оказаться недоступными. Возможно, такая команда была переименована, перемещена в другое меню или она вообще не используется в Microsoft Access 97. Макрокоманда **КомандаМеню (DoMenuItem)**, имеющая в качестве аргумента недопустимую команду, не может быть преобразована в макрокоманду **ВыполнитьКоманду (RunCommand)**. Если пользователь открывает макрос, содержащий недопустимую команду, то Microsoft Access выводит на экран макрокоманду **ВыполнитьКоманду** с пустым значением аргумента «Команда». В этом случае необходимо отредактировать макрос, чтобы задать допустимое значение для аргумента «Команда» или удалить макрокоманду **ВыполнитьКоманду**. Список команд из предыдущих версий Microsoft Access, которые не используются в Microsoft Access 97, содержится в разделе Аргументы макрокоманды **КомандаМеню (DoMenuItem)**, недопустимые в макрокоманде **ВыполнитьКоманду (RunCommand)**.

Чтобы запустить макрокоманду **ВыполнитьКоманду** в программе Visual Basic, следует использовать метод **RunCommand** объекта **Application**. (Это эквивалентно применению метода **RunCommand** объекта **DoCmd**).

Аргументы макрокоманды «КомандаМеню» (DoMenuItem), недопустимые в макрокоманде «ВыполнитьКоманду» (RunCommand)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acconOldDoMenuItemCmdC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acconOldDoMenuItemCmdX":1}

При использовании базы данных из предыдущей версии Microsoft Access некоторые команды могут оказаться недоступными. Возможно, такая команда была переименована, перемещена в другое меню или она вообще не используется в Microsoft Access 97. Макрокоманда **КомандаМеню (DoMenuItem)**, имеющая в качестве аргумента недопустимую команду, не может быть преобразована в макрокоманду **ВыполнитьКоманду (RunCommand)**.

Все команды, используемые в макрокоманде Microsoft Access 95 **КомандаМеню**, являются допустимыми и для макрокоманды Microsoft Access 97 **ВыполнитьКоманду**.

Следующие команды меню, допустимые для макрокоманды Microsoft Access 2.0 **КомандаМеню**, не применяются в методах или в макрокоманде **ВыполнитьКоманду** Microsoft Access 97.

Режим	Меню	Команда меню	Замена
База данных, конструктор таблиц, конструктор запросов, конструктор макросов, схема данных	**Файл**	**Печать описаний**	Аналогичная команда отсутствует. Вместо команды для просмотра сведений об объектах базы данных используется окно **Архивариус**, которое вызывается открывается при выборе подкоманды **Архивариус** команды **Анализ** меню **Сервис**. Для вывода информации об открытом или выделенном в окне базы данных макросе удобно использовать также команду **Предварительный просмотр** в меню **Файл**.
База данных	**Защита**	**Печать сведений о защите**	Аналогичная команда отсутствует. Защита базы данных устанавливается и проверяется с помощью подкоманд команды **Защита** в меню **Сервис**. Для вывода на печать сведений о защите используется кнопка **Отчет о защите** в диалоговом окне **Пользователи и группы**, которое открывается при выборе подкоманды

			Пользователи и группы команды **Защита** в меню **Сервис**.
База данных	**Файл**	**Настройка импорта/экспорта**	Аналогичная команда отсутствует. Для создания спецификаций импорта/экспорта текстовых файлов используются мастера по импорту, экспорту и связыванию текста, которые вызываются с помощью подкоманды **Во внешнем файле или базе данных** команды **Сохранить как/Экспорт** в меню **Файл**.
Конструктор таблиц	**Справка**	**Карточки подсказки**	Аналогичная команда отсутствует.
Конструктор таблиц	**Справка**	**Поддержка пользователей**	Аналогичная команда отсутствует.
Конструктор запросов	**Запрос**	**Объединить таблицы**	Аналогичная команда отсутствует.
Конструктор форм/отчетов	**Вид**	**Палитра**	Аналогичная команда отсутствует. Вместо команды используются кнопки на панели инструментов **Форматирование**, например: **Цвет заливки/фона**, **Цвет текста** или **Цвет линии/границы**.
Конструктор модулей	**Вид**	**Следующая процедура**	Аналогичная команда отсутствует. Перемещайтесь по процедуре в окне модуля.
Конструктор модулей	**Вид**	**Предыдущая процедура**	Команда недоступна. Перемещайтесь по процедурк в окне модуля.
Конструктор модулей	**Запуск**	**Изменить команды**	Аналогичная команда отсутствует. Вместо команды используется поле **Аргументы командной строки** на вкладке **Другие** в диалоговом окне **Параметры**, которое открывается при выборе команды **Параметры** в меню **Сервис**.

Примечание. В версии Microsoft Access 2.0 макрокоманда **КомандаМеню (DoMenuItem)** использовалась в группе макросов, определяющих команды специальной строки меню, для добавления команды **Надстройки** в меню **Файл** строки меню. При этом в пункт меню **Надстройки** автоматически добавлялись соответствующие подкоманды. Аналогично, в Microsoft Access 95 команда **Надстройки** включалась с помощью макрокоманды **КомандаМеню** в меню **Сервис** специальной строки меню, и Microsoft Access добавлял нужные подкоманды. Ни в одной их этих двух версий непосредственное выполнение команды **Надстройки** с помощью макрокоманды **КомандаМеню** не допускалось. В Microsoft Access 97 при открытом диалоговом окне **Настройка** возможно перетаскивание посредством мыши команды **Надстройки** из меню **Сервис** строки меню на любую панель инструментов. При этом макрокоманда **ВыполнитьКоманду (RunCommand)** не имеет аргументов, соответствующих команде **Надстройки**.

Перечни встроенных констант в Microsoft Access 97

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconUsingEnumsAccess97C"}

По сравнению с предыдущими версиями, в Microsoft Access 97 добавлены новые или изменены некоторые старые встроенные константы. Для удобства пользователей созданы стандартные перечни встроенных констант, которые выводятся в окне модуля в списке **Список констант**. Константы из этих перечней используются как аргументы различных методов, функций и свойств Microsoft Access или в качестве значений свойств. Теперь легко можно найти нужную константу в соответствующем перечне в окне модуля вместо того, чтобы задавать ее имя по памяти или искать его в разделе справки.

Ниже приводятся сведения о встроенных константах, входящих в перечни.

- Перечень констант для каждого аргумента метода, функции или свойства имеет имя, которое указывается в строке синтаксиса метода, функции или свойства в окне модуля (в отличие от этого, перечни констант для значений свойств не имеют имен). Для вывода на экран имени перечня следует установить флажок **Краткие сведения** на вкладке **Модуль** диалогового окна **Параметры**, которое открывается при выборе команды **Параметры** в меню **Сервис**. Например, строка синтаксиса для метода **OpenForm** объекта **DoCmd**, когда метод имеет аргумент *вид*, выглядит как **[View as AcFormView = acNormal]**. Здесь **AcFormView** — имя перечня констант, а **acNormal** — значение аргумента по умолчанию. В окне просмотра объектов в списке **Классы** также выводятся имена перечней констант, а в списке **Компонент** выводятся сами встроенные константы, содержащиеся в каждом из этих перечней.
- Константы из предыдущих версий Microsoft Access, имена которых изменились, по-прежнему могут применяться. Например, ранее в качестве значения аргумента *сохранение* метода **Close** объекта **DoCmd** использовалась встроенная константа **acPrompt**. В Microsoft Access 97вместо нее используется константа **acSavePrompt**, однако константа **acPrompt** также является допустимой.
- В предыдущих версиях Microsoft Access в ряде случаев допускалось использование пустых значений аргументов, при этом выполнялись действия, установленные по умолчанию для пустых аргументов. Например, если аргумент метода **Close** *типОбъекта* (или *имяОбъекта*) получал пустое значение, то Microsoft Access закрывал активное окно. Теперь вместо пустых значений применяются новые константы, используемые в качестве значений по умолчанию. Так, аргумент *типОбъекта* метода **Close** теперь принимает по умолчанию значение, задаваемое новой константой **acDefault**. Присвоение аргументу значения, равного такой новой константе, имеет те же последствия, что и присвоение ему пустого значения в предыдущих версиях Microsoft Access. Более того, если в Microsoft Access 97 аргументу все же присвоено пустое значение, оно считается равным новой константе, используемой по умолчанию
- Имеется одно исключение из правила, описанного в предшествующем пункте. Если программа, созданная в более ранней версии Visual Basic, выполняется в Microsoft Access с использованием программирования объектов, то присвоение пустых значений аргументам, для которых определены новые, используемые по умолчанию константы, приведет к ошибке. Если программа из более ранней версии Visual Basic или Visual Basic для приложений выполняется в Microsoft Access непосредственно, то ошибка не возникает.

В следующей таблице перечислены все входящие в стандартные перечни встроенные константы, которые были добавлены или изменены в Microsoft Access 97.

Элемент языка	Аргумент	Старая константа	Новая константа
Close (метод)	*сохранение*	**acPrompt**	**acSavePrompt**
Close (метод)	*типОбъекта*	(нет)	**acDefault**
CopyObject (метод)	*типИсточника*	(нет)	**acDefault**
DeleteObject (метод)	*типОбъекта*	(нет)	**acDefault**

GoToRecord (метод)	*типОбъекта*	**acTable**	**acDataTable**
GoToRecord (метод)	*типОбъекта*	**acQuery**	**acDataQuery**
GoToRecord (метод)	*типОбъекта*	**acForm**	**acDataForm**
GoToRecord (метод)	*типОбъекта*	(нет)	**acActiveDataObject**
OpenForm (метод)	*режимДанных*	**acAdd**	**acFormAdd**
OpenForm (метод)	*режимДанных*	**acEdit**	**acFormEdit**
OpenForm (метод)	*режимДанных*	**acReadOnly**	**acFormReadOnly**
OpenForm (метод)	*режимДанных*	(нет)	**acFormPropertySettings**
OpenForm (метод)	*режимОкна*	**acNormal**	**acWindowNormal**
OpenQuery (метод)	*вид*	**acNormal**	**acViewNormal**
OpenQuery (метод)	*вид*	**acDesign**	**acViewDesign**
OpenQuery (метод)	*вид*	**acPreview**	**acViewPreview**
OpenReport (метод)	*вид*	**acNormal**	**acViewNormal**
OpenReport (метод)	*вид*	**acDesign**	**acViewDesign**
OpenReport (метод)	*вид*	**acPreview**	**acViewPreview**
OpenTable (метод)	*вид*	**acNormal**	**acViewNormal**
OpenTable (метод)	*вид*	**acDesign**	**acViewDesign**
OpenTable (метод)	*вид*	**acPreview**	**acViewPreview**
OutputTo (метод)	*типОбъекта*	**acTable**	**acOutputTable**
OutputTo (метод)	*типОбъекта*	**acQuery**	**acOutputQuery**
OutputTo (метод)	*типОбъекта*	**acForm**	**acOutputForm**
OutputTo (метод)	*типОбъекта*	**acReport**	**acOutputReport**
OutputTo (метод)	*типОбъекта*	**acModule**	**acOutputModule**
Quit (метод объекта **DoCmd**)	*параметры*	**acPrompt**	**acQuitPrompt**
Quit (метод объекта **DoCmd**)	*параметры*	**acSaveYes** или **acSave**	**acQuitSaveAll**
Quit (метод объекта **DoCmd**)	*параметры*	**acSaveNo** или **acExit**	**acQuitSaveNone**
Quit (метод объекта **Application**)	*параметры*	**acPrompt**	**acQuitPrompt**
Quit (метод объекта **Application**)	*параметры*	**acSaveYes** или **acSave**	**acQuitSaveAll**
Quit (метод объекта **Application**)	*параметры*	**acExit**	**acQuitSaveNone**
Rename (метод)	*типОбъекта*	(нет)	**acDefault**
RepaintObject (метод)	*типОбъекта*	(нет)	**acDefault**
Save (метод)	*типОбъекта*	(нет)	**acDefault**
SendObject (метод)	*типОбъекта*	**acTable**	**acSendTable**
SendObject (метод)	*типОбъекта*	**acQuery**	**acSendQuery**
SendObject (метод)	*типОбъекта*	**acForm**	**acSendForm**
SendObject (метод)	*типОбъекта*	**acReport**	**acSendReport**
SendObject (метод)	*типОбъекта*	**acModule**	**acSendModule**
SendObject (метод)	*типОбъекта*	(нет)	**acSendNoObject**
TransferSpreadsheet (метод)	*типТаблицы*	(нет)	**acSpreadsheetTypeExcel3**
TransferSpreadsheet	*типТаблицы*	(нет)	**acSpreadsheetTypeExcel4**

(метод)			
TransferSpreadsheet (метод)	*типТаблицы*	(нет)	**acSpreadsheetTypeExcel5**
TransferSpreadsheet (метод)	*типТаблицы*	(нет)	**acSpreadsheetTypeExcel7**
TransferSpreadsheet (метод)	*типТаблицы*	(нет)	**acSpreadsheetTypeExcel97**
TransferSpreadsheet (метод)	*типТаблицы*	(нет)	**acSpreadsheetTypeLotusWK1**
TransferSpreadsheet (метод)	*типТаблицы*	(нет)	**acSpreadsheetTypeLotusWK3**
TransferSpreadsheet (метод)	*типТаблицы*	(нет)	**acSpreadsheetTypeLotusWJ2** — только в японской версии
TransferSpreadsheet (метод)	*типТаблицы*	(нет)	**acSpreadsheetTypeLotusWK4**
TransferText (метод)	*типПреобразования*	(нет)	**acImportHTML**
TransferText (метод)	*типПреобразования*	(нет)	**acExportHTML**
TransferText (метод)	*типПреобразования*	(нет)	**acLinkHTML**
HyperlinkPart (функция)	*часть*	(нет)	**acAddress**
HyperlinkPart (функция)	*часть*	(нет)	**acDisplayedValue**
HyperlinkPart (функция)	*часть*	(нет)	**acDisplayName**
HyperlinkPart (функция)	*часть*	(нет)	**acSubAddress**
Type (свойство)	[значение]	(нет)	**acClassModule**
Type (свойство)	[значение]	(нет)	**acStandardModule**

Специальные методы и свойства

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconCustomMethodsPropertiesC;vaconCreatingObjVar"}

Для определения нового специального объекта используются модули классов. При создании нового экземпляра класса создается новый объект и возвращается ссылка на него.

Любые общие процедуры, описанные в данном модуле класса, становятся методами и свойствами нового объекта. Для описания метода, который не возвращает значения, используется инструкция **Sub**, а для описания метода, возвращающего значение определенного типа, применяется инструкция **Function**.

Новые процедуры **Property Let**, **Property Get** и **Property Set** позволяют определить специальные свойства для объекта, определяемого пользователем. Процедура **Property Get** возвращает значение свойства. Процедура **Property Let** устанавливает значение свойства, не принадлежащего объекту. Процедура **Property Set** присваивает значение свойству объекта.

Например, модуль класса используется для создания интерфейса между приложением и набором функций интерфейса прикладного программирования Windows (API). Для этого необходимо создать набор простых процедур, которые вызывают более сложные процедуры из библиотеки динамической компоновки (DLL). При создании экземпляра этого класса написанные процедуры становятся методами нового объекта. Эти методы могут использоваться также, как и все другие методы, но при их запуске будут вызываться функции API.

Повышение производительности ядра базы данных Microsoft Jet

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acconJetPerformanceImprovementsC;dahowCustomizingDataAccess"}

В версию 3.5 ядра базы данных Microsoft Jet включены следующие дополнительные средства повышения производительности, которые не входили в ядро версии 3.0, применявшейся в Microsoft Access 95:

- Новый параметр системного реестра MaxLocksPerFile позволяет повысить производительность выполнения больших запросов по сравнению с серверами NetWare и Windows NT с помощью буферизации выполнения транзакций. Для установки значения этого параметра используется метод объектов доступа к данным (DAO) **SetOption**. Метод **SetOption** также позволяет изменять параметры системного реестра на стадии выполнения для настройки приложения в особой среде.
- Индексированные столбцы обеспечивают улучшенные показатели взаимодействия нескольких пользователей. Это означает, что большее количество пользователей может считывать и изменять индексированные столбцы без получения сообщений о конфликте блокировки.
- Благодаря усовершенствованному механизму транзакций для инструкций языка управления данными (DML) SQL и новым параметрам системного реестра, которые вызывают выполнение транзакций при достижении определенного порога блокировки, объемные запросы выполняются быстрее.
- Значительно повышена скорость выполнения запросов, содержащих оператор неравенства (**<>**) в условных выражениях.
- Последовательное считывание ускорено за счет того, что ядро базы данных Jet позволяет заранее единовременно выделить до 64 килобайт дисковой памяти.
- Ускорено выполнение временных запросов.
- Удаление таблицы производится быстрее при использовании инструкций SQL DROP или DELETE без предиката.

В версию 3.5 ядра базы данных Jet включены следующие средства повышения производительности, которые не входили в ядро, используемое в версиях 1.*x* и 2.0 Microsoft Access.

- Новая структура индексов снижает необходимый для сохранения индексов объем памяти и уменьшает время, требуемое для создания сложных индексов. При сжатии базы данных индексы оптимизируются по быстродействию. Для поддержания оптимизации по быстродействию необходимо регулярно выполнять сжатие базы данных.
- На индексированные страницы более не накладывается блокировка изменений. В результате повышается производительность при одновременной работе с общими данными нескольких пользователей в сетевой базе данных.
- Быстрее выполняются операции, включающие инструкцию DELETE Можно сразу удалить часть страницы, а не строки по одной.
- Механизм распределения данных по страницам ядра Jet усовершенствован таким образом, чтобы повысить вероятность сохранения данных из одной таблицы на соседних страницах. В результате повышается способность опережающего считывания данных ядром Jet.
- В ядре базы данных Jet поддерживаются многоканальные операции. По умолчанию, по одному каналу выполняется опережающее считывание данных, по другому - запись результатов, а по третьему - операции с временным буфером.
- В ядро базы данных Jet встроен механизм неявных транзакций. В результате пользователь получает преимущества повышения скорости, предоставляемые транзакциями, без использования в программе методов **BeginTrans** и **CommitTrans.** Однако для полного контроля над моментом записи данных на диск необходимо создавать явные транзакции с

помощью методов **BeginTrans** и **CommitTrans**.

- В ядро базы данных Jet включены новые механизмы сортировки, повышающие производительность.
- В ядро базы данных Jet включен временный буфер с динамической конфигурацией, который формируется при запуске приложения на основании доступных ресурсов системной памяти. Помещение во временный буфер последних использовавшихся данных приводит к повышению быстродействия.
- В ядро базы данных Jet включены новые драйверы ISAM, поддерживающие язык HTML, Microsoft Excel версии 5.0, 7.0 и 8.0, а также Paradox 5.x, dBASE 5.0, FoxPro 3.0, текстовые файлы и электронные таблицы Lotus 1-2-3 версии 3.0.

Дополнительные сведения по доступу к внешним базам данных см. в главе 18 «Работа с внешними данными» книги *Разработка приложений для Microsoft Access 97*.

Новые возможности редактора модулей

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acconUsingModuleEditorC"}

Новые возможности в Microsoft Access 97

Вкладка **Другие** в диалоговом окне **Параметры** (которое открывается командой **Параметры** в меню **Сервис**) содержит новую группу **Перехват ошибок**. Эта группа включает два новых параметра: **Останов в модуле класса** и **Останов при необрабатываемых ошибках**. Параметр **Останов при любой ошибке**, находившийся в Microsoft Access 95 на вкладке **Модули** диалогового окна **Параметры**, теперь включен в группу **Перехват ошибок**.

Эти параметры определяют вход Visual Basic в режим останова при возникновении ошибки в программе.

- **Останов при любой ошибке**. Выполнение программы Visual Basic прерывается при возникновении любых обрабатываемых и необрабатываемых ошибок как в стандартных модулях, так и в модулях класса. Активным остается модуль, содержащий процедуру, в которой возникла ошибка, а строка программы, вызвавшая ошибку, выделяется. Для продолжения выполнения программы или для выполнения следующего шага пользователь имеет возможность нажать клавиши ALT+F8 или ALT+F5.
- **Останов в модуле класса**. Выполнение программы Visual Basic прерывается при возникновении любых необрабатываемых ошибок в модулях класса и необрабатываемых ошибок в стандартных модулях. При возникновении ошибки в процедуре в модуле класса этот модуль остается активным, а строка программы, вызвавшая ошибку, выделяется. Для продолжения выполнения программы или для выполнения следующего шага пользователь имеет возможность нажать клавиши ALT+F8 или ALT+F5. Этот параметр используется для отладки программ в модулях класса.
- **Останов при необрабатываемых ошибках**. Выполнение программы Visual Basic прерывается при возникновении любых необрабатываемых ошибок в стандартных модулях. Если ошибка возникает в процедуре в модуле класса, программа Visual Basic прерывается на строке, из которой вызывалась процедура в модуле класса. При выборе этого параметра никогда не осуществляется вход в режим останова из модуля класса.

Пользователь имеет возможность задавать значения этих параметров в программах с помощью метода **SetOption** и возвращать их значения с помощью метода **GetOption**.

Новое меню **Отладка** содержит как некоторые команды из меню **Запуск** и **Сервис** Microsoft Access 95, так и две новые команды. Меню **Отладка** доступно только в режиме конструктора.

- По команде **Компилировать и сохранить все модули** производится компиляция всех модулей в проекте и сохранение их в откомпилированном состоянии.
- Команда **Шаг с выходом** выполняет программу во всех вложенных процедурах, начиная с той, для которой эта команда выполнялась. Затем управление передается в процедуру, вызвавшую первую процедуру. Предположим, например, что процедуры с «А» по «Д» вложены, «А» является первой процедурой дерева вызовов. Команда **Шаг с выходом** выполняется с процедуры «Б». «Б» вызывает «В», «В» вызывает «Г» и «Г» вызывает «Д». Процедура «Б» выполняется полностью, а управление передается процедуре «А». Данная команда полезна в том случае, когда дальнейшая трассировка процедуры не требуется.

Microsoft Access 97 включает также несколько новых режимов для написания программ в окне модуля, некоторые из которых описаны ниже. Для установки этих параметров выберите команду **Параметры** в меню **Сервис** и выберите вкладку **Модуль**.

- Параметр **Список компонентов** обеспечивает отображение списка объектов, свойств и методов, необходимых в процессе написания программы. Когда пользователь вводит имя объекта, Microsoft Access автоматически пытается помочь завершить инструкцию, отображая список объектов, методов и свойств, которые могут следовать за именем объекта. Для завершения инструкции следует выбрать элемент из списка или продолжить обычный ввод

текста. При продолжении ввода данных в списке отображается наиболее подходящий вариант. Чтобы ввести в инструкцию выбранный в списке элемент, нажмите клавиши CTRL+ENTER или TAB. Ввод выбранного элемента и переход к следующей строке осуществляется при нажатии клавиши ENTER. Чтобы закрыть список, нажмите клавишу ESC.

- Параметр **Краткие сведения** обеспечивает информацию по синтаксису при вводе текста программы в окне модуля. При вводе имени процедуры или метода, за которым следует пробел или открывающаяся скобка, под текущей строкой программы автоматически выводится подсказка. В этой подсказке содержится информация о процедуре, включая сведения об обязательных аргументах.
- При выбранном параметре **Подсказки значений данных** выводится значение переменной или выражения в режиме останова программы. Для отображения текущего значения переместите указатель на переменную или выражение.

Новые возможности Microsoft Access 95

Новый редактор модулей в дополнение к параметрам форматирования текста поддерживает выделение различных синтаксических конструкций с помощью цвета. Соответствующие настройки задаются на вкладке **Модуль** в диалоговом окне **Параметры**, которое открывается командой **Параметры** из меню **Сервис**. Кроме того, редактор модулей поддерживает символ продолжения программной строки, позволяющий разбивать одну логическую программную строку на несколько физических строк. Символ продолжения программной строки образуется из пробела с последующим символом подчеркивания (_), как показано в следующем примере:

```
MsgBox " Будет удалено 10 записей.  " _
   & " Продолжить?"
```

На вкладке **Модуль** в диалоговом окне **Параметры** пользователю предоставляется широкий выбор других параметров настройки среды. Допускается выбор режима просмотра всего модуля (при установленном флажке **Полный модуль**) или просмотра процедур по одной (при снятом флажке). Допускается указание параметров командной строки и параметров условной компиляции, а также включение других параметров программирования, например, **Останов при любой ошибке** или **Явное описание переменных**.

Для получения дополнительных сведений о каждом из параметров в диалоговом окне **Параметры** нажмите кнопку ? в правом верхнем углу диалогового окна, а затем выберите нужный элемент.

Некоторые новые характеристики редактора модулей предназначены для помощи в отладке программ. Окно диалога **Контрольное значение** позволяет в любой момент увидеть текущее значение выбранной переменной или выражения. Данное окно диалога выводится на экран в режиме останова программы. Выделите переменную или выражение и нажмите клавиши SHIFT+F9 или выберите в меню **Сервис** команду **Контрольное значение**.

В меню **Запуск** появились две новые команды, **Следующая инструкция** и **Показать следующую инструкцию**, которые становятся доступными после останова программы. Команда **Следующая инструкция** указывает инструкцию, с которой будет продолжено выполнение программы. Это позволяет обойти часть программных инструкций или повторно выполнить какой-либо блок программы при отладке. Команда **Показать следующую инструкцию** позволяет увидеть, какая инструкция будет выполнена первой после возобновления выполнения программы.

В новой среде разработчика имеются кнопки **Завершить, Сброс, Продолжить** и соответствующие им одноименные команды в меню **Запуск**. При завершении выполнения программы с помощью кнопки **Завершить** ■ на панели инструментов или кнопки **Завершить** в диалоговом окне ошибок выполнения переменные, описанные на уровне модуля, сохраняют свои значения. Для сброса значений переменных, описанных на уровне модуля, следует нажать кнопку **Сброс**

 на панели инструментов.

Новые возможности средства просмотра модели объектов

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconUsingObjectBrowserC"}

Средство просмотра модели объектов является инструментом разработчика, позволяющим просматривать составные части библиотеки типов компонентов ActiveX. Для перехода в средство просмотра объектов необходимо открыть модуль и нажать кнопку **Просмотр объектов**  на панели инструментов или выбрать **Просмотр объектов** в меню **Вид**. Кроме того окно просмотра объектов выводится при нажатии клавиши F2 в режиме конструктора.

Библиотеки типов содержат описания объектов, их свойств, методов, событий, констант, а также связанных с ними функций и инструкций. Вся информация сгруппирована в модули классов и стандартные модули. Имена модулей класса и стандартных модулей выводятся в списке **Модули/классы**. Связанные с модулями методы, свойства, события и константы отображаются в поле **Компоненты**.

Следующие новые особенности включены в средство просмотра модели объектов Microsoft Access 97:

- На первой строке в поле **Классы** находится особый элемент **<глобальные>**. Эта запись является верхней для каждой библиотеки в средстве просмотра объектов. При выборе в поле **Классы** элемента **<глобальные>** в списке **Компонент <имя>** появятся все общедоступные компоненты, включая все константы.
- Для поиска адресуемой библиотеки типов для любого класса или компонента необходимо ввести текст в поле **Образец поиска** и нажать кнопку **Поиск**  на панели инструментов.
- В средстве просмотра модели объектов предусмотрена вставка программы в модуль. Для этого необходимо выбрать метод или свойство и нажать кнопку **Копировать**  на панели инструментов, затем переключиться в окно модуля и нажать кнопку **Вставить**  на панели инструментов **Visual Basic**.
- Из окна просмотра объектов может быть получена справка по любому методу, свойству или событию. Для этого выберите имя в поле **Компонент <имя>** и нажмите кнопку **Справка**  на панели инструментов или нажмите клавишу F1.
- Для просмотра описания процедуры, определенной пользователем, сначала необходимо выбрать имя проекта из списка **Проект/Библиотека**, а затем выбрать имя стандартного модуля или модуля класса, содержащего нужную процедуру, из списка **Классы**. После этого следует выбрать имя процедуры в поле **Компонент <имя>** и нажать кнопку **Описания**  на панели инструментов. Microsoft Access откроет модуль, содержащий эту процедуру, и разместит в ней курсор.
- Для перехода к предыдущему элементу нажмите кнопку **Назад**  на панели инструментов. Чтобы вернуться к текущему элементу нажмите кнопку **Вперед**  .
- В средстве просмотра модели объектов некоторые встроенные константы сгруппированы согласно методам, их использующим. Например, выберите **Access** в поле **Проект/Библиотека** и перемещайте список **Классы** по всем классам объектов. Появятся все наборы констант, доступные в Microsoft Access. Для просмотра всех констант, которые могут быть использованы с методом **RunCommand** выберите **AcCommand**

Примечание. Для просмотра всех объектов из библиотеки Microsoft Office 8.0 в окне просмотра, включая панели команд, необходимо установить ссылку на эту библиотеку. Дополнительные сведения см. в разделе Определение ссылок на библиотеки типов.

Определение ссылок на библиотеки типов

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acconSettingReferencesC"}

Для использования в программе объектов из другого приложения необходимо установить ссылку на библиотеку типов этого приложения. Например, установка ссылки из Microsoft Access на библиотеку Microsoft Excel позволяет использовать объекты Microsoft Excel через механизм программирования объектов. При наличии ссылки на проект Visual Basic из базы данных Microsoft Access становится возможным вызов общих процедур этого проекта. Для использования элемента ActiveX в формах Microsoft Access необходима установка ссылки на этот элемент.

Установка ссылки из Microsoft Access возможна при открытом окне модуля или из программы Visual Basic.

Определение ссылки из Microsoft Access

Для того чтобы установить ссылку на библиотеку типов приложения:

1. Выберите в меню **Сервис** команду **Ссылки**. Эта команда доступна только в том случае, когда окно модуля открыто и находится в активном состоянии в режиме конструктора.
2. Установите флажок рядом с именем нужной библиотеки типов приложения.

Определение ссылки из Visual Basic

Для установки ссылки из Visual Basic необходимо создать объект **Reference**, представляющий нужную ссылку. Семейство **References** содержит все доступные в настоящий момент ссылки.

Для создания нового объекта **Reference** используется метод **AddFromFile** или **AddFromGUID** семейства **References**. Удаление объекта **Reference** производится с помощью метода **Remove**.

Преимущества определения ссылок

Программа, созданная с использованием механизма программирования объектов и работающая с объектами другого приложения, при наличии ссылки на библиотеку типов будет выполняться быстрее. Ссылка на библиотеку типов позволяет описывать переменную, представляющую объект из другого приложения, с типом, соответствующим именно этому объекту. Например, если создается программа, работающая с объектом Microsoft Excel, описание объектной переменной с помощью следующего синтаксиса становится возможным только при созданной ссылке на библиотеку типов Microsoft Excel:

```
Dim appXL As New Excel.Application
```

Если ссылка на библиотеку типов Microsoft Excel не создана, придется описывать эту переменную с встроенным типом **Object**. Программа, содержащая следующее описание, будет выполняться существенно медленнее.

```
Dim appXL As Object
```

Кроме того, после создания ссылки на библиотеку типов приложения все объекты из этой библиотеки, а также их методы и свойства становятся доступными в средстве просмотра модели объектов. При этом пользователь всегда может видеть список свойств и методов, доступных для каждого объекта.

Поскольку Microsoft Access является также компонентом ActiveX, который поддерживает программирование объектов, допускается создание в другом приложении ссылки на библиотеку типов Microsoft Access и управление объектами Microsoft Access из другого приложения.

Работа в окне отладки

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acconUsingDebugWindowC;acdecDebugVBCodeS;achowSetWatchS;vaobjDebug"}

Окно отладки в Microsoft Access 97 состоит из трех панелей, панели отладки, функционально аналогичной окну отладки в предыдущих версиях Microsoft Access, панели локальных значений, на которой выводятся все переменные текущего кадра стека и их значения, панели контрольных значений. Переключение между панелью локальных значений и панелью контрольных значений осуществляется путем выбора вкладок **Локальные** и **Контрольные** в окне отладки. Панель отладки остается на экране независимо от выбранной вкладки.

Для того чтобы открыть окно отладки следует открыть модуль и нажать кнопку **Окно отладки**  на панели инструментов. Кроме того, окно отладки открывается из любого режима Microsoft Access путем выбора команды **Окно отладки** из меню **Вид**.

Панель локальных значений

На панели локальных значений выводится список переменных в трех столбцах: «Выражение», «Значение» и «Тип». Некоторые переменные (например, определяемые пользователем типы, массивы, объекты) могут содержать иерархическую информацию. Для управления отображением этой информации используется кнопка слева от имен переменных. Если панель локальных значений является видимой, то ее содержимое автоматически изменяется при переходе из режима выполнения в режим останова. Переход осуществляется при достижении точки останова или при пошаговом выполнении программы.

Первой переменной в списке является специальная переменная модуля. Для модуля класса это определяемая системой переменная **Me**. Данная переменная принимает значение ссылки на текущий экземпляр класса, определенный в текущем модуле. Так как эта переменная является ссылкой на объект, то через нее могут быть выведены все свойства и данные, входящие в текущий экземпляр класса. Для стандартного модуля первым в списке отображается имя текущего модуля. Это имя используется для вывода списка всех переменных уровня модуля в текущем модуле. На панели локальных значений допускается изменение значения переменной, но параметры «Выражение» и «Тип» изменять нельзя.

Панель контрольных значений

Панель контрольных значений позволяет следить за изменением значения выражения или переменной в ходе выполнения программы. Для того чтобы определить контрольное выражение, выберите в меню **Отладка** команду **Добавить контрольное значение**. Для панели контрольных значений добавлены следующие функции.

- Развертывание/свертывание иерархической структуры сведений.
- Изменение размера заголовков столбцов.
- Изменение значений по месту.

Панель отладки

В панели отладки может быть запущена любая процедура типа **Sub** или **Function**, включая процедуры обработки событий.

Для запуска процедуры, определенной внутри модуля класса, из панели отладки необходимо описать процедуру с именем модуля класса, если в этом модуле не включен режим останова. Если же режим останова включен, то нет необходимости определять процедуру, так как данный модуль находится вне области видимости.

В следующем примере демонстрируется вызов из панели проверки окна отладки процедуры «СписокФамилий», определенной в форме «Сотрудники»:

```
Form_Сотрудники.СписокФамилий
```

В следующем примере демонстрируется запуск процедуры обработки события нажатия кнопки «ЛичныеДанные» в форме «Сотрудники»:

```
Form_Сотрудники.ЛичныеДанные_Click
```

Прочие функции окна отладки

В новом окне отладки предусмотрен автоматический вывод информации о состоянии программы. Если программа не выполняется, в строке состояния под заголовком окна отладки выводится сообщение "<Готово>". После начала выполнения программы в строке состояния окна отладки выводится имя текущей базы данных, имя модуля, в котором находится текущая выполняемая процедура, и имя самой процедуры.

Допускается также открытие из окна отладки окна вызовов с помощью кнопки **Построить**  .

Программы, содержащие модули классов

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acconProgrammingFormsReportsC;vaobjCollection"}

В Microsoft Access 97 существует два типа модулей: стандартные модули и модули классов. В Microsoft Access 95 модули классов всегда были связаны с формой или отчетом. В Microsoft Access 97 они также присутствуют во вкладке **Модули** окна базы данных.

Новые возможности Microsoft Access 97: создание специальных объектов с модулями классов

Для определения специального объекта может использоваться модуль класса. Имя, под которым сохраняется модуль класса, становится именем специального объекта. Общие процедуры типа **Sub** и **Function**, определенные в модуле класса, будут специальными методами объекта. Общие процедуры **Property Let**, **Property Get** и **Property Set** становятся свойствами объекта.

При определении процедуры внутри модуля класса может быть создан новый объект, в качестве экземпляра этого класса. Для создания нового экземпляра класса объявляется переменная такого типа, который задан этим классом. Например, если имя класса «ОсновнойКласс», то новый экземпляр этого класса может быть создан следующим образом:

```
Dim abc As New ОсновнойКласс
```

При выполнении программы, содержащей это описание, Visual Basic создаст новый экземпляр. Обращение к его методам и свойствам осуществляется с помощью переменной `abc`. Например, если определен специальный метод «СписокИмен», то его вызов может выглядеть следующим образом:

```
abc.СписокИмен
```

Новые возможности Microsoft Access 95: создание стандартного экземпляра класса формы

При открытии формы в режиме формы, как с помощью интерфейса пользователя, так и из программы Visual Basic, создается новый экземпляр этой формы. Другими словами, пользователь выделяет место в памяти, в которое помещается. объект. После этого становится возможным вызов методов объекта, а также задание и возвращение значений свойств объекта программным образом, аналогично вызовам методов и определению значений свойств встроенных объектов. Это же справедливо и для отчетов, открываемых в режиме предварительного просмотра.

При ссылках на форму в программе Visual Basic обычно производится работа со стандартным экземпляром формы. Класс формы имеет только один стандартный экземпляр. Кроме того, в программах Visual Basic допускается создание нескольких экземпляров класса формы. При создании нескольких экземпляров объекта приходится работать с экземплярами, не являющимися стандартными.

Существуют четыре способа создания стандартного экземпляра формы. Существующая форма может быть открыта с помощью интерфейса пользователя, при вызове метода **OpenForm** объекта **DoCmd,** при вызове функции **CreateForm** с последующим переводом новой формы в режим формы, а также при создании в программе Visual Basic переменной типа **Form**, представляющей стандартный экземпляр формы. Следующая программа открывает форму «Сотрудники» и связывает с ней переменную типа **Form**.

```
Dim frm As Form
DoCmd.OpenForm "Сотрудники"
Set frm = Forms!Сотрудники
```

В Microsoft Access определен новый способ, позволяющий быстро открыть форму и сослаться на метод или свойство этой формы или одного из содержащихся в ней элементов управления в

одной инструкции. Это делается путем ссылки на модуль класса, как показано в следующем примере.

```
Form_Сотрудники.Visible = True
Form_Сотрудники.Caption = "Новые сотрудники"
```

При выполнении этой программы Microsoft Access открывает форму «Сотрудники» в режиме формы, если эта форма еще не открыта, и выводит новый заголовок формы. Форма не является видимой до тех пор, пока для ее свойства **Вывод на экран (Visible)** не задано значение **True** (-1). После завершения выполнения процедуры, в которой вызывается эта программа, экземпляр формы уничтожается, т.е. форма закрывается.

При попытке выполнить эту программу, когда форма «Сотрудники» открыта в режиме конструктора, Microsoft Access генерирует ошибку выполнения. Необходимо, чтобы форма была открыта в режиме формы или вообще не была открыта.

Если данный синтаксис применяется для изменения свойства формы или одного из ее элементов управления, новые значения будут потеряны после уничтожения текущего экземпляра формы. Это относится к любым изменениям свойств формы в режиме формы. Для сохранения нового значения свойства необходимо изменить это значение в режиме конструктора и сохранить измененный макет формы.

Новые возможности Microsoft Access 95: Создание множественных нестандартных экземпляров формы

Для того чтобы одновременно открыть несколько экземпляров одной формы, необходимо создание множественных нестандартных экземпляров класса формы. Например, может потребоваться вывод на экран как записи о сотруднике, так и записи о руководителе этого сотрудника. Для этого следует открыть два экземпляра формы в каждом из которых выводится нужная запись.

Для создания новых нестандартных экземпляров формы в программе Visual Basic необходимо описать переменную, типом которой является имя класса модуля формы. В описание переменной необходимо включить ключевое слово **New.** Например, следующая инструкция создает новый экземпляр формы «Сотрудники» и связывает его с переменной типа **Form**.

```
Dim frm As New Form_Сотрудники
```

Этот нестандартный экземпляр формы не будет видимым до тех пор, пока не будет в явном виде определено его свойство **Вывод на экран (Visible)**.

После завершения выполнения процедуры, в которой создается этот экземпляр формы, он будет уничтожен, если представляющая его переменная не описана как переменная уровня модуля. Поскольку переменные, описанные на уровне модуля, сохраняют свои значения до явного сброса по команде **Сброс** из меню **Запуск** или при нажатии кнопки **Сброс**  на панели инструментов, форма, представляемая переменной, описанной на уровне модуля, будет оставаться открытой.

Любые значения свойств, задаваемые пользователем, будут относиться только к текущему экземпляру класса формы и не будут сохранены вместе с формой. Кроме того, не допускается создание новых экземпляров формы, если форма открыта в режиме конструктора.

Не допускаются ссылки на нестандартные экземпляры формы по ее имени в семействе **Forms**. Такие ссылки возможны только по индексам в семействе. Поскольку при создании множественных нестандартных экземпляров формы все экземпляры в семействе **Forms** будут иметь одинаковые имена, индекс является единственным средством, позволяющим различать эти экземпляры.

Программирование объектов в Microsoft Access

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acconAccessAutomationControllerC;vaconUnderstandingOleAutomation;vaconWorkingAcrossApps"}

Microsoft Access является компонентом ActiveX, который поддерживает программирование объектов, ранее называемое «механизмом управления OLE». В Microsoft Access предусмотрены два способа программирования объектов. Во-первых, возможна работа с объектами других компонент. Во-вторых, Microsoft Access предоставляет собственные объекты другим компонентам ActiveX.

В предыдущих версиях Microsoft Access для связывания переменной с экземпляром приложения использовались функции **CreateObject** или **GetObject.** В Microsoft Access 97 кроме этого допускается создание с помощью ключевого слова **New** новых экземпляров некоторых приложений. Описание синтаксиса, поддерживаемого данным приложением, следует искать в документации приложения.

В Microsoft Access имеется возможность создавать ссылки на библиотеку типов приложения, повышающие производительность при управлении данным приложением с помощью программирования объектов. Кроме того, в Microsoft Access существует средство просмотра модели объектов, позволяющее просматривать списки объектов других приложений, а также списки их методов и свойств.

Библиотека типов Microsoft Access предоставляет другим компонентам информацию по объектам Microsoft Access. Предусмотрена возможность определения ссылки на библиотеку типов Microsoft Access из какого-либо компонента и просмотра объектов этой библиотеки с помощью средства просмотра модели объектов.

Для того чтобы работать с объектами Microsoft Access с помощью механизма программирования объектов, следует создать экземпляр объекта Microsoft Access **Application**. Предположим, например, что требуется вывести в форме или отчете Microsoft Access данные из электронной таблицы Microsoft Excel. Для запуска Microsoft Access из Microsoft Excel используется ключевое слово **New** для создания объекта **Application**. Альтернативным путем является применение функции **CreateObject** для создания экземпляра объекта Microsoft Access **Application** или функции **GetObject** для связывания объектной переменной с существующим экземпляром Microsoft Access. Описание синтаксиса, поддерживаемого данным компонентом, следует искать в документации компонента.

Если после запуска экземпляра Microsoft Access требуется выполнить работу с любым объектом Microsoft Access, необходимо открыть базу данных в окне Microsoft Access с помощью метода **OpenCurrentDatabase** или **NewCurrentDatabase.** Если Microsoft Access был открыт только как средство использования объектов доступа к данным, то открывать базу данных в окне Microsoft Access не обязательно. Для управления объектами библиотеки «Microsoft DAO 3.5» с помощью механизма программирования объектов следует использовать свойство **DBEngine** объекта Microsoft Access **Application**.

Повышение производительности при компиляции

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acconNewCompilationC"}

Для ускорения компиляции и выполнения программ в Microsoft Access улучшены характеристики загрузки и компиляции модулей.

Модули форм и отчетов, созданные по требованию

При создании формы или отчета в Microsoft Access 97 автоматического назначения связанного с ними модуля не происходит. Модуль создается после нажатия кнопки **Программа**  на панели инструментов, которая служит для просмотра модуля формы или отчета. Для создания модуля из Visual Basic проводится ссылка на свойство формы **Module**, при этом форма или отчет должны находится в режиме конструктора. Создание формы возможно также путем установления значения **True** (–1) для свойства **HasModule.**

Значение свойства **HasModule** указывает на наличие модуля, связанного с формой или отчетом.

Так как модуль формы или отчета не создается, пока есть необходимость добавления в них программ, в проекте содержится меньше модулей, предназначенных для компиляции. Это приводит к повышению производительности. Формы и отчеты без модулей загружаются значительно быстрее форм и отчетов с модулями.

Команды компиляции и сохранения модулей

В меню **Запуск** включены команды **Компилировать загруженные модули**, **Компилировать все модули** и **Компилировать и сохранить все модули.** По команде **Компилировать все модули** выполняется компиляция всех модулей в базе данных, вне зависимости от того, загружены они или нет. Команда **Компилировать загруженные модули** предназначена для компиляции только загруженных модулей. По команде **Компилировать и сохранить все модули** все программы базы данных сохраняются в откомпилированном состоянии.

Компиляция по запросу

Явная компиляция модулей с помощью описанных выше команд является полезной, но необязательной. Если модуль не был откомпилирован, то перед запуском процедуры из этого модуля Microsoft Access откомпилирует его.

При загрузке модуля для запуска Microsoft Access проверяет был ли он уже откомпилирован. Если нет, то модуль компилируется непосредственно перед запуском содержащейся в нем процедуры. Процесс компиляции замедляет работу программы. Таким образом заранее откомпилированная программа будет работать быстрее.

Правила компиляции частично определяются значением параметра **Компиляция по запросу** во вкладке **Модуль** диалогового окна **Параметры**, которое выводится по команде **Параметры** из меню **Сервис**. По умолчанию установлен режим компиляции по запросу. Для повышения производительности рекомендуется оставить этот режим включенным.

При установленном параметре **Компиляция по запросу** компиляция неоткомпилированных модулей происходит при их загрузке для выполнения. Если данный параметр не установлен, то неоткомпилированный модуль компилируется сразу при загрузке модуля, который содержит вызов процедуры из неоткомпилированного модуля. Предположим, что имеется три неоткомпилированных модуля: «МодульА», «МодульБ» и «МодульВ». «ПроцедураА» модуля «МодульА» содержит вызов процедуры «ПроцедураБ» из «МодульБ». Другая процедура из «МодульА» содержит вызов процедуры «ПроцедураВ» из «МодульВ».

Если параметр **Компиляция по запросу** установлен при вызове процедуры «ПроцедураА», то «МодульА» и «МодульБ» загружаются и компилируются. Так как «ПроцедураА» не содержит вызова процедуры «ПроцедураВ», то «МодульВ» не загружается и не компилируется.

Если параметр **Компиляция по запросу** не установлен при вызове процедуры «ПроцедураА», то «МодульА», «МодульБ» и «МодульВ» загружаются и компилируются. Так как «МодульА» содержит вызов процедуры «ПроцедураВ», то «МодульВ» загружается и компилируется, хотя «ПроцедураА» и не вызывает процедуру «ПроцедураВ».

В Microsoft Access 95 при запуске процедуры в одном модуле загружались все модули в потенциальном дереве вызовов, хотя по умолчанию они не компилировались до вызова содержащейся в них процедуры. Microsoft Access 97 загружает модули только по необходимости, поэтому во многих случаях программы будут работать быстрее.

Объединение процедур в модули также приводит к повышению производительности за счет сокращения числа ненужных компиляций. Следует группировать в один модуль вызывающие и вызываемые процедуры, и избегать занесения в один модуль несвязанных процедур.

Определение ссылок на проект Visual Basic в другой базе данных Microsoft Access

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acconSetReferenceToDatabaseC"}

Каждая база данных Microsoft Access включает проект Visual Basic. Проект Visual Basic является набором всех модулей в проекте, включая стандартные модули и модули классов. Каждая база данных Microsoft Access, библиотечная база данных или надстройка, содержащаяся в файле .mde, включает проект Visual Basic.

Имя базы данных и имя проекта могут не совпадать. Имя базы данных определяется именем файла .mdb (или .mda или .mde), а имя проекта - параметром **Имя проекта** во вкладке **Другие** диалогового окна **Параметры**, которое выводится по команде **Параметры** из меню **Сервис**. При создании базы данных ее имя и имя проекта по умолчанию совпадают. Однако при переименовании базы данных имя проекта не изменяется автоматически. Аналогично изменение имени проекта не влияет на имя базы данных.

Ссылка устанавливается из проекта Visual Basic в одной базе данных Microsoft Access на проект в другой базе данных, библиотечной базе данных, или надстройке, содержащейся в файле .mde. После установки ссылки становится возможен запуск процедур Visual Basic из адресуемого проекта. Например, демонстрационная база данных «Борей» включает модуль «Служебные функции», в котором содержится функция IsLoaded. Можно создать ссылку на проект базы «Борей» из проекта текущей базы данных, после чего вызывать функцию IsLoaded так же, как если бы она была описана в текущей базе данных.

Для создания ссылки на проект «Борей» из другого проекта:

1. Откройте окно модуля.
2. Выберите в меню **Сервис** команду **Ссылки** и нажмите кнопку **Обзор** в диалоговом окне **Ссылки**.
3. Выберите элемент с расширением имени, соответствующим базе данных (*.mdb, *.mda, *.mde) в поле со списком **Тип файла**.
4. Найдите файл Борей.mdb. Если этот файл был установлен, он по умолчанию должен находиться в каталоге \Program Files\Microsoft Office\Office\Samples или в том каталоге, куда он был перемещен после этого.
5. Нажмите кнопку **OK**.

Файл Борей.mdb будет добавлен в список ссылок, выводящихся в диалоговом окне **Ссылки**.

Примечания

- Устанавливайте ссылки на проект в другой базе данных Microsoft Access, если требуется вызвать общую процедуру, описанную в стандартном модуле этой базы данных. Не допускается вызов процедур, описанных в модулях форм или отчетов, а также процедур, описанных с ключевым словом **Private.**
- Ссылку на проект в базе данных можно установить из другой базы данных Microsoft Access.
- Ссылки на конкретный проект базы данных Microsoft Access 97 допускаются только в других базах данных Microsoft Access. Для того, чтобы установить ссылку на проект в базе данных, созданной в предыдущей версии Microsoft Access, необходимо предварительно преобразовать эту базу к формату Microsoft Access 97.
- При установке ссылки на проект или библиотеку типов из Microsoft Access и последующем переносе файла, который содержит этот проект или библиотеку, в другую папку Microsoft Access попытается найти файл и переопределить ссылку. Если в системном реестре присутствует запись **RefLibPaths**, то поиск начнется с этого пути. Если такой записи в реестре нет, то поиск будет проводиться сначала в текущей папке, а затем во всех папках на диске. Запись **RefLibPaths** создается с помощью редактора реестра Windows в разделе \ HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\8.0\Access. Дополнительные сведения по использованию редактора системного реестра см. в документации Windows.

Программирование панелей инструментов и строк меню

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acconProgrammingToolbarsC;ofhowOverviewofCommandBars;ofhowUsingCommandBars"}

Microsoft Access 97 включает панели команд, которые являются программируемыми панелями инструментов, и строки меню. С помощью панелей команд создаются специальные панели инструментов и меню для конкретного приложения.

Для программирования с использованием панелей команд необходимо установить ссылку на библиотеку объектов Microsoft Office 8.0. Выберите в режиме конструктора команду **Ссылки** в меню **Сервис** и установите флажок **Microsoft Office 8.0 Object Library**.

Семейство **CommandBars** включает все панели команд, существующие в приложении. Для добавления нового объекта **CommandBar** в семейство **CommandBars** используется метод **Add** этого семейства. Например, следующая программа создает новую панель команд с именем «МояПанельКоманд». Для того, чтобы вывести эту панель на экран следует задать значение **True** (–1) для свойства **Вывод на экран (Visible)**.

```
Dim cmb As CommandBar
Set cmb = Application.CommandBars.Add("МояПанельКоманд")
cmb.Visible = True
```

В каждый объект **CommandBar** входит семейство **CommandBarControls**, которое содержит все элементы управления панели команд. Элементы управления панели команд отличаются от элементов управления формы. Для панели управления возможно создание различных типов элементов управления, включая кнопки, поля со списком и всплывающие меню. Комбинации этих элементов используются для создания специальных панелей инструментов или строк меню.

Для ссылок на семейство **CommandBarControls** используется свойство **Controls** объекта **CommandBar**. Для добавления элемента управления в панель команд применяется метод **Add** семейства **CommandBarControls**, определяющий тип создаваемого элемента. Следующий пример демонстрирует добавление новой кнопки в панель команд, созданную в предыдущем примере:

```
Dim cbc As CommandBarControl
Set cbc = cmb.Controls.Add(msoControlButton)
cbc.Caption = "Кнопка1"
cbc.Style = msoButtonCaption
```

Можно определить выражение для оценки или макрос для запуска, который будет выполняться при выборе пользователем элемента управления панели команд. В свойстве **Действие (OnAction)** следует указать имя макроса или выражение содержащее встроенную или определяемую пользователем функцию. Например, следующая строка программы задает в качестве значения свойства **Действие (OnAction)** объекта **CommandBarControl** выражение, содержащее функцию **MsgBox**. В этом примере используются объекты **CommandBar** и **CommandBarControl**, созданные в двух предыдущих примерах:

```
CommandBars("МояПанельКоманд").Controls("Кнопка1").OnAction = _
   "=MsgBox(""УХ!"")"
```

Примечание. В отличие от большей части остальных семейств Microsoft Access, семейство **CommandBars** и все семейства, которые в нем содержатся, индексированы начиная с 1, а не с 0.

Создание частичных реплик

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acconPartialReplicationC;daidxWhatsNewInDAO"}

Частичные реплики

Частичными называются реплики, которые содержат только некоторое подмножество записей полной реплики. Для создания частичной реплики определяется фильтр, ограничивающий синхронизированные данные подмножеством полной базы данных. С помощью частичных реплик осуществляется синхронизация реплик только с необходимыми данными, а не с полной базой.

Например, в приложении для бизнеса в головной фирме может сохранятся полная база данных по продажам, а копирование будет осуществляться только по региональным данным в офисы на местах. Для каждого регионального офиса может быть создана отдельная реплика, которая содержит данные только по этому региону. База данных в головном офисе будет в таком случае полной репликой, с которой синхронизированы все частичные реплики.

Преимущества применения частичных реплик

Частичные реплики уменьшают размер базы данных, так как возникает необходимость копирования только часто используемых фрагментов базы. Синхронизация частичной реплики производится быстрее, а сама реплика занимает меньше места на диске.

Частичная реплика может использоваться для ограничения доступа к данным. В случае с базой данных по продажам использование частичной реплики гарантирует, что работники региональных офисов не будут просматривать данные по всем офисам. Хотя частичные реплики и могут использоваться для ограничения доступа, они не подходят для надежной системы безопасности.

Частичные реплики также имеют преимущества при копировании данных через локальные (LAN) или глобальные (WAN) сети. Ограничение данных дает сокращение объема информации, передаваемой по сети. Это может снизить интенсивность передачи по сети и затраты на передачу.

Создание частичной реплики с объектом доступа к данным (DAO)

Для создания частичной реплики:

1. Используйте метод **MakeReplica** объекта **Database**, задав константу **dbRepMakePartial** в качестве аргумента *параметры*. Частичная реплика будет создана.
2. Установите необходимые фильтры и связи, определяющие какие данные будут копироваться из полной реплики. Для этого задайте соответствующие значения свойств **ReplicaFilter** и **PartialReplica.**
3. Используйте метод **PopulatePartial** для передачи всех записей, отвечающих новому условию фильтра реплики, из полной реплики.

При изменении данных или условия фильтра следует совместно использовать методы **Synchronize** и **PopulatePartial,** чтобы убедиться, что все данные переданы полной реплике, и что частичная реплика повторно заполнена на основе текущего условия фильтра.

Примечание. Созданная частичная реплика не может быть преобразована в полную реплику. Если удалить все фильтры и связи в частичной реплике, то она будет содержать все записи полной реплики, но у нее все еще будут те же ограничения частичной реплики.

Частичные реплики и целостность данных

В реплике, имеющей связь, которая усиливает целостность данных, многие из таблиц базы данных могут быть связаны с таблицей, на которой основан фильтр. При создании частичной реплики следует убедиться, что в нее включены все таблицы, связанные с таблицей,

содержащей необходимые данные.

Например, в демонстрационной базе данных «Борей», на таблицу «Сотрудники» может быть наложен фильтр [КодСотрудника]=1, который осуществляет синхронизацию только одной записи. Так как таблица «Сотрудники» связана с таблицами «Заказы» и «Заказано», требуется включить эти таблицы. Это необходимо сделать для того, чтобы в частичную реплику вошли соответствующие записи по сотрудникам из таблиц «Заказы» и «Заказано».

Может потребоваться включение других таблиц, которые связаны с фильтром только косвенным образом. Предположим, например, что необходимо, чтобы частичная реплика содержала список всех новых товаров. Таким образом новые заказы смогут включать любой доступный товар. В этом случае следует включить таблицу «Товары» в частичную реплику.

Необходимо также учитывать таблицы базы данных, которые не имеют связей с наложенным условием обеспечения целостности данных, такие как таблицы подстановок или таблицы со связями, где целостность данных не обеспечивается. Например, таблица, в которую входит поле со списком, не может участвовать ни в каких связях. Если такая таблица выбрана для включения, то все записи будут восстановлены, иначе ни одна запись восстановлена не будет.

Дополнительные сведения по частичным репликам и способам их создания см. в главе 20 «Репликация баз данных» книги *Разработка приложений для Microsoft Access 97*, а также в Web в разделе «Technical Information» на узле Microsoft Access Developer Forum по адресу http://www.microsoft.com/accessdev/.

Новые возможности Интернета для разработчиков

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acconInternetFeaturesC"}

Гиперссылки как средство связи приложений с сетью Интернет

В Microsoft Access 97 включены гиперссылки для взаимодействия приложений с Интернетом или с корпоративной сетью. С помощью гиперссылки может осуществляться переход по адресу в сети Интернет или в корпоративной сети к объекту базы данных или к документу на данном компьютере или на компьютере, подключенном к сети.

В Microsoft Access вошел новый тип данных гиперссылка, который может содержать адрес гиперссылки. Допускается описание поля таблицы с этим типом данных для хранения гиперссылок в качестве данных таблицы.

Предусмотрено также включение гиперссылок в формы и отчеты. Гиперссылки могут содержаться в элементах управления следующих трех типов: кнопка, надпись и рисунок. Каждый из этих элементов имеет свойство **Гиперссылка (Hyperlink)**, возвращающее объект **Hyperlink**, представляющий гиперссылку, которая содержится в данном элементе управления. Каждый элемент управления имеет также свойства **Адрес гиперссылки (HyperlinkAddress)** и **Дополнительный адрес (HyperlinkSubAddress)**, которые используются для установки и возврата адреса или дополнительного адреса гиперссылки.

Выбор элемента, содержащего гиперссылку, приводит к переходу по гиперссылке и запускает событие элемента управления **Нажатие кнопки (Click)**. При переходе по гиперссылке открываются документы или объекты, заданные свойствами **Адрес гиперссылки (HyperlinkAddress)** и **Дополнительный адрес (HyperlinkSubAddress)**. Например, если в свойстве **Адрес гиперссылки (HyperlinkAddress)** содержится ссылка на форму базы данных Microsoft Access, то эта форма открывается. Если это свойство указывает на документ Microsoft Word на другом компьютере, доступ к которому осуществляется по сети, то запускается Microsoft Word и открывается соответствующий документ. Если записана ссылка на адрес Web, то средство просмотра Web (например, Microsoft Internet Explorer) открывает и выводит на экран странницу Web.

Предусмотрена эмуляция выбора гиперссылки из Visual Basic с помощью метода **Follow** объекта **Hyperlink**, или метода **FollowHyperlink** объекта **Application**. Например, можно включить программу, которая осуществляет переход по заданной ссылке при открытии формы, в событие формы **Загрузка (Load)**.

Просмотр страниц Web с помощью элемента управления «Средство просмотра Web Microsoft»

Элемент управления «Средство просмотра Web Microsoft» (WebBrowser) является элементом ActiveX, который позволяет просматривать страницы Web и другие документы в сети Интернет или в корпоративной сети из формы Microsoft Access. Средство просмотра Web поставляется вместе с Microsoft Internet Explorer 3.0, который включен в Microsoft Office 97. Microsoft Internet Explorer можно бесплатно скопировать с узла Web корпорации Microsoft (http://www.microsoft.com/). Документация по элементу WebBrowser доступна по адресу http://www.microsoft.com/intdev/sdk/docs/iexplore/. Пользователи, устанавливающие Microsoft Office 97 с компакт-диска, могут просмотреть файл справки, содержащий эти сведения, скопировав его из пакета Office 97 ValuPack. Более подробно см. в разделе Пакет Office 97 ValuPack.

Элемент управления «Средство просмотра Web Microsoft» автоматически регистрируется операционной системой при установке Internet Explorer. Таким образом он может быть вызван из Microsoft Access без предварительной регистрации. Для добавления элемента «Средство просмотра Web Microsoft» в форму выберите команду **Элемент ActiveX** из меню **Вставка**, а затем выберите **Элемент средства просмотра WebMicrosoft** из списка элементов ActiveX.

После добавления элемента управления «Средство просмотра Web Microsoft» в форму для

открытия страницы Web в окне средства просмотра применяется метод **Navigate**. Например, если в форму добавлен элемент управления с именем «ActiveXCtl0», то возможно создание следующей процедуры обработки события формы **Загрузка (Load)**.

```
Private Sub Form_Load
   Me!ActiveXCtl0.Navigate "http://www.microsoft.com/"
End Sub
```

Сохранение данных в формате HTML

Существует несколько способов записи объектов Microsoft Access в формате HTML. Дополнительную информацию по этому вопросу см. в разделе Экспорт таблиц, запросов, форм и отчетов в формат HTML.

Импорт или присоединение данных из таблиц в формате HTML

Данные в формате HTML таблицы могут быть импортированы или присоединены к базе данных Microsoft Access. Дополнительную информацию по этому вопросу см. в разделе Импорт и присоединение таблиц HTML и списков.

Синхронизация реплик базы данных на сервере Интернета

Для синхронизации реплик базы данных по сети Интернет необходимо установить переключатель **dbRepSyncInternet** при вызове метода **Synchronize**.

Дополнительные сведения по работе в Microsoft Access с Интернетом или внутренней сетью содержатся в главе 21 «Разработка приложений для Интернета и Web» книги *Разработка приложений для Microsoft Access 97*. Информация по работе с элементами ActiveX содержится в главе 16 «Использование элементов ActiveX» книги *Разработка приложений*.

Использование технологии ODBCDirect в Microsoft Access

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acconUsingODBCDirectC;daconMSJetDatabaseEngine35;dahowDataAccessOverview;daidxMethodsReferenceODBC;daidxPropertiesReferenceODBC;daidxWhatsNewInDAO"}

ODBCDirect

ODBCDirect является технологией, позволяющей работать с серверами баз данных ODBC без загрузки ядра базы данных Microsoft Jet. ODBCDirect опирается на объектную модель «Microsoft DAO 3.5», что позволяет воспользоваться преимуществами этой технологии с внесением минимальных изменений в существующие программы для объектов доступа к данным (DAO). Библиотека Microsoft DAO 3.5 включает новые объекты, методы и свойства для поддержки ODBCDirect.

Преимущества использования ODBCDirect при доступе к данным ODBC

ODBCDirect представляет следующие преимущества при проведении операций ODBC.

- Использование ODBCDirect может улучшить производительность и эффективность выполнения программ за счет непосредственного доступа к источникам данных ODBC. ODBCDirect требует меньше ресурсов со стороны клиента, так как не требуется загрузка ядра базы данных Microsoft Jet. Сервер ODBC выполняет всю обработку запросов.
- Технология ODBCDirect предоставляет расширенные возможности по работе сервера по сравнению с использованием ODBC через Microsoft Jet. Например, для серверов, которые поддерживают управление указателями наборов записей, ODBCDirect позволяет определить находится ли указатель наборов записей на сервере или на локальной машине. Кроме того для взаимодействия с хранимыми на уровне сервера процедурами могут быть заданы значения ввода и проверены возвращаемые значения. Такие действия невозможно провести при использовании Microsoft Jet.
- ODBCDirect поддерживает асинхронные запросы. При запуске запроса необязательно дожидаться его завершения для того, чтобы выполнить следующую операцию. Для отслеживания работы запросов проверяется свойство **StillExecuting**.
- Технология ODBCDirect поддерживает пакетное изменение. Это позволяет кэшировать изменения объекта **Recordset** на месте, а затем передавать серверу все изменения в едином пакете.
- С помощью ODBCDirect становится возможным создание простых результирующих наборов записей или более сложных указателей наборов записей. Допускается запуск запросов, которые возвращают любое количество результирующих наборов записей. Предусмотрена возможность ограничения количества возвращаемых строк и просмотра всех сообщений и ошибок, образованных удаленным источником данных, без влияния на производительность выполнения запроса.

Создание рабочей области ODBCDirect

Для создания рабочей области ODBCDirect следует задать константу **dbUseODBC** в качестве аргумента *тип* для метода **CreateWorkspace**. Установка свойства **DefaultType** объекта **DBEngine** приводит к созданию рабочей области ODBCDirect по умолчанию.

После создания рабочей области ODBCDirect становится возможным использование особых объектов доступа к данным (DAO), свойств и методов для работы с данными на сервере базы данных ODBC.

Преимущества использования Microsoft Jet при доступе к данным ODBC

Рабочие области Microsoft Jet и ODBCDirect обеспечивают различные, но дополняющие друг друга функциональные возможности. Рабочую область Microsoft Jet следует использовать для доступа к файлам .mdb и данным в формате ISAM, таким как текст и электронные таблицы.

Microsoft Jet предоставляет следующие уникальные возможности, недоступные через ODBCDirect:

- **Обновляемые объединения**. Рабочую область Microsoft Jet следует использовать для изменения данных в объектах **Recordset**, основанных на объединении нескольких таблиц.
- **Неоднородные объединения**. Рабочая область Microsoft Jet необходима для проведения объединения таблиц из различных источников данных.
- **Операции языка определения данных (DDL)**. Рабочую область Microsoft Jet следует использовать для проведения операций языка определения данных (DDL) с объектами доступа к данным (DAO). ODBCDirect не поддерживает объект **TableDef**, таким образом нельзя создавать и изменять таблицы, используя объекты DAO. Однако с помощью ODBCDirect возможно выполнение операций языка определения данных для запуска инструкций SQL из этой библиотеки.
- **Привязка форм и элементов управления**. Если необходимо, чтобы формы или элементы управления приложения были связаны с данными из источника данных ODBC, то следует использовать Microsoft Jet. Данные, к которым осуществляется доступ с помощью рабочей области ODBCDirect, не могут быть привязаны к формам или элементам управления.

Если перечисленные выше возможности не требуются, то лучше использовать рабочую область ODBCDirect.

Примечание. Предусмотрено совместное использование рабочих областей Microsoft Jet и ODBCDirect в приложении. При этом объединение может проводиться любым образом. Например, в одной функции возможно определение рабочей области Microsoft Jet для проведения операций библиотеки динамической компоновки (DDL) с использованием объекта доступа к данным (DAO) и определение рабочей области ODBCDirect для выполнения асинхронных запросов.

Новые возможности для разработчиков: элемент управления «Набор вкладок»

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acconTabControlC"}

В Microsoft Access 97 включен новый элемент управления «Набор вкладок», который предназначен для объединения набора элементов управления в единый элемент. Например, возможно создание диалогового окна с вкладками, в котором различные параметры объединены на различных вкладках.

Отдельные вкладки набора вкладок связаны с объектами **Page** семейства **Pages**. Каждый объект **Page** имеет собственное семейство **Controls**, которое содержит объекты **Control**, присутствующие на этой вкладке.

Для добавления новой вкладки в набор используется метод **Add** семейства **Pages**. Для удаления вкладки используется метод **Remove**.

Свойство **Значение (Value)** набора вкладок указывает на текущую вкладку. Оно возвращает целое число, которое является индексом выбранной вкладки в семействе **Pages**. Например, если выбрана первая вкладка набора вкладок, то свойство **Значение (Value)** возвращает 0 (индекс первой вкладки в семействе **Pages**). Если выбрана вторая вкладка, то свойство **Значение (Value)** возвращает 1 и т.д. Позиция вкладки в семействе связана со значением свойства **PageIndex** вкладки.

Перехват ошибок

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"achowErrorTrappingC;dacolError;daidxRealDataAccessErrors;daobjError;vafctIsError;vamsgTrappableErrors;vamthRaise;vaobjErr;vaproDescription;vaproNumber;vastmErr;vastmExit;vastmOnError;vastmResume"}

Инструкция **On Error GoTo** используется для перехвата ошибок и передачи управления в подпрограмму обработки ошибок внутри процедуры. Например, следующая инструкция передает управление на метку «ОбработкаОшибок:».

```
On Error GoTo ОбработкаОшибок
```

Во избежание конфликтов с другими компонентами процедуры каждая метка подпрограммы обработки ошибок в процедуре должна иметь уникальное имя. Для того, чтобы исключить обработку ошибок в тех случаях, когда ошибки не возникают, в теле процедуры перед меткой подпрограммы обработки ошибок необходимо поместить инструкцию **Exit Sub** или **Exit Function**.

```
Sub CausesAnError()
   ' Нормальное выполнение процедуры.
   On Error GoTo ОбработкаОшибок
   ' Создает ошибку деления на нуль.
   Err.Raise 11
   Exit Sub

ОбработкаОшибок:
   ' Выводит сообщение об ошибке.
   MsgBox "Ошибка номер " & Err.Number & ": " & Err.Description
   ' Возобновляет выполнение с инструкции, следующей
   ' за инструкцией, в которой возникла ошибка.
   Resume Next
End Sub
```

Метод **Raise** объекта **Err** создает указанную ошибку. Свойство **Number** объекта **Err** возвращает код, соответствующий последней возникшей ошибке выполнения; свойство **Description** возвращает текст сообщения об ошибке для данной ошибки.

Примечания

- В версиях Microsoft Access 1.*x* и 2.0 для генерации ошибки использовалась инструкция **Error**, функция **Err** возвращала код ошибки, а функция **Error** описание ошибки. Существующие программы обработки ошибок, использующие инструкцию **Error** и функцию **Error**, по-прежнему будут выполняться. Однако в новых программах рекомендуется использовать объект **Err** и его методы и свойства.
- В версиях Microsoft Access 1.*x* и 2.0 все ошибки программирования объектов рассматривались как одна ошибка. В отличие от этого, теперь приложение, являющееся компонентом ActiveX, при возникновении в нем ошибки возвращает такое сообщение об ошибке, которое получил бы пользователь при работе с этим приложением. В связи с этим для корректной обработки новых ошибок программирования объектов, возможно, потребуется изменение существующих подпрограмм обработки ошибок.
- Если требуется получить строку, содержащую описание ошибки Microsoft Access или ошибки объектов доступа к данным (DAO), при отсутствии реальной ошибки в программе, следует использовать метод **AccessError**, возвращающий такую строку.

Более подробное описание способов перехвата ошибок и написания подпрограмм обработки ошибок содержится в главе 8 -- «Обработка ошибок выполнения» книги *Разработка приложений Microsoft Access 97*.

Вызов процедур в подчиненных формах и отчетах

{ewc HLP95EN.DLL,DYNALINK,"пїSпїS. пїSпїSпїSпїSпїS":"achowCallingProceduresInASubformOrSubreportC"}

Имеется два способа вызова процедуры в модуле, связанном с подчиненной формой или подчиненным отчетом. Если форма, содержащая подчиненную форму, открыта в режиме формы, то процедура может рассматриваться как метод подчиненной формы. В следующем примере демонстрируется вызов процедуры ListNames в подчиненной форме «Новые заказы», присоединенной к элементу управления-подчиненной форме в форме «Заказы».

```
Forms!Заказы![Новые заказы].Form.ListNames
```

Второй способ заключается в создании нового экземпляра формы, которая используется в качестве подчиненной формы, (даже когда ни одна из форм, ни главная, ни подчиненная, не открыта) и в последующем вызове процедуры. Это возможно для любой формы, вне зависимости от ее использования в качестве подчиненной. В следующем примере демонстрируется создание копии формы «Новые заказы» и вызов процедуры ListNames.

```
Dim frm As New [Form_Новые заказы]
frm.ListNames
```

Примечание. При создании новой копии формы с именем, состоящим из нескольких слов, следует заключать имя класса этой формы в прямые скобки, как показано в предыдущем примере.

Элементы обработки ошибок выполнения

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS.
nїSnїSnїSnїSnїS":"achowRuntimeErrorHandlingC;daproDescription;daproNumber;vamthClear;vamthRaise;vaproDescription;vaproNumber;vastmError;vastmExit;vastmResume"}

Ошибки и их обработка

При создании приложения необходимо учитывать возможные последствия возникновения ошибок. Ошибки в приложении возникают по одной из следующих причин. Во-первых, во время работы приложения некоторые условия могут выйти за допустимые пределы. Например, при попытке открыть таблицу, которая была удалена пользователем, возникнет ошибка. Во-вторых, программа может быть построена неверно с точки зрения логики. Это приводит к неожиданным результатам. Примером такой ошибки служит деление на нуль.

Если механизм обработки ошибок не применяется, то при возникновении ошибки Visual Basic прервет выполнение программы и выведет сообщение об ошибке. Такой результат работы неприемлем для пользователя приложения. Для решения этой и других проблем следует включать в программу процедуры обработки для всех ошибок.

При добавлении процедуры обработки следует учитывать способ передачи управления процедуре при возникновении ошибки. Первым этапом задания маршрута передачи управления является подключение обработчика ошибок путем включения некоторой формы инструкции **On Error** в процедуру. Инструкция **On Error** передает управление процедуре обработки события данной ошибки. Если инструкция **On Error** отсутствует, то при возникновении ошибки Visual Basic прерывает выполнение программы и выводит сообщение об ошибке.

При возникновении ошибки в процедуре с подключенным обработчиком ошибок Visual Basic не выводит обычного сообщения об ошибке. Вместо этого управление передается в обработчик (если он присутствует). После передачи управления обработчику он становится активным. В активном обработчике ошибок может определяться тип ошибки и осуществляться произвольная обработка. В Microsoft Access входят два объекта, содержащие информацию по возникшим ошибкам: объект Visual Basic **Err** и объект доступа к данным (DAO) **Error**.

Дополнительные сведения по обработке ошибок см. в главе 8 «Обработка ошибок при выполнении программ» книги Разработка приложений для Microsoft Access 97.

Передача управления при возникновении ошибки

Обработчик ошибок определяет действия при возникновении ошибки в процедуре. Например, может возникнуть необходимость завершения выполнения процедуры при возникновении определенной ошибки или исправления условий и повторного запуска. Инструкции **On Error** и **Resume** определяют способ передачи управления при возникновении ошибки.

Инструкция On Error

Инструкция **On Error** служит для подключения и отключения процедуры обработки ошибок. Если такая процедура подключена, то при возникновении ошибки ей передается управление.

Есть три формы инструкции **On Error**: **On Error GoTo** *метка*, **On Error GoTo 0** и **On Error Resume Next**. Инструкция **On Error GoTo** *метка* подключает процедуру обработки ошибок, начиная с той строки, на которой она находится. Подключить обработчик следует перед первой строкой, которая может привести к возникновению ошибки. Если обработчик активен, то при возникновении ошибки управление передается строке, заданной в аргументе *метка*.

Строка, заданная в качестве аргумента *метка* должна быть первой строкой процедуры обработки ошибок. Например, следующая процедура определяет, что при возникновении ошибки управление передается на строку с меткой «ОбработкаОшибок»:

```
Function MayCauseAnError()
```

```
    ' Подключение обработчика ошибок.
    On Error GoTo ОбработкаОшибок
    .         ' Содержит программу, которая может вызвать ошибку.
    .
    .

ОбработкаОшибок:
    .         ' Содержит программу обработки ошибок.
    .
    .
End Function
```

Инструкция **On Error GoTo 0** отключает обработку ошибок внутри процедуры. Эта инструкция не задает строку 0 в качестве начала программы обработки ошибок, даже если существует строка с номером 0. Если инструкции **On Error GoTo 0** в программе нет, то обработка ошибок отключается автоматически при завершении выполнения процедуры. Инструкция **On Error GoTo 0** сбрасывает свойства объекта **Err** аналогично методу **Clear** этого объекта.

Инструкция **On Error Resume Next** приводит к игнорированию строки, вызвавшей ошибку, и к передаче управления следующей строке. Выполнение не прерывается. Полезно использовать инструкцию **On Error Resume Next** для проверки свойств объекта **Err** сразу за строкой, в которой ожидается возникновение ошибки, и для обработки ошибки внутри самой процедуры, а не в обработчике.

Инструкция Resume

Инструкция **Resume** передает управление обратно в процедуру из обработчика ошибок. Эту инструкцию следует включать в обработчик, если требуется передача управления в определенную точку процедуры. Однако инструкция **Resume** не является обязательной. После завершения работы обработчика процедура может быть также завершена.

Существует три формы инструкции **Resume**. Инструкции **Resume** или **Resume 0** возвращают управление строке, при выполнении которой произошла ошибка. Инструкция **Resume Next** возвращает управление строке, непосредственно следующей за строкой, вызвавшей ошибку. Инструкция **Resume** *метка* передает управление строке, заданной в качестве аргумента *метка*. Аргумент *метка* может указывать номер строки или метку.

Инструкции **Resume** и **Resume 0** обычно используются, когда необходимо, чтобы пользователь внес исправления. Например, у пользователя запрашивается имя таблицы для открытия, а он вводит имя несуществующей таблицы. В этом случае запрос можно повторить и продолжить выполнение программы с инструкции, вызвавшей ошибку.

Инструкция **Resume Next** используется для обработки ошибок внутри обработчика, без повторного выполнения команды, вызвавшей ошибку, при возврате управления. Для передачи управления в другую точку процедуры применяется инструкция **Resume** *метка*. Например, управление может быть передано процедуре выхода, как описано в следующем разделе.

Выход из процедуры

При включении процедуры обработки ошибок необходимо также включить процедуру выхода таким образом, чтобы обработчик выполнялся только при возникновении ошибки. Процедура выхода отмечается меткой строки точно также как и процедура обработки ошибок.

Например, процедуру выхода можно добавить в пример из предыдущего раздела. Если ошибка не возникнет, то после выполнения тела процедуры будет запущена процедура выхода. Если же ошибка произойдет, то управление будет передано в процедуру выхода после выполнения процедуры обработки ошибок. Процедура выхода содержит инструкцию **Exit.**

```
Function MayCauseAnError()
    ' Подключение обработчика ошибок.
```

```
   On Error GoTo ОбработкаОшибок
   .        ' Содержит программу, которая может вызвать ошибку.
   .
   .

Exit_MayCauseAnError:
   Exit Function

ОбработкаОшибок:
   .        ' Содержит программу обработки ошибок.
   .
   .
   ' Возвращает управление функции exit в процедуре выхода.
   Resume Exit_MayCauseAnError
End Function
```

Обработка ошибок во вложенных процедурах

При возникновении ошибки во вложенной процедуре, в которой нет подключенного обработчика ошибок, Visual Basic не просто останавливает выполнение программы, а осуществляет поиск подключенного обработчика по списку вызовов. Это позволяет исправлять ошибки в другой процедуре. Предположим, например, что «ПроцедураА» вызывает процедуру «ПроцедураБ», а «ПроцедураБ» вызывает процедуру «ПроцедураВ». Если ошибка произошла в процедуре «ПроцедураВ», и в ней не подключен обработчик ошибок, то Visual Basic проверит на наличие обработчика процедуру «ПроцедураБ», а затем и «ПроцедураА». Если подключенный обработчик будет найден, то ему будет передано управление. Если же нет, то программа будет остановлена и будет выведено сообщение об ошибке.

Visual Basic осуществляет поиск подключенного обработчика ошибок по списку вызовов при возникновении ошибки в активном обработчике ошибок. Метод **Raise** объекта **Err** вызывает проведение поиска подключенного обработчика ошибок по списку вызовов. Это бывает полезно для обработки непредвиденных ошибок внутри обработчика. Если такая ошибка возникла, то повторение ее в обработчике ошибок приведет к передаче управления вверх по списку вызовов для поиска другого обработчика, который мог быть настроен для обработки этой ошибки.

Предположим, например, что в процедуре «ПроцедураВ» имеется подключенный обработчик ошибок, но он не способен исправить возникшую ошибку. После проверки всех ожидаемых ошибок он повторяет исходную ошибку. Управление передается вверх по списку вызовов обработчику ошибок в процедуре «ПроцедураБ», если он существует. Если обработчик в процедуре «ПроцедураБ» отсутствует или он не может исправить ошибку и повторил ее снова, то управление передается обработчику ошибок процедуры «ПроцедураА», если он существует.

Для иллюстрации этого подхода предположим, что имеется вложенная процедура, которая включает обработчик ошибок для ожидаемой ошибки несовпадения типов. В некоторый момент в процедуре возникла неожиданная ошибка деления на нуль. Если в эту процедуру была включена инструкция для повторения исходной ошибки, то управление будет передано вверх по списку вызовов другому подключенному обработчику ошибок, если такой существует. Если исправление ошибки деления на нуль предусмотрено в другой процедуре из списка вызовов, то ошибка будет исправлена. Если в программе не предусмотрено повторение ошибок, то процедура продолжит работу без исправления ошибки деления на нуль. Это может привести к другим ошибкам в наборе вложенных процедур.

Итак Visual Basic осуществляет поиск подключенного обработчика ошибок вверх по списку вызовов в следующих случаях:

- Ошибка возникла в процедуре, не имеющей подключенного обработчика.
- Ошибка возникла в активном обработчике ошибок. Метод **Raise** объекта **Err** вызывает проведение поиска подключенного обработчика ошибок по списку вызовов.

Получение информации по ошибкам

После передачи управления процедуре обработки ошибок программа должна определить тип возникшей ошибки и адресовать ее. Visual Basic и Microsoft Access обеспечивают различные элементы языка, которые могут быть использованы для получения информации по определенной ошибке. Каждый из них применяется для ошибок различных типов. Так как ошибки могут произойти в различных частях приложения, элементы языка необходимо выбирать, основываясь на ожидаемых типах ошибок.

Для обработки ошибок предусмотрены следующие элементы языка:

- Объект **Err**.
- Объект **Error** и семейство **Errors**.
- Метод **AccessError**.
- Событие Error.

Объект Err

Объект **Err** является объектом Visual Basic. При возникновении ошибки Visual Basic информация по ней записывается в объект **Err**. Этот объект может содержать сведения только по одной ошибке. При возникновении новой ошибки информация объекта **Err** обновляется.

Свойства и методы объекта **Err** применяются для получения информации по отдельной ошибке. Свойство **Number** является стандартным свойством объекта **Err**; оно возвращает идентификационный номер возникшей ошибки. Свойство **Описание (Description)** объекта **Err** возвращает строку описания, связанную с ошибкой Visual Basic. Метод **Clear** удаляет информацию по текущей ошибке из объекта **Err**. Метод **Raise** генерирует особую ошибку и заполняет свойства объекта **Err** информацией о ней.

Следующий пример демонстрирует использование объекта **Err** в процедуре, которая может вызвать ошибку несовпадения типов:

```
Function MayCauseAnError()
   ' Описание константы для представления вероятной ошибки.
   Const conTypeMismatch As Integer = 13

   On Error GoTo ОбработкаОшибок
      .        ' Содержит программу, которая может вызвать ошибку.
      .
      .

Exit_MayCauseAnError:
   Exit Function

ОбработкаОшибок:
   ' Проверяет свойства объекта Err.
   If Err = conTypeMismatch Then
      .        ' Содержит программу обработки ошибок.
      .
      .
   Else
      ' Повторяет исходную ошибку.
      Dim intErrNum As Integer
      intErrNum = Err
      Err.Clear
      Err.Raise intErrNum
   End If
   ' Передает управление функции exit процедуры выхода.
   Resume Exit_MayCauseAnError
```

```
End Function
```

В предыдущем примере метод **Raise** использовался для повторения исходной ошибки. Если произойдет ошибка, отличная от ошибки несовпадения типов, то управление будет передано вверх по списку вызовов другому подключенному обработчику, если такой существует.

Объект **Err** предоставляет всю необходимую информацию по ошибкам Visual Basic. Однако он не дает полной информации по ошибкам Microsoft Access или ошибкам ядра базы данных Microsoft Jet. Microsoft Access и объекты доступа к данным (DAO) обеспечивают дополнительные элементы языка для работы с такими ошибками.

Объект Error и семейство Errors

Объект **Error** и семейство **Errors** предоставлены механизмом объектов доступа к данным (DAO). Объект **Error** представляет ошибку DAO. Отдельная операция доступа к данным (DAO) может привести к возникновению нескольких ошибок особенно, если это операция ODBC. Каждой ошибке, возникшей в результате выполнения отдельной операции доступа к данным назначается соответствующий объект **Error**. Все объекты **Error**, связанные с отдельной операцией DAO, содержатся в семействе **Errors**. Ошибка самого низкого уровня становится первым элементом семейства, а самого высокого уровня - последним.

При возникновении ошибки доступа к данным объект Visual Basic **Err** содержит номер ошибки для первого объекта в семействе **Errors**. Для определения возникновения дополнительных ошибок следует проверять семейство **Errors**. Значения свойств **Number** и **Описание (Description)** первого объекта **Error** семейства **Errors** должно соответствовать значениям свойств **Number** и **Описание (Description)** объекта Visual Basic **Err**.

Метод AccessError

Метод **Raise** объекта **Err** может применяться для генерации ошибки Visual Basic, которая на самом деле не имела место, и определения строки-описания, связанной с этой ошибкой. Однако этот метод неприменим для генерации ошибок Microsoft Access или ошибок доступа к данным (DAO). Для определения строки-описания, связанной с ошибкой Microsoft Access или DAO, которая не имела место, используется метод **AccessError**.

*Событие **Ошибка (Error**)*

Событие **Ошибка (Error)** используется для перехвата ошибок, возникших в форме или отчете Microsoft Access. Например, ввод пользователем текста в поле с типом данных «Date/Time» вызовет событие **Ошибка (Error)**. Если добавить процедуру обработки события **Ошибка (Error)** в форму «Сотрудники», а затем ввести текстовое значение в поле «ДатаНайма», то запустится процедура обработки события **Ошибка (Error)**.

Процедура обработки события **Ошибка (Error)** получает в качестве аргумента целое число, DataErr. При запуске этой процедуры аргумент DataErr содержит количество возникших ошибок Microsoft Access. Единственный способ определить количество ошибок - проверить этот аргумент. Объект **Err** не заполняется информацией об ошибке после возникновения события **Ошибка (Error)**. Совместное использование аргумента DataErr и метода **AccessError** может быть полезным для определения количества ошибок и их строк-описаний.

Примечание. Инструкция **Error** и функция **Error** поддерживаются только для совместимости с предыдущими версиями. При написании новых программ для получения информации по ошибке рекомендуется использовать объекты **Err** и **Error**, функцию **AccessError** и событие **Ошибка (Error)**.

Событие «Инициализация» (Initialize)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtInitializeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtInitializeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtInitializeMO"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïS":"acevtInitializeEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïSnïS":"acevtInitializeA"}

Событие **Инициализация (Initialize)** для модуля класса возникает при создании экземпляра этого модуля класса.

Примечание. Для запуска программы при возникновении события **Инициализация (Initialize)** необходимо поместить ее в процедуру обработки события **Инициализация (Initialize)** в модуле класса.

Примечания.

Событие **Инициализация** возникает в следующих случаях.

- При непосредственном создании нового экземпляра класса с помощью ключевого слова **New**.
- При косвенном создании экземпляра класса путем установки или возврата свойства или с помощью применения метода, определенного в модуле класса.

Событие «Инициализация» (Initialize) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtInitializeEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtInitializeEPX":1}

Для создания процедуры обработки события **Инициализация (Initialize)** следует открыть модуль класса и выбрать **Класс** в поле **Объект**. Затем выбрать **Initialize** в поле **Процедура**.

Синтаксис

Private Sub Class_Initialize( )

Примечания.

Эта процедура обработки события используется для запуска программы при создании экземпляра класса, или для инициализации любых данных, необходимых экземпляру класса.

Событие **Инициализация (Initialize)** возникает при создании нового экземпляра класса с помощью ключевого слова **New**. Предположим, что имеется класс с именем «СпециальныйОбъект». Инструкция описания может быть добавлена в процедуру стандартного модуля, которая создает новый экземпляр класса «СпециальныйОбъект,» как показано в следующем примере:

```
Dim obj As New СпециальныйОбъект
```

Запуск процедуры, которая содержит такое описание, приведет к созданию нового экземпляра класса «СпециальныйОбъект» и к возникновению события **Инициализация (Initialize)**.

Событие **Инициализация (Initialize)** также возникает при создании экземпляра класса путем установки или возвращения свойств или с помощью применения метода, определенного в модуле класса. Пусть, например, определена процедура типа **Function** с именем «ИменаСписков» в модуле класса «СпециальныйОбъект». Для запуска функции из другого модуля необходимо указать имя модуля класса. Например:

```
СпециальныйОбъект.ИменаСписков
```

Указание функции с именем модуля класса создает экземпляр модуля класса и приводит к возникновению события **Инициализация (Initialize)**.

Событие «Завершение» (Terminate)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtTerminateC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtTerminateX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtTerminateMO"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїS":"acevtTerminateEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtTerminateA"}

Событие **Завершение (Terminate)** для модуля класса возникает, когда все ссылки на экземпляр этого класса удалены из памяти.

Примечание. Для запуска программы при возникновении события **Завершение (Terminate)** необходимо поместить ее в процедуру обработки события **Завершение (Terminate)** в модуле класса.

Примечания.

Это событие происходит только при удалении из памяти какой-либо ссылки на экземпляр класса.

Событие «Завершение» (Terminate) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtTerminateEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtTerminateEPX":1}

Для создания процедуры обработки события **Завершение (Terminate)** следует открыть модуль класса и выбрать **Класс** в поле **Объект**, а затем выбрать **Terminate** в поле **Процедура**.

Синтаксис.

Private Sub Class_Terminate( )

Примечания.

Для удаления из памяти ссылки на экземпляр класса следует установить значение **Nothing** для переменной, содержащей эту ссылку. Предположим, например, что создан экземпляр класса «СпециальныйОбъект» и ссылка на него присвоена объектной переменной `obj`. При установке значения **Nothing** для переменной `obj` ссылка удаляется из памяти. Если из памяти удалены все ссылки на экземпляр класса, то возникнет событие **Завершение (Terminate)**.

Ссылка на экземпляр класса также будет удалена из памяти, когда объектная переменная, которой присвоена ссылка на этот экземпляр, выходит из области видимости. Например, если переменная определена в процедуре, то она выходит из области видимости при завершении процедуры. Если переменная определена на уровне модуля в стандартном модуле, то она выходит из области видимости при сбросе программы. Если все ссылки на экземпляр класса будут удалены подобным образом, то событие **Завершение (Terminate)** также будет иметь место.

Событие **Завершение (Terminate)** не возникает, если экземпляры класса удаляются из памяти при нормальном завершении работы Microsoft Access. Например, если в программе вызывается метод **Quit** объекта Microsoft Access **Application** перед удалением из памяти всех существующих экземпляров класса, то событие **Завершение (Terminate)** не возникнет для этого класса.

Событие «Применение фильтра» (ApplyFilter)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtApplyFilterC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtApplyFilterX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS":"acevtApplyFilterMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtApplyFilterEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtApplyFilterA"}

Событие **Применение фильтра (ApplyFilter)** возникает при выполнении пользователем одного из следующих действий.

- Выбор команды **Применить фильтр** меню **Записи** в режиме формы или выбор команды **Применить фильтр** меню **Фильтр** в окне фильтра или нажатие кнопки **Применить фильтр** ... на панели инструментов. При этом применяется последний фильтр (созданный с помощью средства обычный фильтр или в окне расширенного фильтра).
- Выбор команды **Фильтр по выделенному** в подменю **Фильтр** меню **Записи** в режиме формы или нажатие кнопки **Фильтр по выделенному** на панели инструментов. При этом применяется фильтр, основанный на текущих выделенных элементах в форме.
- Выбор команды **Исключить выделенное** в подменю **Фильтр** меню **Записи** в режиме формы. При этом применяется фильтр, исключающий текущие выделенные элементы в форме.
- Выбор команды **Удалить фильтр** меню **Записи** в режиме формы или нажатие кнопки **Удалить фильтр** ... на панели инструментов. При этом удаляется любой текущий фильтр (или условие сортировки), примененный к форме.
- Выбор команды **Фильтр по выделенному**, **Исключить выделенное** или **Удалить фильтр**, а также ввод значения или выражения в окне **Фильтр** контекстного меню в момент, когда фокус имеет присоединенный элемент управления.
- Закрытие окна расширенного фильтра или окна обычного фильтра.
- Выбор в меню **Фильтр** команды **Расширенный фильтр** при открытом окне обычного фильтра или команды **Изменить фильтр** при открытом окне расширенного фильтра. При этом, когда закрывается открытое окно фильтра, возникает событие **Применение фильтра (ApplyFilter)**, а затем, когда открывается окно другого фильтра, возникает событие **Фильтрация (Filter)**.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя макроса или элемент [Процедура обработки события] в качестве значения свойства **Применение фильтра (OnApplyFilter)**.

Событие **Применение фильтра (ApplyFilter)** используется для выполнения следующих действий.

- Проверка правильности примененного фильтра. Предположим, например, что требуется проверить, включено ли в фильтр, примененный к форме «Заказы», условие на поле «Размещен». Для этого необходимо проверить значение свойства **Фильтр (Filter)** формы и убедиться, что выражение, определяющее предложение WHERE, содержит данное условие.
- Изменение вида формы перед применением фильтра. Например, при применении фильтра некоторые поля могут стать ненужными. Такие поля отключают или делают скрытыми.
- Отмена или изменение действий, выполняемых в ответ на событие **Фильтрация (Filter)**. Например, при создании фильтра часто возникает необходимость отключить или скрыть некоторые элементы управления в форме, которые не включаются в условия фильтра. После применения фильтра эти элементы управления могут быть снова включены или выведены на экран.

Макрокоманды в макросе или процедуре обработки события **Применение фильтра (ApplyFilter)** выполняются перед применением или удалением фильтра; они также выполняются после закрытия окна расширенного фильтра или окна обычного фильтра, но до

обновления формы на экране. Условия, введенные в новый фильтр, становятся доступными для макроса или процедуры обработки события **Применение фильтра (ApplyFilter)** в качестве значения свойства **Фильтр (Filter)**.

Примечание. Событие **Применение фильтра (ApplyFilter)** не возникает при выполнении пользователем одного из следующих действий.

- Применение или удаление фильтра с помощью макрокоманд **ПрименитьФильтр (ApplyFilter)**, **ОткрытьФорму (OpenForm)** или **ПоказатьВсеЗаписи (ShowAllRecords)** в макросе или соответствующих им методов объекта **DoCmd** в программе Visual Basic.
- Закрытие окна расширенного фильтра или окна обычного фильтра с помощью макрокоманды **Закрыть (Close)** или метода **Close** объекта **DoCmd**.
- Задание значений свойств **Фильтр (Filter)** или **Фильтр включен (FilterOn)** в макросе или в программе Visual Basic (хотя задание значений этих свойств в макросе или в процедуре обработки события **Применение фильтра (ApplyFilter)** допускается).

Событие «Применение фильтра» (ApplyFilter) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtApplyFilterMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtApplyFilterMOX":1}

Чтобы запустить макрос при возникновении события **Применение фильтра (ApplyFilter),** следует указать имя макроса в качестве значения свойства **Применение фильтра (OnApplyFilter)**.

Дополнительные сведения

Для проверки фильтра или для изменения введенных условий следует проверить значение свойства **Фильтр (Filter)** в макросе или задать новое значение данного свойства с помощью макрокоманды **ЗадатьЗначение (SetValue)** в макросе, связанном с событием **Применение фильтра (ApplyFilter)**.

Макрокоманда **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Применение фильтра (ApplyFilter)**, позволяет отменить применение фильтра. Существует возможность создать условие, в котором проверяются значения одного или нескольких элементов управления в форме, и применить фильтр только при выполнении этого условия. Например, если в форме «Заказы» установлен флажок **КрупныеЗаказы**, то можно применить фильтр, отбирающий крупные заказы из всех записей формы. В противном случае следует отменить применение фильтра и вывести сообщение о том, что крупных заказов нет.

Событие «Применение фильтра» (ApplyFilter) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtApplyFilterEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtApplyFilterEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Применение фильтра (ApplyFilter)**, следует выбрать в качестве значения свойства **Применение фильтра (OnApplyFilter)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_ApplyFilter(Cancel **As Integer,** ApplyType **As Integer)**

Процедура обработки события **Применение фильтра (ApplyFilter)** использует следующие аргументы.

Аргумент	Описание
Cancel	Определяет применение фильтра. Заданное для аргумента «Cancel» значение **True** (–1) отменяет применение фильтра. Для отмены применения фильтра используется также метод **CancelEvent** объекта **DoCmd**.
ApplyType	Действие, вызывающее возникновение события **Применение фильтра (ApplyFilter)**. Значение аргумента «ApplyType» задается с помощью одной из следующих встроенных констант:

Константа	Значение
acShowAllRecords	0
acApplyFilter	1
acCloseFilterWindow	2

Дополнительные сведения

Процедура обработки события **Применение фильтра (ApplyFilter)** используется для вывода на экран или скрытия, а также для включения или отключения определенных элементов управления в форме при применении или удалении фильтра. Например, когда к форме «Заказы» применяется фильтр, отбирающий только оплаченные заказы, можно сделать скрытыми поля «Сумма», «Налог» и «Итого», если аргумент «ApplyType» имеет значение **acApplyFilter**, и снова вывести их на экран, если «ApplyType» имеет значение **acShowAllRecords**.

Допускается также использование процедуры обработки события **Применение фильтра (ApplyFilter)** для отмены действий, выполняемых при возникновении события **Фильтрация (Filter)**. Это особенно полезно, если пользователь, не создав новый фильтр, закрывает окно фильтра при значении аргумента «ApplyType» **acCloseFilterWindow**.

Событие «Применение фильтра» (ApplyFilter) - Пример процедуры обработки события

В следующем примере демонстрируется скрытие полей «Сумма», «Налог» и «Итого» в форме «Заказы» при применении фильтра, отбирающего только записи, содержащие оплаченные заказы.

Для проверки данного примера следует включить процедуру обработки события в форму «Заказы», содержащую элементы управления «Сумма», «Налог» и «Итого», а затем применить фильтр, отбирающий только оплаченные заказы.

```
Private Sub Form_ApplyFilter(Cancel As Integer, ApplyType As Integer)
   If Not IsNull(Me.Filter) And (InStr(Me.Filter, "Заказы.Оплачено = -1")>0
_
         Or InStr(Me.Filter, "Заказы.Оплачено = True")>0)Then
      If ApplyType = acApplyFilter Then
         Forms!Заказы!Сумма.Visible = False
         Forms!Заказы!Налог.Visible = False
         Forms!Заказы!Итого.Visible = False
      ElseIf ApplyType = acShowAllRecords Then
         Forms!Заказы!Сумма.Visible = True
         Forms!Заказы!Налог.Visible = True
         Forms!Заказы!Итого.Visible = True
      End If
   End If
End Sub
```

Событие «Фильтрация» (Filter)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtFilterC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtFilterX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acevtFilterMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïS":"acevtFilterEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acevtFilterA"}

Событие **Фильтрация (Filter)** возникает при выполнении пользователем одного из следующих действий.

- Выбор команды **Изменить фильтр** в подменю **Фильтр** меню **Записи** в режиме формы или нажатие кнопки **Изменить фильтр**  на панели инструментов. При этом открывается окно обычного фильтра, позволяющее быстро создать фильтр на основе значений полей формы.
- Выбор команды **Расширенный фильтр** в подменю **Фильтр** меню **Записи** в режиме формы. При этом открывается окно расширенного фильтра, в котором создаются сложные фильтры для формы.
- Выбор в меню **Фильтр** команды **Расширенный фильтр** при открытом окне обычного фильтра или команды **Изменить фильтр** при открытом окне расширенного фильтра. При этом, когда закрывается открытое окно фильтра, возникает событие **Применение фильтра (ApplyFilter)**, а затем, когда открывается окно другого фильтра, возникает событие **Фильтрация (Filter)**.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя макроса или элемент [Процедура обработки события] в качестве значения свойства **Фильтрация (OnFilter)**.

Событие **Фильтрация (Filter)** используется для выполнения следующих действий.

- Удаление любого ранее определенного для формы фильтра. Для этого необходимо задать пустую строку ("") в качестве значения свойства **Фильтр (Filter)** формы в макросе или в процедуре обработки события **Фильтрация**. Это особенно полезно, если необходимо исключить попадание в новый фильтр любых лишних условий. Например, при каждом создании фильтра по выделенному используемое условие (текст, выделенный в форме) добавляется в выражение, определяющее предложение WHERE для свойства **Фильтр (Filter)**. Добавленное условие выводится как в окне обычного фильтра, так и в окне расширенного фильтра. Событие **Фильтрация** позволяет удалить такие старые условия.
- Ввод в новый фильтр значений, используемых по умолчанию. Для этого стандартные условия включаются в значение свойства **Фильтр (Filter)**. Например, можно потребовать, чтобы каждый фильтр по умолчанию отбирал в форме «Товары» записи только о тех товарах, поставки которых продолжаются (т.е. для которых в форме «Товары» не установлен флажок «ПоставкиПрекращены»).
- Использование специального окна фильтра вместо одного из стандартных окон фильтра Microsoft Access. Пользователь имеет возможность при возникновении события **Фильтрация** открыть специальную форму и использовать значения, вводящиеся в эту форму, для определения свойства **Фильтр** и отбора данных исходной формы. После закрытия специальной формы, чтобы применить фильтр, следует задать для свойства исходной формы **Фильтр включен (FilterOn)** значение **True** (–1). Отмена события **Фильтрация** предотвращает открытие стандартного окна фильтра Microsoft Access.
- Отмена вывода на экран некоторых элементов управления в форме или их использования в окне обычного фильтра. Если скрыть или отключить элемент управления в макросе или процедуре обработки события **Фильтрация**, этот элемент останется отключенным или скрытым в окне обычного фильтра и недоступным для включения в условия отбора. В дальнейшем такие элементы управления снова могут быть включены или сделаны

видимыми в ответ на событие **Применение фильтра (ApplyFilter)** после применения фильтра или его удаления из формы.

Событие «Фильтрация» (Filter) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtFilterMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtFilterMOX":1}

Чтобы запустить макрос при возникновении события **Фильтрация (Filter)**, следует указать имя макроса в качестве значения свойства **Фильтрация (OnFilter)**.

Дополнительные сведения

Допускается задание значения свойства **Фильтр (Filter)** с помощью макрокоманды **ЗадатьЗначение (SetValue)** в макросе, связанном с событием **Фильтрация (Filter)**. Это позволяет изменить условия отбора, которые выводятся в окне обычного фильтра или в окне расширенного фильтра).

Макрокоманда **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Фильтрация**, позволяет отменить открытие окна фильтра.

Событие «Фильтрация» (Filter) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtFilterEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtFilterEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Фильтрация (Filter)**, следует выбрать в качестве значения свойства **Фильтрация (OnFilter)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_Filter(Cancel **As Integer,** FilterType **As Integer)**

Процедура обработки события **Фильтрация (Filter)** использует следующие аргументы.

Аргумент	Описание
Cancel	Определяет, будет ли открыто окно фильтра. Заданное для аргумента «Cancel» значение **True** (–1) отменяет открытие окна фильтра. Допускается также отмена открытия окна фильтра с помощью метода **CancelEvent** объекта **DoCmd**.
FilterType	Определяет тип окна фильтра. Значение аргумента «FilterType» задается с помощью одной из следующих встроенных констант:

Константа	Значение
acFilterByForm	0
acFilterAdvanced	1

Дополнительные сведения

Процедура обработки события **Фильтрация (Filter)** позволяет скрыть или отключить определенные элементы управления в окне обычного фильтра, сделав их недоступными для включения в условия отбора. Например, можно потребовать, чтобы в форме «Заказы» всегда выводились записи, имеющие значение в поле «Итого», вне зависимости от величины этого значения. Для этого следует отключить или скрыть это поле в процедуре обработки события **Фильтрация (Filter)**, а затем включить его или сделать видимым в процедуре обработки события **Применение фильтра (ApplyFilter)**. Процедура обработки события **Фильтрация** удобнее для отключения или скрытия элементов управления в окне обычного фильтра, чем макрос, связанный с событием **Фильтрация**, поскольку аргумент «FilterType» позволяет проверить, открыто ли окно обычного фильтра. В окне расширенного фильтра выводятся все поля формы, даже отключенные или скрытые.

Событие «Фильтрация» (Filter) - Пример процедуры обработки события

В следующем примере демонстрируется, как отключить элемент управления «Итого» в форме «Заказы» при попытке пользователя создать фильтр по этому полю. После применения фильтра в форме будут выведены записи, имеющие значение в поле «Итого» и удовлетворяющие другим условиям фильтра. В данном примере показано, как заставить пользователя использовать окно обычного фильтра, а не окно расширенного фильтра.

Для проверки данного примера включите следующую процедуру обработки события в форму «Заказы», содержащую элемент управления «Итого». Попытайтесь создать фильтр с помощью окна расширенного фильтра, использующего элемент управления «Итого». Попробуйте также создать тот же самый фильтр с помощью окна обычного фильтра.

```
Private Sub Form_Filter(Cancel As Integer, FilterType As Integer)
   If FilterType = acFilterByForm Then
      Forms!Заказы!Итого.Enabled = False
   ElseIf FilterType = acFilterAdvanced Then
      MsgBox "Для создания фильтра выберите команду или кнопку " _
         & "'Изменить фильтр'.", vbOKOnly + vbInformation
      Cancel = True
   End If
End Sub
```

Событие «Отсутствие данных» (NoData)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtNoDataC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtNoDataX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtNoDataMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtNoDataEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtNoDataA"}

Событие **Отсутствие данных (NoData)** возникает после форматирования отчета для печати, если отчет не содержит данных (присоединен к пустому набору записей), но до вывода отчета на печать. Это событие используют для отмены печати пустого отчета.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя макроса или элемент [Процедура обработки события] в качестве значения свойства **Отсутствие данных (OnNoData)**.

Если отчет не присоединен к таблице или запросу (с помощью значения свойства отчета **Источник записей (RecordSource)**), событие **Отсутствие данных (NoData)** не возникает.

Данное событие возникает в отчете после событий **Форматирование (Format)**, но до первого события **Печать (Print)**.

Это событие на возникает для подчиненных отчетов. Если требуется сделать скрытыми элементы управления подчиненного отчета в случае, когда он не содержит данных, чтобы не выводить эти элементы управления на печать, следует использовать свойство **HasData** в макросе или процедуре обработки события **Форматирование** или **Печать**.

Событие **Отсутствие данных** возникает до первого события **Страница (Page)** в отчете.

Событие «Отсутствие данных» (NoData) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtNoDataMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtNoDataMOX":1}

Чтобы запустить макрос при возникновении события **Отсутствие данных (NoData)**, следует указать имя макроса в качестве значения свойства **Отсутствие данных (OnNoData)**.

Дополнительные сведения

Макрокоманда **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Отсутствие данных**, позволяет отменить печать пустого отчета.

Событие «Отсутствие данных» (NoData) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtNoDataEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtNoDataEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Отсутствие данных (NoData)**, следует выбрать в качестве значения свойства **Отсутствие данных (OnNoData)** элемент [Процедура обработки события] и нажать кнопку **Построить** 

Синтаксис

Private Sub Report_NoData(Cancel **As Integer)**

Процедура обработки события **Отсутствие данных** использует следующий аргумент.

Аргумент	**Описание**
Cancel	Определяет, выводится ли отчет на печать. Заданное для аргумента «Cancel» значение **True** (–1) отменяет печать отчета. Допускается также отмена печати отчета с помощью метода **CancelEvent** объекта **DoCmd**.

Событие «Отсутствие данных» (NoData) - Пример процедуры обработки события

В данном примере демонстрируется отмена печати отчета, если он не содержит данных. Для пользователя выводится сообщение об отмене печати.

Для проверки данного примера добавьте следующую процедуру обработки события в отчет. Попытайтесь выполнить отчет, когда он не содержит данных.

```
Private Sub Report_NoData(Cancel As Integer)
   MsgBox "Отчет не содержит данных." _
      & "@Печать отчета отменена. " _
      & "@Проверьте источник данных и убедитесь, что указаны " _
      & "правильные условия (например, допустимый диапазон " _
      & "дат).", vbOKOnly + vbInformation
   Cancel = True
End Sub
```

Событие «Страница» (Page)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtPageC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtPageX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtPageMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS":"acevtPageEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acevtPageA"}

Событие **Страница (Page)** возникает после форматирования страницы отчета для печати, но до вывода страницы на печать. Это событие используют для создания рамки вокруг страницы или для добавления на страницу других элементов оформления.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя макроса или элемент [Процедура обработки события] в качестве значения свойства **Страница (OnPage)**.

Данное событие возникает после всех событий **Форматирование (Format)** отчета и после всех событий **Печать (Print)** страницы, но до физического вывода этой страницы на печать.

Для создания на странице графических элементов обычно используются методы **Line**, **Circle** или **PSet** в процедуре обработки события **Страница (Page)**.

Событие **Отсутствие данных (NoData)** возникает в отчете до первого события **Страница**.

Событие «Страница» (Page) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtPageMOC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtPageMOX":1}

Чтобы запустить макрос при возникновении события **Страница (Page)**, следует указать имя макроса в качестве значения свойства **Страница (OnPage)**.

Дополнительные сведения

Поскольку для создания на странице графических элементов обычно используют методы **Line**, **Circle** или **PSet** в процедуре обработки события **Страница**, с данным событием чаще всего связывают процедуры обработки события, а не макросы.

Не допускается использование макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Страница**.

Событие «Страница» (Page) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtPageEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtPageEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Страница (Page)**, следует выбрать в качестве значения свойства **Страница (OnPage)** элемент [Процедура обработки события] и нажать кнопку **Построить** 

Синтаксис

Private Sub Report_Page( )

Дополнительные сведения

Отмена события **Страница** не допускается.

Событие «Страница» (Page) - Пример процедуры обработки события

В данном примере демонстрируется проведение прямоугольной рамки вокруг страницы отчета с помощью метода **Line**. Свойства **ScaleWidth** и **ScaleHeight** по умолчанию возвращают внутреннюю ширину и высоту отчета.

```
Private Sub Report_Page()
   Me.Line(0, 0)-(Me.ScaleWidth, Me.ScaleHeight), , B
End Sub
```

Событие ItemAdded

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtItemAddedC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtItemAddedX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtItemAddedMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtItemAddedEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtItemAddedA"}

Событие **ItemAdded** возникает при добавлении ссылки в проект из Visual Basic.

Примечания

- Событие **ItemAdded** применимо к семейству **References**. Оно не может быть связано с элементом управления, формой или отчетом, как большинство остальных событий. Поэтому для описания процедуры обработки события **ItemAdded** используется особый синтаксис.
- Событие **ItemAdded** вызывается только в явном виде, и не может быть вызвано с помощью макроса.

Дополнительные сведения

Данное событие возникает только при добавлении ссылки из программы. Оно не вызывается, если ссылка добавляется из диалогового окна **Ссылки**, доступного по команде **Ссылки** из меню **Сервис** в открытом окне модуля.

Событие ItemAdded - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtItemAddedEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtItemAddedEPX":1}

Для создания процедуры обработки события, которая запускается при возникновении события ItemAdded необходимо выполнить действия, указанные ниже в разделе «Дополнительные сведения».

Синтаксис

Private Sub *evtReferences*_**ItemAdded(***reference***)**

Процедура обработки события ItemAdded получает следующие аргументы:

Аргумент	Описание
evtReferences	Объектная переменная, представляющая семейство **References**. Эта переменная должна быть объявлена в модуле класса с помощью ключевого слова **WithEvents**.
reference	Объект **Reference**, который был добавлен в семейство **References**.

Дополнительные сведения

Процедура обработки события ItemAdded может содержать программу, которая будет запускаться при добавлении ссылки на библиотеку типов в проект. Для добавления объекта **Reference** в семейство **References** из Visual Basic используется методы **AddFromFile** или **AddFromGUID**.

Для создания процедуры обработки события ItemAdded необходимо.

1. Объявить объектную переменную типа **References** в разделе описаний модуля класса с помощью ключевого слова **WithEvents**. В Microsoft Access эта объектная переменная используется для определения процедуры обработки события ItemAdded. Следующая строка программы демонстрирует типичное описание **WithEvents**:

   ```
   Public WithEvents evtReferences As References
   ```

2. Создать описание процедуры обработки события внутри модуля класса, выбрав имя объектной переменной **References** в поле **Объект** окна модуля. Например, если используется описание из предыдущего этапа, то в поле **Объект** появится `evtReferences`. Затем выбрать событие ItemAdded в поле **Процедура**.
3. Написать программу, которая будет выполняться при возникновении события. Например, если необходимо вывести сообщение о добавлении ссылки в проект, то в процедуру обработки события следует внести соответствующую программу.
4. Присвоить семейство **References** объектной переменной **References**, которая была объявлена с ключевым словом **WithEvents**. Инструкция присвоения должна содержаться в процедуре, которая запускается перед возникновением события **ItemAdded**.

   Например, данная инструкция может быть включена в процедуру для события **Инициализация (Initialize)** модуля класса, в котором объявлена объектная переменная. В таком случае при создании экземпляра класса объектная переменная будет инициализироваться автоматически. Если используется объектная переменная `evtReferences`, то процедура обработки может выглядеть следующим образом:

   ```
   Private Sub Class_Initialize()
      Set evtReferences = Application.References
   End Sub
   ```

Примечание. Если описанные выше этапы уже были выполнены при создании процедуры обработки события **ItemRemoved**, то нет необходимости в их повторном проведении для события **ItemAdded**. Одна объектная переменная **References** может использоваться для обоих

событий.

Для запуска события **ItemAdded** необходимо создать новую ссылку с помощью объектной переменной **References**, объявленной с ключевым словом **WithEvents**.

Для добавления ссылки и запуска события необходимо:

1. Внутри стандартного модуля создать новый экземпляр класса, в котором была объявлена объектная переменная **References**. Например, если имя класса RefEvents, то новый экземпляр класса может быть создан и присвоен объектной переменной с помощью следующей инструкции описания. Эта инструкция записывается на уровне модуля или внутри процедуры.
   ```
   Dim objRefEvents As New RefEvents
   ```
2. Внутри процедуры типа **Sub** или **Function**, создать новый объект **Reference** с помощью методов **AddFromFile** или **AddFromGUID** объектной переменной, которая представляет семейство **References** (`evtReferences`). Так как событие **ItemAdded** в модуле класса определено в связи с определенной объектной переменной **References**, необходимо создать новый объект **Reference** с помощью этой переменной. В противном случае событие не возникнет.

Объектную переменную **References** следует определять с именем класса, в котором она описана.

После выполнения всех предыдущих шагов новая ссылка может быть добавлена следующим образом:

```
Function AddReference(strFileName As String)
   Dim ref As Reference
   ' Создается новый экземпляр класса RefEvents.
   Dim objRefEvents As New RefEvents

   ' Создается ссылка на семейство References
   ' переменная определяется в классе RefEvents.
   Set ref = objRefEvents.evtReferences.AddFromFile(strFileName)
End Function
```

Примечания

- Процедура, инициализирующая объект **References**, должна находится в той же области видимости, что и описание объектной переменной. Например, если объектная переменная **References** описана как **Private** внутри модуля класса, то необходимо инициализировать переменную в процедуре в том же модуле класса.
- Процедура обработки события **ItemAdded** может быть включена только в модуль класса.
- После описания переменной уровня модуля для представления семейства **References**, ее можно использовать для создания описания процедуры обработки для событий **ItemAdded** и **ItemRemoved**.
- Если в одном проекте описано несколько процедур обработки события **ItemAdded**, то все они будут запущены при возникновении события в произвольной последовательности. Поэтому рекомендуется создавать одну процедуру обработки события **ItemAdded** в проекте и включать в нее все инструкции, которые следует выполнить при возникновении события.
- Событие **ItemAdded** возникает после добавления ссылки в проект.

Пример процедуры обработки событий «Инициализация» (Initialize), ItemAdded, ItemRemoved, «Завершение» (Terminate)

Следующий пример содержит процедуру обработки для событий **ItemAdded** и **ItemRemoved**. Чтобы применить этот пример, сначала следует создать новый класс. Для этого выполните команду **Модуль** в меню **Вставка**. Восстановите следующую программу в модуль класса и сохраните его под именем RefEvents:

```
' Объявление объектной переменной для представления семейства References.
Public WithEvents evtReferences As References

' При создании экземпляра класса происходит инициализация переменной
evtReferences.
Private Sub Class_Initialize()
   Set evtReferences = Application.References
End Sub

' При удалении экземпляра переменной evtReferences присваивается значение
Nothing.
Private Sub Class_Terminate()
   Set evtReferences = Nothing
End Sub

' При добавлении ссылки отображается сообщение.
Private Sub evtReferences_ItemAdded(ByVal Reference As Access.Reference)
   MsgBox "Добавлена ссылка на " & Reference.Name
End Sub

' При удалении ссылки отображается сообщение.
Private Sub evtReferences_ItemRemoved(ByVal Reference As Access.Reference)
   MsgBox "Удалена ссылка на " & Reference.Name
End Sub
```

Следующая процедура типа **Function** добавляет заданную ссылку. При добавлении запускается процедура обработки события **ItemAdded**, определенная в классе RefEvents.

Например, для установки ссылки на элемент календарь следует передать в качестве параметра строку "C:\Windows\System\Mscal.ocx" (если указанный маршрут действительно задает расположение календаря).

```
' Создание нового экземпляра класса RefEvents.
Dim objRefEvents As New RefEvents

' Процедуре передается имя файла и путь к библиотеке типов для этой
процедуры.
Function AddReference(strFileName As String) As Boolean
   Dim ref As Reference

   On Error GoTo Error_AddReference
   ' Создается новая ссылка на объектную переменную References.
   Set ref = objRefEvents.evtReferences.AddFromFile(strFileName)
   AddReference = True

Exit_AddReference:
   Exit Function

Error_AddReference:
   MsgBox Err & ": " & Err.Description
```

```
   AddReference = False
   Resume Exit_AddReference
End Function
```

Следующая процедура типа **Function** удаляет заданную ссылку. При удалении запускается процедура обработки события **ItemRemoved**, определенная в классе RefEvents.

Например, для удаления ссылки на элемент календарь следует передать в качестве параметра строку "MSACAL", которая является именем объекта **Reference**, представляющего этот элемент.

```
Function RemoveReference(strRefName As String) As Boolean
   Dim ref As Reference

   On Error GoTo Error_RemoveReference
   ' Возвращает объект, представляющий существующую ссылку.
   Set ref = objRefEvents.evtReferences(strRefName)
   ' Удаляет ссылку из семейства.
   objRefEvents.evtReferences.Remove ref
   RemoveReference = True

Exit_RemoveReference:
   Exit Function

Error_RemoveReference:
   MsgBox Err & ": " & Err.Description
   RemoveReference = False
   Resume Exit_RemoveReference
End Function
```

Событие ItemRemoved

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtItemRemovedC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtItemRemovedX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtItemRemovedM":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїS":"acevtItemRemovedEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acevtItemRemovedA"}

Событие **ItemRemoved** возникает при удалении ссылки из проекта.

Примечания

- Событие **ItemRemoved** применимо к семейству **References**. Оно не может быть связано с элементом управления, формой или отчетом, как большинство остальных событий. Поэтому для описания процедуры обработки события **ItemRemoved** используется особый синтаксис.
- Событие **ItemAdded** запускается только при его действительном появлении, и не может быть запущено с помощью макроса.

Дополнительные сведения

Данное событие возникает только при удалении ссылки из программы. Оно не вызывается, если ссылка удаляется из диалогового окна **Ссылки**, доступного по команде **Ссылки** из меню **Сервис** в открытом окне модуля.

Событие ItemRemoved - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtItemRemovedEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtItemRemovedEPX":1}

Для создания процедуры обработки события, которая запускается при возникновении события **ItemRemoved** необходимо выполнить действия, указанные в пункте «Примечания» данного раздела.

Синтаксис

Private Sub *evtReferences*_**ItemRemoved(***reference***)**

Процедура обработки события **ItemRemoved** получает следующие аргументы:

Аргумент	Описание
evtReferences	Объектная переменная, представляющая семейство **References**. Эта переменная должна быть объявлена в модуле класса с помощью ключевого слова **WithEvents**.
reference	Объект **Reference**, который был добавлен в семейство **References**.

Дополнительные сведения

Процедура обработки события **ItemRemoved** может содержать программу, которая будет запускаться при удалении ссылки на библиотеку типов из проекта. Для удаления объекта **Reference** из семейства **References** в программе Visual Basic используется метод **Remove**.

Для создания процедуры обработки события ItemRemoved необходимо.

1. Объявить объектную переменную типа **References** в разделе описаний модуля класса с помощью ключевого слова **WithEvents**. В Microsoft Access эта объектная переменная используется для определения процедуры обработки события ItemRemoved. Следующая строка программы демонстрирует типичное описание **WithEvents**:

   ```
   Public WithEvents evtReferences As References
   ```

2. Создать описание процедуры обработки события внутри модуля класса, выбрав имя объектной переменной **References** в поле **Объект** окна модуля. Например, если используется описание из предыдущего этапа, то в поле **Объект** появится `evtReferences`. Затем выбрать событие ItemRemoved в поле **Процедура**.
3. Написать программу, которая будет выполняться при возникновении события. Например, если необходимо вывести сообщение о том, что ссылка была удалена из проекта, то в процедуру обработки события следует внести соответствующую программу.
4. Присвоить семейство **References** объектной переменной **References**, которая была объявлена с ключевым словом **WithEvents**. Инструкция присвоения должна содержаться в процедуре, которая запускается перед возникновением события **ItemRemoved**.

   Например, данная инструкция может быть включена в процедуру для события **Инициализация (Initialize)** модуля класса, в котором объявлена объектная переменная. В таком случае при создании экземпляра класса объектная переменная будет инициализироваться автоматически. Если используется объектная переменная `evtReferences`, то процедура обработки может выглядеть следующим образом:

   ```
   Private Sub Class_Initialize()
      Set evtReferences = Application.References
   End Sub
   ```

Примечание. Если описанные выше этапы уже были выполнены при создании процедуры обработки события **ItemAdded**, то нет необходимости в их повторном проведении для события **ItemRemoved**. Одна объектная переменная **References** может использоваться для обоих событий.

Для запуска события **ItemRemoved** необходимо создать новую ссылку с помощью объектной переменной **References**, объявленной с ключевым словом **WithEvents**.

Для удаления ссылки и запуска события необходимо.

1. Внутри стандартного модуля создать новый экземпляр класса, в котором была объявлена объектная переменная **References**. Например, если имя класса RefEvents, то новый экземпляр класса может быть создан и присвоен объектной переменной с помощью следующей инструкции описания. Эта инструкция записывается на уровне модуля или внутри процедуры.

   ```
   Dim objRefEvents As New RefEvents
   ```

2. Внутри процедуры типа **Sub** или **Function**, создать новый объект **Reference** с помощью метода **Remove** объектной переменной, которая представляет семейство **References** (`evtReferences`). Так как событие **ItemRemoved** в модуле класса определено в связи с определенной объектной переменной **References**, необходимо создать новый объект **Reference** с помощью этой переменной. В противном случае событие не возникнет.

Объектную переменную **References** следует определять с именем класса, в котором она описана.

После выполнения всех предыдущих шагов новая ссылка может быть добавлена следующим образом:

```
Function RemoveReference(strRefName As String)
   Dim ref As Reference
   ' Создается новый экземпляр класса RefEvent.
   Dim objRefEvents As New RefEvents

   ' Создается ссылка на семейство References.
   ' Переменная определяется в классе RefEvents.
   Set ref = objRefEvents.evtReferences(strRefName)
   objRefEvents.evtReferences.Remove ref
End Function
```

Примечания

- Процедура, инициализирующая объект **References**, должна находится в той же области видимости, что и описание объектной переменной. Например, если объектная переменная **References** описана как **Private** внутри модуля класса, то необходимо инициализировать переменную в процедуре в том же модуле класса.
- Процедура обработки события **ItemRemoved** может быть включена только в модуль класса.
- После описания переменной уровня модуля для представления семейства **References**, ее можно использовать для создания описания процедуры обработки для событий **ItemAdded** и **ItemRemoved**.
- Если в одном проекте описано несколько процедур обработки события **ItemRemoved**, то все они будут запущены при возникновении события в произвольной последовательности. Поэтому рекомендуется создавать одну процедуру обработки события **ItemRemoved** в проекте и включать в нее все инструкции, которые следует выполнить при возникновении события.
- Событие **ItemRemoved** возникает после удаления ссылки из проекта.

Функция Eval

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctEvalC;vafctChoose;vafctIIf;vafctSwitch"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctEvalX":1}

Функция **Eval** позволяет найти значение выражения, сводящегося к строковому или числовому значению.

Строка, которая собирается в результате определенных действий, обрабатывается функцией **Eval** как реальное выражение. Функция **Eval** сначала находит значение строкового выражения, а потом возвращает соответствующее значение. Например, `Eval("1 + 1")` возвращает `2`.

Если передать в функцию **Eval** строку, содержащую имя и аргументы функции, то функция **Eval** возвращает значение функции. Например, `Eval("Chr$(65)")` возвращает «A».

Синтаксис

Eval(*строковоеВыражение***)**

Аргумент *строковоеВыражение* представляет выражение, задаваемое строкой. Например, *строковоеВыражение* может задавать функцию, возвращающую строку или число, или являться ссылкой на элемент управления в форме. Необходимо, чтобы значением аргумента *строковоеВыражение* были строка или число; значением этого аргумента не может быть объект Microsoft Access.

Примечание При передаче имени функции в качестве аргумента *строковоеВыражение* в функцию **Eval** необходимо заключить это имя в кавычки. Например:

```
Debug.Print Eval("СписокИмен()")   ' СписокИмен - функция, определенная пользователем.

Debug.Print Eval("StrComp(""Joe"",""joe"", 1)")

Debug.Print Eval("Date()")
```

Дополнительные сведения

Функцию **Eval** используют в вычисляемых элементах управления в формах и отчетах, а также в макросах и модулях. Функция **Eval** возвращает значение типа **Variant** принадлежащее либо к строковому, либо к числовому типу.

Аргумент *строковоеВыражение* с необходимость должен представлять выражение, сохраненное в виде строки. Попытка передать в функцию **Eval** обычное строковое значение, не представляющее выражение, например, **Eval**("Smith"), приведет к ошибке при выполнении .

Функция **Eval** позволяет определить значение, сохраняемое в свойстве **Value** элемента управления. В следующем примере полная ссылка на элемент управления передается в функцию **Eval**. Функция **Eval** возвращает значение, сохраняемое в свойстве **Value** этого элемента управления.

```
Dim strCtl As String
Set ctl = Forms!Сотрудники!Фамилия
strCtl = "Forms!Сотрудники!Фамилия"
MsgBox ("Текущее значение поля '" & strCtl & "': " & Eval(strCtl))
```

Вызов некоторых операторов в Visual Basic возможен только с помощью функции **Eval**. Например, не допускается прямое использование в программах операторов SQL **Between...And** или **In,** однако, они могут включаться в выражения, передаваемые в функцию **Eval**.

В следующем примере определяется, является ли значение в поле «ОбластьДоставки» аббревиатурой одного из штатов США. Если это так, переменная `intState` получает значение **True** (-1). Отметим, что для указания подстроки внутри строки применяются одинарные кавычки (').

```
Dim intState As Integer
intState = Eval("Forms!Заказы!ОбластьПолучателя In "
   & "('AK', 'CA', 'ID', 'WA', 'MT', 'NM', 'OR')")
```

Функция Eval, примеры

В данном примере предполагается, что имеется набор из 50 функций с именами A1, A2 и т.д. С помощью функции **Eval** осуществляется вызов каждой функции.

```
Sub CallSeries()
   Dim intI As Integer

   For intI = 1 To 50
      Eval("A" & intI & "()")
   Next intI
End Sub
```

В следующем примере при нажатии пользователем кнопки в форме создается событие **Нажатие кнопки (Click)**. Если первым символом в значении свойства кнопки **Нажатие кнопки (OnClick)** является символ знака равенства (=), указывающий, что значение представляет имя функции, то функция **Eval** вызывает функцию, отвечающую событию **Нажатие кнопки (Click)**. Если это значение не начинается со знака равенства, то оно должно представлять имя макроса, для выполнения которого вызывается метод **RunMacro** объекта **DoCmd**.

```
Dim ctl As Control, varTemp As Variant

Set ctl = Forms!ИмяФормы!ИмяКнопки
If (Left(ctl.OnClick, 1) = "=") Then
   varTemp = Eval(Mid(ctl.OnClick,2))
Else
   DoCmd.RunMacro ctl.OnClick
End If
```

Функция CurrentUser

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctCurrentUserC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctCurrentUserX":1}

Функция **CurrentUser** возвращает имя текущего пользователя базы данных.

Например, функцию **CurrentUser** используют в процедурах, регистрирующих пользователей, которые вносят изменения в базы данных.

Синтаксис

CurrentUser

Дополнительные сведения

Функция **CurrentUser** возвращает строку, содержащую имя текущей учетной записи пользователя.

Если не создана защищенная рабочая группа, то функция **CurrentUser** возвращает имя используемой по умолчанию учетной записи пользователя «Admin». Учетная запись «Admin» дает пользователю полные права на все объекты базы данных.

Если включена защита рабочей группы, функция **CurrentUser** возвращает имя текущей учетной записи пользователя. Для всех учетных записей пользователя кроме «Admin» необходимо указать разрешения на объекты базы данных/

Следующая инструкция определяет имя текущего пользователя и выводит его в окно диалога.

```
MsgBox("Текущий пользователь: " & CurrentUser)
```

Функция Command

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctCommandC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acfctCommandX":1}

Функция **Command** возвращает параметры командной строки, указанные при запуске Microsoft Access.

Синтаксис

Command

Дополнительные Сведения

При запуске Microsoft Access из командной строки вся часть командной строки, расположенная после ключа **/cmd**, передается в программу как аргумент командной строки. Функция **Command** возвращает переданный аргумент командной строки.

Для того чтобы изменить аргумент командной строки после открытия базы данных, выберите в меню **Сервис** команду **Параметры**. На вкладке **Другие** введите новое значение аргумента в поле **Параметры командной строки**. После этого функция **Command** будет возвращать новое значение аргумента.

При вызове функции **Command** в лбом месте, кроме программы в модуле Visual Basic, следует помещать пустые скобки после имени функции. Например, при вызове функции **Command** из поля в форме следует ввести в ячейку свойства **Данные** (**ControlSource**) такое выражение:

```
=Command()
```

Функция Command, пример

В следующем примере демонстрируется запуск Microsoft Access с параметрами командной строки и возвращение этого аргумента с помощью функции **Command**.

Для проверки данного примера выберите в главном меню Windows команду **Выполнить**. Введите следующее выражение в виде одной строки в поле **Открыть** в диалоговом окне **Запуск программы**.

```
"C:\Program Files\Microsoft Office\Office\Msaccess.exe" "C:\Program Files\
Microsoft Office\Office\Samples\Борей.mdb" /cmd "Заказы"
```

Далее, откройте новый модуль в демонстрационной базе данных «Борей» и добавьте в него следующую функцию.

```
Function ПроверкаКоманднойСтроки()
   ' Проверяет значение, возвращаемое функцией Command.
   If Command = "Заказы" Then
      DoCmd.OpenForm "Заказы"
   ElseIf Command = "Сотрудники" Then
      DoCmd.OpenForm "Сотрудники"
   Else
      Exit Function
   End If
End Function
```

При вызове этой функции будет открыта форма «Заказы».

Данная функция может быть вызвана автоматически при открытии базы данных из макроса AutoExec. Подробнее об автоматическом вызове макрокоманд см. в разделе Выполнение макрокоманды при открытии базы данных.

Функция CurrentDb

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctCurrentDbC;dacolDatabase;daobjDatabase"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctCurrentDbX":1}

Функция **CurrentDb** возвращает объектную переменную типа **Database,** представляющую текущую базу данных, открытую в окне Microsoft Access.

Синтаксис

CurrentDb

Дополнительные Сведения

При необходимости прямого изменения структуры базы данных и обращения к данным из программы Visual Basic следует использовать объекты доступа к данным. Функция **CurrentDb** предоставляет способ доступа к текущей базе данных из программы Visual Basic, при котором не требуется знание имени конкретной базы данных. После создания переменной, указывающей на текущую базу данных, становится возможным доступ к объектам и семействам, входящим в иерархию объектов доступа к данным.

Функция **CurrentDb** позволяет создавать несколько объектных переменных, представляющих одну и ту же базу данных. В следующем примере обе переменные, `dbsA` и `dbsB`, указывают на текущую базу данных.

```
Dim dbsA As Database, dbsB As Database
Set dbsA = CurrentDb
Set dbsB = CurrentDb
```

Примечание. В предыдущих версиях Microsoft Access для возвращения указателя текущей базы данных использовался синтаксис `DBEngine.Workspaces(0).Databases(0)` или `DBEngine(0)(0)`. В Microsoft Access для Windows 95 вместо этого следует использовать функцию **CurrentDb**. Функция **CurrentDb** создает новый экземпляр текущей базы данных, тогда как конструкция `DBEngine(0)(0)` представляет ссылку на открытую копию текущей базы данных. С помощью функции **CurrentDb** пользователь имеет возможность создать несколько объектных переменных типа **Database**, представляющих текущую базу данных. Microsoft Access по-прежнему поддерживает синтаксис `DBEngine(0)(0)`, однако, рекомендуется внести изменения в имеющиеся программы во избежание возможных конфликтов при работе с сетевой базой данных.

При необходимости работать с другой базой данных в то время, когда текущая база данных открыта в окне Microsoft Access, следует использовать метод **OpenDatabase** объекта **Workspace.** Метод **OpenDatabase** не открывает вторую базу данных в окне Microsoft Access; в нем просто возвращается переменная типа **Database**, представляющая вторую базу данных. В следующем примере возвращается указатель на текущую базу данных и на базу данных Contacts.mdb.

```
Dim dbsCurrent As Database, dbsContacts As Database
Set dbsCurrent = CurrentDb
Set dbsContacts = DBEngine.Workspaces(0).OpenDatabase("Contacts.mdb")
```

Функция CurrentDb, пример

В следующем примере с помощью функции **CurrentDb** возвращается объектная переменная типа **Database**, представляющая текущую базу данных. Далее в программе выводится перечень всех полей в таблице «Сотрудники» из этой базы данных.

```
Sub ListFields()
   Dim dbs As Database, tdf As TableDef, fld As Field

   ' Возвращает объектную переменную типа Database, представляющую текущую
базу данных.
   Set dbs = CurrentDb
   ' Возвращает объектную переменную типа TableDef, представляющую таблицу
"Сотрудники".
   Set tdf = dbs.TableDefs!Сотрудники
   ' Выводит перечень всех полей в таблице "Сотрудники".
   For Each fld In tdf.Fields
      Debug.Print fld.Name
   Next fld
End Sub
```

Функция HyperlinkPart

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctHyperlinkPartC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctHyperlinkPartX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acfctHyperlinkPartA"}

Функция **HyperlinkPart** возвращает сведения о данных, сохраняемых с типом данных гиперссылки. Поле гиперссылки может содержать до трех частей в следующем формате:

*отображаемыйТекст***#***адрес***#***допАдрес*

Синтаксис

объект.**HyperlinkPart(***гиперссылка As Variant*[**,** *часть As Integer*]**)**

Функция **HyperlinkPart** имеет следующие аргументы.

Аргумент	Описание
объект	Необязательный аргумент. Объект **Application**.
гиперссылка	Значение типа **Variant**, представляющее данные, сохраняемые в поле гиперссылки.
часть	Необязательный аргумент. Значением аргумента *часть* является встроенная константа, представляющая сведения о данных, возвращаемые функцией **HyperlinkPart**:

Константа	Значение	Описание
acDisplayedValue	0	Используется по умолчанию. Подчеркнутый текст, который отображается в гиперссылке.
acDisplayText	1	Часть поля гиперссылки *отображаемыйТекст*.
AcAddress	2	Часть поля гиперссылки *адрес*.
AcSubAddress	3	Часть поля гиперссылки *допАдрес*.

Дополнительные сведения

Функция **HyperlinkPart** используется для возвращения одного из трех значений из поля гиперссылки или отображаемого значения. Аргумент *часть* определяет, какое именно значение будет возвращено. Этот аргумент является необязательным. Если он не используется, то возвращается значение, которое Microsoft Access отображает для данной гиперссылки (оно соответствует значению аргумента *часть* по умолчанию, представляемому константой **acDisplayedValue**). Возвращаемые значения представляют одну из трех частей поля гиперссылки (*отображаемыйТекст*, *адрес* или *допАдрес*) или отображаемое значение, которое в Microsoft Access выводится для гиперссылки.

Примечание. Если функция **HyperlinkPart** используется в запросе, то аргумент *часть* является обязательным. При этом использование вышеперечисленных констант невозможно, и вместо них задаются фактические значения.

Если возвращается значение из части поля гиперссылки *отображаемыйТекст*, то отображаемое Microsoft Access значение совпадает с содержимым этой части поля. Если часть *отображаемыйТекст* не содержит значения, то отображаемое значение совпадает с содержимым первой заполненной части поля, т.е. части *адрес* или *допАдрес*.

В следующей таблице приводятся значения, возвращаемые функцией **HyperlinkPart** для данных, сохраняемых в поле гиперссылки.

Данные в поле гиперссылки	Возвращаемые HyperlinkPart значения

#http://www.microsoft.com#	**acDisplayedValue**: http://www.microsoft.com **acDisplayText**: **acAddress**: http://www.microsoft.com **acSubAddress**:
Microsoft#http://www.microsoft.com#	**acDisplayedValue**: Microsoft **acDisplayText**: Microsoft **acAddress**: http://www.microsoft.com **acSubAddress**:
Клиенты##Form Клиенты	**acDisplayedValue**: Клиенты **acDisplayText**: Клиенты **acAddress**: **acSubAddress**: Form Клиенты
##Form Клиенты	**acDisplayedValue**: Form Клиенты **acDisplayText**: **acAddress**: **acSubAddress**: Form Клиенты

Если пользователь включает часть *адрес* в поле гиперссылки с помощью диалогового окна **Вставить гиперссылку** (которое открывается при выборе команды **Гиперссылка** в меню **Вставка**) или непосредственно вводит адрес в поле, то Microsoft Access добавляет два символа #, которые разделяют части гиперссылки.

Для добавления или изменения в поле гиперссылки части *отображаемыйТекст* следует выделить гиперссылку в таблице с помощью правой кнопки мыши, а затем выбрать в контекстном меню команду **Гиперссылка** и ввести текст в поле **Отображать текст**.

Если данные вводятся в поле гиперссылки непосредственно или добавляются с помощью методов объектов доступа к данным (DAO), то обязательно должны быть вставлены два символа #, разделяющие части гиперссылки.

Функция HyperlinkPart, пример

В следующем примере все четыре константы аргумента *часть* используются для отображения сведений о данных, возвращаемых функцией **HyperlinkPart** для каждой записи в таблице, содержащей поле гиперссылки. Для проверки работы данного примера вставьте процедуру DisplayHyperlinkParts в раздел описаний модуля. Возможен вызов процедуры DisplayHyperlinkParts, например, в окне отладки с передачей в процедуру имени таблицы «ТаблицаГиперссылки», содержащей гиперссылки, и имени поля «ПолеГиперссылки», содержащего данные гиперссылки.

```
DisplayHyperlinkParts "ТаблицаГиперссылки", "ПолеГиперссылки"

Sub DisplayHyperlinkParts(strTable As String, strField As String)
   Dim dbs As Database, rst As Recordset
   Dim strMsg As String

   Set dbs = CurrentDb
   Set rst = dbs.OpenRecordset(strTable)

   While Not rst.EOF    ' Для каждой записи.
      strMsg = "ОтображаемоеЗначение = " & HyperlinkPart(rst(strField),
acDisplayedValue) _
         & vbCrLf & "ОтображаемыйТекст = " & HyperlinkPart(rst(strField),
acDisplayText) _
         & vbCrLf & "Адрес = " & HyperlinkPart(rst(strField), acAddress) _
         & vbCrLf & "ДопАдрес = " & HyperlinkPart(rst(strField),
acSubAddress)
      ' Части гиперссылки, возвращаемые функцией HyperlinkPart.
      MsgBox strMsg
      rst.MoveNext
   Wend
End Sub
```

Если функция **HyperlinkPart** используется в запросе, то аргумент *часть* является обязательным. Например, в следующей инструкции SQL функция **HyperlinkPart** используется для возвращения сведений о данных, сохраняемых с типом данных гиперссылки в поле адреса «URL» в таблице «Связи»:

```
SELECT Связи.URL, HyperlinkPart([URL],0)
AS Display, HyperlinkPart([URL],1)
AS Name, HyperlinkPart([URL],2)
AS Addr, HyperlinkPart([URL],3) AS SubAddr
FROM Связи;
```

Классификация макрокоманд по применению

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acidxActionTaskC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acidxActionTaskX":1}

Следующая таблица содержит список макрокоманд, сгруппированных по их применению.

Категория	Назначение	Макрокоманда
Данные в формах и отчетах	Отбор данных	ПрименитьФильтр (ApplyFilter)
	Перемещение по данным	СледующаяЗапись (FindNext), НайтиЗапись(FindRecord), КЭлементуУправления (GoToControl), НаСтраницу (GoToPage), НаЗапись(GoToRecord)
Выполнение	Выполнение команды	ВыполнитьКоманду (RunCommand)
	Выход из Microsoft Access	Выход (Quit)
	Выполнение макроса, процедуры или запроса	ОткрытьЗапрос (OpenQuery), ЗапускПрограммы (RunCode), ЗапускМакроса(RunMacro), ЗапускЗапросаSQL (RunSQL)
	Выполнение другого приложения	ЗапускПриложения (RunApp)
	Прерывание выполнения	ОтменитьСобытие (CancelEvent), Выход (Quit), ОстановитьВсеМакросы (StopAllMacros), ОстановитьМакрос (StopMacro)
Импорт/экспорт	Передача объектов Microsoft Access в другие приложения	ВывестиВФормате (OutputTo), ОтправитьОбъект (SendObject)
	Преобразование данных между Microsoft Access и другими форматами данных	ПреобразоватьБазуДанных (TransferDatabase), ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet), ПреобразоватьТекст (TransferText)
Работа с объектами	Копирование, переименование или сохранение объекта	КопироватьОбъект (CopyObject), Переименовать (Rename),, Сохранить (Save)
	Удаление объекта	УдалитьОбъект (DeleteObject)
	Изменение размеров или положения окна	Развернуть (Maximize), Свернуть (Minimize), СдвигРазмер (MoveSize), Восстановить (Restore)
	Открытие или закрытие объекта	Закрыть (Close), ОткрытьФорму (OpenForm), ОткрытьМодуль (OpenModule), ОткрытьЗапрос (OpenQuery), ОткрытьОтчет (OpenReport), ОткрытьТаблицу

		(OpenTable)
	Печать объекта	ОткрытьФорму (OpenForm), ОткрытьЗапрос (OpenQuery), ОткрытьОтчет (OpenReport), Печать (PrintOut)
	Выделение объекта	ВыделитьОбъект (SelectObject)
	Указание значения поля, элемента управления или свойства	ЗадатьЗначение (SetValue)
	Обновление данных или экрана	ОбновитьОбъект (RepaintObject), Обновление (Requery), ПоказатьВсеЗаписи (ShowAllRecords)
Другие вопросы	Создание специальной или общей строки меню, специального или глобального контекстного меню	ДобавитьМеню (AddMenu)
	Задание состояния пунктов меню в специальной или общей строке меню	ЗадатьКомандуМеню (SetMenuItem)
	Вывод информации на экран	ВыводНаЭкран (Echo), ПесочныеЧасы (Hourglass), Сообщение (MsgBox), УстановитьСообщения (SetWarnings)
	Генерация нажатий клавиш	КомандыКлавиатуры (SendKeys)
	Вывод на экран или скрытие встроенной или специальной панели инструментов	ПанельИнструментов (ShowToolbar)
	Подача звукового сигнала	Сигнал (Beep)

Перечень событий и свойств событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproEventsRefC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproEventsRefX":1}

Ниже приводится классификация событий по типам задач и перечень событий в алфавитном порядке (по английским именам событий).

По назначению

События данных (Data Events)	События клавиатуры (Keyboard Events)
События ошибки и таймера (Error and Timing Events)	События мыши (Mouse Events)
События фильтра (Filter Events)	События печати (Print Events)
События фокуса (Focus Events)	События окна (Window Events)

По алфавиту

Событие	Свойство события
Включение (Activate)	**Включение (OnActivate)**
После подтверждения Del (AfterDelConfirm)	**После подтверждения Del (AfterDelConfirm)**
После вставки (AfterInsert)	**После вставки (AfterInsert)**
После обновления (AfterUpdate)	**После обновления (AfterUpdate)**
Применение фильтра (ApplyFilter)	**Применение фильтра (OnApplyFilter)**
До подтверждения Del (BeforeDelConfirm)	**До подтверждения Del (BeforeDelConfirm)**
До вставки (BeforeInsert)	**До вставки (BeforeInsert)**
До обновления (BeforeUpdate)	**До обновления (BeforeUpdate)**
Изменение (Change)	**Изменение (OnChange)**
Нажатие кнопки (Click)	**Нажатие кнопки (OnClick)**
Закрытие (Close)	**Закрытие (OnClose)**
Текущая запись (Current)	**Текущая запись (OnCurrent)**
Двойное нажатие кнопки (DblClick)	**Двойное нажатие кнопки (OnDblClick)**
Отключение (Deactivate)	**Отключение (OnDeactivate)**
Удаление (Delete)	**Удаление (OnDelete)**
Вход (Enter)	**Вход (OnEnter)**
Ошибка (Error)	**Ошибка (OnError)**
Выход (Exit)	**Выход (OnExit)**
Фильтрация (Filter)	**Фильтрация (OnFilter)**
Форматирование (Format)	**Форматирование (OnFormat)**
Получение фокуса (GotFocus)	**Получение фокуса (OnGotFocus)**
Инициализация (Initialize)	*нет*
Добавление элемента (ItemAdded)	*нет*
Удаление элемента (ItemRemoved)	*нет*
Клавиша вниз (KeyDown)	**Клавиша вниз (OnKeyDown)**
Нажатие клавиши (KeyPress)	**Нажатие клавиши (OnKeyPress)**
Клавиша вверх (KeyUp)	**Клавиша вверх (OnKeyUp)**
Загрузка (Load)	**Загрузка (OnLoad)**
Потеря фокуса (LostFocus)	**Потеря фокуса (OnLostFocus)**
Кнопка вниз (MouseDown)	**Кнопка вниз (OnMouseDown)**

Перемещение указателя (MouseMove)	**Перемещение указателя (OnMouseMove)**
Кнопка вверх (MouseUp)	**Кнопка вверх (OnMouseUp)**
Отсутствие данных (NoData)	**Отсутствие данных (OnNoData)**
Отсутствие в списке (NotInList)	**Отсутствие в списке (OnNotInList)**
Открытие (Open)	**Открытие (OnOpen)**
Страница (Page)	**Страница (OnPage)**
Печать (Print)	**Печать (OnPrint)**
Изменение размера (Resize)	**Изменение размера (OnResize)**
Возврат (Retreat)	**Возврат (OnRetreat)**
Завершение (Terminate)	*нет*
Таймер (Timer)	**Таймер (OnTimer)**
Выгрузка (Unload)	**Выгрузка (OnUnload)**
При обновлении (Updated)	**При обновлении (OnUpdated)**

События данных

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxDataEventsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxDataEventsX":1}

События данных возникают при вводе, удалении или изменении данных в форме или элементе управления, а также при перемещении фокуса с одной записи на другую.

Событие	Свойство события	Событие возникает
После подтверждения Del (AfterDelConfirm)	**После подтверждения Del (AfterDelConfirm)** (формы)	После подтверждения пользователем удаления записей и фактического удаления записей или после отмены удаления.
После вставки (AfterInsert)	**После вставки (AfterInsert)** (формы)	После добавления новой записи в базу данных.
После обновления (AfterUpdate)	**После обновления (AfterUpdate)** (формы, элементы управления)	После обновления данных в элементе управления или записи. Данное событие возникает при потере фокуса элементом управления или записью, а также при выборе пользователем команды **Сохранить запись** в меню **Записи**. Событие возникает для новых и существующих записей.
До подтверждения Del (BeforeDelConfirm)	**До подтверждения Del (BeforeDelConfirm)** (формы)	После удаления одной или нескольких записей, но до открытия диалогового окна Microsoft Access с приглашением подтвердить или отменить удаление. Данное событие возникает после события **Удаление (Delete)**.
До вставки (BeforeInsert)	**До вставки (BeforeInsert)** (формы)	При вводе первого символа в новую запись, но до добавления записи в базу данных.
До обновления (BeforeUpdate)	**До обновления (BeforeUpdate)** (формы, элементы управления)	До обновления данных в элементе управления или записи. Данное событие возникает при потере фокуса элементом или при выборе пользователем команды **Сохранить запись** в меню **Записи**. Событие возникает для новых и существующих записей.
Изменение (Change)	**Изменение (OnChange)** (элементы управления)	При изменении содержимого в поле или в текстовом поле поля со списком; например, при вводе символа в элемент управления или при изменении значения свойства **Текст** элемента управления с помощью макроса или программы Visual Basic.
Текущая запись (Current)	**Текущая запись (OnCurrent)** (формы)	При переводе фокуса на запись (которая при этом становится текущей) или при выполнении

		повторного запроса к источнику данных формы. Данное событие возникает при первом открытии формы, а также при каждом переводе фокуса с одной записи на другую. Кроме того, это событие возникает при выполнении запроса к источнику данных для формы — например, при выборе команды **Удалить фильтр** в меню **Записи**, а также макрокоманды **ПоказатьВсеЗаписи** или **Обновление**.
Удаление (Delete)	**Удаление (OnDelete)** (формы)	При удалении записи пользователем, но до подтверждения удаления и до фактического удаления записи.
Отсутствие в списке (NotInList)	**Отсутствие в списке (OnNotInList)** (элементы управления)	При вводе в поле со списком значения, отсутствующего в списке поля со списком.
При обновлении (Updated)	**При обновлении (OnUpdated)** (элементы управления)	При изменении данных в объекте OLE.

События ошибки и таймера

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxErrorEventsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxErrorEventsX":1}

Следующие события используются при обработке ошибок и синхронизации данных в формах или отчетах.

Событие	Свойство события	Событие возникает
Ошибка (Error)	**Ошибка (OnError)** (формы, отчеты)	При возникновении ошибки выполнения Microsoft Access, когда фокус находится в форме или отчете. В число таких ошибок включаются ошибки ядра базы данных Microsoft Jet, но не включаются ошибки выполнения Visual Basic. (Поскольку макросы не позволяют идентифицировать возникающую ошибку, с данным событием обычно связывают процедуры обработки события Visual Basic).
Таймер (Timer)	**Таймер (OnTimer)** (формы)	По истечении промежутка времени, указанного в свойстве **Интервал таймера (TimerInterval)** формы. Событие **Таймер (OnTimer)** используется для регулярной синхронизации данных при работе в сети с помощью обновления данных или обновления экрана через определенные интервалы.

События фильтра

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxFilterEventsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxFilterEventsX":1}

События фильтра возникают при создании или применении фильтра в форме.

Событие	Свойство события	Событие возникает
Применение фильтра (ApplyFilter)	**Применение фильтра (OnApplyFilter)** (формы)	При выборе команды **Применить фильтр** в меню **Записи** или при нажатии кнопки **Применить фильтр** на панели команд. При этом применяется последний фильтр (созданный в окне обычного фильтра или в окне расширенного фильтра). При нажатии кнопки **Фильтр по выделенному** на панели команд или при выборе команды **Фильтр по выделенному** в подменю **Фильтр** меню **Записи**. Это приводит к применению фильтра, определяемого текущими выделенными элементами в форме. При выборе команды **Удалить фильтр** в меню **Записи** или при нажатии кнопки **Удалить фильтр** на панели команд. Это приводит к удалению любого текущего фильтра (или условия сортировки) в форме. При закрытии окна обычного фильтра или окна расширенного фильтра.
Фильтрация (Filter)	**Фильтрация (OnFilter)** (формы)	При выборе команды **Изменить фильтр** в подменю **Фильтр** меню **Записи** или при нажатии кнопки **Изменить фильтр** на панели команд. При этом открывается окно обычного фильтра, которое позволяет быстро создать фильтр по значениям полей в форме. При выборе команды **Расширенный фильтр** в подменю **Фильтр** меню **Записи**. При этом открывается окно расширенного фильтра, которое позволяет создать сложный фильтр для формы.

События фокуса

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxFocusEventsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxFocusEventsX":1}

События фокуса возникают, когда форма или элемент управления теряют или получают фокус, а также в момент, когда форма или отчет становятся активными или неактивными.

Событие	Свойство события	Событие возникает
Включение (Activate)	**Включение (OnActivate)** (формы, отчеты)	Когда окно формы или отчета становится активным.
Отключение (Deactivate)	**Отключение (OnDeactivate)** (формы, отчеты)	При выполнении действия, в результате которого активным должно стать другое окно Microsoft Access, но до того, когда это окно действительно станет активным. Событие **Отключение (Deactivate)** не возникает при переводе фокуса в окно другого приложения, в диалоговое окно или в окно всплывающей формы.
Вход (Enter)	**Вход (OnEnter)** (элементы управления)	Перед фактическим получением фокуса элементом управления от другого элемента управления в той же форме или при открытии формы. Данное событие возникает до события **Получение фокуса (GotFocus)**.
Выход (Exit)	**Выход (OnExit)** (элементы управления)	Непосредственно перед переводом фокуса на другой элемент управления в той же форме. Данное событие возникает до события **Потеря фокуса (LostFocus)**.
Получение фокуса (GotFocus)	**Получение фокуса (OnGotFocus)** (формы, элементы управления)	При получении фокуса элементом управления или формой, не имеющей активных или доступных элементов управления. Форма может получить фокус только в том случае, когда все ее видимые элементы управления недоступны или когда в форме нет элементов управления.
Потеря фокуса (LostFocus)	**Потеря фокуса (OnLostFocus)** (формы, элементы управления)	При потере фокуса элементом управления или формой. Форма может получить фокус только в том случае, когда все ее видимые элементы управления недоступны или когда в форме нет элементов управления.

События клавиатуры

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxKeyboardEventsC;vastmSendKeys"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxKeyboardEventsX":1}

События клавиатуры возникают при вводе с клавиатуры, а также при передаче нажатий клавиш с помощью макрокоманды **КомандыКлавиатуры (SendKeys)** или инструкции **SendKeys**.

Событие	Свойство события	Событие возникает
Клавиша вниз (KeyDown	**Клавиша вниз (OnKeyDown)** (формы, элементы управления)	При нажатии пользователем любой клавиши на клавиатуре в тот момент, когда элемент управления или форма имеют фокус. Форма может получить фокус только в том случае, когда все ее видимые элементы управления недоступны или когда в форме нет элементов управления. Событие **Клавиша вниз (KeyDown)** возникает также при передаче нажатий клавиш в форму или элемент управления с помощью макрокоманды **КомандыКлавиатуры (SendKeys)** в макросе или инструкции **SendKeys** в программе Visual Basic. Форма принимает все события **Клавиша вниз (KeyDown)** (даже события элементов управления) до их возникновения в элементах управления, если для свойства формы **Перехват нажатия клавиш (KeyPreview)** задано значение «Да». Если пользователь нажимает и удерживает клавишу, то событие **Клавиша вниз (KeyDown)** возникает многократно.
Нажатие клавиши (KeyPress)	**Нажатие клавиши (OnKeyPress)** (формы, элементы управления)	Если пользователь нажимает и отпускает клавишу или сочетание клавиш, соответствующие стандартному символу ANSI, в момент, когда элемент управления или форма имеют фокус. Форма может получить фокус только в том случае, когда все ее видимые элементы управления недоступны или когда в форме нет элементов управления. Например, событие **Нажатие клавиши (KeyPress)** позволяет организовать отклик на ввод символов в поле или в поле со списком. Для нажатий клавиш, которые не распознаются как событие **Нажатие клавиши**, таких как функциональные клавиши или клавиши перемещения, следует использовать события **Клавиша вниз (KeyDown)** или **Клавиша вверх (KeyUp)**.

		Событие **Нажатие клавиши** возникает также при передаче нажатий клавиш, соответствующих стандартному символу ANSI, в форму или элемент управления с помощью макрокоманды **КомандыКлавиатуры (SendKeys)** в макросе или инструкции **SendKeys** в программе Visual Basic. Форма принимает все события **Нажатие клавиши** (даже события элементов управления) до их возникновения в элементах управления, если для свойства формы **Перехват нажатия клавиш (KeyPreview)** задано значение «Да». Если пользователь нажимает и удерживает клавишу, то событие **Нажатие клавиши** возникает многократно.
Клавиша вверх (KeyUp)	**Клавиша вверх (OnKeyUp)** (формы, элементы управления)	Если пользователь отпускает нажатую клавишу в момент, когда элемент управления или форма имеют фокус. Объект, имеющий фокус, принимает все нажатия клавиш. Форма может получить фокус только в том случае, когда все ее видимые элементы управления недоступны или когда в форме нет элементов управления. Событие **Клавиша вверх (KeyUp)** возникает также при передаче нажатий клавиш в форму или элемент управления с помощью макрокоманды **КомандыКлавиатуры (SendKeys)** в макросе или инструкции **SendKeys** в программе Visual Basic. Форма принимает все события **Клавиша вверх** (даже события элементов управления) до их возникновения в элементах управления, если для свойства формы **Перехват нажатия клавиш (KeyPreview)** задано значение «Да». Если пользователь нажимает и удерживает клавишу, то событие **Клавиша вверх** возникает после всех событий **Клавиша вниз** и **Нажатие клавиши**.

События мыши

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acidxMouseEventsC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acidxMouseEventsX":1}

События мыши возникают при действиях с мышью, например при нажатии кнопки мыши или при удерживании кнопки в нажатом положении.

Событие	Свойство события	Событие возникает
Нажатие кнопки (Click)	**Нажатие кнопки (OnClick)** (формы, элементы управления)	Для элемента управления это событие возникает, когда пользователь нажимает и быстро отпускает левую кнопку мыши при указателе, установленном на элементе управления. Для формы это событие возникает, когда пользователь выбирает при помощи мыши область выделения записи, а также область формы вне раздела или элемента управления.
Двойное нажатие кнопки (DblClick)	**Двойное нажатие кнопки (OnDblClick)** (формы, элементы управления)	Для элемента управления это событие возникает, когда пользователь дважды быстро нажимает и отпускает левую кнопку мыши при указателе, установленном на элементе управления или присоединенной к нему надписи. Для формы это событие возникает, когда пользователь дважды быстро нажимает и отпускает кнопку мыши при указателе, установленном в области выделения записи или в пустой области формы.
Кнопка вниз (MouseDown)	**Кнопка вниз (OnMouseDown)** (формы, элементы управления)	В момент нажатия кнопки мыши при указателе, установленном на форме или на элементе управления. Для формы или элемента управления отмена события **Кнопка вниз (MouseDown)** с помощью макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе приводит к тому, что при нажатии правой кнопки мыши в форме или на элементе управления контекстные меню не выводятся.
Перемещение указателя (MouseMove)	**Перемещение указателя (OnMouseMove)** (формы, элементы управления)	При перемещении указателя по форме, разделу формы или элементу управления.
Кнопка вверх (MouseUp)	**Кнопка вверх (OnMouseUp)** (формы, элементы управления)	В момент отпускания нажатой кнопки мыши при указателе, установленном на форме или на элементе управления.

События печати

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxPrintEventsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxPrintEventsX":1}

События печати возникают при печати отчета или при его форматировании для печати.

Событие	Свойство	Событие возникает
Форматирование (Format)	**Форматирование (OnFormat)** (отчеты)	В момент, когда Microsoft Access определяет, какие данные попадают в раздел отчета, но до форматирования раздела для предварительного просмотра или печати. Допускается использование данных из текущей записи в макросе или процедуре обработки события для изменения макета страницы.
Отсутствие данных (NoData)	**Отсутствие данных (OnNoData)** (отчеты)	После форматирования для печати отчета, не содержащего данных (связанного с пустым набором записей), но до вывода отчета на печать. Это событие используется для отмены печати пустого отчета.
Страница (Page)	**Страница (OnPage)** (отчеты)	После форматирования страницы для печати, но до вывода страницы на печать.
Печать (Print)	**Печать (OnPrint)** (отчеты)	После форматирования раздела отчета для печати, но до вывода раздела на печать.
Возврат (Retreat)	**Возврат (OnRetreat)** (отчеты)	При возвращении назад на один или несколько разделов отчета на странице при форматировании в несколько проходов. Данное событие возникает после события раздела **Форматирование (Format)**, но до события **Печать (Print)**. Событие **Возврат (Retreat)** возникает для каждого раздела по мере прохода назад по разделам. Это позволяет отменить любые изменения, внесенные при обработке события раздела **Форматирование**.

События окна

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxWindowEventsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxWindowEventsX":1}

События окна возникают при открытии, изменении размеров или закрытии формы или отчета.

Событие	Свойство	Событие возникает
Закрытие (Close)	**Закрытие (OnClose)** (формы, отчеты)	При закрытии формы или отчета и их удалении с экрана.
Загрузка (Load)	**Загрузка (OnLoad)** (формы)	При открытии формы и выводе в ней записей. Данное событие возникает до события **Текущая запись (Current)**, но после события **Открытие (Open)**.
Открытие (Open)	**Открытие (OnOpen)** (формы, отчеты)	При открытии формы, но до вывода первой записи. При открытии отчета, но до начала печати отчета.
Изменение размера (Resize)	**Изменение размера (OnResize)** (формы)	При изменении размеров формы. Данное событие возникает также при первом открытии формы.
Выгрузка (Unload)	**Выгрузка (OnUnload)** (формы)	При закрытии формы и выгрузке ее записей, но до удаления формы с экрана. Данное событие возникает до события **Закрытие (Close)**.

Перечень инструкций

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxStatementsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxStatementsX":1}

Ниже перечислены в порядке, соответствующем латинскому алфавиту, все инструкции (за исключением инструкций SQL). Инструкции, используемые в Visual Basic для приложений (Visual Basic for Applications) имеют обозначение (VBA), а инструкции объектов доступа к данным (Data Access Objects) имеют обозначение (DAO).

A–M

AppActivate (VBA)
Beep (VBA)
Call (VBA)
ChDir (VBA)
ChDrive (VBA)
Close (VBA)
Const (VBA)
Date (VBA)
DDE
DDEExecute
DDEPoke
DDETerminate
DDETerminateAll
Declare (VBA)
DefBool (VBA)
DefByte (VBA)
DefCur (VBA)
DefDate (VBA)
DefDbl (VBA)
DefInt (VBA)
DefLng (VBA)
DefObj (VBA)
DefSng (VBA)
DefStr (VBA)
DefType (VBA)
DefVar (VBA)
DeleteControl
DeleteReportControl
DeleteSetting (VBA)
Dim (VBA)
Do...Loop (VBA)
End (VBA)
Erase (VBA)
Error (VBA)
Exit (VBA)
FileCopy (VBA)
For Each...Next (VBA)
For...Next (VBA)
Function (VBA)
Get (VBA)
GoSub...Return (VBA)
GoTo (VBA)
If...Then...Else (VBA)
Input # (VBA)
Kill (VBA)
Let (VBA)
Line Input # (VBA)
Lock (VBA)
LSet (VBA)
Mid (VBA)
MidB (VBA)
MkDir (VBA)

N–Z

Name (VBA)
On Error (VBA)
On...GoSub (VBA)
On...GoTo (VBA)
Open (VBA)
Option Base (VBA)
Option Compare (VBA)
Option Explicit (VBA)
Resume (VBA)
Return (VBA)
RmDir (VBA)
RSet (VBA)
SaveSetting (VBA)
Seek (VBA)
SelectCase (VBA)
SendKeys (VBA)

Option Private (VBA)
Print # (VBA)
Private (VBA)
Property Get (VBA)
Property Let (VBA)
Property Set (VBA)
Public (VBA)
Put (VBA)
Randomize (VBA)
ReDim (VBA)
Rem (VBA)
Reset (VBA)
Set (VBA)
SetAttr (VBA)
Static (VBA)
Stop (VBA)
Sub (VBA)
Time (VBA)
Type (VBA)
Unlock (VBA)
While...Wend (VBA)
Width # (VBA)
With (VBA)
Write # (VBA)

Перечень функций

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxFunctionsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxFunctionsX":1}

Ниже приводится полный список функций в порядке, соответствующем латинскому алфавиту. Двойной звездочкой (*) отмечены функции, добавленные в Microsoft Access 97. Функции, используемые в Visual Basic для приложений (Visual Basic for Applications) имеют обозначение (VBA), а функции объектов доступа к данным (Data Access Objects) имеют обозначение (DAO).

A–C

Abs (VBA)
Array (VBA)
Asc (VBA)
Atn (VBA)
Avg (DAO)
CBool (VBA)
CByte (VBA)
CCur (VBA)
CDate (VBA)
CDbl (VBA)
CDec (VBA)
Choose (VBA)
Chr (VBA)
CInt (VBA)
CLng (VBA)
CodeDb
Command
Cos (VBA)
Count (DAO)
CreateControl
CreateForm
CreateGroupLevel
CreateObject (VBA)
CreateReport
CreateReportControl
CSng (VBA)
CStr (VBA)
CurDir (VBA)
CurrentDb
CurrentUser
CVar (VBA)
CVDate (VBA)
CVErr (VBA)

D–H

Date (VBA)
DateAdd (VBA)
DateDiff (VBA)
DatePart (VBA)
DateSerial (VBA)
DateValue (VBA)
DAvg
Day (VBA)
DCount
DDB (VBA)
DDE
DDEInitiate
DDERequest
DDESend
DFirst
Dir (VBA)
DLast
DVar
DVarP
Environ (VBA)
EOF (VBA)
Error (VBA)
Eval
Exp (VBA)
FileAttr (VBA)
FileDateTime (VBA)
FileLen (VBA)
First (DAO)
Fix (VBA)
Format (VBA)
FreeFile (VBA)
FV (VBA)
GetAllSettings (VBA)
GetAttr (VBA)

DLookup
DMax
DMin
DoEvents (VBA)
DStDev
DStDevP
DSum

I–R

Iif (VBA)
IMEStatus (VBA)
Input (VBA)
InputBox (VBA)
InStr (VBA)
Int (VBA)
IPmt (VBA)
IRR (VBA)
IsArray (VBA)
IsDate (VBA)
IsEmpty (VBA)
IsError (VBA)
IsMissing (VBA)
IsNull (VBA)
IsNumeric (VBA)
IsObject (VBA)
Last (DAO)
LBound (VBA)
LCase (VBA)
Left (VBA)
Len (VBA)
LoadPicture
Loc (VBA)
LOF (VBA)
GetObject (VBA)
GetSetting (VBA)
GUIDFromString**
Hex (VBA)
Hour (VBA)
HyperlinkPart

Log (VBA)
LTrim (VBA)
Max (DAO)
Mid (VBA)
Min (DAO)
Minute (VBA)
MIRR (VBA)
Month (VBA)
MsgBox (VBA)
Now (VBA)
NPer (VBA)
NPV (VBA)
Nz
Oct (VBA)
Partition (VBA)
Pmt (VBA)
PPmt (VBA)
PV (VBA)
QBColor (VBA)
Rate (VBA)
RGB (VBA)
Right (VBA)
Rnd (VBA)
RTrim (VBA)

S–Z

Second (VBA)
Seek (VBA)
Sgn (VBA)
Shell (VBA)
Sin (VBA)
SLN (VBA)
Space (VBA)
Spc (VBA)
Sqr (VBA)
StDev (DAO)
SYD (VBA)
SysCmd
Tab (VBA)
Tan (VBA)
Time (VBA)
Timer (VBA)
TimeSerial (VBA)
TimeValue (VBA)
Trim (VBA)
TypeName (VBA)

StDevP (DAO)
Str (VBA)
StrComp (VBA)
StrConv (VBA)
String (VBA)
StringFromGUID**
Sum (DAO)
Switch (VBA)
UBound (VBA)
UCase (VBA)
Val (VBA)
Var (DAO)
VarP (DAO)
VarType (VBA)
Weekday (VBA)
Year (VBA)

Перечень методов

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxMethodsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxMethodsX":1}

Ниже приводится классификация всех методов по объектам или семействам и полный список методов в порядке, соответствующем латинскому алфавиту. Двойной звездочкой (*) отмечены методы, добавленные в Microsoft Access 97. Методы, используемые в Visual Basic для приложений (Visual Basic for Applications) имеют обозначение (VBA), а методы объектов доступа к данным (Data Access Objects) имеют обозначение (DAO).

По объектам или семействам

Application
Collection (VBA)
Connection, **Connections** (DAO)
Container, **Containers** (DAO)
Control
Controls
Database, **Databases** (DAO)
DBEngine (DAO)
Debug (VBA)
DoCmd
Document, **Documents** (DAO)
Err (VBA)
Error, **Errors** (DAO)
Field, **Fields** (DAO)
Form
Group, **Groups** (DAO)
Index, **Indexes** (DAO)
ItemsSelected
Module
Modules
Page
Pages
Parameter, **Parameters** (DAO)
Property, **Properties** (DAO)
QueryDef, **QueryDefs** (DAO)
Recordset, **Recordsets** (DAO)
Reference
References
Relation, **Relations** (DAO)
Report
Screen
TableDef, **TableDefs** (DAO)
User, **Users** (DAO)
Workspace, **Workspaces** (DAO)

По алфавиту

A–C

AccessError**
Add (VBA)
Add (семейство **Pages**)**
AddFromFile (объект **Module**)**
AddFromFile (семейство **References**)**
AddFromGUID**
AddFromString**
AddNew (DAO)
AddToFavorites**
Append (DAO)
AppendChunk (DAO)
ApplyFilter
Beep
BeginTrans (DAO)
BuildCriteria
Clone (DAO)
Close (DAO)
Close
CloseCurrentDatabase
CommitTrans (DAO)
CompactDatabase (DAO)
CopyObject
CopyQueryDef (DAO)
CreateDatabase (DAO)
CreateEventProc**
CreateField (DAO)
CreateGroup (DAO)
CreateIndex (DAO)
CreateProperty (DAO)
CreateQueryDef (DAO)

Cancel (DAO)
CancelEvent
CancelUpdate (DAO)
Circle
Clear (VBA)
CreateRelation (DAO)
CreateTableDef (DAO)
CreateUser (DAO)
CreateWorkspace (DAO)

D–F

DefaultControl
DefaultWorkspaceClone
Delete (DAO)
DeleteLines**
DeleteObject
DoMenuItem
Dropdown
Echo (объект **DoCmd**)
Echo (объект **Application**)
Edit (DAO)
Execute (DAO)
FillCache (DAO)
Find**
FindFirst (DAO)
FindLast (DAO)
FindNext (DAO)
FindNext
FindPrevious (DAO)
FindRecord
Follow**
FollowHyperlink**

G–M

GetChunk (DAO)
GetOption
GetRows (DAO)
GoToControl
GoToPage (объект **DoCmd**)
GoToPage (объект **Form**)
GoToRecord
Hourglass
Idle (DAO)
InsertLines**
InsertText**
Item (VBA)
Line
MakeReplica (DAO)
Maximize
Minimize
Move (DAO)
MoveFirst (DAO)
MoveLast (DAO)
MoveNext (DAO)
MovePrevious (DAO)
MoveSize

N–Q

NewCurrentDatabase
NewPassword (DAO)
NextRecordSet (DAO)
OpenConnection (DAO)
OpenCurrentDatabase
OpenDatabase (DAO)
OpenForm
OpenModule
OpenQuery
OpenRecordset (DAO)
OpenReport
OpenTable
OutputTo
PopulatePartial (DAO)
Print (VBA)
Print
PrintOut
PSet
Quit (объект **DoCmd**)
Quit (объект **Application**)

R

Raise (VBA)
RepaintObject

Recalc
Refresh (DAO)
Refresh
RefreshDatabaseWindow**
RefreshLink (DAO)
RefreshTitleBar
RegisterDatabase (DAO)
Remove (VBA)
Remove (семейство **Pages**)**
Remove (семейство **References**)**
Rename
Repaint
RepairDatabase (DAO)
ReplaceLine**
Requery (DAO)
Requery (объект **DoCmd**)
Requery (объекты **Control** или **Form**)
Restore
Rollback (DAO)
Run
RunCommand**
RunMacro
RunSQL

S–Z

Save
Scale
Seek (DAO)
SelectObject
SendObject
SetFocus
SetMenuItem
SetOption (DAO)
SetOption
SetWarnings
ShowAllRecords
ShowToolbar
SizeToFit
Synchronize (DAO)
TextHeight
TextWidth
TransferDatabase
TransferSpreadsheet
TransferText
Undo
Update (DAO)

Перечень свойств

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxPropertiesX":1}

Ниже приводится классификация свойств по объектам и семействам, а также список свойств в алфавитном порядке (по английским именам свойств). Звездочкой (*) отмечены свойства, добавленные в Microsoft Access 95. Двойная звездочка (**) указывает, что свойство добавлено в Microsoft Access 97. Свойства, используемые в Visual Basic для приложений (Visual Basic for Applications) имеют обозначение (VBA), а свойства объектов доступа к данным (Data Access Objects) имеют обозначение (DAO).

По объектам или семействам

Application
Collection (VBA)
Connection, **Connections** (DAO)
Container, **Containers** (DAO)
Control
Controls
Database, **Databases** (DAO)
DBEngine (DAO)
Debug (VBA)
DoCmd
Document, **Documents** (DAO)
Err (VBA)
Error, **Errors** (DAO)
Field, **Fields** (DAO)
Form
Forms
Group, **Groups** (DAO)
Index, **Indexes** (DAO)
ItemsSelected
Module
Modules
Parameter, **Parameters** (DAO)
Property, **Properties** (DAO)
Query
QueryDef, **QueryDefs** (DAO)
Recordset, **Recordsets** (DAO)
Reference
References
Relation, **Relations** (DAO)
Report
Reports
Screen
Table
TableDef, **TableDefs** (DAO)
User, **Users** (DAO)
Workspace, **Workspaces** (DAO)

По алфавиту

A

AbsolutePosition (DAO)*
Action
ActiveControl
ActiveDatasheet*
ActiveForm
ActiveReport
Подписи с двоеточием (AddColon)
После подтверждения Del (AfterDelConfirm)
После вставки (AfterInsert)
После обновления (AfterUpdate)
Псевдоним (Alias)
Разрешить добавление (AllowAdditions)*
Разрешить автозамену (AllowAutoCorrect)*
AllowFullMenus*
AllowShortcutMenus*
AllowSpecialKeys*
AllowToolbarChanges*
Пустые строки (AllowZeroLength) (DAO)
Пустые строки (AllowZeroLength)
AllPermissions (DAO)*
AppIcon*
Application*
AppTitle*
Assistant**
Attributes (DAO)
Автоматический запуск (AutoActivate)

AllowBreakIntoCode*
AllowBuiltInToolbars*
AllowBypassKey*
Разрешить удаление (AllowDeletions)*
Разрешить изменение (AllowEdits)*
Применение фильтров (AllowFilters)

B

Цвет фона (BackColor)
Тип фона (BackStyle)
BatchCollisionCount (DAO)
BatchCollisions (DAO)
BatchSize (DAO)
До подтверждения Del (BeforeDelConfirm)
До вставки (BeforeInsert)
До обновления (BeforeUpdate)
BOF (DAO)
Выравнивание по центру (AutoCenter)
Автоподстановка (AutoExpand)
Добавление подписи (AutoLabel)
Автоматический повтор (AutoRepeat)
Автоматический размер (AutoResize)
Автопереход по Tab (AutoTab)
Bookmark (DAO)
Bookmarkable (DAO)
Bookmark
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Присоединенный столбец (BoundColumn)
BuiltIn**

C

CacheSize (DAO)
CacheStart (DAO)
Отмена (Cancel)
Расширение (CanGrow)
Сжатие (CanShrink)
Подпись (Caption)
Класс (Class)
Кнопка закрытия (CloseButton)*
Clustered (DAO)
CodeContextObject*
CollatingOrder (DAO)
Column
Число столбцов (ColumnCount)
Заголовки столбцов (ColumnHeadings)
Заглавия столбцов (ColumnHeads)
ColumnHidden
ColumnOrder
ColumnWidth
Ширина столбцов (ColumnWidths)
CommandBars**
ConflictTable (DAO)*
Connect (DAO)
Connection (DAO)**
Container (DAO)
Кнопка оконного меню (ControlBox)
Данные (ControlSource)
Всплывающая подсказка (ControlTipText)*
ControlType*
Count (DAO)
Count (VBA)
Count
CountOfDeclarationLines**
CountOfLines**
CurrentObjectName
CurrentObjectType
CurrentRecord*
CurrentSectionLeft
CurrentSectionTop
CurrentView
CurrentX
CurrentY
Цикл табуляции (Cycle)*

D

Database Properties*
Database (DAO)
Ввод данных (DataEntry)*
DatasheetBackColor**
DefaultUser (DAO)*
Значение по умолчанию (DefaultValue) (DAO)
Значение по умолчанию (DefaultValue)
Режим по умолчанию (DefaultView)

DatasheetCellsEffect*
DatasheetFontHeight
DatasheetFontItalic
DatasheetFontName
DatasheetFontUnderline
DatasheetFontWeight
DatasheetForeColor**
DatasheetGridLinesBehavior*
DatasheetGridLinesColor*
Тип данных (DataType)
DataUpdatable (DAO)
DateCreated (DAO)
Группировка по датам (DateGrouping)
DBEngine*
Число десятичных знаков (DecimalPlaces)
По умолчанию (Default)
DefaultCursorDriver (DAO)**
DefaultType (DAO)**
Описание (Description) (DAO)*
Описание (Description) (VBA)*
Описание (Description)
DesignMasterID (DAO)
Строка подключения-получатель (DestConnectStr)
База данных-получатель (DestinationDB)
Таблица-получатель (DestinationTable)
Direction (DAO)**
Dirty
Тип элемента управления (DisplayControl)*
Тип вывода (DisplayType)
Режим вывода (DisplayWhen)
DistinctCount (DAO)*
Разделительные линии (DividingLines)*
DrawMode
DrawStyle
DrawWidth

E–F

EditMode (DAO)*
Доступ (Enabled)
Поведение по Enter (EnterKeyBehavior)
EOF (DAO)
EventProcPrefix
FailOnError**
Для лазерного принтера (FastLaserPrinting)
Имя поля (FieldName)
Размер поля (FieldSize) (DAO)**
Размер поля (FieldSize)
FillColor
FillStyle
Фильтр (Filter) (DAO)
Фильтр (Filter)
Применение автофильтра (FilterLookup)*
Фильтр включен (FilterOn)*
FontBold
Наклонный (FontItalic)
Шрифт (FontName)
Размер шрифта (FontSize)
Подчеркнутый (FontUnderline)
Насыщенность (FontWeight)
Конец страницы (ForceNewPage)
Цвет текста (ForeColor)
Foreign (DAO)
ForeignName (DAO)
ForeignTable (DAO)
Form
Формат поля (Format)
Формат поля (Format)—Дата/время
Формат поля (Format)—Числовой и денежный
Формат поля (Format)—Текстовый и поле MEMO
Формат поля (Format)—Логический
FormatCount
FrozenColumns
FullPath**

G–H

Число делений по X (GridX)
Число делений по Y (GridY)
Высота (Height)
HelpContext (DAO)*

Примечание группы (GroupFooter)
Заголовок группы (GroupHeader)
Интервал (GroupInterval)
Уровень группировки (GroupLevel)
Группировка (GroupOn)
Неразрывная группа (GrpKeepTogether)*
GUID**
HasContinued*
HasData*
HasModule**
HelpContext (VBA)
HelpContextID (Идентификатор справки)
Файл справки (HelpFile) (DAO)*
Файл справки (HelpFile)
Не выводить повторы (HideDuplicates)
hWndAccessApp*
hWnd
Hyperlink**
HyperlinkAddress**
HyperlinkSubAddress**

I–K

IgnoreNulls (DAO)
IgnoreNulls
ImageHeight*
ImageWidth*
Index (DAO)
Индексированное поле (Indexed)
Inherit (DAO)*
Inherited (DAO)
IniPath (DAO)*
Маска ввода (InputMask)
InSelection
InsideHeight*
InsideWidth*
IsBroken**
IsolateODBCTrans (DAO)*
IsVisible**
Элемент (Item) (элемент управления-рамка объекта)
Элемент (Item) (Семейства)**
ItemData
KeepLocal (DAO)*
Не разрывать (KeepTogether) — Группы
Не разрывать (KeepTogether) — Разделы
Перехват нажатия клавиш (KeyPreview)*
Kind**

L

Выравнивание подписи (LabelAlign)
Позиция подписи X (LabelX)
Позиция подписи Y (LabelY)
LastDLLError (VBA)
LastModified (DAO)
LastUpdated (DAO)
Формат для печати (LayoutForPrint)
От левого края (Left)
Ограничиться списком (LimitToList)
Lines**
Наклон линии (LineSlant)
Подчиненные поля (LinkChildFields)
Основные поля (LinkMasterFields)
ListCount
ListIndex
Число строк списка (ListRows)
Ширина списка (ListWidth)
Блокировка (Locked)
LockEdits (DAO)
LoginTimeout (DAO)
Таблица сообщений (LogMessages) (DAO)
Таблица сообщений (LogMessages)
lpOLEObject

M–N

Major**
MaxRecords (DAO)**
MaxRecords**
MDE**
Me
MultiRow**
Несвязное выделение (MultiSelect)*
Имя (Name) (DAO)
Имя (Name)
Поле номера записи (NavigationButtons)

Строка меню (MenuBar)
Кнопки размеров окна (MinMaxButtons)*
Minor**
Модальное окно (Modal)
Module
MousePointer**
MoveLayout

O

Object Properties*
Object
ObjectPalette
ObjectVerbs
ObjectVerbsCount
Строка подключения ODBC (ODBCConnectStr)
Время ожидания ODBC (ODBCTimeout) (DAO)
Время ожидания ODBC (ODBCTimeout)
OldValue
Класс OLE (OLEClass)
OLEData **
Тип OLE (OLEType)
Допустимый тип OLE (OLETypeAllowed)
Включение (OnActivate)
Изменение (OnChange)
Нажатие кнопки (OnClick)
Закрытие (OnClose)
Текущая запись (OnCurrent)
Двойное нажатие кнопки (OnDblClick)
OnDeactivate (Отключение)
Удаление (OnDelete)
Вход (OnEnter)
Ошибка (OnError)
Выход (OnExit)
NewRecord*
Новая строка или столбец (NewRowOrCol)
Новые значения (NewValues)*
NoMatch (DAO)
Number (DAO)*
Number (VBA)
Форматирование (OnFormat)
Потеря фокуса (OnLostFocus)
Кнопка вниз (OnMouseDown)
Перемещение указателя (OnMouseMove)
Кнопка вверх (OnMouseUp)
Отсутствие данных (OnNoData)
Отсутствие в списке (OnNotInList)
Открытие (OnOpen)
Страница (OnPage)
Печать (OnPrint)
Изменение размера (OnResize)
Возврат (OnRetreat)
Таймер (OnTimer)
Выгрузка (OnUnload)
При обновлении (OnUpdated)
OpenArgs
Значение параметра (OptionValue)
Порядок сортировки (OrderBy)*
Сортировка включена (OrderByOn)*
OrdinalPosition (DAO)
OriginalValue (DAO)
Вывод всех полей (OutputAllFields)
Owner (DAO)

P

Page
Нижний колонтитул (PageFooter)
Верхний колонтитул (PageHeader)
PageIndex**
Pages
Painting
PaintPalette
Источник палитры (PaletteSource)
Parent
PartialReplica (DAO)
Тип рисунка (PictureType)*
PID (DAO)
Всплывающее окно (PopUp)
Prepare (DAO)
PreviousControl
Primary (DAO)
Primary
PrintCount
PrintSection
ProcBodyLine**

Password (DAO)*
PercentPosition (DAO)
Permissions (DAO)
Рисунок (Picture)
Выравнивание рисунка (PictureAlignment)*
PictureData*
Страницы с рисунком (PicturePages)*
Масштабы рисунка (PictureSizeMode)*
Мозаичное заполнение (PictureTiling)*
ProcCountLines**
ProcOfLine**
ProcStartLine**
ProjectName**
Properties**
PrtDevMode
PrtDevNames
PrtMip

Q–R

QueryTimeout (DAO)
RecordCount (DAO)
Блокировка записей (RecordLocks)
RecordsAffected (DAO)*
Область выделения (RecordSelectors)
RecordsetClone
Тип набора записей (RecordsetType)*
Источник записей (RecordSource)
Повторение раздела (RepeatSection)*
Replicable (DAO)*
ReplicaFilter (DAO)*
ReplicaID (DAO)*
ReplicationConflictFunction*
Report
Обязательное поле (Required) (DAO)
Обязательное поле (Required)
Restartable (DAO)
Возврат записей (ReturnsRecords) (DAO)
Возврат записей (ReturnsRecords)
RowHeight
Источник строк (RowSource)
Тип источника строк (RowSourceType)
Сумма с накоплением (RunningSum)
При запуске представляют (RunPermissions)

S

ScaleHeight
ScaleLeft
ScaleMode
ScaleTop
ScaleWidth
Полосы прокрутки (ScrollBars)
Section
Selected*
SelHeight*
SelLeft*
SelLength
SelStart
SelText
SelTop*
SelWidth*
Контекстные меню (ShortcutMenu)
Контекстное меню (ShortcutMenuBar)*
Size (DAO)
Установка размеров (SizeMode)
Sort (DAO)
Источник (Source) (VBA)
Источник (Source)
Строка подключения-источник (SourceConnectStr)
База данных-источник (SourceDatabase)
Документ-источник (SourceDoc)
SourceField (DAO)
Источник данных (SourceItem)
Объект-источник (SourceObject)
SourceTable (DAO)
SourceTableName (DAO)
Оформление (SpecialEffect)
SQL (DAO)
StartupForm*
StartupMenuBar*
StartupShortcutMenuBar*
StartupShowDBWindow*
StartupShowStatusBar*
Текст строки состояния (StatusBarText)
StillExecuting (DAO)**
Style**

Сортировка (SortOrder)
Источник (Source) (DAO)*
SystemDB (DAO)*

T–U

TabFixedHeight**
TabFixedWidth**
Индекс перехода по Tab (TabIndex)
Table (DAO)
Переход по Tab (TabStop)
Дополнительные сведения (Tag)
Текст (Text)
Выравнивание текста (TextAlign)
Интервал таймера (TimerInterval)
Toolbar**
От верхнего края (Top)
Набор значений (TopValues)
Transactions (DAO)**
Прозрачный (Transparent)
Тройное состояние (TripleState)*
Type (DAO)
Type
Unique (DAO)
Unique
Уникальные записи (UniqueRecords)
Уникальные значения (UniqueValues)
Updatable (DAO)
Параметры обновления (UpdateOptions) (DAO)
Параметры обновления (UpdateOptions)
UserControl*
UserName (DAO)
UseTransaction**

V–Z

V1xNullBehavior (DAO)
ValidateOnSet (DAO)
Условие на значение (ValidationRule) (DAO)
Условие на значение (ValidationRule)
Сообщение об ошибке (ValidationText) (DAO)
Сообщение об ошибке (ValidationText)
Значение (Value) (DAO)
Значение (Value)
Команда (Verb)
Version (DAO)
Допустимые режимы (ViewsAllowed)
Вывод на экран (Visible)
Вывод на экран (Visible) (объект **Application**)*
VisibleValue (DAO)
Кнопка контекстной справки (WhatsThisButton) (VBA)*
Кнопка контекстной справки (WhatsThisButton)*
Ширина (Width)
WillContinue*
WindowHeight
WindowWidth

Свойства запросов

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxQueryPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxQueryPropertiesX":1}

Каждому запросу соответствует набор свойств, с помощью которых пользователь определяет функциональные характеристики и представление результатов запроса.

Для того чтобы получить подробное описание свойства запроса, выберите имя свойства в приведенном ниже списке. Звездочкой (*) отмечены свойства полей запроса или списков полей. Свойства, имена которых приведены без звездочки, относятся к запросу в целом.

Псевдоним (Alias)*
Подпись (Caption)*
Заголовки столбцов (ColumnHeadings)
Описание (Description)
Строка подключения-получатель (DestConnectStr)
База данных-получатель (DestinationDB)
Таблица-получатель (DestinationTable)
Тип элемента управления (DisplayControl)*
FailOnError
Фильтр (Filter)
Формат поля (Format)*
FrozenColumns
Маска ввода (InputMask)*
Таблица сообщений (LogMessages)
MaxRecords
Строка подключения ODBC (ODBCConnectStr)
Время ожидания ODBC (ODBCTimeout)
Порядок сортировки (OrderBy)
Сортировка включена (OrderByOn)
Вывод всех полей (OutputAllFields)
Блокировка записей (RecordLocks)
Тип набора записей (RecordsetType)
Возврат записей (ReturnsRecords)
При запуске представляют (RunPermissions)
Источник (Source)*
Строка подключения-источник (SourceConnectStr)
База данных-источник (SourceDatabase)
Набор значений (TopValues)
Уникальные записи (UniqueRecords)
Уникальные значения (UniqueValues)
UseTransaction

Свойства полей

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxFieldPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxFieldPropertiesX":1}

Каждому полю таблицы соответствует набор свойств, с помощью которых пользователь определяет вид и функциональные характеристики поля. Набор свойств конкретного поля определяется типом данных этого поля.

Для того чтобы получить подробное описание свойства поля, выберите имя свойства в приведенном ниже списке.

Пустые строки (AllowZeroLength)
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Присоединенный столбец (BoundColumn)
Подпись (Caption)
Число столбцов (ColumnCount)
Заглавия столбцов (ColumnHeads)
ColumnHidden
ColumnOrder
ColumnWidth
Ширина столбцов (ColumnWidths)
DatasheetCellsEffect
DatasheetGridlinesBehavior
Тип данных (DataType)
Число десятичных знаков (DecimalPlaces)
Значение по умолчанию (DefaultValue)
Описание (Description)
Тип элемента управления (DisplayControl)
Имя поля (FieldName)
Размер поля (FieldSize)
ForeColor
Формат поля (Format)
FrozenColumns
IgnoreNulls
Индексированное поле (Indexed)
Маска ввода (InputMask)
Ограничиться списком (LimitToList)
Число строк списка (ListRows)
Ширина списка (ListWidth)
Новые значения (NewValues)
Primary
PrtDevModes
PrtDevNames
PrtMip
Обязательное поле (Required)
Источник строк (RowSource)
Тип источника строк (RowSourceType)
SelHeight
SelTop
Сортировка (SortOrder)
Оформление (SpecialEffect)
Unique
Условие на значение (ValidationRule)
Сообщение об ошибке (ValidationText)

Свойства таблиц

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxTablePropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxTablePropertiesX":1}

Каждой таблице соответствует набор свойств, с помощью которых пользователь определяет вид и функциональные характеристики таблицы.

Для того чтобы получить подробное описание свойства таблицы, выберите имя свойства в приведенном ниже списке.

Описание (Description)

Фильтр (Filter)

Порядок сортировки (OrderBy)

Условие на значение (ValidationRule)

Сообщение об ошибке (ValidationText)

Свойства элементов управления

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxControlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxControlPropertiesX":1}

Каждому элементу управления соответствует набор свойств, с помощью которых пользователь определяет вид и функциональные характеристики элемента управления.

Пользователь имеет возможность изменить используемые по умолчанию значения свойств элемента управления перед добавлением элемента управления в форму или отчет. Допускается также изменение значений свойств элементов управления, размещенных в форме или отчете.

Для получения подробного описания свойств элемента управления выберите в следующем списке имя этого элемента управления.

ActiveX
Выключатель
Группа
Диаграмма
Кнопка
Конец страницы
Линия
Набор вкладок
Надпись
Переключатель
Подчиненная форма/отчет
Поле
Поле со списком
Присоединенная рамка объекта
Прямоугольник
Рисунок
Свободная рамка объекта
Список
Флажок

Свойства присоединенной рамки объекта

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxBoundObjectFramePropC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxBoundObjectFramePropX":1}

В следующем списке содержатся имена свойств элемента управления присоединенная рамка объекта. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в следующем списке имя свойства. Для каждого свойства будут выведены условия, при которых оно применяется, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

A-O

Подписи с двоеточием (AddColon)
После обновления (AfterUpdate) (формы)
Автоматический запуск (AutoActivate) (формы)
Добавление подписи (AutoLabel)
Цвет фона (BackColor)
Тип фона (BackStyle)
До обновления (BeforeUpdate) (формы)
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Класс (Class)
Данные (ControlSource)
Всплывающая подсказка (ControlTipText)
ControlType
Тип вывода (DisplayType)
Режим вывода (DisplayWhen) (формы)
Доступ (Enabled) (формы)
Высота (Height)
Идентификатор справки (HelpContextID) (формы)
InSelection (формы)
Выравнивание подписи (LabelAlign)
Позиция подписи X (LabelX)
Позиция подписи Y (LabelY)
От левого края (Left)
Блокировка (Locked) (формы)
Имя (Name)
Допустимый тип OLE (OLETypeAllowed)
Нажатие кнопки (OnClick) (формы)
Двойное нажатие кнопки (OnDblClick) (формы)
Вход (OnEnter) (формы)
Выход (OnExit) (формы)
Получение фокуса (OnGotFocus) (формы)
Клавиша вниз (OnKeyDown) (формы)
Нажатие клавиши (OnKeyPress) (формы)
Клавиша вверх (OnKeyUp) (формы)
Потеря фокуса (OnLostFocus) (формы)
Кнопка вниз (OnMouseDown) (формы)
Перемещение указателя (OnMouseMove) (формы)
Кнопка вверх (OnMouseUp) (формы)
При обновлении (OnUpdated) (формы)

P-Z

Контекстное меню (ShortcutMenuBar)
Установка размеров (SizeMode)
Документ-источник (SourceDoc)
Источник данных (SourceItem)
Оформление (SpecialEffect)
Текст строки состояния (StatusBarText) (формы)
Индекс перехода по Tab (TabIndex) (формы)
Переход по Tab (TabStop) (формы)
Дополнительные сведения (Tag)
От верхнего края (Top)
Параметры обновления (UpdateOptions)
Команда (Verb)
Вывод на экран (Visible)
Ширина (Width)

Свойства флажка

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxCheckBoxPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxCheckBoxPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-флажка. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Подписи с двоеточием (AddColon)

После обновления (AfterUpdate) (формы)

Добавление подписи (AutoLabel)

До обновления (BeforeUpdate) (формы)

Цвет границы (BorderColor)

Тип границы (BorderStyle)

Ширина границы (BorderWidth)

Данные (ControlSource)

Всплывающая подсказка (ControlTipText)

ControlType

Значение по умолчанию (DefaultValue) (формы)

Режим вывода (DisplayWhen) (формы)

Доступ (Enabled) (формы)

Высота (Height)

Идентификатор справки (HelpContextID) (формы)

InSelection (формы)

Выравнивание подписи (LabelAlign)

Позиция подписи X (LabelX)

Позиция подписи Y (LabelY)

От левого края (Left)

Блокировка (Locked) (формы)

Имя (Name)

Нажатие кнопки (OnClick) (формы)

Двойное нажатие кнопки (OnDblClick) (формы)

Вход (OnEnter) (формы)

Выход (OnExit) (формы)

Получение фокуса (OnGotFocus) (формы)

Клавиша вниз (OnKeyDown) (формы)

Нажатие клавиши (OnKeyPress) (формы)

Клавиша вверх (OnKeyUp) (формы)

Потеря фокуса (OnLostFocus) (формы)

Кнопка вниз (OnMouseDown) (формы)

Перемещение указателя (OnMouseMove) (формы)

Кнопка вверх (OnMouseUp) (формы)

Контекстное меню (ShortcutMenuBar)

Оформление (SpecialEffect)

Текст строки состояния (StatusBarText) (формы)

Индекс перехода по Tab (TabIndex) (формы)

Переход по Tab (TabStop) (формы)

Дополнительные сведения (Tag)

От верхнего края (Top)

Тройное состояние (TripleState)

Условие на значение (ValidationRule) (формы)

Сообщение об ошибке (ValidationText) (формы)

Вывод на экран (Visible)

Ширина (Width)

Свойства поля со списком

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxComboBoxPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acidxComboBoxPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления поле со списком. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

A–M

Подписи с двоеточием (AddColon)
После обновления (AfterUpdate) (формы)
Разрешить автозамену (AllowAutoCorrect)
Автоподстановка (AutoExpand)
Добавление подписи (AutoLabel)
Цвет фона (BackColor)
Тип фона (BackStyle)
До обновления (BeforeUpdate) (формы)
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Присоединенный столбец (BoundColumn)
Column
Число столбцов (ColumnCount)
Заглавия столбцов (ColumnHeads)
Ширина столбцов (ColumnWidths)
Данные (ControlSource)
Всплывающая подсказка (ControlTipText) (формы)
ControlType
Число десятичных знаков (DecimalPlaces)
Значение по умолчанию (DefaultValue) (формы)
Режим вывода (DisplayWhen) (формы)
Доступ (Enabled) (формы)
Наклонный (FontItalic)
Шрифт (FontName)
Размер шрифта (FontSize)
Подчеркнутый (FontUnderline)
Насыщенность (FontWeight)
Цвет текста (ForeColor)
Формат поля (Format)
Высота (Height)
Идентификатор справки (HelpContextID) (формы)
Маска ввода (InputMask)
InSelection (формы)
ItemData
Выравнивание подписи (LabelAlign)
Позиция подписи X (LabelX)
Позиция подписи Y (LabelY)
От левого края (Left)
Ограничиться списком (LimitToList)
Число строк списка (ListRows)
Ширина списка (ListWidth)
Блокировка (Locked) (формы)

N–Z

Имя (Name)
Изменение (OnChange) (формы)
Нажатие кнопки (OnClick) (формы)
Двойное нажатие кнопки (OnDblClick) (формы)
Вход (OnEnter) (формы)
Выход (OnExit) (формы)
Получение фокуса (OnGotFocus) (формы)
Источник строк (RowSource)
Тип источника строк (RowSourceType)
Контекстное меню (ShortcutMenuBar)
Оформление (SpecialEffect)
Текст строки состояния (StatusBarText) (формы)
Индекс перехода по Tab (TabIndex) (формы)
Переход по Tab (TabStop) (формы)

Клавиша вниз (OnKeyDown) (формы)
Нажатие клавиши (OnKeyPress) (формы)
Клавиша вверх (OnKeyUp) (формы)
Потеря фокуса (OnLostFocus) (формы)

Кнопка вниз (OnMouseDown) (формы)

Перемещение указателя (OnMouseMove) (формы)
Кнопка вверх (OnMouseUp) (формы)
Отсутствие в списке (OnNotInList) (формы)

Дополнительные сведения (Tag)
Выравнивание текста (TextAlign)
От верхнего края (Top)
Условие на значение (ValidationRule) (формы)

Сообщение об ошибке (ValidationText) (формы)

Вывод на экран (Visible)

Ширина (Width)

Свойства кнопки

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxCommandButtonPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxCommandButtonPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-кнопки. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Подписи с двоеточием (AddColon)
Добавление подписи (AutoLabel)
Автоматический повтор (AutoRepeat) (формы)
Отмена (Cancel) (формы)
Подпись (Caption)
Всплывающая подсказка (ControlTipText) (формы)
ControlType
По умолчанию (Default)
Режим вывода (DisplayWhen) (формы)
Доступ (Enabled) (формы)
Наклонный (FontItalic)
Шрифт (FontName)
Размер шрифта (FontSize)
Подчеркнутый (FontUnderline)
Насыщенность (FontWeight)
Цвет текста (ForeColor)
Высота (Height)
Идентификатор справки (HelpContextID) (формы)
HyperlinkAddress
HyperlinkSubAddress
InSelection (формы)
Выравнивание подписи (LabelAlign)
Позиция подписи X (LabelX)
Позиция подписи Y (LabelY)
От левого края (Left)
Имя (Name)
Нажатие кнопки (OnClick) (формы)
Двойное нажатие кнопки (OnDblClick) (формы)
Вход (OnEnter) (формы)
Выход (OnExit) (формы)
Получение фокуса (OnGotFocus) (формы)
Клавиша вниз (OnKeyDown) (формы)
Нажатие клавиши (OnKeyPress) (формы)
Клавиша вверх (OnKeyUp) (формы)
Потеря фокуса (OnLostFocus) (формы)
Кнопка вниз (OnMouseDown) (формы)
Перемещение указателя (OnMouseMove) (формы)
Кнопка вверх (OnMouseUp) (формы)
Рисунок (Picture)
Тип рисунка (PictureType)
Контекстное меню (ShortcutMenuBar)
Текст строки состояния (StatusBarText) (формы)
Индекс перехода по Tab (TabIndex) (формы)
Переход по Tab (TabStop) (формы)
Дополнительные сведения (Tag)
От верхнего края (Top)
Прозрачный (Transparent)
Вывод на экран (Visible)
Ширина (Width)

Свойства диаграммы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxGraphControlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxGraphControlPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-диаграммы. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Подписи с двоеточием (AddColon)
После обновления (AfterUpdate) (формы)
Автоматический запуск (AutoActivate) (формы)
Добавление подписи (AutoLabel)
Цвет фона (BackColor)
Тип фона (BackStyle)
До обновления (BeforeUpdate) (формы)
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Класс (Class)
Данные (ControlSource)
Всплывающая подсказка (ControlTipText)
ControlType
Тип вывода (DisplayType)
Режим вывода (DisplayWhen) (формы)
Доступ (Enabled) (формы)
Высота (Height)
Идентификатор справки (HelpContextID) (формы)
InSelection (формы)
Выравнивание подписи (LabelAlign)
Позиция подписи X (LabelX)
Позиция подписи Y (LabelY)
От левого края (Left)
Блокировка (Locked) (формы)
Имя (Name)
Допустимый тип OLE (OLETypeAllowed)
Нажатие кнопки (OnClick) (формы)
Двойное нажатие кнопки (OnDblClick) (формы)
Вход (OnEnter) (формы)
Выход (OnExit) (формы)
Получение фокуса (OnGotFocus) (формы)
Клавиша вниз (OnKeyDown) (формы)
Нажатие клавиши (OnKeyPress) (формы)
Клавиша вверх (OnKeyUp) (формы)
Потеря фокуса (OnLostFocus) (формы)
Кнопка вниз (OnMouseDown) (формы)
Перемещение указателя (OnMouseMove) (формы)
Кнопка вверх (OnMouseUp) (формы)
При обновлении (OnUpdated) (формы)
Контекстное меню (ShortcutMenuBar)
Установка размеров (SizeMode)
Документ-источник (SourceDoc)
Источник данных (SourceItem)
Оформление (SpecialEffect)
Текст строки состояния (StatusBarText) (формы)
Индекс перехода по Tab (TabIndex) (формы)
Переход по Tab (TabStop) (формы)
Дополнительные сведения (Tag)
От верхнего края (Top)
Параметры обновления (UpdateOptions)
Команда (Verb)
Вывод на экран (Visible)
Ширина (Width)

Свойства рисунка

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acidxImagePropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acidxImagePropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-рисунка. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Цвет фона (BackColor)

Тип фона (BackStyle)

Цвет границы (BorderColor)

Тип границы (BorderStyle)

Ширина границы (BorderWidth)

Всплывающая подсказка (ControlTipText) (формы)

ControlType

Режим вывода (DisplayWhen) (формы)

Высота (Height)

Идентификатор справки (HelpContextID)

HyperlinkAddress

HyperlinkSubAddress

InSelection (формы)

От левого края (Left)

Имя (Name)

Нажатие кнопки (OnClick) (формы)

Двойное нажатие кнопки (OnDblClick) (формы)

Кнопка вниз (OnMouseDown) (формы)

Перемещение указателя (OnMouseMove) (формы)

Кнопка вверх (OnMouseUp) (формы)

Рисунок (Picture)

Выравнивание рисунка (PictureAlignment)

Мозаичное заполнение (PictureTiling) (формы)

Тип рисунка (PictureType)

Контекстное меню (ShortcutMenuBar)

Установка размеров (SizeMode)

Оформление (SpecialEffect)

Дополнительные сведения (Tag)

От верхнего края (Top)

Вывод на экран (Visible)

Ширина (Width)

Свойства надписи

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxLabelPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxLabelPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-надписи. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Цвет фона (BackColor)
Тип фона (BackStyle)
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Подпись (Caption)
Всплывающая подсказка (ControlTipText) (формы)
ControlType
Режим вывода (DisplayWhen) (формы)
Наклонный (FontItalic)
Шрифт (FontName)
Размер шрифта (FontSize)
Подчеркнутый (FontUnderline)
Насыщенность (FontWeight)
Цвет текста (ForeColor)
Высота (Height)
Идентификатор справки (HelpContextID)
HyperlinkAddress
HyperlinkSubAddress
InSelection (формы)
От левого края (Left)
Имя (Name)
Нажатие кнопки (OnClick) (формы)
Двойное нажатие кнопки (OnDblClick) (формы)
Кнопка вниз (OnMouseDown) (формы)
Перемещение указателя (OnMouseMove) (формы)
Кнопка вверх (OnMouseUp) (формы)
Контекстное меню (ShortcutMenuBar)
Оформление (SpecialEffect)
Дополнительные сведения (Tag)
Выравнивание текста (TextAlign)
От верхнего края (Top)
Вывод на экран (Visible)
Ширина (Width)

Свойства линии

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxLinePropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxLinePropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-линии. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
ControlType
Режим вывода (DisplayWhen) (формы)
Высота (Height)
InSelection (формы)
От левого края (Left)
Наклон линии (LineSlant)
Имя (Name)
Оформление (SpecialEffect)
Дополнительные сведения (Tag)
От верхнего края (Top)
Вывод на экран (Visible)
Ширина (Width)

Свойства списка

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxListBoxControlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxListBoxControlPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-списка. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

A–M

Подписи с двоеточием (AddColon)
После обновления (AfterUpdate) (формы)
Добавление подписи (AutoLabel)
Цвет фона (BackColor)
До обновления (BeforeUpdate) (формы)
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Присоединенный столбец (BoundColumn)
Column (формы)
Число столбцов (ColumnCount)
Заглавия столбцов (ColumnHeads)
Ширина столбцов (ColumnWidths)
Данные (ControlSource)
Всплывающая подсказка (ControlTipText) (формы)
ControlType
Значение по умолчанию (DefaultValue) (формы)
Режим вывода (DisplayWhen) (формы)
Доступ (Enabled) (формы)
Наклонный (FontItalic)
Шрифт (FontName)
Размер шрифта (FontSize)
Подчеркнутый (FontUnderline)
Насыщенность (FontWeight)
Цвет текста (ForeColor)
Высота (Height)
Идентификатор справки (HelpContextID) (формы)
InSelection (формы)
ItemData
Выравнивание подписи (LabelAlign)
Позиция подписи X (LabelX)
Позиция подписи Y (LabelY)
От левого края (Left)
Блокировка (Locked) (формы)
Несвязное выделение (MultiSelect)

N–Z

Имя (Name)
Нажатие кнопки (OnClick) (формы)
Двойное нажатие кнопки (OnDblClick) (формы)
Вход (OnEnter) (формы)
Выход (OnExit) (формы)
Получение фокуса (OnGotFocus) (формы)
Клавиша вниз (OnKeyDown) (формы)
Нажатие клавиши (OnKeyPress) (формы)
Клавиша вверх (OnKeyUp) (формы)
Потеря фокуса (OnLostFocus) (формы)
Источник строк (RowSource)
Тип источника строк (RowSourceType)
Контекстное меню (ShortcutMenuBar)
Оформление (SpecialEffect)
Текст строки состояния (StatusBarText) (формы)
Индекс перехода по Tab (TabIndex) (формы)
Переход по Tab (TabStop) (формы)
Дополнительные сведения (Tag)
От верхнего края (Top)
Условие на значение (ValidationRule) (формы)

Кнопка вниз (OnMouseDown) (формы)

Перемещение указателя (OnMouseMove) (формы)

Кнопка вверх (OnMouseUp) (формы)

Сообщение об ошибке (ValidationText) (формы)

Вывод на экран (Visible)

Ширина (Width)

Свойства переключателя

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxOptionButtonControlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxOptionButtonControlPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-переключателя. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Подписи с двоеточием (AddColon)

После обновления (AfterUpdate) (формы)

Добавление подписи (AutoLabel)

До обновления (BeforeUpdate) (формы)

Цвет границы (BorderColor)

Тип границы (BorderStyle)

Ширина границы (BorderWidth)

Данные (ControlSource)

Всплывающая подсказка (ControlTipText) (формы)

ControlType

Значение по умолчанию (DefaultValue) (формы)

Режим вывода (DisplayWhen) (формы)

Доступ (Enabled) (формы)

Высота (Height)

Идентификатор справки (HelpContextID) (формы)

InSelection (формы)

Выравнивание подписи (LabelAlign)

Позиция подписи X (LabelX)

Позиция подписи Y (LabelY)

От левого края (Left)

Блокировка (Locked) (формы)

Имя (Name)

Нажатие кнопки (OnClick) (формы)

Двойное нажатие кнопки (OnDblClick) (формы)

Вход (OnEnter) (формы)

Выход (OnExit) (формы)

Получение фокуса (OnGotFocus) (формы)

Клавиша вниз (OnKeyDown) (формы)

Нажатие клавиши (OnKeyPress) (формы)

Клавиша вверх (OnKeyUp) (формы)

Потеря фокуса (OnLostFocus) (формы)

Кнопка вниз (OnMouseDown) (формы)

Перемещение указателя (OnMouseMove) (формы)

Кнопка вверх (OnMouseUp) (формы)

Контекстное меню (ShortcutMenuBar)

Оформление (SpecialEffect)

Текст строки состояния (StatusBarText) (формы)

Индекс перехода по Tab (TabIndex) (формы)

Переход по Tab (TabStop) (формы)

Дополнительные сведения (Tag)

От верхнего края (Top)

Тройное состояние (TripleState)

Условие на значение (ValidationRule) (формы)

Сообщение об ошибке (ValidationText) (формы)

Вывод на экран (Visible)

Ширина (Width)

Свойства группы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxOptionGroupControlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acidxOptionGroupControlPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-группы. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Подписи с двоеточием (AddColon)
После обновления (AfterUpdate) (формы)
Добавление подписи (AutoLabel)
Цвет фона (BackColor)
Тип фона (BackStyle)
До обновления (BeforeUpdate) (формы)
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Данные (ControlSource)
Всплывающая подсказка (ControlTipText) (формы)
ControlType
Значение по умолчанию (DefaultValue) (формы)
Режим вывода (DisplayWhen) (формы)
Доступ (Enabled) (формы)
Высота (Height)
Идентификатор справки (HelpContextID) (формы)
InSelection
Выравнивание подписи (LabelAlign)
Позиция подписи X (LabelX)
Позиция подписи Y (LabelY)
От левого края (Left)
Блокировка (Locked) (формы)
Имя (Name)
Нажатие кнопки (OnClick) (формы)
Двойное нажатие кнопки (OnDblClick) (формы)
Вход (OnEnter) (формы)
Выход (OnExit) (формы)
Кнопка вниз (OnMouseDown) (формы)
Перемещение указателя (OnMouseMove) (формы)
Кнопка вверх (OnMouseUp) (формы)
Контекстное меню (ShortcutMenuBar)
Оформление (SpecialEffect)
Текст строки состояния (StatusBarText) (формы)
Индекс перехода по Tab (TabIndex) (формы)
Переход по Tab (TabStop) (формы)
Дополнительные сведения (Tag)
От верхнего края (Top)
Условие на значение (ValidationRule) (формы)
Сообщение об ошибке (ValidationText) (формы)
Вывод на экран (Visible)
Ширина (Width)

Свойства конца страницы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxPageBreakControlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxPageBreakControlPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления конец страницы. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

ControlType

Имя (Name)

InSelection (формы)

Дополнительные сведения (Tag)

От левого края (Left)

От верхнего края (Top)

Свойства прямоугольника

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxRectangleControlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxRectangleControlPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-прямоугольника. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Цвет фона (BackColor)

Тип фона (BackStyle)

Цвет границы (BorderColor)

Тип границы (BorderStyle)

Ширина границы (BorderWidth)

ControlType

Режим вывода (DisplayWhen) (формы)

Высота (Height)

InSelection (формы)

От левого края (Left)

Имя (Name)

Нажатие кнопки (OnClick) (формы)

Двойное нажатие кнопки (OnDblClick) (формы)

Кнопка вниз (OnMouseDown) (формы)

Перемещение указателя (OnMouseMove) (формы)

Кнопка вверх (OnMouseUp) (формы)

Оформление (SpecialEffect)

Дополнительные сведения (Tag)

От верхнего края (Top)

Вывод на экран (Visible)

Ширина (Width)

Свойства подчиненной формы/отчета

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxSubformSubreportCtrlPropC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxSubformSubreportCtrlPropX":1}

В следующем списке содержатся имена свойств элемента управления подчиненная форма/отчет. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Подписи с двоеточием (AddColon)
Добавление подписи (AutoLabel)
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Расширение (CanGrow)
Сжатие (CanShrink)
ControlType
Режим вывода (DisplayWhen) (формы)
Доступ (Enabled) (формы)
Высота (Height)
InSelection (формы)
Выравнивание подписи (LabelAlign)
Позиция подписи X (LabelX)
Позиция подписи Y (LabelY)
От левого края (Left)
Подчиненные поля (LinkChildFields)
Основные поля (LinkMasterFields)
Блокировка (Locked) (формы)
Имя (Name)
Вход (OnEnter) (формы)
Выход (OnExit) (формы)
Объект-источник (SourceObject)
Оформление (SpecialEffect)
Текст строки состояния (StatusBarText) (формы)
Индекс перехода по Tab (TabIndex) (формы)
Переход по Tab (TabStop) (формы)
Дополнительные сведения (Tag)
От верхнего края (Top)
Вывод на экран (Visible)
Ширина (Width)

Свойства элемента управления-поля

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxTextBoxPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxTextBoxPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-поля. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

A–N

Подписи с двоеточием (AddColon)

После обновления (AfterUpdate) (формы)

Разрешить автозамену (AllowAutoCorrect)

Добавление подписи (AutoLabel)

Автопереход по Tab (AutoTab) (формы)

Цвет фона (BackColor)

Тип фона (BackStyle)

До обновления (BeforeUpdate) (формы)

Цвет границы (BorderColor)

Тип границы (BorderStyle)

Ширина границы (BorderWidth)

Расширение (CanGrow)

Сжатие (CanShrink)

Данные (ControlSource)

Всплывающая подсказка (ControlTipText) (формы)

ControlType

Число десятичных знаков (DecimalPlaces)

Значение по умолчанию (DefaultValue) (формы)

Режим вывода (DisplayWhen) (формы)

Доступ (Enabled) (формы)

Поведение по Enter (EnterKeyBehavior) (формы)

Применение автофильтра (FilterLookup)

FontBold

Наклонный (FontItalic)

Шрифт (FontName)

Размер шрифта (FontSize)

Подчеркнутый (FontUnderline)

Насыщенность (FontWeight)

Цвет текста (ForeColor)

Формат поля (Format)

Высота (Height)

Идентификатор справки (HelpContextID) (формы)

Маска ввода (InputMask)

InSelection (формы)

Выравнивание подписи (LabelAlign)

Позиция подписи X (LabelX)

Позиция подписи Y (LabelY)

От левого края (Left)

Блокировка (Locked) (формы)

Имя (Name)

O–P

Изменение (OnChange) (формы)

Нажатие кнопки (OnClick) (формы)

Двойное нажатие кнопки (OnDblClick) (формы)

Вход (OnEnter) (формы)

Выход (OnExit) (формы)

Получение фокуса (OnGotFocus) (формы)

Клавиша вниз (OnKeyDown) (формы)

Нажатие клавиши (OnKeyPress) (формы)

Полосы прокрутки (ScrollBars) (формы)

Контекстное меню (ShortcutMenuBar)

Оформление (SpecialEffect)

Текст строки состояния (StatusBarText) (формы)

Индекс перехода по Tab (TabIndex) (формы)

Переход по Tab (TabStop) (формы)

Дополнительные сведения (Tag)

Выравнивание текста (TextAlign)

Клавиша вверх (OnKeyUp) (формы)

Потеря фокуса (OnLostFocus) (формы)

Кнопка вниз (OnMouseDown) (формы)

Перемещение указателя (OnMouseMove) (формы)

Кнопка вверх (OnMouseUp) (формы)

От верхнего края (Top)

Условие на значение (ValidationRule) (формы)

Сообщение об ошибке (ValidationText) (формы)

Вывод на экран (Visible)

Ширина (Width)

Свойства выключателя

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxToggleButtonCtrlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxToggleButtonCtrlPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления-выключателя. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Подписи с двоеточием (AddColon)

После обновления (AfterUpdate) (формы)

Добавление подписи (AutoLabel)

До обновления (BeforeUpdate) (формы)

Подпись (Caption)

Данные (ControlSource)

Всплывающая подсказка (ControlTipText) (формы)

ControlType

Значение по умолчанию (DefaultValue) (формы)

Режим вывода (DisplayWhen) (формы)

Доступ (Enabled) (формы)

Наклонный (FontItalic)

Шрифт (FontName)

Размер шрифта (FontSize)

Подчеркнутый (FontUnderline)

Насыщенность (FontWeight)

Цвет текста (ForeColor)

Высота (Height)

Идентификатор справки (HelpContextID) (формы)

InSelection (формы)

Выравнивание подписи (LabelAlign)

Позиция подписи X (LabelX)

Позиция подписи Y (LabelY)

От левого края (Left)

Блокировка (Locked) (формы)

Имя (Name)

Нажатие кнопки (OnClick) (формы)

Двойное нажатие кнопки (OnDblClick) (формы)

Вход (OnEnter) (формы)

Выход (OnExit) (формы)

Получение фокуса (OnGotFocus) (формы)

Клавиша вниз (OnKeyDown) (формы)

Нажатие клавиши (OnKeyPress) (формы)

Клавиша вверх (OnKeyUp) (формы)

Потеря фокуса (OnLostFocus) (формы)

Кнопка вниз (OnMouseDown) (формы)

Перемещение указателя (OnMouseMove) (формы)

Кнопка вверх (OnMouseUp) (формы)

Рисунок (Picture)

Тип рисунка (PictureType)

Контекстное меню (ShortcutMenuBar)

Текст строки состояния (StatusBarText) (формы)

Индекс перехода по Tab (TabIndex) (формы)

Переход по Tab (TabStop) (формы)

Дополнительные сведения (Tag)

От верхнего края (Top)

Тройное состояние (TripleState)

Условие на значение (ValidationRule) (формы)

Сообщение об ошибке (ValidationText) (формы)

Вывод на экран (Visible)

Ширина (Width)

Свойства свободной рамки объекта

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxUnboundObjFrameCtrlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxUnboundObjFrameCtrlPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления свободная рамка объекта. Свойства, применимые только для элемента управления в форме, имеют обозначение (формы), что означает «только для форм».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Автоматический запуск (AutoActivate) (формы)
Цвет фона (BackColor)
Тип фона (BackStyle)
Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Класс (Class)
Число столбцов (ColumnCount)
Заглавия столбцов (ColumnHeads)
Всплывающая подсказка (ControlTipText) (формы)
ControlType
Тип вывода (DisplayType)
Режим вывода (DisplayWhen) (формы)
Доступ (Enabled) (формы)
Высота (Height)
Идентификатор справки (HelpContextID) (формы)
InSelection (формы)
От левого края (Left)
Подчиненные поля (LinkChildFields)
Основные поля (LinkMasterFields)
Блокировка (Locked) (формы)
Имя (Name)
Класс OLE (OLEClass)
Тип OLE (OLEType)
Допустимый тип OLE (OLETypeAllowed)
Нажатие кнопки (OnClick) (формы)
Двойное нажатие кнопки (OnDblClick) (формы)
Вход (OnEnter) (формы)
Выход (OnExit) (формы)
Получение фокуса (OnGotFocus) (формы)
Потеря фокуса (OnLostFocus) (формы)
Кнопка вниз (OnMouseDown) (формы)
Перемещение указателя (OnMouseMove) (формы)
Кнопка вверх (OnMouseUp) (формы)
При обновлении (OnUpdated) (формы)
Источник строк (RowSource)
Тип источника строк (RowSourceType)
Контекстное меню (ShortcutMenuBar)
Установка размеров (SizeMode)
Документ-источник (SourceDoc)
Источник данных (SourceItem)
Оформление (SpecialEffect)
Текст строки состояния (StatusBarText)
Индекс перехода по Tab (TabIndex) (формы)
Переход по Tab (TabStop) (формы)
Дополнительные сведения (Tag)
От верхнего края (Top)
Параметры обновления (UpdateOptions)
Команда (Verb)
Вывод на экран (Visible)
Ширина (Width)

Свойства элемента управления «Набор вкладок»

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxTabCtrlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxTabCtrlPropertiesX":1}

В следующем списке содержатся имена свойств элемента управления «Набор вкладок».

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Тип фона (BackStyle)
ControlType
Режим вывода (DisplayWhen)
Доступ (Enabled)
Наклонный (FontItalic)
Шрифт (FontName)
Размер шрифта (FontSize)
Подчеркнутый (FontUnderline)
Насыщенность (FontWeight)
Высота (Height)
Идентификатор справки (HelpContextID)
InSelection
От левого края (Left)
MultiRow
Имя (Name)
Изменение (OnChange)
Нажатие кнопки (OnClick)
Двойное нажатие кнопки (OnDblClick)
Клавиша вниз (OnKeyDown) (формы)
Нажатие клавиши (OnKeyPress) (формы)
Клавиша вверх (OnKeyUp) (формы)
Кнопка вниз (OnMouseDown)
Перемещение указателя (OnMouseMove)
Кнопка вверх (OnMouseUp)
Контекстное меню (ShortcutMenuBar)
Текст строки состояния (StatusBarText)
Style
TabFixedHeight
TabFixedWidth
Индекс перехода по Tab (TabIndex)
Переход по Tab (TabStop)
Дополнительные сведения (Tag)
От верхнего края (Top)
Вывод на экран (Visible)
Ширина (Width)

Свойства элемента ActiveX

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acidxActiveXControlPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acidxActiveXControlPropertiesX":1}

В следующем списке содержатся имена свойств элемента ActiveX.

Для получения подробных сведений о конкретном свойстве выберите в списке имя свойства. Для каждого свойства будут выведены условия применения, а также информация о возможности задавать значение этого свойства в окне свойств, в макросе или в программе Visual Basic.

Цвет границы (BorderColor)
Тип границы (BorderStyle)
Ширина границы (BorderWidth)
Класс (Class)
Данные (ControlSource)
Всплывающая подсказка (ControlTipText)
ControlType
Специальные (Custom)
Режим вывода (DisplayWhen) (формы)
Доступ (Enabled) (формы)
Высота (Height)
Идентификатор справки (HelpContextID) (формы)
InSelection (формы)
От левого края (Left)
Блокировка (Locked) (формы)
Имя (Name)
Object
Класс OLE (OLEClass)
Вход (OnEnter) (формы)
Выход (OnExit) (формы)
Получение фокуса (OnGotFocus) (формы)
Потеря фокуса (OnLostFocus) (формы)
При обновлении (OnUpdated) (формы)
Оформление (SpecialEffect)
Индекс перехода по Tab (TabIndex) (формы)
Переход по Tab (TabStop) (формы)
Дополнительные сведения (Tag)
От верхнего края (Top)
Команда (Verb)
Вывод на экран (Visible)
Ширина (Width)

Функция LoadPicture

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctLoadPictureC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctLoadPictureX":1}

Функция **LoadPicture** загружает рисунок в элемент ActiveX.

Синтаксис

LoadPicture(*имяФайла***)**

Функция **LoadPicture** использует следующий аргумент.

Аргумент	Описание
имяФайла	Строковое выражение, определяющее имя загружаемого файла. Допустимыми являются точечные рисунки (bitmap) (.bmp), значки (.ico), файлы в формате .rle или метафайлы (.wmf).

Дополнительные сведения

Для динамической загрузки рисунка в элемент ActiveX следует присвоить значение, возвращаемое функцией **LoadPicture**, свойству **Рисунок (Picture)** этого элемента. В следующем примере точечный рисунок загружается в специальный элемент управления с именем «ЭлементOLE» в форме «Заказы».

```
Set Forms!Заказы!ЭлементOLE.Picture = LoadPicture("Stars.bmp")
```

Функция **LoadPicture** возвращает объект типа **Picture**. Это значение может быть присвоено переменной типа **Object** с помощью инструкции **Set**.

Хотя объект **Picture** и не является объектом Microsoft Access, он доступен для процедур в Microsoft Access.

Примечание. Не допускается использование функции **LoadPicture** для определения свойства **Рисунок (Picture)** элемента управления-рисунка. Данная функция применима только к элементам ActiveX. Чтобы задать значение свойства **Рисунок** для элемента управления-рисунка, достаточно просто ввести строку, указывающую имя и путь нужного файла рисунка.

Функция LoadPicture, пример

В следующем примере функция **LoadPicture** загружает метафайл в элемент ActiveX с именем «СпецЭлемент» в форме «Сотрудники».

```
Sub DisplayGraphic()
   Const strConPathToBitmaps = "C:\Program Files\Microsoft Office\Office\
Bitmaps\Styles\"

   ' Описывает объектные переменные типа Picture и Control.
   Dim objPicture As Object, ctl As Control

   ' Связывает переменную Control с элементом ActiveX в форме.
   Set ctl = Forms!Сотрудники!СпецЭлемент
   ' Присваивает возвращаемое LoadPicture значение объекту Picture.
   Set objPicture = LoadPicture(strConPathToBitmaps & "Globe.wmf")
   ' Задает значение свойства "Рисунок" (Picture) элемента ActiveX.
   Set ctl.Picture = objPicture
End Sub
```

Изменения в именах свойств

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproChangedNamesC;dahowObsoleteFeatures"}

В следующей таблице перечислены свойства, получившие новые имена или выполняющие другие функции, чем в предыдущих версиях Microsoft Access 95. Эти свойства включены в настоящую версию только для совместимости с предыдущими. Вместо них рекомендуется использовать свойства данной версии.

Добавленные в Microsoft Access 95 свойства отмечены звездочкой (*) в указателе свойств. Свойства, добавленные в Microsoft Access 97, в указателе свойств отмечены двумя звездочками (**).

Свойство предыдущей версии Microsoft Access	Текущее свойство Microsoft Access
AllowEditing, **DefaultEditing**	**Разрешить добавление (AllowAdditions)**, **Разрешить удаление (AllowDeletions)**, **Разрешить изменение (AllowEdits)**, **Ввод данных (DataEntry)**
MaxButton, **MinButton**	**Кнопки размеров окна (MinMaxButtons)**
BorderLineStyle	**Тип границы (BorderStyle)**
AllowUpdating	**Тип набора записей (RecordsetType)**
Dynaset	**RecordsetClone**
ShowGrid	**DatasheetGridLinesBehavior**

Задание значений свойств в макросах

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"achowSettingPropsUsingMacrosC"}

В макросах допускается определение свойств объектов **Form** (форма), **Report** (отчет) и **Control** (элемент управления), а также значений свойств разделов формы или отчета. Значения свойств задаются с помощью макрокоманды **ЗадатьЗначение (SetValue)**.

Невозможно задать в макросе значения свойств других объектов Microsoft Access и объектов доступа к данным, а также определить стандартные свойства элемента управления. Эти действия выполняются либо в программе Visual Basic, либо в окне свойств объекта в режиме конструктора.

Задание в макросе значения свойства формы, отчета или элемента управления.

1. Добавьте в макрос макрокоманду **ЗадатьЗначение (SetValue)**.
2. Введите в ячейку аргумента «Элемент» макрокоманды **ЗадатьЗначение (SetValue)** выражение, содержащее ссылку на определяемое свойство:
   - Для ссылок на свойства формы или отчета используется синтаксис **Forms!** *имяФормы***.***имяСвойства* или **Reports!***имяОтчета***.***имяСвойства*. Например, следующее выражение определяет ссылку на свойство **Вывод на экран (Visible)** формы «Клиенты».

     ```
     Forms!Клиенты.Visible
     ```

   - Для ссылок на свойства элемента управления в форме или отчете используется синтаксис **Forms!***имяФормы***!***имяЭлемента***.***имяСвойства* или **Reports!***имяОтчета***!** *имяЭлемента***.***имяСвойства.* Например, следующее выражение определяет ссылку на свойство **Вывод на экран (Visible)** элемента управления «СкрытыйКонецСтраницы» в отчете «Счет»:

     ```
     Reports!Счет!СкрытыйКонецСтраницы.Visible
     ```

   Совет. Если макрос, содержащий макрокоманду **ЗадатьЗначение (SetValue)**, запускается из тех форм или отчетов, для которых задается значение свойства, то допускается ссылка на свойство с помощью синтаксиса *имяСвойства*. Рекомендуется, однако, всегда использовать полный синтаксис ссылок во избежание возможных конфликтов с именами элементов управления или ключевыми словами Visual Basic. Например, **Имя (Name)** является именем свойства Microsoft Access; поэтому при наличии в форме элемента управления с именем «Name» полный синтаксис является обязательным как при ссылках на свойство, так и при ссылках на элемент управления.

3. Введите в ячейку аргумента «Выражение» макрокоманды **ЗадатьЗначение (SetValue)** значение, которое задается для свойства. Если задается строковое значение, не забудьте заключить его в прямые кавычки ("). Например, для того чтобы указать в свойстве формы **Подпись (Caption)** заголовок «Заказы», необходимо ввести слово «Заказы» в прямых кавычках.

Задание в макросе значения свойства раздела

1. Добавьте в макрос макрокоманду **ЗадатьЗначение (SetValue)**.
2. Введите в ячейку аргумента «Элемент» следующую ссылку на определяемое свойство **Forms!***имяФормы***.Section(***n***).***имяСвойства.* Целое число *n* определяет конкретный раздел формы или отчета, как это описано в разделе справки для свойства **Section.** Например, следующее выражение определяет ссылку на свойство **Вывод на экран (Visible)** верхнего колонтитула формы «Клиенты»:

   ```
   Forms!Клиенты.Section(acPageHeader).Visible
   ```

3. Определите аргумент «Выражение», как это описано выше.

Примечания. В указателе справочной системы можно найти разделы, содержащие сведения по следующим вопросам.

- В каких случаях допускается задание значения свойства в макросе.
- Режимы, в которых разрешается задавать значение свойства. Не каждое свойство допускает задание значения во всех режимах. Например, значение свойства формы **Тип границы (BorderStyle)** может быть задано только в режиме конструктора формы.
- Допустимые значения свойства. В программах Visual Basic часто требуются значения, отличные от задаваемых в интерактивном режиме в окне свойств. Например, если значение свойства в окне свойств выбирается из раскрывающегося списка, то в программе следует задать числовой эквивалент выбираемого значения.

Задание значений свойств в программах Visual Basic

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"achowSettingPropsUsingCodeC"}

Для большинства свойств допускается задание значения в программе Visual Basic. Способ, с помощью которого задается значение свойства, зависит от того, задается ли это значение для объекта **Form** (форма), **Report** (отчет) и **Control** (элемент управления) или для объекта доступа к данным. Конкретные действия, требующиеся при задании значений свойств объектов различных типов описаны в следующих разделах:

Задание значений свойств форм, отчетов и элементов управления в программах Visual Basic

Задание значений свойств объектов доступа к данным в программах Visual Basic

Задание значений свойств форм, отчетов и элементов управления в программах Visual Basic

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"achowSettingFormReportAndControlPropertiesInVisualBasicC"}

Объекты **Form** (форма), **Report** (отчет) и **Control** (элемент управления) являются объектами Microsoft Access. Значения свойств этих объектов могут быть заданы в процедурах **Sub, Function** или процедурах обработки событий. Кроме того, допускается задание значений свойств разделов форм и отчетов.

Задание значения свойства формы или отчета

Для того чтобы задать значение свойства формы или отчета, входящих в семейство, следует указать семейство **Forms** или **Reports,** имя формы или отчета, а также имя и значение свойства. Например, значение **True** (-1) задается для свойства **Вывод на экран (Visible)** формы "Клиенты" с помощью следующей программной строки:

```
Forms!Клиенты.Visible = True
```

Возможна также установка значения свойства формы или отчета из модуля объекта с помощью свойства объекта **Me.** Программа, использующая свойство объекта **Me** выполняется быстрее, чем использующая полный синтаксис имени объекта. Например, чтобы установить значение свойства **RecordSource (Источник записей)** формы «Клиенты» в инструкцию SQL, возвращающую все записи из модуля формы «Клиенты», в которых поле «ИмяКомпании» начинается с буквы "А", следует написать следующую программную строку:

```
Me!RecordSource = "SELECT * FROM Клиенты " _
   & "WHERE ИмяКомпании Like 'A*'"
```

Задание значения свойства элемента управления

Для того чтобы задать значение свойства элемента управления, входящего в семейство, следует указать семейство **Controls** объекта **Form** или **Report**, в котором находится данный элемент управления. Допускаются как явные, так и неявные ссылки на семейство **Controls**, но неявные ссылки выполняются быстрее. В следующих примерах задается значение свойства **Вывод на экран (Visible)** поля «КодКлиента» в форме «Клиенты»:

```
' Более быстрый метод.
Me!КодКлиента.Visible = True

' Более медленный метод.
Forms!Клиенты.Controls!КодКлиента.Visible = True
```

Самый быстрый способ задания значения свойства элемента управления - задание из модуля объекта с помощью свойства обхекта **Me**. Например, следующий код переключает значение свойства **Вывод на экран (Visible)** поля «КодКлиента» формы «Клиенты»:

```
With Me!КодКлиента
   .Visible = Not .Visible
End With
```

Задание значений свойства раздела формы или отчета

Для того чтобы задать значение свойства раздела формы или отчета, следует указать семейство **Forms** или **Reports**, свойство **Section** и индекс, определяющий конкретный раздел. В следующем примере для свойства **Вывод на экран (Visible)** верхнего колонтитула формы «Клиенты» задается значение **False** (0):

```
Forms!Клиенты.Section(3).Visible = False

Me!Section(acPageHeader).Visible = False
```

Примечания

- В указателе справочной системы можно найти разделы, содержащие сведения по следующим вопросам.
  - В каких случаях допускается задание значения свойства в программе Visual Basic.
  - Режимы, в которых разрешается задавать значение свойства. Не каждое свойство допускает задание значения во всех режимах. Например, значение свойства формы **Тип границы (BorderStyle)** может быть задано только в режиме конструктора формы.
  - Допустимые значения свойства. В программах Visual Basic часто требуются значения, отличные от задаваемых в интерактивном режиме в окне свойств. Например, если значение свойства в окне свойств выбирается из раскрывающегося списка, то в программе следует задать числовой эквивалент выбираемого значения.
- Для того чтобы задать стандартные свойства элемента управления в программе Visual Basic, следует использовать метод **DefaultControl.**

Задание значений свойств объектов доступа к данным в программах Visual Basic

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS.
nїSnїSnїSnїSnїS":"achowSettingPropertiesOfDataAccessObjectsInVisualBasicC;daconMSJetDatabaseEngine25"}

Объекты доступа к данным (DAO) обеспечивают управление структурой базы данных и содержащимися в ней данными из программы Visual Basic. Многие объекты доступа к данным соответствуют объектам, выводящимся в базе данных например, объект **TableDef** соответствует таблице Microsoft Access. Объект **Field** соответствует полю в таблице.

Большинство из свойств, значения которых задаются пользователем для объектов доступа к данным, являются свойствами доступа к данным. Эти свойства определены в ядре базы данных Microsoft Jet и их значения задаются одинаково в любом приложении, использующем ядро базы данных Jet. Некоторые из свойств объектов доступа к данным определены в Microsoft Access и не распознаются автоматически ядром базы данных Jet. Значения свойств объектов доступа к данным, определенных в ядре Jet и в Microsoft Access, задаются по-разному.

Задание значений свойств объектов доступа к данным

Для того чтобы задать значение свойства, которое определено в ядре базы данных Jet, требуется ссылка на иерархию объектов доступа к данным. Самым быстрым и простым способом является создание объектных переменных, представляющих различные объекты, с последующими ссылками на объектные переменные в программе. Например, следующая программа создает объект **TableDef** и определяет его свойство **Name**:

```
Dim dbs As Database, tdf As TableDef
Set dbs = CurrentDb
Set tdf = dbs.CreateTableDef
tdf.Name = "Контакты"
```

Задание для объектов доступа к данным значений свойств, определенных в Microsoft Access

Свойство, определенное в Microsoft Access, которое применяется к объектам доступа к данным, не распознается автоматически ядром базы данных Jet как допустимое свойство. Если значение данного свойства задается впервые, необходимо создать свойство и добавить его в семейство **Properties** объекта, к которому оно будет применено. После добавления данного свойства в семейство **Properties** его значение задается точно так же, как и любое свойство объектов доступа к данным.

Если в первый раз значение такого свойства задается через интерфейс пользователя, это свойство автоматически добавляется в семейство **Properties** и дальнейшая работа с ним выполняется обычным образом.

При разработке процедур, в которых задаются значения свойств, определенных в Microsoft Access, необходимо включать в программу блок обработки ошибок, который должен проверять, добавлено ли уже данное свойство в семейство **Properties**. Более подробное описание см. в разделах справки для метода **CreateProperty** или для конкретных свойств.

Необходимо помнить, что при создании свойства необходимо корректно указать его тип в свойстве **Type** перед добавлением свойства в семейство **Properties**. При определении значения свойства **Type** следует ознакомиться со сведениями, содержащимися в части «Значения» раздела справки для соответствующего свойства. Ниже формулируются наиболее общие правила определения свойства **Type**.

Значение свойства	Константа, определяющая свойство Type
Строковое	**dbText**

Логическое	**dbBoolean**
Целое	**dbInteger**

В следующей таблице перечислены некоторые определенные в Microsoft Access свойства, которые применяются к объектам доступа к данным.

Объект	**Свойства Microsoft Access**
Database	**AppTitle**, **AppIcon**, **StartupShowDBWindow**, **StartupShowStatusBar**, **AllowShortcutMenus**, **AllowFullMenus**, **AllowBuiltInToolbars**, **AllowToolbarChanges**, **AllowBreakIntoCode**, **AllowSpecialKeys**, **Replicable**, **ReplicationConflictFunction**
Container SummaryInfo	**Title** (Название), **Subject** (Тема), **Author** (Автор), **Manager** (Должность), **Company** (Организация), **Category** (Группа), **Keywords** (Ключевые слова), **Comments** (Примечания), **База гиперрссылки (Hyperlink Base)** (См. Вкладку **Документ** в диалоговом окне **Свойства:<*база данных*>**, которое открывается командой **Свойства базы данных** из меню **Файл**).
Container UserDefined	(См. вкладку **Прочие** в диалоговом окне **Свойства:<*база данных*>**, которое открывается командой **Свойства базы данных** из меню **Файл**).
TableDef	**DatasheetBackColor**, **DatasheetCellsEffect**, **DatasheetFontHeight**, **DatasheetFontItalic**, **DatasheetFontName**, **DatasheetFontUnderline**, **DatasheetFontWeight**, **DatasheetForeColor**, **DatasheetGridlinesBehavior**, **DatasheetGridlinesColor**, **Description**, **FrozenColumns**, **RowHeight**, **ShowGrid**
QueryDef	**DatasheetBackColor**, **DatasheetCellsEffect**, **DatasheetFontHeight**, **DatasheetFontItalic**, **DatasheetFontName**, **DatasheetFontUnderline**, **DatasheetFontWeight**, **DatasheetForeColor**, **DatasheetGridlinesBehavior**, **DatasheetGridlinesColor**, **Description**, **FailOnError**, **FrozenColumns**, **LogMessages**, **MaxRecords**, **RecordLocks**, **RowHeight**, **ShowGrid,** **UseTransaction**
Field	**Подпись (Caption)**, **Число десятичных знаков (DecimalPlaces)**, **ColumnHidden**, **Описание (Описание)**, **ColumnOrder**, **Формат поля (Format)**, **ColumnWidth**, **Маска ввода (InputMask)**

Задание значений свойств форм, отчетов и элементов управления

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"achowSetFormReportControlPropertiesC"}

Каждая форма, отчет, раздел формы или отчета и элемент управления имеют набор свойств, с помощью которых пользователь изменяет вид или характеристики данного объекта. Эти свойства доступны для просмотра и изменения в окне свойств, макросе или программе Visual Basic.

Задание значений свойств

1. В режиме конструктора формы или режиме конструктора отчета выделите элемент управления, раздел, форму или отчет, для которых требуется задать значение свойства. Пользователь имеет возможность выделить:
   - Один или несколько элементов управления. Для того чтобы выделить несколько элементов управления, следует выбирать их при нажатой клавише SHIFT или, удерживая нажатой кнопку мыши, провести указатель по выделяемым элементам управления. При выборе нескольких элементов в окне свойств будут выведены лишь их общие свойства.
   - Один раздел. Выберите область выделения раздела необходимого раздела
   - Всю форму или весь отчет. Выберите область выделения формы или область выделения отчета в верхнем левом углу формы или отчета.
2. Откройте окно свойств нажатием правой кнопки мыши на объекте или разделе и выбором команды **Свойства** в контекстного меню или нажатием кнопки **Свойства** на панели инструментов.
3. Выберите ячейку свойства, для которого требуется задать значение, и выполните одно из следующих действий:
   - Введите в ячейку свойства соответствующее значение или выражение..
   - Если ячейка свойства содержит раскрывающийся список, выберите значение в списке.
   - Если справа от ячейки свойства появляется кнопка построителя , нажмите эту кнопку, которая вызовет соответствующий построитель или откроет диалоговое окно для выбора построителя. Например, значения некоторых свойств могут быть заданы с помощью построителя программы, построителя макроса или построителя запроса.

Если окно свойств пусто или не содержит ожидаемых свойств, следует обратиться к разделу Разрешение вопросов, связанных с окнами свойств

Советы

- Длинные значения или выражения удобнее вводить в окно **Область ввода.** Для того чтобы открыть это окно, выберите ячейку в окне свойств, а затем нажмите клавиши SHIFT+F2 или нажмите правую кнопку мыши и в контекстном меню выберите команду **Область ввода**.
- Для некоторых элементов управления значение свойства **Данные (ControlSource)** задается путем ввода значения с клавиатуры непосредственно в элемент управления. Более подробное описание см. в разделе Создание вычисляемого элемента управления.
- Пользователь имеет возможность изменить стандартные значения свойств. После этого создаваемые элементы управления по умолчанию будут получать новые стандартные значения свойств.
- Значения свойств присоединенного элемента управления могут отличаться от соответствующих настроек для поля в базовой таблице или запросе, с которым связан элемент управления. В этом случае значения свойств формы или отчета обычно имеют приоритет над значениями свойств таблицы или запроса. Более подробные сведения о наследовании свойств содержаться в разделе Свойства элементов управления и наследование свойств.

Определение параметров запуска и параметров приложения в программе

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"achowSetStartupPropsUsingCodeC"}

Параметры запуска определяют вид окна приложения при его открытии. Например, эти параметры позволяют изменить заголовок окна приложения, меню, панели инструментов, а также указать форму, открываемую при запуске. Их значения задаются в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**.

Параметры приложения, значения которых задаются в диалоговом окне **Параметры**, определяют различные характеристики среды приложения для данного пользователя. Например, в этом окне пользователь задает параметры форм, отчетов таблиц и запросов, используемые по умолчанию. Для того чтобы открыть диалоговое окно **Параметры**, выберите в меню **Сервис** команду **Параметры**.

Подробное описание способов определения параметров запуска и параметров приложения в программах приводится в следующих разделах.

Определение параметров запуска в программе Visual Basic

Определение параметров приложения в программе Visual Basic

Определение параметров запуска в программе Visual Basic

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"achowSettingStartupPropertiesFromVisualBasicC"}

Параметры запуска соответствуют свойствам объекта **Database (база данных).** Объект **Database** является объектом доступа к данным, определенным в ядре базы данных Microsoft Jet, однако, свойства, соответствующие параметрам запуска, определены в Microsoft Access и не распознаются автоматически ядром базы данных Jet. Если значение параметра запуска не было задано ранее, необходимо создать соответствующее свойство и добавить его в семейство **Properties** объекта **Database**. Если свойства, определяющие параметры запуска, задаются в программе Visual Basic, необходимо включить в программу блок обработки ошибок, проверяющий существование данного свойства в семействе **Properties**. Более подробное описание см. в разделе Задание значений свойств объектов доступа к данным в программе Visual Basic.

Имена свойств, определяющих параметры запуска, отличаются от имен соответствующих элементов управления в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Имена параметров и соответствующие свойства, используемые в программах Visual Basic, перечислены в следующей таблице.

Окно «Параметры запуска»	**Имя свойства**
Заголовок приложения	**AppTitle**
Значок приложения	**AppIcon**
Форма	**StartupForm**
Окно базы данных	**StartupShowDBWindow**
Строка состояния	**StartupShowStatusBar**
Строка меню	**StartupMenuBar**
Контекстное меню	**StartupShortcutMenuBar**
Полный набор меню Access	**AllowFullMenus**
Стандартные контекстные меню	**AllowShortcutMenus**
Стандартные панели инструментов	**AllowBuiltInToolbars**
Изменение панелей инструментов	**AllowToolbarChanges**
Просмотр программы после ошибки	**AllowBreakIntoCode**
Специальные клавиши	**AllowSpecialKeys**

Определение параметров приложения в программе Visual Basic

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"achowSettingOptionsFromVisualBasicC"}

Методы **SetOption** и **GetOption** задают и возвращают в программах значения параметров приложения, доступных в диалоговом окне **Параметры**. Для того чтобы открыть это диалоговое окно, выберите в меню **Сервис** команду **Параметры**.

В следующих таблицах перечислены все параметры приложения, доступные из программ, а также вкладки, на которых эти параметры выводятся в диалоговом окне, и соответствующие строковые аргументы (русские или английские), которые необходимо передать в методах **SetOption** или **GetOption**.

Примечания

- Невозможно задать в программе Visual Basic значения параметров, выводящихся в диалоговом окне **Параметры** на вкладке **Модуль**; эти значения задаются только интерактивно в диалоговом окне **Параметры**.
- Если предполагается выполнение базы данных в языковой версии Microsoft Access, отличной от исходной, необходимо задать английские значения для аргументов методов **GetOption** и **SetOption**.

Вкладка «Вид»

Параметр	Строковый аргумент
Строка состояния	Строка состояния (Show Status Bar)
Окно запуска	Окно запуска (Show Startup Dialog Box)
Скрытые объекты	Скрытые объекты (Show Hidden Objects)
Системные объекты	Системные объекты (Show System Objects)
Столбец имен	Столбец имен (Show Macro Names Column)
Столбец условий	Столбец условий (Show Conditions Column)

Вкладка «Общие»

Параметр	Строковый аргумент
Левое поле	Левое поле (Left Margin)
Правое поле	Правое поле (Right Margin)
Верхнее поле	Верхнее поле (Top Margin)
Нижнее поле	Нижнее поле (Bottom Margin)
Рабочий каталог	Рабочий каталог (Default Database Directory)
Порядок сортировки базы данных	Порядок сортировки базы данных (New Database Sort Order)
Звуковое сопровождение событий	Звуковое сопровождение событий (Provide Feedback with Sound)

Вкладка «HTML»

Параметр	Строковый аргумент
Цвет гиперссылки	Цвет гиперссылки (Hyperlink Color)
Цвет просмотренной гиперссылки	Цвет просмотренной гиперссылки (Followed Hyperlink Color)

Подчеркивание	Подчеркивание (Underline Hyperlinks)
Вывод адреса гиперссылки в строку состояния	Вывод адреса гиперссылки в строку состояния (Show Hyperlink Addresses in Status Bar)
Шаблон HTML	Шаблон HTML (HTML Template)
Название источника	Название источника (Data Source Name)
Вход под именем	Вход под именем (User Name)
Пароль	Пароль (Password)
Адрес URL сервера	Адрес URL сервера ASP (Active Server Pages URL)
Период ожидания (мин)	Период ожидания ASP (Active Server Pages Session Timeout)

Вкладка «Правка/поиск»

Параметр	Строковый аргумент
Поиск/замена по умолчанию	Поиск/замена по умолчанию (Default Find/Replace Behavior)
Подтверждение: Изменения записей	Подтверждение изменения записей (Confirm Record Changes)
Подтверждение: Удаления документов	Подтверждение удаления документов (Confirm Document Deletions)
Подтверждение: Запросов на изменение	Подтверждение запросов на изменение (Confirm Action Queries)
Отображать список значений: В локальные индексированные полях	Локальные индексированные поля для автофильтра (Show Values in Indexed)
Отображать список значений: В локальные неиндексированные полях	Локальные неиндексированные поля для автофильтра (Show Values in Non-indexed)
Отображать список значений: В полях ODBC	Внешние индексированные поля для автофильтра (Show Values in Remote)
Не отображать списки, содержащие более заданного числа строк	Максимальный размер списка автофильтра (Show Values Limit)

Вкладка «Клавиатура»

Параметр	Строковый аргумент
Переход при нажатии Enter	Переход при нажатии Enter (Move After Enter)
Переход по клавише со стрелкой	Переход по клавише со стрелкой (Arrow Key Behavior)
Поведение при входе в поле	Поведение при входе в поле (Behavior Entering Field)
Останов на первом/последнем поле	Останов на первом/последнем поле (Cursor Stops at First/Last Field)

Вкладка «Режим таблицы»

Параметр	Строковый аргумент
Цвет по умолчанию: Шрифт	Цвет текста по умолчанию (Default Font Color)
Цвет по умолчанию: Фон	Цвет фона по умолчанию (Default

	Background Color)
Цвет по умолчанию: Сетка	Цвет сетки по умолчанию (Default Gridlines Color)
Шрифт по умолчанию: Шрифт	Шрифт по умолчанию (Default Font Name)
Шрифт по умолчанию: Насыщенность	Насыщенность шрифта по умолчанию (Default Font Weight)
Шрифт по умолчанию: Размер	Размер шрифта по умолчанию (Default Font Size)
Шрифт по умолчанию: Наклонный	Наклонный по умолчанию (Default Font Italic)
Шрифт по умолчанию: Подчеркнутый	Подчеркивание по умолчанию (Default Font Underline)
Линии сетки по умолчанию: По горизонтали	Горизонтальные линии сетки по умолчанию (Default Gridlines Horizontal)
Линии сетки по умолчанию: По вертикали	Вертикальные линии сетки по умолчанию (Default Gridlines Vertical)
Ширина столбца по умолчанию	Ширина столбца по умолчанию (Default Column Width)
Вид сетки по умолчанию	Вид сетки по умолчанию (Default Cell Effect)
Визуализация изменений	Визуализация изменений (Show Animations)

Вкладка «Таблицы/запросы»

Параметр	Строковый аргумент
Размеры полей по умолчанию: Текстовое	Размер текстовых полей по умолчанию (Default Text Field Size)
Размеры полей по умолчанию: Числовое	Размер числовых полей по умолчанию (Default Number Field Size)
Тип поля по умолчанию	Тип поля по умолчанию (Default Field Type)
Автоиндекс при импорте/создании	Автоиндекс при импорте/создании (Autoindex on Import/Create)
Вывод имен таблиц	Вывод имен таблиц (Show Table Names)
Вывод всех полей	Вывод всех полей (Output All Fields)
Автоматическое объединение	Автоматическое объединение (Enable AutoJoin)
При запуске предоставляются права	При запуске предоставляются права (Run Permissions)

Вкладка «Формы/отчеты»

Параметр	Строковый аргумент
Выделение объектов	Выделение объектов (Selection Behavior)
Шаблон формы	Шаблон формы (Form Template)
Шаблон отчета	Шаблон отчета (Report Template)
Всегда использовать процедуры обработки событий	Всегда использовать процедуры обработки событий (Always Use Event Procedures)

Вкладка «Другие»

Параметр	Строковый аргумент
Блокировка по умолчанию	Блокировка по умолчанию (Default Record Locking)

Режим открытия по умолчанию	Режим открытия по умолчанию (Default Open Mode for Databases)
Пропуск команд DDE	Пропуск команд DDE (Ignore DDE Requests)
Обновление связей DDE	Обновление связей DDE (Enable DDE Refresh)
Время ожидания OLE/DDE (с)	Время ожидания OLE/DDE (с) (OLE/DDE Timeout (sec))
Число повторов обновления	Число повторов обновления (Number of Update Retries)
Период обновления ODBC (с)	Период обновления ODBC (с) (ODBC Refresh Interval (sec))
Период обновления (с)	Период обновления (с) (Refresh Interval (sec))
Период повтора обновления (мс)	Период повтора обновления (мс) (Update Retry Interval (msec))
Аргументы командной строки	Аргументы командной строки (Command-Line Arguments)
Аргументы условной компиляции	Аргументы условной компиляции (Conditional Compilation Arguments)
Имя проекта	Имя проекта (Project Name)
Перехват ошибок	Перехват ошибок (Error Trapping)

Примечание. При разработке приложения базы данных, надстройки, библиотечной базы данных или адресуемой базы данных следует установить для параметра **Перехват ошибок (Error Trapping)** значение 2 (**Останов при необрабатываемых ошибках**) после завершения отладки программы.

Значение, которое передается в метод **SetOption** в аргументе *значение,* зависит от типа определяемого параметра. В следующей таблице приведены рекомендации по заданию параметров.

Представление параметра	**Содержимое аргумента *значение***
Поле	Строка
Флажок	Логическое значение – «Да»/«Нет» или **True** (–1) / **False** (0)
Переключатель в группе или элемент поля со списком или списка	Целое число, соответствующее положению параметра в группе или значения в списке (отсчет от 0).

Включение переменных и элементов управления в инструкции SQL

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"achowSQLStringVBAC;dahowBuildingSQLStatements"}

При работе с объектами доступа к данным (DAO) в программе часто требуется создать инструкцию SQL. Например, при создании нового объекта **QueryDef** (запрос) значением его свойства **SQL** должна быть допустимая строка SQL. Инструкция SQL обычно требуется также при создании объекта **Recordset** (набор записей).

Простейший способ разработки инструкции SQL заключается в создании запроса в бланке запроса,переключении в режим SQL и копировании соответствующей запросу инструкции SQL в собственную программу.

Часто при выполнении запросов используют значения, вводящиеся или изменяемые пользователем. В подобных случаях в запрос необходимо включить переменные или элементы управления. В ядре базы данных Microsoft Jet обрабатываются все инструкции SQL, но не имена переменных и элементов управления. Таким образом, инструкцию SQL следует разрабатывать таким образом, чтобы Microsoft Access мог вначале определить эти переменные и включить их имена в инструкцию SQL, которая передается в ядро базы данных Jet.

В данном примере демонстрируется создание объекта **QueryDef** с помощью простой инструкции SQL. Этот запрос возвращает из таблицы «Заказы» все заказы, размещенные после 31.03.96:

```
Dim dbs As Database, qdf As QueryDef, strSQL As String
Set dbs = CurrentDb
strSQL = "SELECT * FROM Заказы WHERE ДатаРазмещения >#31.03.96#;"
Set qdf = dbs.CreateQueryDef("ВторойКвартал", strSQL)
```

В следующем примере создается аналогичный объект **QueryDef**, в котором используется значение, сохраняемое в переменной. Обратите внимание на способ включения в строку символов (#), обозначающих значение даты.

```
Dim dbs As Database, qdf As QueryDef, strSQL As String
Dim dteStart As Date
dteStart = #31.03.96#
Set dbs = CurrentDb
strSQL = "SELECT * FROM Заказы WHERE ДатаРазмещения" _
   & "> #" & dteStart & "#;"
Set qdf = dbs.CreateQueryDef("ВторойКвартал", strSQL)
```

В третьем примере создается объект **QueryDef**, в котором используется значение элемента управления «ДатаРазмещения» из формы «Заказы». Обратите внимание на полную ссылку на элемент управления и на способ включения в строку символов (#), обозначающих значение даты.

```
Dim dbs As Database, qdf As QueryDef, strSQL As String
Set dbs = CurrentDb
strSQL = "SELECT * FROM Заказы WHERE ДатаРазмещения " _
   & "> #" & Forms!Заказы!ДатаРазмещения & "#;"
Set qdf = dbs.CreateQueryDef("ВторойКвартал ", strSQL)
```

Параметры, которые были изменены или удалены в Microsoft Access 97

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"aclanChangedRemovedOptionsC"}

Следующие параметры, существовавшие в версиях 1.*x* и 2.0 были изменены:

Параметр	**Примечания**
Проверка синтаксиса (Syntax Checking) Интервал табуляции (Tab Stop Width) Окно диалога сортировки/группировки (Show Sort/Group Box)	Данные параметры оставлены для совместимости с предыдущими версиями Microsoft Access. Указание значений этих параметров не будет иметь последствий, но не приведет к возникновению ошибки.
Привязка объектов к сетке (Objects Snap To Grid) Сетка (Show Grid) Линейка (Show Ruler) Макрос назначения клавиш (Key Assignment Macro) Перемещение вложенных элементов (Move Enclosed Controls) Мастера элементов управления (Control Wizards)	Значения этих параметров могут быть заданы в программе, но не выводятся в диалоговом окне **Параметры**.
Стандартные панели инструментов (Built-In Toolbars Available) Изменение панелей инструментов (Can Customize Toolbars)	Эти параметры были заменены на параметры **Стандартные панели инструментов** и **Изменение панелей инструментов** в диалоговом окне **Параметры запуска** (вызываемого с помощью одноименной команды из меню **Сервис**). Однако разрешается по-прежнему использовать этот параметр в программах.
Линии сетки по умолчанию (Default Gridlines Behavior)	Данный параметр был заменен на параметры **Горизонтальные линии сетки по умолчанию (Default Gridlines Horizontal)** и **Вертикальные линии сетки по умолчанию (Default Gridlines Vertical)**. Однако разрешается по-прежнему использовать этот параметр в программах.

Следующие параметры в Microsoft Access 97 были удалены. При попытке передать следующие строковые значения в методы **SetOption** или **GetOption** в Microsoft Access генерируется ошибка выполнения.

First Week

First Weekday

Определение зарезервированных кодов ошибок Visual Basic

{ewc HLP95EN.DLL,DYNALINK,"пїSпїS. пїSпїSпїSпїSпїS":"acexaDeterminingErrorCodesX;vamsgTrappableErrors":1}

Следующая процедура создает таблицу Microsoft Access, содержащую коды ошибок и строки с описанием ошибок, зарезервированные в Visual Basic. В эту таблицу не включаются ошибки объектов доступа к данным(DAO).

Visual Basic резервирует часть из первых 1000 возможных кодов ошибок, поэтому в данном примере рассматриваются только номера от 1 до 1000. Другие коды ошибок резервируются ядром базы данных Microsoft Jet или являются допустимыми для описания специальных ошибок.

```
Sub CreateErrorsTable()
   Dim dbs As Database, tdf As TableDef, fld As Field
   Dim rst As Recordset, lngCode As Long
   Const conAppObjectError = "Ошибка, определяемая приложением или
объектом"

' Создает таблицу "Ошибки" с полями "КодОшибки" и "ОписаниеОшибки".
   Set dbs = CurrentDb
   Set tdf = dbs.CreateTableDef("Ошибки")
   Set fld = tdf.CreateField("КодОшибки", dbLong)
   tdf.Fields.Append fld
   Set fld = tdf.CreateField("ОписаниеОшибки", dbText, 255)
   tdf.Fields.Append fld
   dbs.TableDefs.Append tdf
   ' Открывает набор записей на базе таблицы "Ошибки".
   Set rst = dbs.OpenRecordset("Ошибки")
   ' Цикл по первой 1000 кодов ошибок Visual Basic.
   For lngCode = 1 To 1000
      On Error Resume Next
      ' Вызывает каждую ошибку.
      Err.Raise lngCode
      DoCmd.Hourglass True
      ' Пропускает коды ошибок приложения или ошибок объектов.
      If Err.Description <> conAppObjectError Then
         ' Добавляет код и описание ошибки в таблицу "Ошибки".
         rst.AddNew
         rst!КодОшибки = Err.Number
         rst!ОписаниеОшибки = Err.Description
         rst.Update
      End If
      ' Очищает объект Err.
      Err.Clear
   Next lngCode
   ' Закрывает набор записей.
   rst.Close
   DoCmd.Hourglass False
   MsgBox "Таблица 'Ошибки' создана."
End Sub
```

Преобразование программ с вызовами библиотек DLL

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"achowConvertDLLC;vastmDeclare;vastmFunction;vastmSub"}

Вызов процедур прикладного программного интерфейса Windows

Если в базе данных, созданной в одной из предыдущих версий Microsoft Access, вызываются процедуры из прикладного программного интерфейса Windows (API), то при преобразовании базы данных из формата версии 2.0 или более ранней в формат Microsoft Access 97 требуются некоторые изменения в программах.

Прикладной программный интерфейс Windows API содержит библиотеки динамической компоновки (DLL), включающие системно-ориентированные процедуры, сообщения, структуры данных, типы данных и инструкции, которые могут использоваться при разработке приложений для Windows 95 и Windows NT . Для вызова таких процедур в программах Visual Basic необходимо предварительно описать их с помощью инструкции **Declare.** После этого становится возможным вызов этих процедур таким же образом, как вызываются любые другие процедуры.

Версии 1.*x* и 2.0 Microsoft Access были 16-разрядными приложениями и выполнялись под управлением 16-разрядных версий Windows. Microsoft Access 97 является 32-разрядным приложением и выполнятся под управлением Windows 95 и Windows NT. Если в программах Access Basic вызывался 16-разрядный интерфейс Windows API, то при переходе к формату Microsoft Access 97 необходимо выполнить преобразование инструкций вызовов. Ниже перечислены некоторые необходимые действия.

- Проверьте все инструкции **Declare** и убедитесь, что они содержат ссылки на правильные библиотеки DLL. В следующей таблице приведены новые имена 32-разрядных библиотек DLL Windows.

16-разрядная DLL	32-разрядная DLL
User.dll	User32.dll
Kernel.dll	Kernel32.dll
GDI.dll	GDI32.dll

- Имена некоторых функций в 32-разрядном интерфейсе Windows API были изменены. Кроме того, функции из 32-разрядного интерфейса Windows API являются чувствительными к регистру. Проверьте, правильно ли указаны имена процедур и псевдонимы.
- Для аргументов некоторых функций из 32-разрядного интерфейса Windows API определены новые типы данных. Обратитесь к перечисленным ниже источникам и проверьте, какие из инструкций, содержащих вызовы функций, требуют изменения.
- Если на вашем компьютере сохранены 16-разрядные версии библиотек DLL с именами, которые совпадают с именами 32-разрядных версий, то Microsoft Access может сделать попытку вызова из этих библиотек в том случае, когда каталог, в котором находятся эти библиотеки, находится на пути пользователя раньше, чем каталог новых библиотек.
- Некоторые 32-разрядные библиотеки DLL содержат функции несколько отличных версий, измененных таким образом, чтобы принимать строки в кодировке Unicode и ANSI. Символ *A* в конце имени функции означает версию ANSI. Символ *W* в конце имени функции означает версию Unicode. Если функция принимает строковые аргументы, попробуйте добавить символ *A* к имени функции.

Более подробное описание работы с 32-разрядным интерфейсом Windows API и способов перевода приложений под 32-разрядную версию Microsoft Access 97 можно найти в следующих источниках:

- Windows API Viewer, входящий в комплект разработчика Microsoft Office 97, Developer Edition, включает синтаксис описаний, типы данных и константы для 32-разрядной версии Visual Basic.

- Комплект Microsoft Office Resource Kit содержит информацию о переносе в 32-разрядную среду программ 16-разрядной версии Microsoft Office, включая приложения, разработанные в предыдущих версиях Microsoft Access.
- Комплект разработчика программ Microsoft Win32 Software Development Kit содержит полный справочник по процедурам 32-разрядного интерфейса Windows API.

Вызов программ из других библиотек DLL

Если в приложении вызываются процедуры из других библиотек DLL, то при преобразовании базы данных к формату Microsoft Access 97 необходимо создать или приобрести 32-разрядные версии этих библиотек и внести требуемые изменения в программы.

Если приобрести 32-разрядну версию библиотеки DLL невозможно, то требуется промежуточная библиотека DLL, которая преобразует 32-разрядные вызовы в 16-разрядные. Более подробное описание см. в перечисленных выше источниках.

Объекты Microsoft Access

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acsumAccessObjHierarchyC"}

Использование национальных форматов даты в инструкциях SQL

{ewc HLP95EN.DLL,DYNALINK,"пїSпїS. пїSпїSпїSпїSпїS":"achowInternationalDateC"}

Visual Basic требует использования в инструкциях SQL англо-американских форматов даты. В бланке запроса допускается использование национальных форматов даты.

Определение зарезервированных кодов ошибок Microsoft Access и ядра базы данных Microsoft Jet

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acexaAccessJetErrorCodesXC"}

Следующая процедура создает таблицу Microsoft Access, содержащую коды ошибок и строки с описанием ошибок, зарезервированные в Microsoft Access и в ядре базы данных Microsoft Jet. В эту таблицу включаются не все ошибки, поскольку некоторые коды ошибок выходят за пределы диапазона кодов, задаваемого данной процедурой (0 — 4500).

```
Function ТаблицаОшибокAccessИJet() As Boolean
   Dim dbs As Database, tdf As TableDef, fld As Field
   Dim rst As Recordset, lngCode As Long
   Dim strAccessErr As String
   Const conAppObjectError = "Ошибка, определяемая приложением или
объектом"

   On Error GoTo Error_ТаблицаОшибокAccessИJet
   ' Создает таблицу "ОшибкиAccessИJet" с полями "КодОшибки" и
"ОписаниеОшибки".
   Set dbs = CurrentDb
   Set tdf = dbs.CreateTableDef("ОшибкиAccessИJet")
   Set fld = tdf.CreateField("КодОшибки", dbLong)
   tdf.Fields.Append fld
   Set fld = tdf.CreateField("ОписаниеОшибки", dbMemo)
   tdf.Fields.Append fld

   dbs.TableDefs.Append tdf
   ' Открывает набор записей таблицы "ОшибкиAccessИJet".
   Set rst = dbs.OpenRecordset("ОшибкиAccessИJet")
   ' Цикл по кодам ошибок.
   For lngCode = 0 To 4500
      On Error Resume Next
      ' Вызывает каждую ошибку.
      strAccessErr = AccessError(lngCode)
      DoCmd.Hourglass True
      ' Пропускает коды ошибок, не имеющих описания ошибки.
      If strAccessErr <> "" Then
         ' Пропускает коды ошибок приложения или ошибок объектов.
         If strAccessErr <> conAppObjectError Then
            ' Добавляет код и описание ошибки в таблицу "ОшибкиAccessИJet".
            rst.AddNew
            rst!КодОшибки = lngCode
            ' Добавляет описание в поле MEMO.
            rst!ОписаниеОшибки.AppendChunk strAccessErr
            rst.Update
         End If
      End If
   Next lngCode
   ' Закрывает набор записей.
   rst.Close
   DoCmd.Hourglass False
   RefreshDatabaseWindow
   MsgBox "Создана таблица ошибок Access и Jet"
   ТаблицаОшибокAccessИJet = True

Exit_ТаблицаОшибокAccessИJet:
```

```
   Exit Function

Error_ТаблицаОшибокAccessИJet:
   MsgBox Err & ": " & Err.Description
   ТаблицаОшибокAccessИJet = False
   Resume Exit_ТаблицаОшибокAccessИJet
End Function
```

Метод Item

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthItemC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthItemX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthItemA"}

Метод **Item** возвращает конкретный компонент семейства, определяемый по номеру или по ключу

Синтаксис

объект.**Item(***индекс***)**

Метод **Item** использует следующие аргументы.

Аргумент	Описание
объект	Семейство **References**.
индекс	Выражение, определяющее положение компонента в семействе, указанного в аргументе *объект*. Если этот аргумент задается с помощью числового выражения, то *индекс* представляет собой число в диапазоне от 1 до значения свойства **Count** семейства. Если аргумент *индекс* определяется строковым выражением, то его значением является имя компонента семейства.

Дополнительные сведения

Если значение аргумента *индекс* не соответствует ни одному из существующих компонентов семейства, возникает ошибка

Метод **Item** по умолчанию является компонентом семейства **References**, поэтому не требуется указывать его явно. Например, две следующие строки эквивалентны:

```
Debug.Print References(1).Name

Debug.Print References.Item(1).Name
```

Свойство Kind

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproKindC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproKindX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproKindA"}

Свойство **Kind** указывает тип ссылки, представленной объектом **Reference**.

Использование свойства Kind

Свойство **Kind** является доступным только для чтения, причем доступ к нему возможен только в программе Visual Basic.

Свойство **Kind** возвращает следующие значения:

Значение	Описание
Project (1)	Объект **Reference** представляет ссылку на проект Visual Basic.
TypeLib (0)	Объект **Reference** представляет ссылку на файл, который содержит библиотеку типов.

Метод ApplyFilter

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthactApplyFilterC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acmthactApplyFilterX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acmthactApplyFilterA"}

Метод **ApplyFilter** выполняет макрокоманду **ПрименитьФильтр (ApplyFilter)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.ApplyFilter [*имяФайла*] [, *условиеWhere*]

Метод **ApplyFilter** использует следующие аргументы.

Аргумент	Описание
имяФайла	Строковое выражение, представляющее допустимое имя запроса в текущей базе данных.
условиеWhere	Строковое выражение, представляющее допустимое предложение SQL WHERE без ключевого слова WHERE.

Дополнительные сведения

Необходимо определить по крайней мере один аргумент метода **ApplyFilter**. Если указаны значения обоих аргументов, то *условиеWhere* применяется к фильтру.

Максимальная длина строки в аргументе *условиеWhere* составляет 32 768 символов (в отличие от аргумента «Условие отбора» в окне макроса, максимальная длина которого составляет 256 символов).

Для того чтобы определить аргумент *условиеWhere* и оставить аргумент *имяФайла* пустым, необходимо ввести запятую, представляющую аргумент *имяФайла*.

Метод ApplyFilter, пример

В данном примере метод **ApplyFilter** используется для вывода только тех записей, которые содержат значение «Иванов» в поле «Фамилия»:

```
DoCmd.ApplyFilter , "Фамилия = 'Иванов'"
```

Метод Beep

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactBeepC;vastmBeep"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactBeepX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactBeepA"}

Метод **Beep** выполняет макрокоманду **Сигнал (Beep)** в программе Visual Basic. Подробное описание макрокоманды см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Beep

Данный метод не требует аргументов.

Дополнительные сведения

Для подачи звукового сигнала через динамик компьютера используется также инструкция **Beep** Visual Basic.

Метод CancelEvent

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactCancelEventC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactCancelEventX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactCancelEventA"}

Метод **CancelEvent** выполняет макрокоманду **ОтменитьСобытие (CancelEvent)** в программе Visual Basic. Подробное описание макрокоманды см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.CancelEvent

Данный метод не требует аргументов.

Дополнительные сведения

Метод **CancelEvent** дает результат только при его вызове в ответ на событие. Данный метод отменяет событие.

Все события, которые допускают отмену в программе Visual Basic, имеют аргумент «Cancel», позволяющий отменить событие без вызова метода **CancelEvent**. Отмена событий **Нажатие клавиши (KeyPress)** и **Кнопка вниз (MouseDown)** (только для нажатия правой кнопки мыши) допускается только в макросах, но не в процедурах обработки события. Для отмены этих событий необходимо вызвать макрокоманду **ОтменитьСобытие (CancelEvent)** в макросе.

Метод Close

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactCloseC;damthClose"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactCloseX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactCloseA"}

Метод **Close** выполняет макрокоманду **Закрыть (Close)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Close [*типОбъекта*, *имяОбъекта*], [*сохранение*]

Метод **Close** использует следующие аргументы.

Аргумент	Описание
типОбъекта	Одна из следующих встроенных констант: **acDefault** (значение по умолчанию) **acForm** (форма) **acMacro** (макрос) **acModule** (модуль) **acQuery** (запрос) **acReport** (отчет) **acTable** (таблица)
имяОбъекта	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*.
сохранение	Одна из следующих встроенных констант: **acSaveNo** (выход без сохранения) **acSavePrompt** (подтверждение, значение по умолчанию) **acSaveYes** (сохранение без подтверждения) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acSavePrompt**).

Дополнительные сведения

Если оставлены пустыми аргументы *типОбъекта* и *имяОбъекта*, (по умолчанию аргументу *типОбъекта* присваивается значение **acDefault**), Microsoft Access закрывает активное окно. Для того чтобы определить аргумент *сохранение* и оставить аргументы *типОбъекта* и *имяОбъекта* пустыми, необходимо ввести запятые, представляющие аргументы *типОбъекта* и *имяОбъекта*.

Метод Close, пример

В данном примере метод **Close** закрывает форму Просмотр заказа и сохраняет все изменения без приглашения подтвердить изменения:

```
DoCmd.Close acForm, "Просмотр заказа", acSaveYes
```

Метод CopyObject

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthactCopyObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthactCopyObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthactCopyObjectA"}

Метод **CopyObject** выполняет макрокоманду **КопироватьОбъект (CopyObject)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.CopyObject [*БДПолучатель*] [, *новоеИмя*] [, *типИсточника*, *имяИсточника*]

Метод **CopyObject** использует следующие аргументы.

Аргумент	Описание
БДПолучатель	Строковое выражение, представляющее допустимые путь и имя файла базы данных, в которую копируется объект. Для того чтобы указать текущую базу данных, оставьте данный аргумент пустым. Если вызвать программу Visual Basic, содержащую метод **CopyObject**, из библиотечной базы данных и оставить данный аргумент пустым, Microsoft Access копирует объект в библиотечную базу данных.
новоеИмя	Строковое выражение, представляющее новое имя копируемого объекта. Для того чтобы скопировать объект в другую базу данных под тем же именем, оставьте данный аргумент пустым.
типИсточника	Одна из следующих встроенных констант: **acDefault** (значение по умолчанию) **acForm** (форма) **acMacro** (макрос) **acModule** (модуль) **acQuery** (запрос) **acReport** (отчет) **acTable** (таблица)
имяИсточника	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типИсточника*. Если вызвать программу Visual Basic, содержащую метод **CopyObject**, из библиотечной базы данных, Microsoft Access проводит поиск объекта с этим именем сначала в библиотечной базе данных, а затем в текущей базе данных.

Дополнительные сведения

Данный метод требует обязательного указания значения либо одного из аргументов *БДПолучатель* или *новоеИмя*, либо обоих.

Если оставлены пустыми аргументы *типИсточника* и *имяИсточника* (по умолчанию аргументу *типИсточника* присваивается значение **acDefault**), копирует объект, выделенный в окне базы данных. Для выделения объекта в окне базы данных следует вызвать макрокоманду **ВыделитьОбъект (SelectObject)** или метод **SelectObject** с заданным для аргумента *вОкнеБД* значением «Да» (**True**).

Для того чтобы определить аргументы *типИсточника* и *имяИсточника,* но оставить при этом любой из аргументов *новоеИмя* или *БДПолучатель* пустым, необходимо ввести запятую,

представляющую аргумент *новоеИмя* или *БДПолучатель*. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Метод CopyObject, пример

В данном примере метод **CopyObject** копирует таблицу «Сотрудники» в текущую базу данных под новым именем:

```
DoCmd.CopyObject, "Сотрудники (копия)", acTable, "Сотрудники"
```

Метод DeleteObject

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactDeleteObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthactDeleteObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthactDeleteObjectA"}

Метод **DeleteObject** выполняет макрокоманду **УдалитьОбъект (DeleteObject)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.DeleteObject [*типОбъекта*, *имяОбъекта*]

Метод **DeleteObject** использует следующие аргументы.

Аргумент	Описание
типОбъекта	Одна из следующих встроенных констант: **acDefault** (значение по умолчанию) **acForm** (форма) **acMacro** (макрос) **acModule** (модуль) **acQuery** (запрос) **acReport** (отчет) **acTable** (таблица)
имяОбъекта	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*. Если вызвать программу Visual Basic, содержащую метод **DeleteObject**, из библиотечной базы данных, Microsoft Access проводит поиск объекта с этим именем сначала в библиотечной базе данных, а затем в текущей базе данных.

Дополнительные сведения

Если оставлены пустыми аргументы *типОбъекта* и *имяОбъекта* (по умолчанию аргументу *типОбъекта* присваивается значение **acDefault**), Microsoft Access удаляет объект, выделенный в окне базы данных. Для выделения объекта в окне базы данных следует вызвать макрокоманду **ВыделитьОбъект (SelectObject)** или метод **SelectObject** с заданным для аргумента *вОкнеБД* значением «Да» (**True**).

Метод DeleteObject, пример

В данном примере удаляется указанная таблица:

```
DoCmd.DeleteObject acTable, "Прежняя таблица Сотрудники "
```

Метод DoMenuItem

{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acmthactDoMenuItemC;vastmSendKeys"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acmthactDoMenuItemX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acmthactDoMenuItemA"}

Метод **DoMenuItem** выполняет макрокоманду **КомандаМеню (DoMenuItem)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Примечание. В Microsoft Access 97 метод **DoMenuItem** заменен на метод **RunCommand.** Метод **DoMenuItem** включен в данную версию Microsoft Access только для совместимости с предыдущими версиями. При запуске существующей программы Visual Basic, содержащей вызов метода **DoMenuItem**, Microsoft Access высветит подходящую команду меню или панели инструментов. Однако в отличие от макрокоманды **КомандаМеню**в макросе вызов метода **DoMenuItem** в программе Visual Basic не преобразуется в вызов метода **RunCommand** при преобразовании базы данных, созданной в предыдущей версии Microsoft Access.

Некоторые команды из предыдущих версий Microsoft Access не доступны в Microsoft Access 97 и вызов метода **DoMenuItem** в программе Visual Basic, запускающего такие команды, приведет к ошибке. Необходимо исправить или удалить все такие вызовы метода **DoMenuItem** в программе Visual Basic. Список исключенных команд можно найти в разделе Аргументы макрокоманды «КомандаМеню» (DoMenuItem), недопустимые в макрокоманде «ВыполнитьКоманду» (RunCommand).

Синтаксис

DoCmd.DoMenuItem *строкаМеню*, *имяМеню*, *команда* [, *подкоманда*] [, *версия*]

Метод **DoMenuItem** использует следующие аргументы.

Аргумент	Описание
строкаМеню	Для строки меню режима формы используйте встроенную константу **acFormBar**. Строка меню, соответствующая другому режиму, определяется с помощью номера, для определения которого следует открыть в окне макроса список значений аргумента «Строка меню» макрокоманды **КомандаМеню** и определить порядковый номер нужного элемента в списке (нумерация элементов списка начинается с 0).
имяМеню	Допускается использование одной из следующих встроенных констант: **acFile** (Файл) **acEditMenu** (Правка) **acRecordsMenu** (Записи) Константа **acRecordsMenu** используется только для строки меню режима формы в базах данных Microsoft Access версии 2.0 и Microsoft Access для Windows 95. Имена меню, соответствующих другому режиму, определяются с помощью номера, для определения которого следует открыть в окне макроса список значений аргумента «Название меню» макрокоманды **КомандаМеню** и определить порядковый номер нужного элемента в списке (нумерация элементов списка начинается с 0).
команда	Допускается использование одной из следующих встроенных констант: **acNew** (Создать)

acSaveForm (Сохранить форму)
acSaveFormAs (Сохранить форму как)
acSaveRecord (Сохранить запись)
acUndo (Отменить)
acCut (Вырезать)
acCopy (Копировать)
acPaste (Вставить)
acDelete (Удалить)
acSelectRecord (Выделить запись)
acSelectAllRecords (Выделить все записи)
acObject (Объект)
acRefresh (Обновить)

Для других команд следует открыть окно макроса и определить порядковый номер нужного элемента в списке значений аргумента «Команда» (нумерация элементов списка начинается с 0).

подкоманда Допускается использование одной из следующих встроенных констант:

acObjectVerb (команда объекта)
acObjectUpdate (обновление объекта)

Константа **acObjectVerb** представляет первую команду в подменю, которое открывается командой **Объект** в меню **Правка**. Эта команда определяется типом объекта. Например, для объекта Paintbrush, допускающего изменение, первой командой является команда **Изменить**.

Для других подкоманд следует открыть окно макроса и определить порядковый номер нужного элемента в списке значений аргумента «Подкоманда» (нумерация элементов списка начинается с 0).

версия Используйте встроенную константу **acMenuVer70** для программ, написанных для баз данных Microsoft Access 95, встроенную константу **acMenuVer20** для программ, написанных для баз данных Microsoft Access версии 2.0, и встроенную константу **acMenuVer1X** для программ, написанных для баз данных Microsoft Access версии 1.x. Данный аргумент является доступным только в программах Visual Basic.

Примечание. Значением по умолчанию для данного аргумента является **acMenuVer1X**, поэтому любые программы из баз данных Microsoft Access версии 1.x будут выполняться без изменений. При разработке программы для базы данных Microsoft Access 95 или версии 2.0, в которой требуется использовать в методе **DoMenuItem** команды меню Microsoft Access 95 или версии 2.0, необходимо задать для этого аргумента значение **acMenuVer70** или **acMenuVer20**.

Кроме того, при определении значений аргументов метода **DoMenuItem** по номерам элементов в списках значений аргументов «Строка меню», «Название меню», «Команда» и «Подкоманда» необходимо при указании в аргументе *версия* константы **acMenuVer70** использовать списки значений Microsoft Access 95, при указании константы **acMenuVer20** использовать списки значений Microsoft Access версии 2.0, а при указании константы **acMenuVer1X** (или пустом значении этого аргумента) - списки значений Microsoft Access версии 1.x.

Примечание. Значения **acMenuVer80** для этого аргумента не существует. Невозможно использовать метод **DoMenuItem** для высвечивания команд Microsoft Access 97 (хотя существующие вызовы метода в программе Visual Basic **DoMenuItem** будут работать). Вместо этого следует использовать метод **RunCommand**.

Дополнительные сведения

Списки значений аргументов «Название меню», «Команда» и «Подкоманда» определяются выбранными значениями предыдущих аргументов. Для каждого из аргументов *строкаМеню*, *имяМеню*, *команда* и *подкоманда* необходимо использовать соответствующие ему встроенные константы или номера.

Если аргумент *подкоманда* оставлен пустым, но требуется задать значение аргумента *версия*, необходимо ввести запятую, представляющую аргумент *подкоманда*. Если оставлены пустыми аргументы *подкоманда* и *версия*, вводить запятые после аргумента *команда* не требуется.

Метод DoMenuItem, пример

В данном примере метод **DoMenuItem** выполняет команду **Вставить** из меню **Правка** режима формы базы данных Microsoft Access для Windows 95:

```
DoCmd.DoMenuItem acFormBar, acEditMenu, acPaste, , acMenuVer70
```

В следующем примере выполняется команда **Мозаика** из меню **Окно** режима формы базы данных Microsoft Access версии 2.0:

```
DoCmd.DoMenuItem acFormBar, 4, 0, , acMenuVer20
```

Метод Echo (объект DoCmd)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthactEchoC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acmthactEchoX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acmthactEchoA"}

Метод **Echo** объекта **DoCmd** выполняет макрокоманду **ВыводНаЭкран (Echo)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Echo *режим* [, *текст*]

Метод **Echo** использует следующие аргументы.

Аргумент	Описание
режим	Значение **True** (-1) включает режим отображения на экране, а значение **False** (0) отключает этот режим.
текст	Строковое выражение, представляющее текст сообщения, выводящегося в строке состояния.

Дополнительные сведения

Если аргумент *текст* оставлен пустым, вводить запятую после аргумента *режим* не требуется.

После отключения режима отображения в программе Visual Basic, этот режим окажется отключенным и после прерывания выполнения программы нажатием клавиш CTRL+BREAK или после прерывания процедуры Visual Basic на точке останова. Удобно создать макрос, включающий режим отображения, и назначить этот макрос на клавиатуру или связать его со специальной командой меню. Это позволит вновь включать режим отображения, отключенный в программе Visual Basic.

Метод **Echo** объекта **DoCmd**, выполняющий макрокоманду **ВыводНаЭкран (Echo)** в программе Visual Basic, был добавлен в Microsoft Access для Windows 95 для совместимости с предыдущими версиями. Рекомендуется использовать в новых программах метод **Echo** объекта **Application.**

Метод Echo (объект DoCmd), пример

В данном примере метод **Echo** отключает режим отображения и выводит сообщение в строке состояния при выполнении программы Visual Basic:

```
DoCmd.Echo False, "Выполняется программа Visual Basic."
```

Метод FindNext

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactFindNextC;damthFindFirst"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactFindNextX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactFindNextA"}

Метод **FindNext** выполняет макрокоманду **СледующаяЗапись (FindNext)** в программе Visual Basic. Подробное описание макрокоманды см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.FindNext

Данный метод не требует аргументов.

Метод FindRecord

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactFindRecordC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactFindRecordX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactFindRecordA"}

Метод **FindRecord** выполняет макрокоманду **НайтиЗапись (FindRecord)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.FindRecord *образец* [, *совпадение]* [, *регистр*] [, *поиск*] [, *формат*] [, *текущееПоле*] [, *сНачала*]

Метод **FindRecord** использует следующие аргументы.

Аргумент	Описание
образец	Выражение, значением которого является текст, число или дата. Данное выражение задает образец для поиска.
совпадение	Одна из следующих встроенных констант: **acAnywhere** (с любой частью поля) **acEntire** (со значением поля) **acStart** (с начальными символами в значении поля) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acEntire**).
регистр	Значение **True** (-1) задает поиск с учетом регистра символов, а **False** (0) поиск без учета регистра. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**False**).
поиск	Одна из следующих встроенных констант: **acDown** (вниз) **acSearchAll** (везде) **acUp** (вверх) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (acSearchAll).
формат	Значение **True** задает поиск с учетом формата полей, а значение **False** поиск данных в том виде, в котором они сохраняются в базе данных. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**False**).
текущееПоле	Одна из следующих встроенных констант: **acAll** (все поля) **acCurrent** (текущее поле) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (acCurrent).
сНачала	Значение **True** задает начало поиска с первой записи. **False** задает начало поиска с текущей записи. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**True**).

Дополнительные сведения

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если опускаются один или несколько последних аргументов, вводить запятые вслед за последним указанным

аргументом не требуется.

Метод FindRecord, пример

Следующая конструкция обнаружит в текущем поле фамилию «Рост». «Ростов» или «рост» найдены не будут.

```
DoCmd.FindRecord "Рост",, True,, True
```

Метод GoToControl

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactGoToControlC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactGoToControlX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactGoToControlA"}

Метод **GoToControl** выполняет макрокоманду **КЭлементуУправления (GoToControl)** в программе Visual Basic. Подробное описание макрокоманды и ее аргумента см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.GoToControl *имяЭлемента*

Метод **GoToControl** использует следующий аргумент.

Аргумент	Описание
имяЭлемента	Строковое выражение, представляющее имя элемента управления в активной форме или в объекте в режиме таблицы.

Дополнительные сведения

В аргументе *имяЭлемента* следует указать только имя элемента управления, полная ссылка не требуется.

Допускается также использование в данном аргументе переменной типа **Control**.

```
Dim ctl As Control
Set ctl = Forms!Форма1!Поле3
DoCmd.GoToControl ctl.Name
```

Метод **SetFocus** позволяет перевести фокус на элемент управления в форме или в любой из ее подчиненных форм, а также на поля в открытых таблице, запросе или форме в режиме таблицы. Этот способ перевода фокуса является наиболее подходящим для использования в программах Visual Basic, особенно при переходах на элементы управления в подчиненных и вложенных формах, поскольку при этом допускается использование полного синтаксиса для указания нужного элемента управления.

Метод GoToControl, пример

В данном примере метод **GoToControl** переводит фокус на поле «КодСотрудника»:

```
DoCmd.GoToControl "КодСотрудника"
```

Метод GoToPage (объект DoCmd)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthactGoToPageC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acmthactGoToPageX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acmthactGoToPageA"}

Метод **GoToPage** объекта **DoCmd** выполняет макрокоманду **НаСтраницу (GoToPage)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.GoToPage [*номерСтраницы*] [, *отЛевогоКрая*, *отВерхнегоКрая*]

Метод **GoToPage** использует следующие аргументы.

Аргумент	Описание
номерСтраницы	Числовое выражение, представляющее допустимый номер страницы активной формы. Если оставить данный аргумент пустым, фокус останется на текущей странице. Аргументы *отЛевогоКрая* и *отВерхнегоКрая* позволяет указать часть страницы, выводящуюся на экран.
отЛевогоКрая	Числовое выражение, представляющее допустимый горизонтальный сдвиг страницы от левого края окна.
отВерхнегоКрая	Числовое выражение, представляющее допустимый вертикальный сдвиг страницы от верхнего края окна.

Дополнительные сведения

Значения аргументов *отЛевогоКрая* и *отВерхнегоКрая* измеряются в единицах твип.

Для того чтобы определить аргумент *отЛевогоКрая* и *отВерхнегоКрая*, оставив аргумент *номерСтраницы* пустым, необходимо ввести запятую, представляющую аргумент *номерСтраницы*. Если аргументы *отЛевогоКрая* и *отВерхнегоКрая* не заданы, вводить запятые после аргумента *номерСтраницы* не требуется.

Метод **GoToPage** объекта **DoCmd**, выполняющий макрокоманду **НаСтраницу (GoToPage)** в программе Visual Basic, был добавлен в Microsoft Access для Windows 95 для совместимости с предыдущими версиями. Рекомендуется использовать в новых программах метод **GoToPage** объекта **Form.**

Метод GoToPage (объект DoCmd), пример

В данном примере метод **GoToPage** переводит фокус на вторую страницу активной формы в указанную позицию:

```
DoCmd.GoToPage 2, 1440, 567
```

Метод GoToRecord

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactGoToRecordC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthactGoToRecordX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthactGoToRecordA"}

Метод **GoToRecord** выполняет макрокоманду **НаЗапись (GoToRecord)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.GoToRecord [*типОбъекта*, *имяОбъекта*] [, *запись*] [, *смещение*]

Метод **GoToRecord** использует следующие аргументы.

Аргумент	Описание
типОбъекта	Одна из следующих встроенных констант: **acActiveDataObject** (активный объект, значение по умолчанию) **acDataForm** (форма) **acDataQuery** (запрос) **acDataTable** (таблица)
имяОбъекта	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*.
запись	Одна из следующих встроенных констант: **acFirst** (первая) **acGoTo** (с указанным номером) **acLast** (последняя) **acNewRec** (новая) **acNext** (следующая) **acPrevious** (предыдущая) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (acNext).
смещение	Числовое выражение, представляющее число записей, на которое отстоит искомая запись от текущей вперед или назад, если в аргументе *запись* заданы константы **acNext** или **acPrevious**, или номер искомой записи, если в аргументе *запись* задана константа **acGoTo**. Значение выражения должно представлять допустимый номер записи.

Дополнительные сведения

Если оставлены пустыми аргументы *типОбъекта* и *имяОбъекта* (по умолчанию аргументу *типОбъекта* присваивается значение **acActiveDataObject**), подразумевается активный объект.

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если опускаются один или несколько последних аргументов, вводить запятые вслед за последним указанным аргументом не требуется.

Метод GoToRecord, пример

В данном примере метод **GoToRecord** делает текущей записью седьмую запись в форме «Сотрудники»:

```
DoCmd.GoToRecord acDataForm, "Сотрудники", acGoTo, 7
```

Метод Hourglass

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactHourglassC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactHourglassX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactHourglassA"}

Метод **Hourglass** выполняет макрокоманду **ПесочныеЧасы (Hourglass)** в программе Visual Basic. Подробное описание макрокоманды и ее аргумента см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Hourglass *значок*

Метод **Hourglass** использует следующие аргументы.

Аргумент	Описание
значок	Значение **True** (-1) задает вывод указателя в виде песочных часов (или другого значка по выбору пользователя). Значение **False** (0) задает стандартный вид указателя.

Метод Hourglass, пример

В данном примере метод **Hourglass** задает вывод указателя в виде песочных часов (или другого значка по выбору пользователя) во время выполнения программы Visual Basic:

```
DoCmd.Hourglass True
```

Метод Maximize

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactMaximizeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactMaximizeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactMaximizeA"}

Метод **Maximize** выполняет макрокоманду **Развернуть (Maximize)** в программе Visual Basic. Подробное описание макрокоманды см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Maximize

Данный метод не требует аргументов.

Метод Minimize

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactMinimizeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactMinimizeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactMinimizeA"}

Метод **Minimize** выполняет макрокоманду **Свернуть (Minimize)** в программе Visual Basic. Подробное описание макрокоманды см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Minimize

Данный метод не требует аргументов.

Метод MoveSize

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthactMoveSizeC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acmthactMoveSizeX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acmthactMoveSizeA"}

Метод **MoveSize** выполняет макрокоманду **СдвигРазмер (MoveSize)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.MoveSize [*отЛевогоКрая*] [, *отВерхнегоКрая*] [, *ширина*] [, *высота*]

Метод **MoveSize** использует следующие аргументы.

Аргумент	Описание
отЛевогоКрая	Числовое выражение.
отВерхнегоКрая	Числовое выражение.
ширина	Числовое выражение.
высота	Числовое выражение.

Дополнительные сведения

В методе **MoveSize** требуется указать значение по крайней мере одного аргумента. Если какой-либо аргумент оставлен пустым, используется текущее значение для окна.

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если опускаются один или несколько последних аргументов, вводить запятые вслед за последним указанным аргументом не требуется.

Значения аргументов измеряются в единицах твип.

Метод MoveSize, пример

В данном примере изменяется положение и высота активного окна; ширина при этом остается неизменной:

```
DoCmd.MoveSize 1440, 2400, , 2000
```

Метод OpenForm

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactOpenFormC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactOpenFormX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactOpenFormA"}

Метод **OpenForm** выполняет макрокоманду **ОткрытьФорму (OpenForm)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.OpenForm *имяФормы* [, *режим*] [, *имяФайла*] [, *условиеWhere*] [, *режимДанных*] [, *режимОкна*] [, *аргументыОткрытия*]

Метод **OpenForm** использует следующие аргументы.

Аргумент	Описание
имяФормы	Строковое выражение, представляющее допустимое имя формы в текущей базе данных. Если вызвать программу Visual Basic, содержащую метод **OpenForm**, из библиотечной базы данных, Microsoft Access проводит поиск формы с указанным именем сначала в библиотечной базе данных, а затем в текущей базе данных.
режим	Одна из следующих встроенных констант:: **acDesign** (конструктор) **acFormDS** (таблица) **acNormal** (форма) **acPreview** (просмотр) Константа **acNormal** задает открытие формы в режиме формы. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (acNormal).
имяФайла	Строковое выражение, представляющее допустимое имя запроса в текущей базе данных.
условиеWhere	Строковое выражение, представляющее допустимое предложение SQL WHERE без ключевого слова WHERE.
режимДанных	Одна из следующих встроенных констант: **acFormAdd** (добавление) **acFormEdit** (изменение) **acFormPropertySettings** (значения свойств) **acFormReadOnly** (только чтение) Если оставить данный аргумент пустым (предполагается значение по умолчанию **acFormPropertySettings**), Microsoft Access открывает форму в режиме данных, который определяется значениями свойств **Разрешить изменение (AllowEdits), Разрешить удаление (AllowDeletions), Разрешить добавление (AllowAdditions)** и **Ввод данных (DataEntry).**
режимОкна	Одна из следующих встроенных констант: **acDialog** (окно диалога) **acHidden** (невидимое) **acIcon** (значок) **acWindowNormal** (обычное) Если оставить данный аргумент пустым, подразумевается значение по умолчанию **acWindowNormal**.

аргументыОткрытия	Строковое выражение, определяющее значение свойства формы **OpenArgs.** В дальнейшем это значение может быть использовано в программе в модуле формы, например, в процедуре обработки события **Открытие (Open)**. Ссылки на свойство **OpenArgs** допускаются также в макросах и выражениях. Предположим, например, что в режиме ленточной формы открывается список клиентов. Для того чтобы при открытии формы установить фокус на запись об определенном клиенте, достаточно указать имя клиента в аргументе *аргументыОткрытия*, а затем вызвать метод **FindRecord**, чтобы переместить фокус на запись для клиента с нужным именем. Данный аргумент является доступным только в программах Visual Basic.

Дополнительные сведения

Максимальная длина строки в аргументе *условиеWhere* составляет 32 768 символов (в отличие от аргумента «Условие отбора» в окне макроса, максимальная длина которого составляет 256 символов).

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Метод OpenForm, пример

В данном примере в форме «Сотрудники», открывающейся в режиме формы, выводятся только записи со значением «Иванов» в поле «Фамилия». Допускается изменение выводящихся записей и добавление новых.

```
DoCmd.OpenForm "Сотрудники", , ,"Фамилия = 'Иванов'"
```

Метод OpenModule

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactOpenModuleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactOpenModuleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactOpenModuleA"}

Метод **OpenModule** выполняет макрокоманду **ОткрытьМодуль (OpenModule)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.OpenModule [*имяМодуля*] [, *имяПроцедуры*]

Метод **OpenModule** использует следующие аргументы.

Аргумент	Описание
имяМодуля	Строковое выражение, представляющее допустимое имя открываемого модуля Visual Basic. Если оставить данный аргумент пустым, Microsoft Access проводит поиск процедуры, указанной в аргументе *имяПроцедуры*, во всех стандартных модулях базы данных и открывает модуль, содержащий эту процедуру, на данной процедуре. Если вызвать программу Visual Basic, содержащую метод **OpenModule**, из библиотечной базы данных, Microsoft Access проводит поиск модуля с указанным именем сначала в библиотечной базе данных, а затем в текущей базе данных.
имяПроцедуры	Строковое выражение, представляющее допустимое имя процедуры, на которой требуется открыть модуль. Если оставить данный аргумент пустым, модуль открывается на разделе описаний.

Дополнительные сведения

Необходимо определить по крайней мере один аргумент метода OpenModule. Если указаны значения обоих аргументов, то Microsoft Access открывает указанный модуль на указанной процедуре.

Если аргумент *имяПроцедуры* оставлен пустым, вводить запятую после аргумента *имяМодуля* не требуется.

Метод OpenModule, пример

В данном примере модуль «Служебные функции» открывается на процедуре **Function** IsLoaded():

```
DoCmd.OpenModule "Служебные функции","IsLoaded"
```

Метод OpenQuery

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactOpenQueryC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactOpenQueryX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactOpenQueryA"}

Метод **OpenQuery** выполняет макрокоманду **ОткрытьЗапрос (OpenQuery)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.OpenQuery *имяЗапроса* [, *режим*] [, *режимДанных*]

Метод **OpenQuery** использует следующие аргументы.

Аргумент	Описание
имяЗапроса	Строковое выражение, представляющее допустимое имя запроса в текущей базе данных. Если вызвать программу Visual Basic, содержащую метод **OpenQuery**, из библиотечной базы данных, Microsoft Access проводит поиск запроса с указанным именем сначала в библиотечной базе данных, а затем в текущей базе данных.
режим	Одна из следующих встроенных констант: **acViewDesign** (конструктор) **acViewNormal** (таблица, значение по умолчанию) **acViewPreview** (просмотр) Если в аргументе *имяЗапроса* указано имя запроса на выборку, перекрестного запроса, запроса на объединение или запроса к серверу, у которого свойство **Возврат записей (ReturnsRecords)** имеет значение -1, то константа **acViewNormal** определяет открытие результирующего набора записей. Если аргумент *имяЗапроса* задает имя запроса на изменение, управляющего запроса или запроса к серверу, у которого свойство **Возврат записей (ReturnsRecords)** имеет значение 0, то константа **acViewNormal** определяет запуск запроса. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acViewNormal**).
режимДанных	Одна из следующих встроенных констант: **acAdd** (добавление) **acEdit** (изменение) **acReadOnly** (только чтение) Если оставить данный аргумент пустым, подразумевается значение по умолчанию **acEdit**.

Дополнительные сведения

Для того чтобы определить аргумент *режимДанных* и оставить аргумент *режим* пустым, необходимо ввести запятую, представляющую аргумент *режим*. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Метод OpenQuery, пример

В данном примере запрос «Выработка сотрудников» открывается в режиме таблицы, в котором пользователю разрешается только просмотр записей:

```
DoCmd.OpenQuery "Выработка сотрудников", , acReadOnly
```

Метод OpenReport

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactOpenReportC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthactOpenReportX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthactOpenReportA"}

Метод **OpenReport** выполняет макрокоманду **ОткрытьОтчет (OpenReport)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.OpenReport *имяОтчета* [, *режим*] [, *имяФайла*] [, *условиеWhere*]]

Метод **OpenReport** использует следующие аргументы.

Аргумент	Описание
имяОтчета	Строковое выражение, представляющее допустимое имя отчета в текущей базе данных. Если вызвать программу Visual Basic, содержащую метод **OpenReport**, из библиотечной базы данных, Microsoft Access проводит поиск отчета с указанным именем сначала в библиотечной базе данных, а затем в текущей базе данных.
режим	Одна из следующих встроенных констант: **acViewDesign** (конструктор) **acViewNormal** (печать, значение по умолчанию) **acViewPreview** (просмотр) Константа **acViewNormal** задает немедленный вывод отчета на печать. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acViewNormal**).
имяФайла	Строковое выражение, представляющее допустимое имя запроса в текущей базе данных.
условиеWhere	Строковое выражение, представляющее допустимое предложение SQL WHERE без ключевого слова WHERE.

Дополнительные сведения

Максимальная длина строки в аргументе *условиеWhere* составляет 32 768 символов (в отличие от аргумента «Условие отбора» в окне макроса, максимальная длина которого составляет 256 символов).

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если опускаются один или несколько последних аргументов, вводить запятые вслед за последним указанным аргументом не требуется.

Метод OpenReport, пример

В данном примере отчет «Продажи» печатается с использованием запроса «Фильтр для отчета»:

```
DoCmd.OpenReport "Продажи", acViewNormal, "Фильтр для отчета"
```

Метод OpenTable

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactOpenTableC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactOpenTableX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactOpenTableA"}

Метод **OpenTable** выполняет макрокоманду **ОткрытьТаблицу (OpenTable)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.OpenTable *имяТаблицы* [, *режим*] [, *режимДанных*]

Метод **OpenTable** использует следующие аргументы.

Аргумент	Описание
имяТаблицы	Строковое выражение, представляющее допустимое имя таблицы в текущей базе данных. Если вызвать программу Visual Basic, содержащую метод **OpenTable**, из библиотечной базы данных, Microsoft Access проводит поиск таблицы с указанным именем сначала в библиотечной базе данных, а затем в текущей базе данных.
режим	Одна из следующих встроенных констант: **acViewDesign** (конструктор) **acViewNormal** (таблица, значение по умолчанию) **acViewPreview** (просмотр) Константа **acViewNormal** задает открытие таблицы в режиме таблицы. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acViewNormal**).
режимДанных	Одна из следующих встроенных констант: **acAdd** (добавление) **acEdit** (изменение, значение по умолчанию) **acReadOnly** (только чтение) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acEdit**).

Дополнительные сведения

Для того чтобы определить аргумент *режимДанных* и оставить аргумент *режим* пустым, необходимо ввести запятую, представляющую аргумент *режим*. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Метод OpenTable, пример

В данном примере таблица «Сотрудники» открывается в режиме предварительного просмотра.

```
DoCmd.OpenTable "Сотрудники", acViewPreview
```

Метод OutputTo

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acmthactOutputToC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acmthactOutputToX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acmthactOutputToA"}

Метод **OutputTo** выполняет макрокоманду **ВывестиВФормате (OutputTo)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.OutputTo *типОбъекта* [, *имяОбъекта*] [, *форматВывода*] [, *имяФайла*] [, *автозагрузка*][**,** *файлШаблона*]

Метод **OutputTo** использует следующие аргументы.

Аргумент	Описание
типОбъекта	Одна из следующих встроенных констант: **acOutputForm** (форма) **acOutputModule** (модуль) **acOutputQuery** (запрос) **acOutputReport** (отчет) **acOutputTable** (таблица)
имяОбъекта	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*. Для того чтобы вывести активный объект, укажите значение аргумента *типОбъекта* и оставьте данный аргумент пустым. Если вызвать программу Visual Basic, содержащую метод **OutputTo**, из библиотечной базы данных, Microsoft Access проводит поиск объекта с этим именем сначала в библиотечной базе данных, а затем в текущей базе данных.
форматВывода	Одна из следующих встроенных констант: **acFormatASP** **acFormatHTML** **acFormatIIS** **acFormatRTF** **acFormatTXT** **acFormatXLS** Если оставить данный аргумент пустым, Microsoft Access выводит приглашение указать формат вывода объекта.
имяФайла	Строковое выражение, представляющее полное имя файла, в который выводится объект. Если оставить данный аргумент пустым, Microsoft Access выводит приглашение указать имя выходного файла.
автозагрузка	Значение **True** (-1) определяет немедленный запуск приложения в среде Windows с загрузкой выходного файла, указанного в аргументе *имяФайла*. Задавайте значение **False** (0), если не требуется запускать приложение. Для форматов Microsoft Internet Information Server (.htx, .idc) и Microsoft ActiveX Server (*.asp) этот аргумент во внимание не принимается. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**False**).

файлШаблона	Строковое выражение, представляющее полное имя файла, используемого в качестве шаблона для вывода в файл в формате HTML, HTX или ASP.

Дополнительные сведения

Для модулей допускается вывод только в текстовом формате MS-DOS, поэтому константе **acOutputModule** в аргументе *типОбъекта* должна соответствовать константа **acFormatTXT** в аргументе *форматВывода*. Форматы Internet Information Server и Microsoft ActiveX Server доступны только для таблиц, запросов и форм, поэтому, если в аргументе *форматВывода* задана константа **acFormatIIS** или **acFormatASP**, в аргументе *типОбъекта* необходимо указать **acOutputTable**, **acOutputQuery** или **acOutputForm**.

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Метод OutputTo, пример

В данном примере таблица «Сотрудники» выводится в файл Employee.rtf, который немедленно открывается в Microsoft Word для Windows:

```
DoCmd.OutputTo acOutputTable, "Сотрудники", acFormatRTF, "Employee.rtf",
True
```

Метод PrintOut

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactPrintC;vamthPrint"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactPrintX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactPrintA"}

Метод **PrintOut** выполняет макрокоманду **Печать (PrintOut)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.PrintOut [*распечатать*] [, *соСтраницы*, *поСтраницу*] [, *разрешение*] [, *числоКопий*] [, *разобратьКопии*]

Метод **PrintOut** использует следующие аргументы.

Аргумент	Описание
распечатать	Одна из следующих встроенных констант: **acPrintAll** (все) **acSelection** (фрагмент) **acPages** (страницы) Если оставить данный аргумент пустым, подразумевается значение по умолчанию **acPrintAll**.
соСтраницы	Числовое выражение, представляющее допустимый номер страницы в активной форме или объекте в режиме таблицы. Данный аргумент является обязательным, если в аргументе *распечатать* задана константа **acPages**.
поСтраницу	Числовое выражение, представляющее допустимый номер страницы в форме или объекте в режиме таблицы. Данный аргумент является обязательным, если в аргументе *распечатать* задана константа **acPages**.
разрешение	Одна из следующих встроенных констант: **acDraft** (черновик) **acHigh** (высокое, значение по умолчанию) **acLow** (низкое) **acMedium** (среднее) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acHigh**).
числоКопий	Числовое выражение. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (1).
разобратьКопии	Значение **True** (-1) определяет печать с раскладкой по копиям, а **False** (0) печать без раскладки. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**True**).

Дополнительные сведения

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если опускаются один или несколько последних аргументов, вводить запятые вслед за последним указанным аргументом не требуется.

Метод PrintOut, пример

В данном примере первые четыре страницы активной формы или объекта в режиме таблицы печатаются в двух экземплярах с раскладкой по копиям:

```
DoCmd.PrintOut acPages, 1, 4, , 2
```

Метод Quit (объект DoCmd)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactQuitC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactQuitX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactQuitA"}

Метод **Quit** объекта **DoCmd** выполняет макрокоманду **Выход (Quit)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Quit [*параметры*]

Метод **Quit** использует следующие аргументы.

Аргумент	Описание
параметры	Одна из следующих встроенных констант: **acQuitPrompt** (подтверждение) **acQuitSaveAll** (сохранить все, значение по умолчанию) **acQuitSaveNone** (не сохранять) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acQuitSaveAll**).

Дополнительные сведения

Метод **Quit** объекта **DoCmd**, выполняющий макрокоманду **Выход (Quit)** в программе Visual Basic, был добавлен для совместимости с Microsoft Access версии 7.0. Рекомендуется использовать в новых программах метод **Quit** объекта **Application.**

Метод Quit (объект DoCmd), пример

В данном примере выводится окно диалога с приглашением подтвердить сохранение любых изменений объектов перед выходом из Microsoft Access:

```
DoCmd.Quit acQuitPrompt
```

Метод Rename

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactRenameC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactRenameX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactRenameA"}

Метод **Rename** выполняет макрокоманду **Переименовать (Rename)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Rename *новоеИмя* [, *типОбъекта*, *староеИмя*]

Метод **Rename** использует следующие аргументы.

Аргумент	Описание
новоеИмя	Строковое выражение, представляющее новое имя объекта. Имя должно удовлетворять соглашениям об именах объектов Microsoft Access.
типОбъекта	Одна из следующих встроенных констант: **acDefault** (значение по умолчанию) **acForm** (форма) **acMacro** (макрос) **acModule** (модуль) **acQuery** (запрос) **acReport** (отчет) **acTable** (таблица)
староеИмя	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*. Если вызвать программу Visual Basic, содержащую метод **Rename**, из библиотечной базы данных, Microsoft Access проводит поиск объекта с указанным именем сначала в библиотечной базе данных, а затем в текущей базе данных.

Дополнительные сведения

Если оставлены пустыми аргументы *типОбъекта* и *староеИмя* (по умолчанию аргументу *типОбъекта* присваивается константа **acDefault**), Microsoft Access переименовывает объект, выделенный в окне базы данных. Для выделения объекта в окне базы данных следует вызвать макрокоманду **ВыделитьОбъект (SelectObject)** или метод **SelectObject** с заданным для аргумента *вОкнеБД* значением «Да» (**True**).

Если оставлены пустыми аргументы *типОбъекта* и *староеИмя*, вводить запятые после аргумента *новоеИмя* не требуется.

Метод Rename, пример

В данном примере задается новой имя для таблицы «Сотрудники»:

```
DoCmd.Rename "Старая таблица Сотрудники", acTable, "Сотрудники"
```

Метод RepaintObject

{ewc HLP95EN.DLL,DYNALINK,"пїSпїSпїSпїSпїS":"acmthactRepaintObjectC"} {ewc HLP95EN.DLL,DYNALINK,"пїSпїSпїSпїSпїS":"acmthactRepaintObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"пїSпїSпїSпїSпїSпїSпїSпїS":"acmthactRepaintObjectA"}

Метод **RepaintObject** выполняет макрокоманду **ОбновитьОбъект (RepaintObject)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.RepaintObject [*типОбъекта*, *имяОбъекта*]

Метод **RepaintObject** использует следующие аргументы.

Аргумент	Описание
типОбъекта	Одна из следующих встроенных констант: **acDefault** (значение по умолчанию) **acForm** (форма) **acMacro** (макрос) **acModule** (модуль) **acQuery** (запрос) **acReport** (отчет) **acTable** (таблица)
имяОбъекта	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*.

Дополнительные сведения

Метод **RepaintObject**, вызванный без аргументов (по умолчанию аргументу *типОбъекта* присваивается константа **acDefault**), обновляет активное окно.

Метод **RepaintObject** объекта **DoCmd**, выполняющий макрокоманду **ОбновитьОбъект (RepaintObject)** в программе Visual Basic, был добавлен в Microsoft Access для Windows 95 для совместимости с предыдущими версиями. Для обновления форм в новых программах рекомендуется использовать метод **Repaint** объекта **Form.**

Метод RepaintObject, пример

В данном примере обновляется таблица «Клиенты»:

```
DoCmd.RepaintObject acTable, "Клиенты"
```

Метод Requery (объект DoCmd)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactRequeryC;damthRequery"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactRequeryX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactRequeryA"}

Метод **Requery** объекта **DoCmd** выполняет макрокоманду **Обновление (Requery)** в программе Visual Basic. Подробное описание макрокоманды и ее аргумента см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Requery [*имяЭлемента*]

Метод **Requery** использует следующий аргумент.

Аргумент	Описание
имяЭлемента	Строковое выражение, представляющее имя элемента управления в активном объекте.

Дополнительные сведения

В аргументе *имяЭлемента* достаточно указать только имя элемента управления; полную ссылку указывать необязательно.

Допускается определение значения этого аргумента с помощью переменной, описанной с типом данных **Control**:

```
Dim ctlLstBox As Control
Set ctlLstBox = Forms!Форма1!Поле3
DoCmd.Requery ctlLstBox.Name
```

Метод **Requery** объекта **DoCmd** отличается от метода **Requery** Visual Basic. Метод **Requery** объекта **DoCmd**, выполняющий макрокоманду **Обновление (Requery)** в программе Visual Basic, был добавлен в Microsoft Access для Windows 95 для совместимости с предыдущими версиями. Для обновления элемента управления, не находящегося в активном объекте необходимо использовать метод **Requery** Visual Basic, а не макрокоманду **Обновление (Requery)** или соответствующие ей метод **Requery** объекта **DoCmd**. Метод **Requery** Visual Basic выполняется быстрее, чем макрокоманда **Обновление (Requery)** или метод **Requery** объекта **DoCmd**. Кроме того, после вызова макрокоманды **Обновление (Requery)** или метод **Requery** объекта **DoCmd** Microsoft Access закрывает запрос и вновь загружает его из базы данных. При вызове метода **Requery** Microsoft Access вновь выполняет запрос без его закрытия и повторной загрузки.

Примечание. Метод **Requery** объектов доступа к данным (DAO) выполняется так же, как и метод **Requery** Microsoft Access.

Метод Requery (объект DoCmd), пример

В данном примере метод **Requery** вызывается для обновления элемента управления «СписокСотрудников»:

```
DoCmd.Requery "СписокСотрудников"
```

Метод Restore

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactRestoreC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactRestoreX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactRestoreA"}

Метод **Restore** выполняет макрокоманду **Восстановить (Restore)** в программе Visual Basic. Подробное описание макрокоманды см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Restore

Данный метод не требует аргументов.

Метод RunMacro

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactRunMacroC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactRunMacroX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactRunMacroA"}

Метод **RunMacro** выполняет макрокоманду **ЗапускМакроса (RunMacro)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.RunMacro *имяМакроса* [, *числоПовторов*] [, *условиеПовтора*]

Метод **RunMacro** использует следующие аргументы.

Аргумент	Описание
имяМакроса	Строковое выражение, представляющее допустимое имя макроса в текущей базе данных. Если вызвать программу Visual Basic, содержащую метод **RunMacro**, из библиотечной базы данных, Microsoft Access проводит поиск макроса с указанным именем в библиотечной базе данных и не проводит поиск макроса в текущей базе данных.
числоПовторов	Числовое выражение, имеющее целочисленное значение, представляющее число повторов выполнения макроса.
условиеПовтора	Условное выражение, которое проверяется при каждом выполнении макроса. Когда это выражение получает значение **False** (0), выполнение макроса прекращается.

Дополнительные сведения

В аргументе *имяМакроса* допускается использование синтаксиса *имяГруппыМакросов.имяМакроса* для запуска конкретного макроса, входящего в группу макросов.

Для того чтобы определить аргумент *условиеПовтора* и оставить аргумент *числоПовторов* пустым, необходимо ввести запятую, представляющую аргумент *числоПовторов*. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Метод RunMacro, пример

В данном примере запускается макрос «Печать продаж»:

```
DoCmd.RunMacro "Печать продаж"
```

Метод RunSQL

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactRunSQLC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthactRunSQLX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acmthactRunSQLA"}

Метод **RunSQL** выполняет макрокоманду **ЗапускЗапросаSQL (RunSQL)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.RunSQL *инструкцияSQL*[**,** *использоватьТранзакцию*]

Метод **RunSQL** использует следующие аргументы.

Аргумент	Описание
инструкцияSQL	Строковое выражение, представляющее допустимую инструкцию SQL для запроса на изменение или управляющего запроса. В этой инструкции используются инструкции INSERT INTO, DELETE, SELECT...INTO, UPDATE, CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX, или DROP INDEX. Для доступа к другой базе данных следует использовать предложение IN.
использовать Транзакцию	Чтобы включить этот запрос в транзакцию следует задать значение **True** (–1). Чтобы отказаться от использования транзакции следует задать значение **False** (0). Если этот аргумент оставлен пустым, используется значение по умолчанию (**True**).

Дополнительные сведения

Максимальная длина строки в аргументе *инструкцияSQL* составляет 32 768 символов (в отличие от значения аргумента макрокоманды «Инструкция SQL» в окне макроса, максимальная длина которого составляет 256 символов).

Если аргумент *использоватьТранзакцию* оставлен пустым, нет необходимости ставить запятую после аргумента *инструкцияSQL*

Метод RunSQL, пример

В данном примере изменяется название должности всех агентов по продажам в таблице «Сотрудники»:

```
DoCmd.RunSQL "UPDATE Сотрудники " & _
   "SET Сотрудники.Title = 'Региональный представитель' " & _
   "WHERE Сотрудники.Title = 'Агент по продажам';"
```

Метод Save

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthactSaveC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acmthactSaveX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acmthactSaveA"}

Метод **Save** выполняет макрокоманду **Сохранить (Save)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.Save [*типОбъекта*, *имяОбъекта*]

Метод **Save** использует следующие аргументы.

Аргумент	Описание
типОбъекта	Одна из следующих встроенных констант: **acDefault** (значение по умолчанию) **acForm** (форма) **acMacro** (макрос) **acModule** (модуль) **acQuery** (запрос) **acReport** (отчет) **acTable** (таблица)
имяОбъекта	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*.

Дополнительные сведения

Если оставлены пустыми аргументы *типОбъекта* и *имяОбъекта* (по умолчанию аргументу *типОбъекта* присваивается константа **acDefault**), Microsoft Access сохраняет активный объект. Если аргумент *типОбъекта* оставлен пустым, но задано значение аргумента *имяОбъекта*, то под указанным именем сохраняется активный объект. При определении типа объекта с помощью аргумента *типОбъекта* необходимо указать в аргументе *имяОбъекта* имя существующего объекта.

Если аргумент *типОбъекта* оставлен пустым, но определен аргумент *имяОбъекта*, необходимо ввести запятую, представляющую аргумент *типОбъекта*.

Примечание. С помощью метода **Save** невозможно сохранить под новым именем:

- форму в режиме формы или в режиме таблицы;
- отчет в режиме предварительного просмотра;
- модуль.

Метод **Save** вне зависимости от его выполнения в текущей базе данных или в библиотечной базе данных всегда сохраняет указанный объект или активный объект в той базе данных, в которой объект был создан.

Метод Save, пример

В данном примере метод **Save** сохраняет активный объект (форму) под именем «Новая форма 'Сотрудники'». Форма должна быть открыта.

```
DoCmd.Save acForm, "Новая форма 'Сотрудники'"
```

Метод SelectObject

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthactSelectObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acmthactSelectObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acmthactSelectObjectA"}

Метод **SelectObject** выполняет макрокоманду **ВыделитьОбъект (SelectObject)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.SelectObject *типОбъекта*, *имяОбъекта* [, *вОкнеБД*]

Метод **SelectObject** использует следующие аргументы.

Аргумент	Описание
типОбъекта	Одна из следующих встроенных констант:: **acForm** (форма) **acMacro** (макрос) **acModule** (модуль) **acQuery** (запрос) **acReport** (отчет) **acTable** (таблица) **Примечание.** Константа **acDefault**, появляющаяся в списке возможных значений для этого аргумента является недопустимой. Необходимо выбрать одну из перечисленных выше констант.
имяОбъекта	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*. Это обязательный аргумент, если только аргумент *вОкнеБД* отличен от **True** (–1). Если аргумент *вОкнеБД* равен **True** и аргумент *имяОбъекта* оставлен пустым, Microsoft Access выделяет вкладку в окне базы данных, соответствующую объекту базы данных, указанному в аргументе *типОбъекта*.
вОкнеБД	Значение **True** (-1) определяет выделение объекта в окне базы данных. Значение **False** (0) следует задать для выделения открытого объекта. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**False**).

Дополнительные сведения

Если значение аргумента *вОкнеБД* установлено в **True** и аргумент *имяОбъекта* опущен, необходимо ввести запятую, представляющую аргумент *имяОбъекта*. Если не указаны последние аргументы, не нужно вводить запятые, их представляющие.

Метод SelectObject, пример

В данном примере выделяется форма «Клиенты» в окне базы данных:

```
DoCmd.SelectObject acForm, "Клиенты", True
```

Метод SendObject

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactSendObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthactSendObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthactSendObjectA"}

Метод **SendObject** выполняет макрокоманду **ОтправитьОбъект (SendObject)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.SendObject [*типОбъекта*] [, *имяОбъекта*] [, *форматВывода*] [, *кому*] [, *копии*] [, *скрытыеКопии*] [, *тема*] [, *сообщение*] [, *изменитьСообщение*][**,** *файлШаблона*]

Метод **SendObject** использует следующие аргументы.

Аргумент	Описание
типОбъекта	Одна из следующих встроенных констант: **acSendForm** (форма) **acSendModule** (модуль) **acSendNoObject** (нет объекта, значение по умолчанию) **acSendQuery** (запрос) **acSendReport** (отчет) **acSendTable** (таблица)
имяОбъекта	Строковое выражение, представляющее допустимое имя объекта, тип которого указан в аргументе *типОбъекта*. Для того чтобы включить в сообщение электронной почты активный объект, укажите тип объекта в аргументе *типОбъекта* и оставьте данный аргумент пустым. Если оба аргумента, *типОбъекта* и *имяОбъекта*, оставлены пустыми (по умолчанию аргументу *типОбъекта* присваивается константа **acSendNoObject**), Microsoft Access отправляет сообщение в приложение электронной почты без объекта базы данных. Если вызвать программу Visual Basic, содержащую метод **SendObject**, из библиотечной базы данных, Microsoft Access проводит поиск объекта с этим именем сначала в библиотечной базе данных, а затем в текущей базе данных.
форматВывода	Одна из следующих встроенных констант: **acFormatHTML** **acFormatRTF** **acFormatTXT** **acFormatXLS** Если оставить данный аргумент пустым, в Microsoft Access выводится приглашение указать выходной формат.
кому	Строковое выражение, содержащее список имен получателей, включаемых в строку «Кому» (To) сообщения электронной почты. Имена получателей сообщения, указанных в данном аргументе, а также в аргументах *копии* и *скрытыеКопии*, следует разделять символом точки с запятой (;) или символом разделителя, заданным на

	вкладке **Числа** в окне **Язык и стандарты** панели управления Windows. Если Microsoft Access не сможет идентифицировать имена получателей, сообщение не будет отослано. Если оставить данный аргумент пустым, Microsoft Access выводит приглашение указать имена получателей.
копии	Строковое выражение, содержащее список имен получателей, включаемых в строку «Копии» (Cc) сообщения электронной почты. Если оставить данный аргумент пустым, строка «Копии» (Cc) сообщения электронной почты окажется пустой.
скрытыеКопии	Строковое выражение, содержащее список имен получателей, включаемых в строку «Скрытые копии» (Bcc) сообщения электронной почты. Если оставить данный аргумент пустым, строка «Скрытые копии» (Bcc) сообщения электронной почты окажется пустой.
тема	Строковое выражение, содержащее текст, помещаемый в строку «Тема» (Subject) сообщения электронной почты. Если оставить данный аргумент пустым, строка «Тема» (Subject) сообщения электронной почты окажется пустой.
сообщение	Строковое выражение, содержащее основной текст сообщения электронной почты, помещаемый после объекта. Если оставить данный аргумент пустым, единственным содержимым основной части сообщения будет передаваемый объект.
изменитьСообщение	Значение **True** (-1) определяет немедленное открытие приложения электронной почты после загрузки сообщения, что позволяет начать редактирование сообщения. Значение **False** (0) определяет отправку сообщения без его редактирования. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**True**).
файлШаблона	Строковое выражение, представляющее полное имя файла, используемого в качестве шаблона для вывода в файл в формате HTML.

Дополнительные сведения

Модули пересылаются только в текстовом формате MS-DOS, поэтому константа **acSendModule** в аргументе *типОбъекта* требует указания константы **acFormatTXT** в аргументе *форматВывода*.

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Метод SendObject, пример

В данном примере таблица «Сотрудники» включается в сообщение электронной почты в формате Microsoft Excel. В сообщении задаются строки «To», «Cc» и «Subject». Сообщение отправляется немедленно без предварительного редактирования.

```
DoCmd.SendObject acSendTable, "Сотрудники", acFormatXLS, _
   "Иван Иванов; Петр Петров", "Сидор Сидоров", , _
   "Текущая электронная таблица 'Сотрудники'", , False
```

Метод SetMenuItem

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactSetMenuItemC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthactSetMenuItemX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthactSetMenuItemA"}

Метод **SetMenuItem** выполняет макрокоманду **ЗадатьКомандуМеню (SetMenuItem)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Примечание. Метод **SetMenuItem** работает только со специальными строками меню и общими строками меню, созданными с помощью макросов создания меню. Метод **SetMenuItem** включен в эту версию Microsoft Access только для совместимости с предыдущими версиями. Он не работает с новыми панелями команд.

Синтаксис

DoCmd.SetMenuItem *индексМеню* [, *индексКоманды*] [, *индексПодкоманды*] [, *флаг*]

Метод **SetMenuItem** использует следующие аргументы.

Аргумент	Описание
индексМеню	Целое значение, отсчитываемое от 0, представляющее допустимый индекс меню в специальной строке меню или в общей строке меню активного окна. Этот индекс определяется в макросе создания меню, который создает специальную или общую строку меню. Если указать индекс меню и оставить пустыми аргументы *индексКоманды* и *индексПодкоманды* (или задать для них значения -1), пользователь имеет возможность включить или отключить (сделать недоступным) само меню. Не допускается, однако, установка или снятие метки рядом с именем меню (для имен меню Microsoft Access игнорирует константы **acMenuCheck** и **acMenuUncheck** в аргументе *флаг*).
индексКоманды	Целое значение, отсчитываемое от 0, представляющее допустимый индекс команды в меню, указанном в аргументе *индексМеню*. Этот индекс определяется в группе макросов, которая создает указанное меню в специальной или общей строке меню активного окна.
индексПодкоманды	Целое значение, отсчитываемое от 0, представляющее допустимый индекс подкоманды в подменю, указанном в аргументе *индексКоманды*. Этот индекс определяется в группе макросов, которая создает указанное подменю в специальной или общей строке меню активного окна.
флаг	Одна из следующих встроенных констант: **acMenuCheck** (помечен) **acMenuGray** (отключен) **acMenuUncheck** (не помечен) **acMenuUngray** (включен, значение по умолчанию) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acMenuUngray**).

Дополнительные сведения

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если пустыми

оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Метод SetMenuItem, пример

В данном примере метод **SetMenuItem** отключает вторую команду в первом меню специальной строки меню активного окна:

```
DoCmd.SetMenuItem 0, 1, , acMenuGray
```

Метод SetWarnings

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthactSetWarningsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactSetWarningsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactSetWarningsA"}

Метод **SetWarnings** выполняет макрокоманду **УстановитьСообщения (SetWarnings)** в программе Visual Basic. Подробное описание макрокоманды и ее аргумента см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.SetWarnings *режим*

Метод **SetWarnings** использует следующие аргументы.

Аргумент	Описание
режим	Значение **True** (-1) задает включение режима вывода системных сообщений; значение **False** (0) отключает вывод сообщений.

Дополнительные сведения

Режим вывода сообщений, отключенный в программе Visual Basic, останется отключенным даже после прерывания программы нажатием клавиш CTRL+BREAK или после прерывания выполнения программы на точке останова Visual Basic. Полезно создать макрос, включающий режим вывода сообщений, и назначить этот макрос на клавиатуру или связать его с командой специального меню. Это позволит в любой момент включить режим вывода сообщений, если он был отключен в программе Visual Basic.

Метод SetWarnings, пример

В данном примере включается вывод системных сообщений:

```
DoCmd.SetWarnings False
```

Метод ShowAllRecords

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthactShowAllRecordsC;damthRequery"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acmthactShowAllRecordsX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acmthactShowAllRecordsA"}

Метод **ShowAllRecords** выполняет макрокоманду **ПоказатьВсеЗаписи (ShowAllRecords)** в программе Visual Basic. Подробное описание макрокоманды см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.ShowAllRecords

Данный метод не требует аргументов.

Метод ShowToolbar

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅS":"acmthactShowToolbarC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acmthactShowToolbarX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acmthactShowToolbarA"}

Метод **ShowToolbar** выполняет макрокоманду **ПанельИнструментов (ShowToolbar)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Примечание. Метод **ShowToolbar** влияет только на панели инструментов, а не на строки меню или контекстные меню.

Синтаксис

DoCmd.ShowToolbar *имяПанели* [, *режим*]

Метод **ShowToolbar** использует следующие аргументы.

Аргумент	Описание
имяПанели	Строковое выражение, представляющее допустимое имя встроенной панели инструментов Microsoft Access или специальной панели инструментов, созданной пользователем.. Если вызвать программу Visual Basic, содержащую метод **ShowToolbar**, из библиотечной базы данных, Microsoft Access проводит поиск панели инструментов с указанным именем сначала в библиотечной базе данных, а затем в текущей базе данных.
режим	Одна из следующих встроенных констант: **acToolbarNo** (нет) **acToolbarWhereApprop** (в обычном режиме) **acToolbarYes** (да, значение по умолчанию) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acToolbarYes**).

Дополнительные сведения

Если аргумент *режим* оставлен пустым, вводить запятые после аргумента *имяПанели* не требуется.

Метод ShowToolbar, пример

Следующая конструкция задает вывод специальной панели инструментов «НоваяПанель» во всех активных окнах Microsoft Access:

```
DoCmd.ShowToolbar "НоваяПанель", acToolbarYes
```

Метод TransferDatabase

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acmthactTransferDatabaseC;damthCreateTableDef;damthRefreshLink;daobjTableDef;daproConnect"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactTransferDatabaseX":1} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactTransferDatabaseA"}

Метод **TransferDatabase** выполняет макрокоманду **ПреобразоватьБазуДанных (TransferDatabase)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.TransferDatabase [*преобразование*], *типБД*, *имяБД* [, *типОбъекта*], *источник*, *адресат* [, *структура*] [, *подключение*]

Метод **TransferDatabase** использует следующие аргументы.

Аргумент	Описание
преобразование	Одна из следующих встроенных констант: **acExport** (экспорт) **acImport** (импорт, значение по умолчанию) **acLink** (связь) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acImport**).
типБД	Строковое выражение, представляющее тип базы данных, для которой выполняются операции импорта, экспорта или связывания данных. Список поддерживаемых типов баз данных доступен для просмотра в окне макроса в ячейке аргумента «Тип базы данных» макрокоманды **ПреобразоватьБазуДанных (TransferDatabase)**.
имяБД	Строковое выражение, представляющее полное имя (включая путь) базы данных, для которой выполняются операции импорта, экспорта или связывания данных.
типОбъекта	Одна из следующих встроенных констант: **acForm** (форма) **acMacro** (макрос) **acModule** (модуль) **acQuery** (запрос) **acReport** (отчет) **acTable** (таблица, значение по умолчанию) Данный аргумент определяет тип импортируемого, экспортируемого или связываемого объекта. Константы, отличные от **acTable**, используются только при импорте или экспорте данных между двумя базами данных Microsoft Access. При экспорте результирующего набора записей запроса на выборку Microsoft Access в базу данных другого типа в данном аргументе необходимо указать константу **acTable**. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acTable**). **Примечание.** Константа **acDefault**, появляющаяся в списке возможных значений для этого аргумента является недопустимой. Необходимо выбрать одну из

	вышеперечисленных констант.
источник	Строковое выражение, представляющее имя объекта, являющегося источником данных при импорте, экспорте или связывании.
адресат	Строковое выражение, представляющее имя импортируемого, экспортируемого или связываемого объекта в базе данных-получателе.
структура	Значение **True** (-1) указывает импорт или экспорт только структуры таблицы из базы данных. Значение **False** (0) указывает импорт или экспорт структуры таблицы вместе с содержащимися в ней данными. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**False**).
подключение	Значение **True** указывает сохранение имени и пароля, используемых для подключения к базе данных ODBC, в строке подключения для связанной таблицы в базе данных. Если эти сведения сохраняются, не возникает необходимость подключаться заново при каждом открытии таблицы. При заданном значении **False** имя и пароль, используемые при подключении не записываются. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**False**). Данный аргумент является доступным только в программах Visual Basic.

Дополнительные сведения

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Администратор базы данных ODBC имеет право отключить настройку, заданную в аргументе *подключение*, и потребовать от всех пользователей указывать имя и пароль при каждом подключении к базе данных ODBC.

Примечание. Хотя в программах Visual Basic допускается использование метода **TransferDatabase** для связывания таблиц, рекомендуется вместо этого использовать объекты доступа к данным (DAO). Для создания связи с помощью DAO следует воспользоваться свойством **Connect** объекта **TableDef.**

Метод TransferDatabase, пример

В данном примере отчет «Продажи за апрель» импортируется из базы данных Microsoft Access NWSales.mdb в отчет «Общие продажи за апрель» в текущей базе данных:

```
DoCmd.TransferDatabase acImport, "Microsoft Access", _
   "C:\DBS\NWSales.mdb", acReport, "Продажи за апрель", _
   "Общие продажи за апрель"
```

В следующем примере таблица "Authors" из базы данных ODBC связывается с текущей базой данных:

```
DoCmd.TransferDatabase acLink, "ODBC Database", _
   "ODBC;DSN=DataSource1;UID=User2;PWD=www;LANGUAGE=us_english;" _
   & "DATABASE=pubs", acTable, "Authors", "dboAuthors"
```

Метод TransferSpreadsheet

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acmthactTransferSpreadsheetC;damthCreateTableDef;damthRefreshLink;daobjTableDef;daproConnect"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthactTransferSpreadsheetX":1} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthactTransferSpreadsheetA"}

Метод **TransferSpreadsheet** выполняет макрокоманду **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.TransferSpreadsheet [*преобразование*] [, *типЭТ*], *имяТаблицы*, *имяФайла* [, *именаПолей*] [, *диапазон*]

Метод **TransferSpreadsheet** использует следующие аргументы.

Аргумент	Описание
преобразование	Одна из следующих встроенных констант: **acExport** (экспорт) **acImport** (импорт, значение по умолчанию) **acLink** (связь) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acImport**).
типЭТ	Одна из следующих встроенных констант или ее числовое значение: 0 **acSpreadsheetTypeExcel3** (значение по умолчанию) 6 **acSpreadsheetTypeExcel4** 5 **acSpreadsheetTypeExcel5** 5 **acSpreadsheetTypeExcel7** 8 **acSpreadsheetTypeExcel97** 2 **acSpreadsheetTypeLotusWK1** 3 **acSpreadsheetTypeLotusWK3** 7 **acSpreadsheetTypeLotusWK4** 4 **acSpreadsheetTypeLotusWJ2** — только японская версия **Примечание.** Допускается связывание файлов электронных таблиц Lotus 1-2-3, но такие данные в Microsoft Access будут доступны только для чтения. Возможен импорт и связывание (только для чтения) файлов Lotus .WK4, но экспорт данных Microsoft Access в этих форматах невозможен. Microsoft Access больше не поддерживает импорт, экспорт или связывание данных для электронных таблиц Lotus .WKS и Microsoft Excel версии 2.0 с помощью этого метода. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acSpreadsheetTypeExcel3**).
имяТаблицы	Строковое выражение, представляющее имя таблицы Microsoft Access, в которую импортируются или связываются данные или из которой данные экспортируются. Допускается также указание имени запроса на выборку Microsoft Access, результаты которого экспортируются в электронную таблицу.
имяФайла	Строковое выражение, представляющее имя и путь файла электронной таблицы.
именаПолей	Значение **True** (-1) задает использование значений из первой строки электронной таблицы в качестве имен полей при импорте или связывании. Значение **False** (0) указывает, что в

	первой строке электронной таблицы содержатся обычные данные. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**False**). При экспорте таблицы Microsoft Access или запроса на выборку в электронную таблицу имена полей помещаются в первую строку электронной таблицы независимо от значения этого аргумента.
диапазон	Строковое выражение, представляющее допустимые адрес или имя диапазона ячеек электронной таблицы. Данный аргумент используется только при импорте. Для импорта всей электронной таблицы следует оставить данный аргумент пустым. При экспорте в электронную таблицу следует оставить этот аргумент пустым. Если в нем будет указан диапазон, операция экспорта завершится неудачей.

Дополнительные сведения

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Примечание. Хотя в программах Visual Basic допускается использование метода **TransferSpreadsheet** для связывания данных электронных таблиц, рекомендуется вместо этого использовать объекты доступа к данным (DAO). Для создания связи с помощью DAO следует воспользоваться свойством **Connect** объекта **TableDef.**

Метод TransferSpreadsheet, пример

В данном примере данные из указанного диапазона электронной таблицы Lotus с именем Newemps.wk3 импортируются в таблицу «Сотрудники» Microsoft Access. Содержимое первой строки электронной таблицы используется как имена полей.

```
DoCmd.TransferSpreadsheet acImport, 3, "Сотрудники","C:\Lotus\Newemps.wk3",
True, "A1:G12"
```

Метод TransferText

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS.
nїSnїSnїSnїSnїS":"acmthactTransferTextC;damthCreateTableDef;damthRefreshLink;daobjTableDef;daproConnect"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acmthactTransferTextX":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acmthactTransferTextA"}

Метод **TransferText** выполняет макрокоманду **ПреобразоватьТекст (TransferText)** в программе Visual Basic. Подробное описание макрокоманды и ее аргументов см. в разделе справки о макрокоманде.

Синтаксис

DoCmd.TransferText [*преобразование*] [, *спецификация*], *имяТаблицы*, *имяФайла* [, *именаПолей*][**,** *имяТаблицыHTML*]

Метод **TransferText** использует следующие аргументы.

Аргумент	Описание
преобразование	Одна из следующих встроенных констант: **acExportDelim** (Экспорт (разделители)) **acExportFixed** (Экспорт (фиксированный)) **acExportHTML** (Экспорт (HTML)) **acExportMerge** (Экспорт слияний Word для Windows) **acImportDelim** (Импорт (разделители), значение по умолчанию) **acImportFixed** (Импорт (фиксированный)) **acImportHTML** (Импорт (HTML)) **acLinkDelim** (Связь (разделители)) **acLinkFixed** (Связь (фиксированный)) **acLinkHTML** (Связь (HTML)) Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**acImportDelim**). **Примечание.** Допускается связывание данных из текстового файла или файла HTML, но в Microsoft Access эти данные будут доступны только для чтения.
спецификация	Строковое выражение, представляющее название спецификации импорта/экспорта, созданной и сохраненной в текущей базе данных. Данный аргумент является обязательным только для текстовых файлов с фиксированной длиной записей. Для текстовых файлов с разделителями и файлов данных составных документов Microsoft Word для Windows можно оставить данный аргумент пустым.
имяТаблицы	Строковое выражение, представляющее имя таблицы Microsoft Access, в которую импортируются или связываются данные или из которой данные экспортируются. Допускается также указание имени запроса на выборку Microsoft Access, результаты которого экспортируются в текстовый файл.
имяФайла	Строковое выражение, представляющее имя и путь текстового файла.
именаПолей	Значение **True** (-1) задает использование значений из первой строки текстового файла в качестве имен полей при импорте, экспорте или связывании. Значение **False** (0) указывает, что в первой строке текстового файла

	содержатся обычные данные. Если оставить данный аргумент пустым, подразумевается значение по умолчанию (**False**). Для файлов данных составных документов Word для Windows этот аргумент игнорируется. Такие файлы всегда содержат в первой строке имена полей.
имяТаблицыHTML	Строковое выражение, представляющее имя импортируемой или связываемой таблицы или списка в файле HTML. Этот аргумент принимается во внимание, только если аргумент *преобразование* равен **acImportHTML** или **acLinkHTML**. Если этот аргумент оставлен пустым, импортируется или связывается первая таблица или список из файла HTML. Имя таблицы или списка из файла HTML задается в команде <CAPTION>, если такая существует. Иначе имя определяется в команде <TITLE>. Если имена нескольких таблиц или списков совпадают, Microsoft Access добавляет в конец каждого имени число; например, «Сотрудники1» и «Сотрудники2».

Дополнительные сведения

Необязательный аргумент посреди списка аргументов разрешается пропустить, однако, при этом необходимо ввести запятую, представляющую пропущенный аргумент. Если пустыми оставлены последние аргументы, вводить запятые вслед за последним указанным аргументом не требуется.

Примечание. Хотя в программах Visual Basic допускается использование метода **TransferText** для связывания текстовых данных, рекомендуется вместо этого использовать объекты доступа к данным (DAO). Для создания связи с помощью DAO следует воспользоваться свойством **Connect** объекта **TableDef.**

Метод TransferText, пример

В данном примере данные из таблицы Microsoft Access с именем «Внешний отчет» экспортируются в текстовый файл с разделителями April.doc с использованием спецификации «Стандартный вывод»:

```
DoCmd.TransferText acExportDelim, "Стандартный вывод", _
   "Внешний отчет", "C:\Txtfiles\April.doc"
```

Метод Echo (Объект Application)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthEchoC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthEchoX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthEchoA"}

Метод **Echo** задает режим обновления экрана Microsoft Access.

Синтаксис

Application.Echo *обновление*[, *сообщение*]

Метод **Echo** использует следующие аргументы.

Аргумент	Описание
обновление	**True** (по умолчанию) указывает на то, что текущий экран обновлен; **False** - текущий экран не обновлен.
сообщение	Строковое выражение, определяющее текст, выводимый в строке состояния при включении и выключении обновления .

Дополнительные сведения

Программы Visual Basic, которые вносят ряд изменений в объекты, выводящиеся на экран, выполнятся быстрее, если отключить обновление экрана на промежуточных стадиях. Кроме того, полезно отключить обновление экрана, чтобы не выводить изменения, которые не несут полезной для пользователя информации или должны быть скрыты от него.

Метод **Echo** не влияет на вывод модальных окон диалога, таких как сообщения об ошибках, или всплывающих форм, таких как окна свойств.

Если обновление экрана было на какое-то время отключено, то следует обязательно включить его обратно, иначе обновление экрана не будет производиться даже после нажатия пользователем клавиш CTRL+BREAK или после прерывания выполнения программы Visual Basic на точке останова. Имеется возможность создать макрос, включающий обновление экрана, и связать этот макрос с сочетанием клавиш или со специальной командой меню. После этого можно будет с помощью этого сочетания клавиш или команды меню включать режим обновления экрана, если обновление экрана было отключено в программе Visual Basic.

Если обновление экрана было отключено, то при выполнении программы в пошаговом режиме результаты выполнения каждой инструкции не будут отображаться на экране до тех пор, пока обновление не будет включено обратно. Однако программа при этом будет выполняться.

Примечание. Не следует путать метод **Echo** с методом **Repaint.** Метод **Echo** включает или отключает обновление экрана. Метод **Repaint** вызывает принудительное обновление экрана.

Метод Echo (объект Application), пример

В данном примере метод **Echo** вызывается для отключения обновления экрана при выполнении некоторых операций. В то время как в процедуре открывается и свертывается окно формы, пользователь видит только значок песочных часов, показывающий, что программы выполняется. После завершения задачи значок песочных часов снова заменяется на обычный указатель и восстанавливается режим обновления экрана.

```
Sub EchoOff()
   Application.Echo False
   DoCmd.Hourglass True
   DoCmd.OpenForm "Сотрудники", acNormal
   DoCmd.Minimize
   Application.Echo True
   DoCmd.Hourglass False
End Sub
```

Методы GetOption, SetOption

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthGetSetOptionC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthGetSetOptionX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthGetSetOptionA"}

- Метод **GetOption** возвращает текущее значение параметра настройки из диалогового окна **Параметры**, открывающегося командой **Параметры** из меню **Сервис**.
- Метод **SetOption** задает текущее значение параметра настройки в диалоговом окне **Параметры**.

Синтаксис

объект.**GetOption(***имяПараметра***)**
объект.**SetOption** *имяПараметра*, *значение*

Методы **GetOption** и **SetOption** используют следующие аргументы.

Аргумент	Описание
объект	Необязательный параметр - объект **Application**.
имяПараметра	Имя параметра настройки. Список значений аргумента *имяПараметра* см. в разделе Установка параметров из Visual Basic.
значение	Значение типа **Variant,** определяющее значение параметра настройки. Конкретный вид аргумента *значение* определяется выбранным параметром.

Дополнительные сведения

Методы **GetOption** и **SetOption** являются средством изменения настроек среды приложения из программы Visual Basic. Эти методы позволят задавать или считывать значения любых параметров настройки, доступных в диалоговом окне **Параметры** за исключением параметров, выводящихся на вкладке **Модуль**.

Конкретные значения параметров определятся типом выбранного параметра. Существуют три основных типа параметров.

- Параметры, имеющие логические значения («Да» или «Нет»), которые определятся установленным или снятым флажком.
- Параметры настройки, значение которых определяется вводящейся строкой или числовым значением.
- Параметры настройки, значение которых выбирается из ограниченного списка встроенных значений в раскрывающемся списке или в группе параметров.

Для параметров, определяющихся с помощью снятого или установленного флажка, метод **GetOption** возвращает значение **True**, если параметр имеет значение «Да» (флажок установлен), или значение **False**, если параметр имеет значение «Нет» (флажок снят). Для указания значения параметра данного типа с помощью метода **SetOption** следует в аргументе *значение* задать значения **True** (установить флажок) или **False** (снять флажок), как показано в следующем примере:

```
Application.SetOption "Строка состояния", True
```

Для параметров, требующих от пользователя ввода строкового или числового значения, метод **GetOption** возвращает значение в том виде, в каком это значение вводится в поле в диалоговом окне. В следующем примере возвращается строковое значение, содержащее ширину левого поля при печати:

```
Dim varSetting As Variant
varSetting = Application.GetOption("Левое поле")
```

Для указания значения параметра данного типа с помощью метода **SetOption** следует в аргументе *значение* задать значение в том виде, в котором оно должно быть введено в поле в диалоговом окне. Следующая инструкция указывает, что стандартным шаблоном форм является «ШаблонЗаказы».

```
Application.SetOption "Шаблон формы", "ШаблонЗаказы"
```

Для параметров, значения которых выбираются в раскрывающихся списках, метод **GetOption** возвращает число, соответствующее положению значения в списке. Нумерация значений начинается с нуля, поэтому для первого значения **GetOption** возвращает 0, для второго значения 1 и т.д. Например, если на вкладке **Таблицы/запросы** для параметра **Тип поля по умолчанию** в списке выбран шестой элемент «Счетчик», то метод **GetOption** возвращает значение 5.

Для указания значения параметра данного типа с помощью метода **SetOption** следует в аргументе *значение* задать значение, определяющее положение элемента в списке. Следующая инструкция задает для параметра **Тип поля по умолчанию** значение «Счетчик».

```
Application.SetOption "Тип поля по умолчанию", 5
```

Значение других параметров, допускающих выбор из ограниченного числа значений, задаются путем выбора из группы параметров в диалоговом окне **Параметры**. В инструкциях Visual Basic значения этих параметров также определятся с помощью порядкового номера конкретного значения в группе. Первому параметру в группе присваивается номер 0, второму 1 и т.д. Например если на вкладке **Формы/отчеты** в группе **Выделение объектов** выбран параметр «Пересечение», метод **GetOption** возвращает значение 0.

```
Debug.Print Application.GetOption("Выделение объектов")
```

Для того чтобы определить параметр, требующий выбора из группы, следует указать порядковый номер элемента в группе. Следующая инструкция задает для параметра **Выделение объектов** значение «Охват».

```
Application.SetOption "Выделение объектов", 1
```

Примечания

- При работе с параметрами в диалоговом окне **Параметры** с помощью методов **GetOption** или **SetOption** нет необходимости указывать конкретную вкладку, на которой находится параметр.
- Не допускается использование методов **GetOption** или **SetOption** для считывания или определения значений параметров, находящихся в диалоговом окне **Параметры** на вкладке **Модуль**.
- Если значение, возвращаемое методом **GetOption**, присваивается переменной, эта переменная должна быть описана с типом **Variant**.
- Для того, чтобы база данных, созданная с помощью версии Microsoft Access для определенного языка, работала с версией для другого языка, необходимо описывать аргументы методов **GetOption** и **SetOption** на английском языке.

При выходе из Microsoft Access пользователь имеет возможность восстановить первоначальные значения параметров настройки, вызывая метод **SetOption** для всех измененных параметров. Для сохранения исходных значений параметров настройки следует создать общие переменные. Инструкции восстановления значений параметров настройки следует поместить в процедуру обработки события **Закрытие (Close)** формы, которая будет закрываться последней, или в специальную процедуру, вызываемую при выходе из приложения.

Методы GetOption, SetOption, пример

В данном примере метод **GetOption** возвращает значение параметра **Тип поля по умолчанию**, выводящегося на вкладке **Таблицы/запросы** в диалоговом окне **Параметры**, и сохраняет это значение в переменной. Если для параметра **Тип поля по умолчанию** не задано значение «Текстовый», то Microsoft Access выводит диалоговое окно с приглашением пользователю ввести значение «Текстовый». Если пользователь нажимает кнопку **Да**, метод **SetOption** вызывается для изменения стандартного типа поля.

```
Sub ChangeFieldSize()
   ' Константа, представляющая значение "Текстовый"
   ' параметра "Тип поля по умолчанию".
   Const conText = 0
   Dim varSize As Variant, intResponse As Integer
   Dim strMsg As String

   ' Определяет текущие настройки.
   varSize = Application.GetOption("Тип поля по умолчанию")
   strMsg = "Определить по умолчанию тип 'Текстовый'?"
   If varSize <> conText Then
   ' Выводит приглашение изменить стандартный тип поля.
      If MsgBox(strMsg, vbYesNo) = vbYes Then
         ' Заменяет значение на строку "Текстовый".
         Application.SetOption "Тип поля по умолчанию", conText
      End If
   End If
End Sub
```

Метод GoToPage (объект Form)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthGoToPageC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthGoToPageX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthGoToPageA"}

Метод **GoToPage** переводит фокус на первый элемент управления указанной страницы активной формы.

Синтаксис

имяФормы.**GoToPage** *страница*[, *вправо*, *вниз*]

Метод **GoToPage** использует следующие аргументы.

Аргумент	Описание
имяФормы	Объект **Form**, представляющий активную форму.
страница	Числовое выражение для допустимого номера страницы активной формы.
вправо	Числовое выражение для смещения по горизонтали (в единицах твип) от левой границы окна до видимой части страницы.
вниз	Числовое выражение для смещения по вертикали (в единицах твип) от верхней границы окна до видимой части страницы.

Дополнительные сведения

При перемещении на указанную страницу с помощью данного метода фокус будет переведен на первый элемент управления на этой странице в порядке заданной для формы последовательности перехода. Для перемещения фокуса на определенный элемент управления в форме следует вместо этого метода применить метод **SetFocus.**

Данный метод применяется к формам, разбитым на страницы для группировки родственных данных. Например, можно создать форму «Сотрудники», в которой на первой странице содержатся сведения о самом сотруднике, на второй странице сведения о его положении в компании, а на третьей странице сведения об объемах продаж данного сотрудника. Метод **GoToPage** позволит перейти сразу на нужную страницу.

Аргументы *вправо* и *вниз* применяются для форм, размер страницы которых превышает размер окна приложения Microsoft Access. В подобном случае аргумент *страница* задает страницу формы, на которую необходимо перейти, после чего аргументы *вправо* и *вниз* позволят вывести на экран нужную часть страницы. Microsoft Access выводит на экран часть страницы, верхний левый угол которой находится на расстоянии *вправо* и *вниз* от верхнего левого угла окна формы.

Метод GoToPage (объект Application), пример

В следующей инструкции метод **GoToPage** переводит фокус на вторую страницу формы «Клиенты» в позицию, задаваемую аргументами *вправо* и *вниз.*

```
Forms!Клиенты.GoToPage 2, 1440, 600
```

Свойство ItemData

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthItemDataC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthItemDataX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acmthItemDataA"}

Свойство **ItemData** возвращает значение, содержащееся в присоединенном столбце указанной строки списка или поля со списком.

Синтаксис

список.**ItemData(***номерСтроки***)**

Свойство **ItemData** использует следующие аргументы.

Аргумент	**Описание**
список	Объект **Control**, представляющий список или поле со списком.
номерСтроки	Строка в списке или поле со списком, содержащая возвращаемые данные. Строки нумеруются начиная с нуля. Например, чтобы возвратить элемент в шестой строке поля со списком, следует задать аргумент *номерСтроки* равный 5 .

Дополнительные сведения

Свойство **ItemData** позволяет просмотреть набор значений в списке или поле со списком. Предположим, например, что требуется просмотреть все значения списка в поисках конкретного значения. Сначала с помощью свойства **ListCount** следует определить число строк в списке, а затем с помощью свойства **ItemData** возвратить значение из связанного столбца в каждой строке.

Свойство **ItemData** позволяет также возвращать значения из выделенных строк списка. Для определения выделенных строк списка следует проверить семейство **ItemsSelected,** после чего вызвать свойство **ItemData** для возвращения содержимого выделенных строк. Для того чтобы сделать возможным одновременное выделение нескольких строк списка, следует указать для свойства списка **Несвязное выделение** (**MultiSelect**) одно из значений «Со связным выбором» или «С несвязным выбором».

Совет. Свойство **Column** позволяет возвращать данные из указанного столбца списка, даже если этот столбец не является связанным.

Свойство ItemData, пример

В данном примере выводится значение из связанного столбца для каждой выделенной строки списка «СписокСотрудников» в форме «Сотрудники». Свойство списка **Несвязное выделение (MultiSelect)** должно иметь значение «Простой» или «Со связным выбором».

```
Sub RowsSelected()
   Dim ctlList As Control, varItem As Variant

   ' Возвращает объектную переменную типа Control, указывающую на список.
   Set ctlList = Forms!Сотрудники!СписокСотрудников
   ' Перебирает выделенные элементы списка.
   For Each varItem in ctlList.ItemsSelected
      ' Выводит значение, содержащееся в связанном столбце.
      Debug.Print ctlList.ItemData(varItem)
   Next varItem
End Sub
```

Метод Quit (объект Application)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthQuitC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acmthQuitX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acmthQuitA"}

Метод **Quit** используется для выхода из Microsoft Access. Аргумент данного метода позволяет выбрать один из нескольких способов сохранения объектов базы данных перед выходом из приложения.

Синтаксис

Application.Quit [*параметр*]

Метод **Quit** использует следующий аргумент.

Аргумент	Описание
параметр	Встроенная константа, которая определяет, что случится с не сохраненными объектами базы данных при выходе из Microsoft Access:

Константа	Описание
acSaveYes	(Значение по умолчанию). Сохраняет все объекты базы данных без вывода диалогового окна.
acPrompt	Выводит диалоговое окно с приглашением сохранить объекты базы данных, которые были изменены, но не были сохранены.
acExit	Определяет выход из Microsoft Access без сохранения каких-либо объектов.

Дополнительные сведения

Выполнение метода **Quit** аналогично выбору команды **Выход** в меню **Файл**. Данный метод используют при создании специальной команды меню или кнопки, при нажатии которой происходит выход из Microsoft Access. Например, в форму можно поместить кнопку **Выход** и определить для события **Нажатие кнопки (Click)** этой кнопки процедуру обработки события, в которой вызывается метод **Quit** с аргументом *параметр*, имеющим значение **acSaveYes**.

Метод Quit (объект Application), пример

В данном примере приведена процедура обработки события **Нажатие кнопки (Click)** для кнопки **Выход**. После нажатия кнопки **Выход** на экране появляется диалоговое окно с приглашением подтвердить сохранение данных, после чего процедура осуществляет выход из Microsoft Access.

```
Private Sub AppExit_Click()
   Application.Quit acPrompt
End Sub
```

Метод Refresh

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthRefreshC;damthRefresh"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthRefreshX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthRefreshA"}

Метод **Refresh** немедленно обновляет записи в базовом источнике данных указанной формы в режиме формы или в режиме таблицы в соответствии с изменениями данных, внесенных текущим и другими пользователями в сети.

Синтаксис

*имяФормы***.Refresh**

Метод **Refresh** использует следующий аргумент.

Аргумент	Описание
имяФормы	Объект **Form**, определяющий форму для обновления.

Дополнительные сведения

Вызов данного метода эквивалентен выбору команды **Обновить** в меню **Записи**.

Microsoft Access автоматически обновляет записи через промежутки времени, указанные в параметре **Период обновления (с)** на вкладке **Другие** в диалоговом окне **Параметры**. Обновление источников данных ODBC определяется параметром **Период обновления ODBC (с)**, значение которого задается на той же вкладке. Вызов метода **Refresh** позволяет немедленно отобразить изменения, которые были внесены в базовый набор записей формы, выведенной в режиме формы или таблицы, со времени последнего автоматического обновления записей.

Данный метод не выполняет отбор записей заново, поэтому отображаются только изменения, внесенные в текущий набор записей. При этом не выводятся добавленные и не удалятся удаленные записи. В выводящемся на экране наборе записей не будут отображены записи из текущей базы данных, добавленные или удаленные после последнего обновления. Не будут также удалены с экрана записи, которые больше не удовлетворят условиям отбора в запросе или фильтре. Для нового отбора записей следует выполнить повторный запрос с помощью метода **Requery.** Метод **Requery** позволяет полностью отобразить результаты в источнике данных.

Примечание

- Операция обновления экрана обычно выполняется быстрее, чем обновление источников данных, в особенности для запросов, занимающих много времени.
- Не следует путать метод **Refresh** с методом **Repaint,** который обновляет изображение объектов на экране.

Метод Refresh, пример

В данном примере метод **Refresh** используется для обновления записей, выводящихся в форме «Клиенты», при каждом получении фокуса формой.

```
Private Sub Form_Activate()
   Me.Refresh
End Sub
```

Метод Repaint

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthRepaintC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthRepaintX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthRepaintA"}

Метод **Repaint** завершает все отложенные операции обновления экрана для заданной формы. При этом выполнятся также отложенные операции пересчета значений элементов управления формы.

Синтаксис

имяФормы.**Repaint**

Метод **Repaint** использует следующий аргумент.

Аргумент	Описание
имяФормы	Объект **Form**, определяющий форму для обновления.

Дополнительные сведения

В Microsoft Access обновления экрана иногда откладывается до тех пор, пока не будет завершены другие задачи. Метод **Repaint** позволяет провести немедленное принудительное обновление элементов управления указанной формы. Данный метод применяют в следующих случаях:

- После изменения значений ряда полей. Если не указано принудительное обновление, то изменения могут не сразу отображаться в окне Microsoft Access, особенно в тех случаях, когда другие поля (например вычисляемые элементы управления) зависят от значений измененных полей.
- При необходимости обеспечить вывод данных во всех полях формы. Например, в полях, содержащих объекты OLE, данные часто не выводятся на экран сразу же после открытия формы.

Данный метод не вызывает выполнения повторного запроса к базе данных и не отображает новые, измененные или удаленные записи из базового источника записей формы. Для выполнения повторного запроса к источнику данных формы или одного из ее элементов управления следует воспользоваться методом **Requery.**

Примечания

- Не следует путать метод **Repaint** с методом **Refresh** или с командой **Обновить** из меню **Записи**. Метод **Refresh** и команда **Обновить** демонстрируют изменения, внесенные вами или другими пользователями в записи в базовой таблице, принадлежащие набору записей, выведенному на экран. Метод **Repaint** просто выполняет операции обновления экрана, отложенные до выполнения других задач Microsoft Access.
- Метод **Repaint** отличается также от метода **Echo.** Вызов метода **Repaint** приводит к немедленному обновлению экрана, тогда как метод **Echo** включает и отключает режим обновления экрана.

Метод Repaint, пример

В данном примере метод **Repaint** вызывается для обновления экранного изображения формы при получении фокуса.

```
Private Sub Form_Activate()
   Me.Repaint
End Sub
```

Метод Requery (объект Control или Form)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthRequeryC;damthRequery":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthRequeryX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthRequeryA"}

Метод **Requery** обновляет данные, выводящиеся в указанной форме или в элементе управления в активной форме с помощью выполнения повторного запроса к источнику данных формы или элемента управления.

Синтаксис

[*имяОбъекта*.]**RequeryRequery**

Метод **Requery** использует следующий аргумент.

Аргумент	Описание
имяОбъекта	Объект **Form** или **Control**, определяющий форму или элемент управления, который необходимо изменить. При выполнении повторного запроса для источника данных активного объекта этот аргумент можно опустить.

Дополнительные сведения

Вызов данного метода обеспечивает вывод в форме или элементе управления самых свежих данных.

Метод **Requery** выполняет одно из следующих действий.

- Повторно выполняет базовый запрос формы или элемента управления.
- Выводит все новые или измененные записи и убирает записи, удаленные из базовой таблицы формы или элемента управления.
- Обновляет выводящиеся записи в соответствии с изменением свойства **Фильтр** (**Filter**) формы.

На основе таблицы или запроса могут быть созданы следующие элементы управления:

- список и поле со списком;
- элемент управления подчиненная форма;
- объекты OLE, такие как диаграммы;
- элементы управления, для которых свойство **Данные** (**ControlSource**) содержит статистические функции или статистические функции SQL.

Если в аргументе *имяОбъекта* указан элемент управления любого другого типа, выполняется повторный запрос для формы

Если элемент управления, указанный в аргументе *имяОбъекта,* не связан с полем таблицы или запроса, то метод **Requery** вызывает немедленный пересчет значения элемента управления.

Вызов метода **Requery** с опущенным аргументом *имяОбъекта* приводит к выполнению повторного запроса для формы или элемента управления, имеющего фокус. Если у элемента управления, на котором находится фокус, имеется базовый источник записей, то выполняется обновление данных; в противном случае для элемента управления выполняется обновление экрана.

Если фокус имеет подчиненная форма, то выполняется повторный запрос только к источнику данных подчиненной формы.

Примечания

- Метод **Requery** отображает результаты добавления или удаления записей, выполненного

после последнего запроса к источнику записей. Метод **Refresh** отображает результаты изменения уже выведенного набора записей и не демонстрирует результаты последнего добавления или удаления записей. Метод **Repaint** обновляет экранное представление формы или элементов управления.

- Метод **Requery** не передает управление операционной системе (что позволяет Microsoft Windows продолжить обработку сообщений). Для того чтобы передать управление операционной системе, следует вызвать инструкцию **DoEvents.**
- Метод **Requery** выполняется быстрее, чем макрокоманда **Обновление (Requery).** При вызове макрокоманды **Обновление (Requery)** Microsoft Access закрывает запрос и повторно загружает его из базы данных, в то время как при вызове метода **Requery** Microsoft Access повторно выполняет запрос без его закрытия и перезагрузки.

Метод Requery (объект Control или Form), пример

Следующая программа вызывает метод **Requery** для обновления данных в списке **СписокСотрудников** в форме «Сотрудники».

```
Sub RequeryList()
   Dim ctlList As Control

   ' Возвращает объект Control, указывающий на список.
   Set ctlList = Forms!Сотрудники!СписокСотрудников
   ' Обновляет список.
   ctlList.Requery
End Sub
```

Метод SetFocus

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthSetFocusC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthSetFocusX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthSetFocusA"}

Метод **SetFocus** переводит фокус на указанную форму или на элемент управления активной формы, а также на поле активного объекта в режиме таблицы.

Синтаксис

имяОбъекта.**SetFocus**

Метод **SetFocus** использует следующий аргумент.

Аргумент	Описание
имяОбъекта	Объект **Form,** представляющий форму, или объект **Control,** представляющий элемент управления в активной форме или в объекте в режиме таблицы.

Дополнительные сведения

Метод **SetFocus** применяют при необходимости перевести фокус на определенное поле или элемент управления для того, чтобы все вводящиеся пользователем данные были адресованы этому объекту.

Для считывания значений некоторых свойств элемента управления необходимо, чтобы этот элемент управления имел фокус. Например, для того, чтобы считать значение свойства **Text** поля, необходимо перевести фокус на это поле.

Существуют также свойства, значения которых можно задать только для элемента управления, имеющего фокус. Например, невозможно задать значение **False** для свойства элемента управления **Вывод на экран (Visible)** или **Доступ (Enabled),** если фокус находится на другом элементе управления.

Метод **SetFocus** позволяет осуществлять перемещение по форме в соответствии с некоторыми условиями. Например, можно проверить выбранное в списке значение и, если пользователь выбрал **Другие**, организовать в программе Visual Basic автоматический переход на другой элемент управления, в котором будет проверяться выбор из другой группы значений.

Допускается перемещение фокуса только на видимый элемент управления или форму. Поскольку форма и ее элементы управления не являются видимыми до завершения события **Загрузка (Load),** метод **SetFocus** можно применять для перевода фокуса на загружаемую форму только после вызова метода **Repaint.**

Нельзя поместить фокус на элемент управления, если свойство **Доступ (Enabled)** этого элемента управления имеет значение **False**. До перевода фокуса на этот элемент управления следует установить для его свойства **Доступ (Enabled)** значение **True**. Однако допускается перевод фокуса на элемент управления, у которого свойство **Блокировка (Locked)** имеет значение **True**.

Кроме этого, нельзя перенести фокус на форму, содержащую доступные и видимые элементы управления; в этом случае фокус можно поместить только на один из элементов управления формы. При попытке использовать метод **SetFocus** для перемещения фокуса на форму фокус будет установлен на элемент управления, котором последним имел фокус в данной форме.

Совет. Метод **SetFocus** позволяет перевести фокус на подчиненную форму, представляющую из себя один из типов элементов управления. Кроме того, фокус можно также поместить на элемент управления подчиненной формы, применив метод **SetFocus** дважды, сначала для перевода фокуса на саму подчиненную форму, а затем на один из ее элементов управления.

Метод SetFocus, пример

В данном примере метод **SetFocus** переводит фокус на поле **КодСотрудника** в форме «Сотрудники».

```
Forms!Сотрудники!КодСотрудника.SetFocus
```

Метод Circle

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthCircleC;vafctQBColor;vafctRGB":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthCircleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthCircleA"}

Метод **Circle** создает в объекте **Report** при возникновении события **Печать (Print)** окружность, эллипс или дугу окружности.

Синтаксис

имяОбъекта.**Circle** [**Step**](*x*, *y*), *радиус*[, [*цвет*][, [*начало*][, [*конец*][, *отношение*]]]]

Данный метод можно использовать только в процедурах обработки события или в макросах, указанных в свойствах событий раздела отчета или в свойстве события отчета **Страница (OnPage)**.

Метод **Circle** использует следующие аргументы.

Аргумент	Описание
имяОбъекта	Объект **Report**, в котором создается окружность или дуга.
Step	Ключевое слово, указывающее на центр окружности, эллипса или дуги, связанное с текущими координатами, заданными значениями свойств **CurrentX** и **CurrentY** аргумента *имяОбъекта*.
x, *y*	Значения типа **Single**, являющиеся координатами центральной точки окружности, эллипса или дуги. Свойства Scale (**ScaleMode**, **ScaleLeft**, **ScaleTop**, **ScaleHeight** и **ScaleWidth**) объекта **Report**, заданного аргументом *имяОбъекта*, определяют единицу измерения.
радиус	Значение типа **Single**, задающее радиус окружности, эллипса или дуги. Используемые единицы измерения определятся значением свойств **ScaleMode**, **ScaleLeft**, **ScaleTop**, **ScaleHeight** и **ScaleWidth** объекта *имяОбъекта.* По умолчанию используются единицы твип.
цвет	Целое значение типа **Long,**, задающее цвет контура окружности в стандарте RGB (красный-зеленый-синий). Если данный аргумент опущен, используется значение свойства **ForeColor** объекта. Указать цвет позволят также функции **RGB** или **QBColor**.
начало, *конец*	Значения типа **Single**. При создании дуги окружности или эллипса данные аргументы задают положение начала и конца дуги в радианах. По умолчанию, значение аргумента *начало* равно 0 радиан, а значение аргумента *конец* равно 2 * Пи радиан. Диапазон допустимых значений для этих аргументов составляет от -2 * Пи радиан до 2 * Пи радиан.
отношение	Значение типа **Single**, задающее эксцентриситет кривой. По умолчанию задается значение 1.0, приводящее к созданию правильной окружности (не эллипса) на любом экране.

Дополнительные сведения

Если при создании сектора окружности или эллипса задано отрицательное значение аргумента *начало,* метод **Circle** проводит радиус в точку, определяемую аргументом *начало*, при этом в качестве нужного угла будет принято абсолютное (положительное) значение аргумента. Аналогично, если отрицательное значение имеет аргумент *конец,* метод **Circle** проводит радиус в точку, определяемую аргументом *конец,* при этом в качестве нужного угла будет

принято абсолютное (положительное) значение аргумента. Метод **Circle** всегда отсчитывает угол в положительном направлении (против часовой стрелки).

Для заполнения круга следует задать значения свойств **FillColor** и **FillStyle** для фигур в отчете. Только замкнутые фигуры допускают заливку. Замкнутыми фигурами являются круги, эллипсы и секторы, представляющие части круга или эллипса, ограниченные дугой и двумя радиусами.

Если при создании сектора необходимо провести радиус вправо, то в аргумента *начало* следует задать маленькое отрицательное значение, например -.00000001.

Отдельные аргументы в середине синтаксической конструкции данного метода можно пропускать, однако в этом случае следует сохранить соответствующую запятую аргумента перед заданием следующего аргумента. При пропуске аргументов в конце конструкции лишние запятые после последнего из указанных аргументов оставлять не требуется.

Толщина линии, применяемой для создания окружности, эллипса или дуги, определяется значением свойства **DrawWidth.** Способ создания окружности по отношению к фону определяется значениями свойств **DrawMode** и **DrawStyle.**

После вызова метода **Circle** свойства **CurrentX** и **CurrentY** получат значения координат центра окружности или эллипса.

Метод Circle, пример

В данном примере с помощью метода **Circle** создается окружность, после чего внутри окружности создается сектор с красной заливкой.

Для выполнения данной программы в Microsoft Access создайте новый отчет нажатием кнопки **Создать** на вкладке **Отчеты** в окне базы данных. В ячейке свойства **Печать (OnPrint)** области данных выберите элемент «[Процедура обработки событий]». Введите следующую программу в окно модуля отчета и переключитесь в режим предварительного просмотра.

```
Private Sub ОбластьДанных_Print(Cancel As Integer, PrintCount As Integer)
   Const conPI = 3.14159265359
   Dim sngHCtr As Single, sngVCtr As Single
   Dim sngRadius As Single
   Dim sngStart As Single, sngEnd As Single

   sngHCtr = Me.ScaleWidth / 2     ' X-координата центра.
   sngVCtr = Me.ScaleHeight / 2    ' Y-координата центра.
   sngRadius = Me.ScaleHeight / 3  ' Радиус окружности.
' Проводит окружность.
   Me.Circle(sngHCtr, sngVCtr), sngRadius
   sngStart = -0.00000001          ' Начало сектора.
   sngEnd = -2 * conPI / 3         ' Конец сектора.
   Me.FillColor = RGB(255,0,0)     ' Красный цвет заливки.
   Me.FillStyle = 0                ' Заливка сектора.
   ' Создает сектор внутри круга.
   Me.Circle(sngHCtr, sngVCtr), sngRadius, , sngStart, sngEnd
End Sub
```

Метод Line

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthLineC;vafctQBColor;vafctRGB":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthLineX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthLineA"}

Метод **Line** создает в объекте **Report** при возникновении события **Печать (Print)** линии и прямоугольники.

Синтаксис

имяОбъекта.**Line** [[**Step**](x1, y1)] - [**Step**](x2, y2)[, [*цвет*][, **B[F**]]]

Данный метод можно использовать только в процедурах обработки события или в макросах, указанных в свойствах событий **Печать (OnPrint)** или **Форматирование (OnFormat)** раздела отчета или в свойстве события отчета **Страница (OnPage)**.

Метод **Line** использует следующие аргументы.

Аргумент	Описание
имяОбъекта	Объект **Report**, в котором создается линия или прямоугольник.
Step	Ключевое слово, определяющее, что аргументы *x1* и *y1* указывают положение начальной точки относительно текущих координат, которые определятся текущими значениями свойств **CurrentX** и **CurrentY** объекта *имяОбъекта*.
x1, *y1*	Значения с плавающей точкой типа **Single,** определяющие координаты начальной точки. Используемые единицы измерения определятся значением свойств **ScaleMode**, **ScaleLeft**, **ScaleTop**, **ScaleHeight** и **ScaleWidth** объекта *имяОбъекта.* Если данные аргументы опущены, координаты начальной точки линии или прямоугольника определятся текущими значениями свойств **CurrentX** и **CurrentY.**
Step	Ключевое слово, указывающее на то, что координаты конечной точки указаны относительно начальной точки линии.
x2, *y2*	Значения с плавающей точкой типа **Single**, определяющие координаты конечной точки линии или прямоугольника. Данная пара аргументов является обязательной.
цвет	Целое значение типа **Long,** задающее цвет линии или контура прямоугольника в стандарте RGB (красный-зеленый-синий). Если данный аргумент опущен, используется значение свойства **ForeColor** объекта. Указать цвет позволят также функции **RGB** или **QBColor**.
B	Если указан параметр **B**, то создается прямоугольник, а координаты его углов задаются начальной и конечной точками.
F	Параметр **F** используется только вместе с параметром **B**. Если параметр **B** указан, указание параметра **F** приводит к заливке прямоугольника тем же цветом, который используется для контура прямоугольника. При использовании параметра **B** без параметра **F** заливка прямоугольника определяется текущими значениями свойств **FillColor** и **Тип фона** (**BackStyle**). Значением по умолчанию свойства **Тип фона (BackStyle)** для прямоугольников является «Обычный».

Дополнительные сведения

При создании ломаных линий необходимо следить за тем, чтобы координаты начала каждого следующего отрезка линии совпадали с координатами конца предыдущего отрезка.

Толщина линии определяется значением свойства **DrawWidth.** Способ создания линии или прямоугольника по отношению к фону определяется значениями свойств **DrawMode** и **DrawStyle.**

После вызова метода **Line** свойства **CurrentX** и **CurrentY** получат значения координат конечной точки линии или прямоугольника *x2* и *y2*.

Метод Line, пример

В данном примере с помощью метода **Line** в отчете «Сведения о сотрудниках» создается красный прямоугольник с шириной контура пять пикселов. Для определения цвета линии вызывается функция **RGB**.

Для выполнения данной программы в Microsoft Access создайте новый отчет. Введите следующую программу в окно модуля отчета и переключитесь в режим предварительного просмотра.

```
Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
   ' Вызывает процедуру Drawline
   DrawLine
End Sub

Sub DrawLine()
   Dim rpt As Report, lngColor As Long
   Dim sngTop As Single, sngLeft As Single
   Dim sngWidth As Single, sngHeight As Single

   Set rpt = Reports![Сведения о сотрудниках]
   ' Задает масштаб в пикселах.
   rpt.ScaleMode = 3
   ' Верхняя граница.
   sngTop = rpt.ScaleTop + 5
   ' Левая граница.
   sngLeft = rpt.ScaleLeft + 5
   ' Ширина.
   sngWidth = rpt.ScaleWidth - 10
   ' Высота.
   sngHeight = rpt.ScaleHeight - 10
   ' Красный цвет.
   lngColor = RGB(255,0,0)
   ' Создает прямоугольник.
   rpt.Line(sngTop, sngLeft) - (sngWidth, sngHeight), lngColor, B
End Sub
```

Метод PSet

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthPSetC;vafctQBColor;vafctRGB":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthPSetX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthPSetA"}

Метод **PSet** задает для точки в объекте **Report** указанный цвет при возникновении события **Печать (Print)**.

Синтаксис

имяОбъекта.**PSet** [**Step**](*x*, *y*)[, *цвет*]

Данный метод можно использовать только в процедурах обработки события или в макросах , указанных в свойствах событий **Печать (OnPrint)** или **Форматирование (OnFormat)** раздела отчета или в свойстве события отчета **Страница (OnPage)**.

Метод **PSet** использует следующие аргументы.

Аргумент	Описание
имяОбъекта	Объект **Report**, в котором рисуется точка.
Step	Ключевое слово, определяющее, что аргументы указывают положение точки относительно текущих координат, которые определятся текущими значениями свойств **CurrentX** и **CurrentY** объекта *имяОбъекта*.
x, *y*	Значения типа **Single**, указывающие координаты отображаемой точки.
цвет	Целое значение типа **Long**, задающее цвет точки в стандарте RGB (красный-зеленый-синий). Если данный аргумент не опущен, то используется значение свойства **ForeColor.** Определить цвет позволят функции **RGB** или **QBColor**.

Дополнительные сведения

Размер выводящейся точки определяется значением свойства **DrawWidth.** Если свойство **DrawWidth** имеет значение 1, то метод **PSet** выводит один пиксел заданного цвета. Если значение свойства **DrawWidth** превышает единицу, то создается точка указанных размеров с координатами центра, определяемыми аргументами *x* и *y.*

Способ создания точки определяется значениями свойств **DrawMode** и **DrawStyle.**

После вызова метода **PSet** свойства **CurrentX** и **CurrentY** получат значения координат точки, заданные аргументами метода.

Для того чтобы стереть точку с помощью метода **PSet**, следует задать координаты нужной точки и указать белый цвет (&HFFFFFF) в аргументе *цвет*.

Совет С помощью метода **Line** линия рисуется быстрее, чем с помощью метода **PSet**.

Метод PSet, пример

В данном примере с помощью метода **PSet** в отчете проводится горизонтальная линия.

Для выполнения данной программы в Microsoft Access создайте новый отчет. В ячейке свойства **Печать (OnPrint)** области данных выберите элемент «[Процедура обработки событий]». Введите следующую программу в окно модуля отчета и переключитесь в режим предварительного просмотра.

```
Sub ОбластьДанных_Print(Cancel As Integer, PrintCount As Integer)
   Dim sngMidPt As Single, intI As Integer
   ' Задает масштаб в пикселах.
   Me.ScaleMode = 3
   ' Середина отчета по высоте.
   sngMidPt = Me.ScaleHeight / 2
   ' В цикле линия строится по пикселам.
   For intI = 1 To Me.ScaleWidth
      Me.PSet(intI, sngMidPt)
   Next intI
End Sub
```

Метод Scale

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthScaleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthScaleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthScaleA"}

Метод **Scale** определяет систему координат в объекте **Report.**

Синтаксис

имяОбъекта.**Scale**[(*x1*, *y1*) **-** (*x2*, *y2*)]

Данный метод можно использовать только в процедурах обработки события или в макросах, указанных в свойствах событий **Печать (OnPrint)** или **Форматирование (OnFormat)** раздела отчета или в свойстве события отчета **Страница (OnPage)**.

Метод **Scale** использует следующие аргументы.

Аргумент	Описание
имяОбъекта	Объект **Report**, в котором определяется используемая система координат.
x1, *y1*	Значение с плавающей точкой типа **Single,** определяющее координаты верхнего левого угла объекта в используемой системе координат.
x2, *y2*	Значение с плавающей точкой типа **Single**, определяющее координаты правого нижнего угла объекта в используемой системе координат.

Дополнительные сведения

Метод **Scale** позволяет определить систему координат с произвольным масштабом. Если метод **Scale** вызывается без аргументов, координаты измеряются в единицах твип. Метод **Scale** определяет координатную систему для метода **Print** и графических методов отчета, таких как **Circle**, **Line** и **PSet**.

Метод Scale, пример

В следующем примере сначала строится окружность с одним масштабом, а затем масштаб изменяется с помощью метода **Scale**, и строиться окружность с новым масштабом.

```
Private Sub ОбластьДанных_Print(Cancel As Integer, PrintCount As Integer)
   DrawCircle
End Sub

Sub DrawCircle()
   Dim sngHCtr As Single, sngVCtr As Single
   Dim sngNewH As Single, sngNewV As Single
   Dim sngRadius As Single

   Me.ScaleMode = 3                 ' Задает масштаб в пикселах.
   sngHCtr = Me.ScaleWidth / 2      ' Центр по горизонтали.
   sngVCtr = Me.ScaleHeight / 2     ' Центр по вертикали.
   sngRadius = Me.ScaleHeight / 3   ' Радиус окружности.
   ' Draw circle.
   Me.Circle (sngHCtr, sngVCtr), sngRadius
   ' Новый масштаб по горизонтали.
   sngNewH = Me.ScaleWidth * 0.9
   ' Новый масштаб по вертикали.
   sngNewV = Me.ScaleHeight * 0.9
   ' Изменяет масштаб.
   Me.Scale(0, 0)-(sngNewH, sngNewV)
   ' Рисует окружность.
   Me.Circle (sngHCtr + 100, sngVCtr), sngRadius, RGB(0, 256, 0)
End Sub
```

Метод TextHeight

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthTextHeightC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthTextHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthTextHeightA"}

Метод **TextHeight** возвращает высоту текстовой строки, которую эта строка будет иметь при ее выводе на печать с использованием текущего шрифта объекта **Report.**

Синтаксис

имяОбъекта.**TextHeight(***строка***)**

Метод **TextHeight** использует следующие аргументы.

Аргумент	Описание
имяОбъекта	Имя объекта **Report**, для которого определены шрифт и размер шрифта.
строка	Текстовая строка, для которой определяется значение высоты.

Дополнительные сведения

Метод **TextHeight** применяется для определения размера по вертикали, необходимого для печати указанной строки текста. Например, для текста с шрифтом Arial Cyr с размером 9 пт требуется меньший размер по вертикали, чем для текста со шрифтом Courier и размером 12 пт. Определить текущие настройки шрифта позволят свойства **Шрифт (FontName) и Размер шрифта (FontSize).**

Значение, возвращаемое методом **TextHeight**, измеряется в единицах текущей системы координат, заданной для отчета с помощью метода **Scale.** Сведения о текущей системе координат, используемой в отчете, возвращает свойство **ScaleMode.**

Если аргумент *строка* содержит символы перевода каретки, то метод **TextHeight** возвращает полную высоту всех строк текста, включая просветы над и под каждой строкой. Это значение можно использовать для расчета места, необходимого для размещения в отчете группы строк.

Методы TextHeight, TextWidth, пример

В данном примере методы **TextHeight** и **TextWidth** используются для определения пространства по вертикали и горизонтали, необходимого для печати в отчете текста с применением текущего шрифта отчета.

Для выполнения данной программы в Microsoft Access создайте новый отчет. В ячейке свойства **Печать (OnPrint)** области данных выберите элемент «[Процедура обработки событий]». Введите следующую программу в окно модуля отчета и переключитесь в режим предварительного просмотра.

```
Private Sub ОбластьДанных_Print(Cancel As Integer, PrintCount As Integer)
   ' Задает использование единиц твип (стандартный масштаб).
   Me.Scalemode = 1
   ' Выводит имя и размер шрифта отчета.
   Debug.Print "Шрифт отчета: "; Me.FontName
   Debug.Print "Размер шрифта: "; Me.FontSize
   ' Выводит размеры текста в единицах твип.
   Debug.Print "Высота текста (твип): "; Me.TextHeight("Товары")
   Debug.Print "Ширина текста (твип): "; Me.TextWidth("Товары")
End Sub
```

Метод TextWidth

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthTextWidthC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthTextWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthTextWidthA"}

Метод **TextWidth** возвращает ширину текстовой строки, которую эта строка будет иметь при ее выводе на печать с использованием текущего шрифта объекта **Report**.

Синтаксис

имяОбъекта.**TextWidth(***строка***)**

Метод **TextWidth** использует следующие аргументы.

Аргумент	Описание
имяОбъекта	Имя объекта **Report**, для которого определены шрифт и размер шрифта.
строка	Текстовая строка, для которой определяется значение ширины.

Дополнительные сведения

Метод **TextWidth** применяется для определения размера по горизонтали, необходимого для печати указанной строки текста. Например, для текста с шрифтом Arial Cyr с размером 9 пт требуется меньший размер по горизонтали, чем для текста со шрифтом Courier и размером 12 пт. Определить текущие настройки шрифта позволят свойства **Шрифт (FontName) и Размер шрифта (FontSize)**.

Значение, возвращаемое методом **TextWidth**, выражается в единицах, заданных для координатной системы отчета с помощью метода **Scale.** Определить установленную для отчета систему координат позволяет метод **ScaleMode.**

Если указанный текст печатается в несколько строк, то значение, возвращаемое методом **TextWidth**, определяется максимальной шириной строки от ее начала до символа возврата каретки. Это значение можно использовать для расчета места, необходимого для размещения в отчете группы строк.

Метод Recalc

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthRecalcC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthRecalcX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acmthRecalcA"}

Метод **Recalc** немедленно обновляет все вычисляемые элементы управления в форме.

Синтаксис

имяФормы.**Recalc**
Метод **Recalc** использует следующий аргумент.

Аргумент	Описание
имяФормы	Объект **Form,** представляющий форму, содержащую элементы управления, значения которых требуется пересчитать.

Дополнительные сведения

Вызов данного метода эквивалентен нажатию клавиши F9 для формы, имеющей фокус. Данный метод позволяет пересчитать значения элементов управления, зависящих от других полей, которых могли быть изменены.

Метод Recalc, пример

В данном примере метод **Recalc** вызывается для обновления элементов управления в форме «Заказы». Форма содержит поле «СтоимостьДоставки» и вычисляемое поле, в котором рассчитывается общая стоимость заказа с учетом доставки. Если конструкция, содержащая метод **Recalc**, включается в процедуру обработки события **После обновления (AfterUpdate)** поля «СтоимостьДоставки», общая стоимость заказа будет пересчитываться сразу после ввода нового значения стоимости доставки.

```
Sub СтоимостьДоставки_AfterUpdate()
   Me.Recalc
End Sub
```

Метод Print

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthPrintC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthPrintX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthPrintA;vaobjDebug"}

Метод **Print** распечатывает текст объекта **Report** с использованием текущего цвета и шрифта.

Синтаксис

имяОбъекта.**Print** [{**Spc(***n***)** | **Tab(***n***)**}][*списокВыражения*][{**;**|**,**}]

Метод **Print** использует следующие аргументы.

Аргумент	Описание
имяОбъекта	Объект **Report**, текст которого будет распечатан.
Spc(*n*)	Функция Visual Basic, предназначенная для вставки *n* пробелов в выводимый текст. Допустимо многократное использование этой функции.
Tab(*n*)	Функция Visual Basic, предназначенная для перемещения точки вставки в зону печати номер *n* перед распечаткой аргумента *списокВыражения*. Зоны печати предваряют каждые 14 столбцов. Ширина каждого столбца определяется как средняя ширина букв выбранного шрифта. Допустимо многократное использование этой функции.
списокВыражения	Числовое или символьное выражение для печати. Если этот аргумент опущен, то напечатается пустая строка. При использовании нескольких выражений они разделяются пробелом, точкой с запятой (;), или запятой. В этом случае пробел эквивалентен точке с запятой. Для разделения выражений можно также применять функции **Spc** и **Tab** .
{;|,}	Определяет место вставки следующего печатного символа. Точка с запятой означает, что место вставки расположено сразу за последним напечатанным символом; запятая означает, что место вставки расположено в начале следующей зоны печати.

Дополнительные сведения

Данный метод можно использовать только в процедурах обработки события или в макросах, указанных в свойствах событий **Печать (OnPrint)** или **Форматирование (OnFormat)** раздела отчета или в свойстве события отчета **Страница (OnPage)**.

Выражения, заданные аргументом *списокВыражения*, печатаются в объекте, начиная с позиции, указанной значениями свойств **CurrentX** и **CurrentY**.

После печати аргумента *списокВыражения* обычно добавляется символ возврата каретки. Таким образом новый метод **Print** начинает печать со следующей строки. Если печатается символ возврата каретки, то значение свойства **CurrentY** увеличивается на высоту строки, заданной аргументом *списокВыражения* (значение, равное величине, возвращаемой методом **TextHeight**), а значение свойства **CurrentX** устанавливается в 0.

Если за аргументом *списокВыражения* следует точка с запятой, то символ возврата каретки не добавляется, и новый метод **Print** будет печатать на той же строке. Значения свойств **CurrentX** и **CurrentY** приравниваются координатам точки, следующей за последним напечатанным символом. Если в аргументе *списокВыражения* содержится символы возврата каретки, то каждый из них устанавливает значения свойств **CurrentX** и **CurrentY** так же как для метода **Print** без указания точки с запятой.

Если за аргументом *списокВыражения* следует запятая, то значения для свойств **CurrentX** и **CurrentY** приравниваются к координатам следующей зоны печати на той же строке.

Если аргумент *списокВыражения* печатается в объекте **Report**, то строки, не помещающиеся в заданную позицию, не будут смещаться. При этом текст обрезается чтобы поместиться в объекте.

Так как метод **Print** обычно выполняет печать с использованием пропорционального шрифта, то важно учитывать, что количество печатаемых символов не связано с количеством столбцов фиксированной ширины, выделяемых на один символ. Например, широкие символы (такие как Ш) занимают более одного столбца фиксированной ширины, а узкие символы (такие как цифра 1) занимают менее одного столбца. Следует убедиться, что границы столбцов размещены достаточно широко, чтобы уместить печатающийся текст. Кроме того, можно выполнить печать с использованием моноширинного шрифта (такого как Courier), в котором каждый символ занимает один столбец.

Метод Print, пример

Следующий пример демонстрирует использование метода **Print** для распечатки текста отчета «Отчет1». При этом применяются методы **TextWidth** и **TextHeight** для центрирования текста по вертикали и горизонтали.

```
Private Sub ОбластьДанных_Format(Cancel As Integer, FormatCount As Integer)
   Dim rpt as Report
   Dim strMessage As String
   Dim intHorSize As Integer, intVerSize As Integer

   Set rpt = Me
   strMessage = "DisplayMessage"
   With rpt
      'Устанавливает масштаб в пикселах, а также значения свойств FontName
и FontSize.
      .ScaleMode = 3
      .FontName = "Courier"
      .FontSize = 24
   End With
   ' Ширина.
   intHorSize = Rpt.TextWidth(strMessage)
   ' Высота.
   intVerSize = Rpt.TextHeight(strMessage)
   ' Calculate location of text to be displayed.
   Rpt.CurrentX = (Rpt.ScaleWidth/2) - (intHorSize/2)
   Rpt.CurrentY = (Rpt.ScaleHeight/2) - (intVerSize/2)
   ' Печатает текст в объекте Report.
   Rpt.Print strMessage
End Sub
```

Метод BuildCriteria

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthBuildCriteriaC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthBuildCriteriaX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthBuildCriteriaA"}

Метод **BuildCriteria** возвращает синтаксически проанализированную строку условий отбора в том виде, который она имела бы в бланке запроса или в фильтре для формы. Например, иногда возникает необходимость определить значение свойства **Фильтр** (**Filter**) формы на основании условий, изменяемых пользователем. В этом случае метод **BuildCriteria** позволяет создать строку условий, определяющую значение свойства **Фильтр (Filter)**.

Синтаксис

*имяПриложения.***BuildCriteria(***поле*, *типПоля*, *выражение***)**

Метод **BuildCriteria** использует следующие аргументы.

Аргумент	Описание
имяПриложения	(Необязательный аргумент) Имя объекта **Application**.
поле	Строковое выражение, определяющее поле, для которого задается условие отбора.
типПоля	Внутренняя константа, задающая тип данных поля. Для просмотра списка возможных типов данных полей используется свойство объектов доступа к данным (DAO) **Type**.
выражение	Строковое выражение, определяющее условие отбора, которое будет синтаксически проанализировано.

Метод **BuildCriteria** возвращает строку.

Дополнительные сведения

Метод **BuildCriteria** позволяет легко определять условия отбора для фильтра на основании условий, вводящихся пользователем. В этом методе аргумент *выражение* анализируется таким же способом, каким это выражение было бы проанализировано при вводе условий отбора в бланк запроса в режиме фильтра для формы.

Например, если требуется, создать запрос по таблице «Заказы», ограничивающий результирующий набор записей заказами, размещенными после 1 января 1995 г., то пользователь может указать условия отбора для поля «ДатаРазмещения», поместив в ячейку условий отбора для этого поля следующее выражение:

>1-1-95

Microsoft Access автоматически проанализирует это выражение и возвратит его в следующем виде:

>#1/1/95#

Метод **BuildCriteria** выполнит тот же анализ в программе Visual Basic. Например, для того чтобы возвратить корректно проанализированную строку, представленную выше, можно указать для метода **BuildCriteria** следующие аргументы

```
Dim strCriteria As String
strCriteria = BuildCriteria("ДатаРазмещения", dbDate, ">1-1-95")
```

Условия отбора, требуемые свойством **Фильтр (Filter)** в синтаксически правильной форме, будут построены в методе **BuildCriteria** именно в таком виде.

Метод **BuildCriteria** позволяет создавать строку условий, содержащую комбинированные условия для одного поля. Например, в следующей конструкции метод **BuildCriteria** создает строку с комбинированными условиями отбора для поля «ДатаРазмещения».

```
strCriteria = BuildCriteria("ДатаРазмещения", dbDate, ">1-1-95 and <5-1-
95")
```

Синтаксически обработанная строка условий отбора будет иметь вид.

ДатаРазмещения>#1/1/95# And ДатаРазмещения<#5/1/95#

Однако, если требуется создать строку условий отбора, включающую условия для нескольких полей, пользователь должен создавать такие строки и объединять их самостоятельно. Например, для отбора записей, относящихся к заказам, размещенным после 1.01.95, стоимость доставки которых превышает 50000р., необходимо вызвать метод **BuildCriteria** дважды, а затем выполнить объединение строк.

Метод BuildCriteria, пример

В следующем примере выводится приглашение пользователю ввести несколько первых букв названия товара, после чего с помощью метода **BuildCriteria** строится строка условий отбора, основанная на введенных символах. Далее в процедуре эта строка условий используется как аргумент для свойства **Фильтр (Filter)** формы «Товары». И наконец, фильтр применяется с помощью свойства **Фильтр включен (FilterOn)**.

```
Sub SetFilter()
   Dim frm As Form, strMsg As String
   Dim strInput As String, strFilter As String

   ' Открывает форму "Товары" в режиме формы.
   DoCmd.OpenForm "Товары"
   ' Возвращает объектную переменную, указывающую на форму "Товары".
   Set frm = Forms!Products
   strMsg = "Введите одну или несколько первых букв в марке товара " _
   & "с последующей звездочкой."
   ' Выводит приглашение пользователю.
   strInput = InputBox(strMsg)
   ' Строит строку условий.
   strFilter = BuildCriteria("Марка", dbText, strInput)
   ' Задает значение свойства "Фильтр" (Filter) для применяемого фильтра.
   frm.Filter = strFilter
   ' Включает фильтр с помощью свойства "Фильтр включен" (FilterOn).
   frm.FilterOn = True
End Sub
```

Метод CloseCurrentDatabase

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ.
пїЅпїЅпїЅпїЅпїЅ":"acmthCloseCurrentDatabaseC;vaconUnderstandingOLEAutomation"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acmthCloseCurrentDatabaseX":1} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acmthCloseCurrentDatabaseA"}

Метод **CloseCurrentDatabase** используется для закрытия текущей базы данных из другого приложения, в котором эта база данных была открыта с помощью механизма программирования объектов. Например, этот метод позволяет закрыть из Microsoft Excel текущую базу данных, которая была открыта в окне Microsoft Access.

Синтаксис

приложение.**CloseCurrentDatabase**

Метод **CloseCurrentDatabase** использует следующий аргумент.

Аргумент	Описание
приложение	Объект **Application**.

Дополнительные сведения

Метод **CloseCurrentDatabase** применяется к базе данных Microsoft Access, открытой из другого приложения с помощью механизма программирования объектов. После создания экземпляра Microsoft Access из другого приложения, может возникнуть необходимость создать новую базу данных или открыть другу базу данных. Эта база данных открывается в окне Microsoft Access.

После закрытия текущей базы данных с помощью метода **CloseCurrentDatabase** становится возможным открытие другой базы данных без необходимости создавать новый экземпляр Microsoft Access.

Примечание. Не следует путать метод **CloseCurrentDatabase** с методом **Close** объектов доступа к данным. Метод **Close** закрывает объектную переменную типа **Database** и выводит эту переменную из области определения, не закрывая в действительности базу данных, открытую в окне Microsoft Access. Метод **CloseCurrentDatabase** закрывает саму базу данных, открытую в окне Microsoft Access.

Метод CloseCurrentDatabase, пример

Следующая программа открывает базу данных Microsoft Access из другого приложения с помощью механизма программирования объектов, создает и сохраняет новую форму, а затем закрывает базу данных.

Данная программа может быть включена в модуль Visual Basic любого приложения, способного выполнять роль компонента ActiveX. Например, эту программу можно запустить из Microsoft Excel или Microsoft Visual Basic.

После того как переменная, представляющая объект **Application** выходит из области определения, экземпляр Microsoft Access, который она представляет, закрывается. Поэтому данную переменную необходимо описать на уровне модуля.

```
' Следующую инструкцию необходимо включить в раздел описаний модуля.
Dim appAccess As Access.Application

Sub НоваяФорма()
   Const strConPathToSamples = "C:\Program Files\Microsoft Office\Office\
Samples\"

   Dim frm As Form, strDB As String

   ' Инициализирует строку, указывающую путь к базе данных.
   strDB = strConPathToSamples & "Борей.mdb"
   ' Создает новый экземпляр Microsoft Access.
   Set appAccess = CreateObject("Access.Application.8")
   ' Открывает базу данных в окне Microsoft Access.
   appAccess.OpenCurrentDatabase strDB
   ' Создает новую форму.
   Set frm = appAccess.CreateForm
   ' Сохраняет новую форму.
   appAccess.DoCmd.Save , "NewForm1"
   ' Закрывает текущую базу данных.
   appAccess.CloseCurrentDatabase
   Set AppAccess = Nothing
End Sub
```

Метод DefaultControl

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthDefaultControlC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthDefaultControlX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthDefaultControlA"}

Метод **DefaultControl** возвращает объект **Control,** с помощью которого задаются стандартные свойства элементов управления определенного типа в конкретной форме. Например, этот метод позволяет определить стандартные свойства полей, создаваемых в этой форме. После этого указанный основной набор свойств будут получать все создаваемые в форме поля без необходимости задавать эти свойства для каждого поля по отдельности.

Синтаксис

Set *элемент* **=** *объект***.DefaultControl(***типЭлемента***)**

Метод **DefaultControl** использует следующие аргументы.

Аргумент	Описание
элемент	Объект **Control**, для которого должны быть установлены стандартные свойства.
объект	Объект **Form** или **Report,** в котором будут создаваться элементы управления. Указанный набор стандартных свойств, определенных для конкретного типа элементов управления, будет применяться только к элементам управления этого типа, создаваемым в данной форме или отчете.
типЭлемента	Встроенная константа, указывающая тип элемента управления, для которого задаются стандартные свойства.

Дополнительные сведения

Метод **DefaultControl** позволяет определять набор стандартных свойств элемента управления программным образом. Набор стандартных свойств, определенных для конкретного типа элементов управления, будет применяться к каждому элементу управления этого типа, создаваемому в данной форме или отчете.

Например, если задать для свойства **Размер шрифта** (**FontSize**) стандартной кнопки значение 12, надписи на каждой кнопке будут создаваться с размером 12 пунктов.

Не все свойства элемента управления допускают определение в качестве стандартных свойств. Конкретный набор стандартных свойств зависит от типа элемента управления.

Метод **DefaultControl** возвращает объект **Control** типа, указанного в аргументе *типЭлемента.* Этот объект **Control** представляет не конкретный элемент управления в форме, а является прототипом всех создаваемых в форме элементов управления данного типа. Значения стандартных свойств объекта **Control**, который возвращается методом **DefaultControl**, задаются теми же способами, что и значения конкретного элемента управления в форме.

Список внутренних констант, которые передаются как значение аргумента *типЭлемента*, см. в разделе справки для функции **CreateControl.** Эти константы также доступны для просмотра в модуле «Constants» библиотеки объектов Microsoft Access в окне **Просмотр объектов**.

Метод **DefaultControl** применим только в режиме конструктора формы или в режиме конструктора отчета. Попытка вызвать этот метод для формы или отчета, не находящихся в режиме конструктора, приведет к ошибке при выполнении.

Попытка задать с помощью метода **DefaultControl** значение свойства, не допускающего определения в качестве стандартного, также приведет к ошибке при выполнении. Для того чтобы узнать, какие свойства могут быть определены как стандартные, следует вывести перечень компонентов семейства **Properties** объекта **Control**, возвращаемого в методе

DefaultControl.

Метод DefaultControl, пример

Следующая программа создает новую форму и с помощью метода **DefaultControl** возвращает объект **Control**, представляющий стандартную кнопку. Далее в процедуре определятся некоторые стандартные свойства кнопки и создается новая кнопка в форме.

```
Sub SetDefaultProperties()
   Dim frm As Form, ctlDefault As Control, ctlNew As Control

   ' Создает новую форму.
   Set frm = CreateForm
   ' Возвращает объект Control, представляющий стандартную кнопку.
   Set ctlDefault = frm.DefaultControl(acCommandButton)
   ' Задает некоторые стандартные свойства.
   With ctlDefault
      .FontWeight = 700
      .FontSize = 12
      .Width = 3000
      .Height = 1000
   End With
   ' Создает новую кнопку.
   Set ctlNew = CreateControl(frm.Name, acCommandButton, , , , 500, 500)
   ' Задает заголовок элемента.
   ctlNew.caption = "Новая кнопка"
   ' Восстанавливает окно формы.
   DoCmd.Restore
End Sub
```

Метод SizeToFit

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthSizeToFitC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthSizeToFitX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthSizeToFitA"}

Метод **SizeToFit** задает изменение размеров элемента управления в соответствии с размерами содержащегося в нем текста. Например, применение метода **SizeToFit** к кнопке, на которой не умещается подпись, позволит полностью поместить на кнопку текст подписи, заданный в свойстве **Подпись** (**Caption**) кнопки.

Синтаксис

элемент.**SizeToFit**

Метод **SizeToFit** использует следующий аргумент.

Аргумент	Описание
элемент	Элемент управления, который отображает текст или рисунок. К таким элементам относятся подпись, выключатель, кнопка, рисунок, свободная рамка объекта и составная форма/подчиненный отчет.

Дополнительные сведения

Вызов метода **SizeToFit** эквивалентен выделению элемента управления в форме или отчете с последующим выбором в меню **Формат** команды **Размер** и подкоманды **по размеру данных**. Метод **SizeToFit** применим к элементам управления только в режиме конструктора формы или в режиме конструктора отчета.

В результате применения метода **SizeToFit** размеры элемента управления могут как увеличиться, так и уменьшиться, в зависимости от размеров содержащихся в них текста или рисунков.

Метод **SizeToFit** обычно применяют вместе с функцией **CreateControl** для указания размеров новых элементов управления, создающихся программным образом.

Примечание. Не все элементы управления, содержащие текст или рисунки, допускают изменение размеров с помощью метода **SizeToFit**. Некоторые элементы управления связываются с данными, размеры которых могут изменяться от записи к записи. Такими элементами управления являются поле, список, поле со списком и присоединенная рамка объекта.

Метод SizeToFit, пример

Следующая программа создает новую форму и кнопку в этой форме. Далее в процедуре определяется значение свойства кнопки **Подпись (Caption)** и задается изменение размеров кнопки по размеру подписи.

```
Sub SizeNewControl()
   Dim frm As Form, ctl As Control

   ' Создает новую форму.
   Set frm = CreateForm
   ' Создает новую кнопку.
   Set ctl = CreateControl(frm.Name, acCommandButton, , , , 500, 500)
   ' Восстанавливает форму.
   DoCmd.Restore
   ' Задает значение свойства "Подпись" (Caption).
   ctl.Caption = "Очень длинный заголовок"
   ' Изменяет размер кнопки по размеру подписи.
   ctl.SizeToFit
End Sub
```

Метод DefaultWorkspaceClone

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ.
пїЅпїЅпїЅпїЅпїЅпїЅS":"acmthDefaultWorkspaceCloneC;damthBeginTrans;daproName;daproPassword"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acmthDefaultWorkspaceCloneX":1} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acmthDefaultWorkspaceCloneA"}

Метод **DefaultWorkspaceClone** создает новый объект **Workspace** (рабочая область), не требуя от пользователя нового подключения. Например, если требуется одновременно выполнить две транзакции в двух независимых рабочих областях, метод **DefaultWorkspaceClone** позволяет создать **Workspace** с тем же именем пользователя и паролем без необходимости вводить эти сведения повторно.

Синтаксис

приложение.**DefaultWorkspaceClone**

Метод **DefaultWorkspaceClone** использует следующий аргумент.

Аргумент	Описание
приложение	Объект **Application**.

Дополнительные сведения

Метод **DefaultWorkspaceClone** создает копии стандартного объекта **Workspace** в Microsoft Access. Значения свойств объекта **Workspace** полностью совпадают со значениями свойств стандартного объекта **Workspace**, за исключением свойства **Name.** Для стандартного объекта **Workspace** свойство **Name** всегда имеет значение #Default Workspace#. Копия объекта **Workspace** всегда получает имя #CloneAccess#.

В свойстве **UserName** стандартного объекта **Workspace** содержится имя, под которым подключился текущий пользователь. Копия объекта **Workspace** эквивалентна объекту **Workspace**, который был бы создан при новом подключении текущего пользователя с тем же именем и паролем.

Метод DefaultWorkspaceClone, пример

В следующем примере создается копия стандартной рабочей области и выводится перечень свойств каждой рабочей области.

```
Sub CloneWorkspace()
   Dim wrkDefault As Workspace, wrkClone As Workspace
   Dim prp As Property
   On Error Resume Next
   ' Определяет стандартную рабочую область.
   Set wrkDefault = DBEngine.Workspaces(0)
   ' Создает копию стандартной рабочей области.
   Set wrkClone = Application.DefaultWorkspaceClone
   ' Выводит перечень свойств каждой рабочей области.
   For Each prp In wrkDefault.Properties
      Debug.Print ">>>"; prp.Name; "   "; prp.Value
   Next prp
   For Each prp In wrkClone.Properties
      Debug.Print ">>>"; prp.Name; "   "; prp.Value
   Next prp
End Sub
```

Функция Nz

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïS. nïSnïSnïSnïSnïS":"acfctNzC;vafctIsEmpty;vafctIsNull;vaoprIs"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctNzX":1}

Функция **Nz** возвращает нуль, пустую строку ("") или другое указанное значение, если переменная типа **Variant** имеет значение **Null.** Например, эту функцию используют для преобразования значений **Null** в другое значение при работе с выражениями, не допускающими пустых значений.

Синтаксис

Nz(*variant*[, *представление*]**)**

Функция **Nz** использует следующие аргументы.

Аргумент	Описание
variant	Переменная с типом данных **Variant**.
представление	Необязательный аргумент (если не используется в запросе). Значение типа **Variant**, которое возвращается, если аргумент *variant* имеет значение **Null**. Данный аргумент позволяет возвращать значение, отличное от нуля или пустой строки. Если функция Nz используется в выражении в запросе без аргумента «Представление», то результатом будет пустая строка в полях, содержащих значения null

Если аргумент *variant* имеет значение **Null**, функция **Nz** возвращает нуль или пустую строку, в зависимости от контекста, требующего числовое или строковое значение. Если указан необязательный аргумент *представление*, то функция **Nz** возвращает это значение при пустом значении аргумента *variant.*

Если аргумент *variant* имеет значение, отличное от значения **Null**, то функция **Nz** возвращает значение аргумента *variant*.

Дополнительные сведения

Функцию **Nz** используют при работе с выражениями, в которых могут оказаться пустые значения. Для того чтобы это выражение возвращало непустое значение при любых значениях входящих в него компонентов, следует с помощью функции **Nz** определить замену пустых значений на нули, пустые строки или любое специальное значение, представляющее пустые значения.

Например, выражение `2 + varX` возвращает значение **Null**, если переменная `varX` типа **Variant** имеет значение **Null**. Однако выражение `2 + Nz(varX)` в этом случае возвращает значение 2.

Функцию **Nz** часто используют как альтернативу функции **IIf.** Например, в следующей конструкции для получения нужных результатов требуются две инструкции с функцией **IIf**. Первое выражение, содержащее функцию **IIf**, используется для проверки на пустые значения и преобразования пустых значений в нулевые.

```
varTemp = IIf(IsNull(доставка), 0, доставка)
varResult = IIf(varTemp > 50, "Больше", "Меньше")
```

В следующей конструкции функция **Nz** позволяет выполнить те же действия в одной программной строке.

```
varResult = IIf(Nz(доставка) > 50, "Больше", "Меньше")
```

Если указать необязательный аргумент *представление*, его значение будет возвращаться вместо пустого значения аргумента *variant.* Использование этого необязательного аргумента позволяет исключить одно выражение, содержащее функцию **IIf**. Например, следующее

выражение использует функцию **IIf** для возвращения строки, если переменная `доставка` имеет значение **Null**.

```
varResult = IIf(IsNull(доставка), "Бесплатно", доставка)
```

В следующем примере необязательный аргумент функции **Nz** указывает строку, которая возвращается, если переменная `доставка` имеет значение **Null**.

```
varResult = Nz(доставка, "Бесплатно")
```

Функция Nz, пример

Следующая программа проверяет значение элемента управления в форме и возвращает одну из двух строк, определяющуюся значением элемента управления. Если элемент управления имеет значение **Null**, то с помощью функции **Nz** значение **Null** преобразуется в пустую строку.

```
Sub CheckValue()
   Dim frm As Form, ctl As Control
   Dim varResult As Variant

   ' Возвращает переменную типа Form, указывающую на форму "Заказы".
   Set frm = Forms!Заказы
   ' Возвращает переменную типа Control, указывающую на ОбластьПолучателя.
   Set ctl = frm!ОбластьПолучателя
   ' Выбирает строку, зависящую от значения элемента управления.
   varResult = IIf(Nz(ctl.Value) = "", "Отсутствует", "Значение " &
ctl.Value)
   ' Выводит результат.
   MsgBox varResult
End Sub
```

Метод RefreshTitleBar

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthRefreshTitleBarC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acmthRefreshTitleBarX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acmthRefreshTitleBarA"}

Метод **RefreshTitleBar** обновляет строку заголовка Microsoft Access после изменения значения свойства **AppTitle** или **AppIcon** в программе Visual Basic. Например, с помощью свойства **AppTitle** можно указать для строки заголовка Microsoft Access текст «База данных 'Контакты'».

Синтаксис

приложение.**RefreshTitleBar**

Метод **RefreshTitleBar** использует следующий аргумент.

Аргумент	Описание
приложение	Объект **Application**.

Дополнительные сведения

Свойства **AppTitle** и **AppIcon** позволяет заменить стандартные заголовок и значок Microsoft Access на собственные заголовок и значок приложения. Изменение значений этих свойств не приводит к автоматическому обновлению окна приложения. Для того чтобы новые значения этих свойств вступили в действие, следует вызвать метод **RefreshTitleBar**.

Для того чтобы вернуть свойствам **AppTitle** и **AppIcon** их стандартные значения, следует удалить эти свойства из семейства **Properties** объекта **Database,** представляющего текущую базу данных. После удаления этих свойств следует вызвать метод **RefreshTitleBar** для восстановления стандартных значка и заголовка Microsoft Access.

Метод RefreshTitleBar, пример

Следующая программа задает значение свойства **AppTitle** текущей базы данных и вызывает метод **RefreshTitleBar** для обновления строки заголовка.

```
Sub ChangeTitle()
   Dim dbs As Database, prp As Property
   Const conPropNotFoundError = 3270

   On Error GoTo ErrorHandler
   ' Возвращает переменную Database,
   ' представляющую текущую базу данных.
   Set dbs = CurrentDb
   ' Изменяет текст строки заголовка.
   dbs.Properties!AppTitle = "Контакты"
   ' Обновляет заголовок на экране.
   Application.RefreshTitleBar
   Exit Sub

ErrorHandler:
   If Err.Number = conPropNotFoundError Then
      Set prp = dbs.CreateProperty("AppTitle", dbText, "Контакты")
      dbs.Properties.Append prp
   Else
      MsgBox "Ошибка: " & Err.Number & vbCrLf & Err.Description
   End If
   Resume Next
End Sub
```

Метод AddToFavorites

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthAddToFavoritesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthAddToFavoritesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthAddToFavoritesA"}

Метод **AddToFavorites** добавляет адрес гиперссылки в папку «Избранное».

Синтаксис

объект.*гиперссылка*.**AddToFavorites**

Метод **AddToFavorites** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Application**, или объект **Control**, содержащий гиперссылку. Элементы управления подпись, кнопка и рисунок могут содержать гиперссылки.
гиперссылка	Гиперссылка объекта **Control**.

Дополнительные сведения

Применение метода **AddToFavorites** к объекту **Application** приводит к добавлению имени текущей базы данных в папку «Избранное». Примером может служить применение метода **AddToFavorites** к объекту **Application** для занесения в папку «Избранное» адреса гиперссылки учебной базы данных «Борей».

Применение метода **AddToFavorites** к объекту **Control** приводит к добавлению адреса гиперссылки, содержащегося в элементе управления, в папку «Избранное». Эта папка по умолчанию создается системой в папке Windows.

Использование метода **AddToFavorites** приводит к тем же последствиям, что и выбор команды **Добавить в папку «Избранное»** в меню **Личные** на панели инструментов **Web** (при этом документ, адрес которого требуется добавить, должен быть открыт).

Метод AddToFavorites, пример

В следующем примере задается значение свойства **Адрес гиперссылки (HyperlinkAddress)** для кнопки. При нажатии кнопки пользователем адрес добавляется в папку «Избранное» с помощью метода **AddToFavorites**.

Для проверки данного примера создайте новую форму и добавьте в нее кнопку с именем «Кнопка0». Включите следующую процедуру в модуль формы. Перейдите в режим формы и нажмите кнопку.

```
Private Sub Form_Load()
   Me!Кнопка0.HyperlinkAddress = "http://www.microsoft.com/"
End Sub

Private Sub Кнопка0_Click()
   Me!Кнопка0.Hyperlink.AddToFavorites
End Sub
```

Метод AddFromFile (объект Module)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthAddFromFileC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthAddFromFileX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthAddFromFileA"}

Метод **AddFromFile** добавляет содержимое текстового файла в объект **Module**. Объект **Module** представляет стандартный модуль или модуль класса.

Синтаксис

объект.**AddFromFile** *имяФайла*

Метод **AddFromFile** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Module**.
имяФайла	Имя и полный путь текстового файла (.txt) или другого файла, содержащего текст в формате ANSI.

Дополнительные сведения

Метод **AddFromFile** помещает содержимое текстового файла в модуле сразу после раздела описаний перед первой процедурой, если модуль содержит другие процедуры.

Применение метода **AddFromFile** позволяет импортировать программные строки и комментарии, содержащиеся в текстовом файле.

Чтобы добавить содержимое текстового файла в модуль формы или отчета, следует открыть форму или отчет, соответственно, в режиме конструктора формы или в режиме конструктора отчета. Чтобы добавить содержимое текстового файла в стандартный модуль или в модуль класса, следует открыть нужный модуль.

Методы AddFromFile, AddFromString, пример

Следующая функция использует методы **AddFromString** и **AddFromFile** для добавления, соответственно, текстовой строки и содержимого текстового файла в стандартный модуль.

```
Function AddTextToModule(strModuleName As String, strFileName As String,
strText As String) As Boolean
   Dim mdl As Module

   On Error GoTo Error_AddTextToModule
   DoCmd.OpenModule strModuleName
   Set mdl = Modules(strModuleName)
   mdl.AddFromFile strFileName
   mdl.AddFromString strText
   AddTextToModule = True

Exit_AddTextToModule:
   Exit Function

Error_AddTextToModule:
   MsgBox Err & ": " & Err.Description
   AddTextToModule = False
   Resume Exit_AddTextToModule
End Function
```

Рассмотренную функцию можно вызвать в процедуре следующим образом. Создайте текстовый файл с именем Functions.txt, добавьте к нему какие-либо процедуры Visual Basic и сохраните все вместе в каталоге «Мои документы». Затем поместите рассмотренную процедуру вместе с нижеследующей процедурой в новый стандартный модуль в учебной базе данных «Борей». Запустите следующую процедуру:

```
Sub AddFunctionsFromText()
   Dim strModuleName As String, strFileName As String
   Dim strText As String

   strModuleName = "Служебные программы"
   strFileName = "C:\Мои документы\Functions.txt"
   strText = "Public intX As Integer" & vbCrLf _
      & "Const conPathName As String = " _
      & """C:\Program Files\Microsoft Office\Office\Samples\"""
   If AddTextToModule(strModuleName, strFileName, strText) = True Then
      Debug.Print "Строка и содержимое файла добавлены."
   Else
      Debug.Print "Строка и содержимое файла не добавлены."
   End If
End Sub
```

В следующем примере создается новая форма, и в ее модуль добавляются текстовая строка и содержимое файла Functions.txt. Запустите следующую процедуру из стандартного модуля:

```
Sub AddTextToFormModule()
   Dim frm As Form, mdl As Module

   Set frm = CreateForm
   Set mdl = frm.Module
   mdl.AddFromString "Public intY As Integer"
   mdl.AddFromFile "C:\Мои документы\Functions.txt"
End Sub
```

Метод AddFromString

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthAddFromStringC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthAddFromStringX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthAddFromStringA"}

Метод **AddFromString** добавляет текстовую строку в объект **Module**. Объект **Module** представляет стандартный модуль или модуль класса.

Синтаксис

*объект.***AddFromString** *выражение*

Метод **AddFromString** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Module**.
выражение	Строковое выражение.

Дополнительные сведения

Метод **AddFromString** помещает содержимое строки символов в модуле после раздела описаний перед первой существующей процедурой, если модуль содержит другие процедуры.

Чтобы добавить строку символов в модуль формы или отчета, следует открыть форму или отчет, соответственно, в режиме конструктора формы или в режиме конструктора отчета. Чтобы добавить строку в стандартный модуль или в модуль класса, следует открыть нужный модуль.

Метод CreateEventProc

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthCreateEventProcC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthCreateEventProcX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthCreateEventProcA"}

Метод **CreateEventProc** создает процедуру обработки событий в модуле класса. Он возвращает значение типа **Long**, указывающее номер первой строки процедуры обработки событий.

Синтаксис

объект.**CreateEventProc(***имяСобытия*, *имяОбъекта***)**

Метод **CreateEventProc** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Module**, свойство **Type** которого возвращает константу **acClassModule**, имеющую значение 1.
имяСобытия	Строковое выражение,значением которого является имя события.
имяОбъекта	Объект, для которого возникает событие, указанное в аргументе *имяСобытия*. Сюда относятся объекты **Form**, **Report** и **Control**, а также раздел формы, раздел отчета и модуль класса.

Дополнительные сведения

Метод **CreateEventProc** создает заготовку программы для процедуры обработки событий указанного объекта. Например, этот метод позволяет создать процедуру обработки события **Нажатие кнопки (Click)** для кнопки в форме. Microsoft Access создает процедуру обработки события **Нажатие кнопки** в модуле, который соответствует форме, содержащей кнопку.

После того, как с помощью метода **CreateEventProc** создана программная заготовка для процедуры обработки событий, в нее можно добавить программные строки с помощью других методов объекта **Module**. Например, для вставки программной строки используется метод **InsertLines**.

Методы CreateEventProc, InsertLines, пример

В следующем примере создается новая форма, в нее добавляется кнопка, и создается процедура обработки события **Нажатие кнопки (Click)** для кнопки:

```
Function ClickEventProc() As Boolean
   Dim frm As Form, ctl As Control, mdl As Module
   Dim lngReturn As Long

   On Error GoTo Error_ClickEventProc
   ' Создает новую форму.
   Set frm = CreateForm
   ' Создает кнопку в форме.
   Set ctl = CreateControl(frm.Name, acCommandButton, , , , 1000, 1000)
   ctl.Caption = "Нажмите кнопку"
   ' Возвращает ссылку на модуль формы.
   Set mdl = frm.Module
   ' Добавляет процедуру обработки события.
   lngReturn = mdl.CreateEventProc("Click", ctl.Name)
   ' Вставляет текст в процедуру.
   mdl.InsertLines lngReturn + 1, vbTab & "MsgBox ""Порядок!"""
   ClickEventProc = True

Exit_ClickEventProc:
   Exit Function

Error_ClickEventProc:
   MsgBox Err & " :" & Err.Description
   ClickEventProc = False
   Resume Exit_ClickEventProc
End Function
```

Метод AddFromFile (семейство References)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthCreateFromFileC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthCreateFromFileX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthCreateFromFileA"}

Метод **AddFromFile** создает ссылку на библиотеку типов, находящуюся в указанном файле.

Синтаксис

Set *объект* **= References.AddFromFile(***имяФайла***)**

Метод **AddFromFile** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Reference**.
имяФайла	Строковое выражение, значением которого является полный путь и имя файла, содержащего библиотеку типов, на которую задается ссылка.

Дополнительные сведения

В следующей таблице перечислены типы файлов, которые обычно содержат библиотеки типов.

Расширение файла	Тип файла
.olb, .tlb	Библиотечный файл
.mdb, .mda, .mde	База данных
.exe, .dll	Исполняемый файл
.ocx	Элемент ActiveX

Метод AddFromFile (семейство References), пример

В следующем примере создается ссылка на указанную библиотеку типов:

```
Function ReferenceFromFile(strFileName As String) As Boolean
   Dim ref As Reference

   On Error GoTo Error_ReferenceFromFile
   ' Создает новую ссылку.
   Set ref = References.AddFromFile(strFileName)
   ReferenceFromFile = True

Exit_ReferenceFromFile:
   Exit Function

Error_ReferenceFromFile:
   MsgBox Err & ": " & Err.Description
   ReferenceFromFile = False
   Resume Exit_ReferenceFromFile
End Function
```

В следующей процедуре данная функция вызывается для создания ссылки на элемент управления-календарь:

```
Sub CreateCalendarReference()
   If ReferenceFromFile("C:\Windows\System\Mscal.ocx") = True Then
      MsgBox "Создана ссылка на элемент управления-календарь."
   Else
      MsgBox "Не создана ссылка на элемент управления-календарь."
   End If
End Sub
```

Метод AddFromGUID

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthCreateFromGuidC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthCreateFromGuidX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acmthCreateFromGuidA"}

Метод **AddFromGUID** создает объект **Reference** на основе кода GUID, определяющего библиотеку типов.

Синтаксис

объект.**AddFromGUID**(*GUID*)

Метод **AddFromGUID** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Reference**.
GUID	Код GUID, определяющий библиотеку типов.

Дополнительные сведения

Свойство **GUID** возвращает код GUID для указанного объекта **Reference**. Предварительно сохраненное значение свойства **GUID** позволяет повторно создавать нарушенные ссылки.

Метод DeleteLines

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthDeleteLinesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthDeleteLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthDeleteLinesA"}

Метод **DeleteLines** удаляет строки из стандартного модуля или из модуля класса.

Синтаксис

объект.**DeleteLines** *строка1*, *количество*

Метод **DeleteLines** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Module**.
строка1	Значение типа **Long**, указывающее номер строки модуля, с которой начинается удаление.
количество	Значение типа **Long**, указывающее количество удаляемых строк.

Дополнительные сведения

Нумерация строк в модуле начинается с 1. Для определения числа строк в модуле используется свойство **CountOfLines**.

Для замены одной строки на другую используется метод **ReplaceLine**.

Метод DeleteLines Method, свойство Lines, пример

В следующем примере из модуля удаляется указанная строка.

```
Function DeleteWholeLine(strModuleName, strText As String) As Boolean
   Dim mdl As Module, lngNumLines As Long
   Dim lngSLine As Long, lngSCol As Long
   Dim lngELine As Long, lngECol As Long
   Dim strTemp As String

   On Error GoTo Error_DeleteWholeLine
   DoCmd.OpenModule strModuleName
   Set mdl = Modules(strModuleName)

   If mdl.Find(strText, lngSLine, lngSCol, lngELine, lngECol) Then
      lngNumLines = Abs(lngELine - lngSLine) + 1
      strTemp = LTrim$(mdl.Lines(lngSLine, lngNumLines))
      strTemp = RTrim$(strTemp)
      If strTemp = strText Then
         mdl.DeleteLines lngSLine, lngNumLines
      Else
         MsgBox "Строка содержит текст в дополнение к '" _
            & strText & "'."
      End If
   Else
      MsgBox "Текст '" & strText & "' не найден."
   End If
   DeleteWholeLine = True

Exit_DeleteWholeLine:
   Exit Function

Error_DeleteWholeLine:
   MsgBox Err & " :" & Err.Description
   DeleteWholeLine = False
   Resume Exit_DeleteWholeLine
End Function
```

Примером использования данной функции может служить следующая процедура, которая находит в модуле Module1 описание константы и удаляет его.

```
Sub DeletePiConst()
   If DeleteWholeLine("Module1", "Const conPi = 3.14") Then
      Debug.Print "Описание константы удалено."
   Else
      Debug.Print "Описание константы не удалено."
   End If
End Sub
```

Метод Find

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthFindC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthFindX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acmthFindA"}

Метод **Find** проводит поиск указанного текста в стандартном модуле или в модуле класса.

Синтаксис

объект.**Find(***образец***,** *строка1***,** *столбец1***,** *строка2***,** *столбец2*[**,** *словоЦеликом***,** *учетРегистра***,** *шаблон*]**)**

Метод **Find** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Module**.
Образец	Строковое выражение, значением которого является текст, который надо найти.
Строка1	Значение типа **Long**, указывающее номер строки модуля, с которой начинается поиск. Если найден текст, совпадающий с образцом поиска, то аргументу *строка1* присваивается номер строки модуля, в которой найден первый символ этого текста.
столбец1	Значение типа **Long**, указывающее номер столбца, с которого начинается поиск. Каждому символу в строке модуля отвечает столбец с определенным номером; столбцы в модуле нумеруются слева направо, начиная с 0. Если найден текст, совпадающий с образцом поиска, то аргументу *столбец1* присваивается номер столбца, в котором найден первый символ этого текста.
строка2	Значение типа **Long**, указывающее номер строки модуля, на которой заканчивается поиск. Если найден текст, совпадающий с образцом поиска, то аргументу *строка2* присваивается номер строки модуля, в которой найден последний символ этого текста.
столбец2	Значение типа **Long**, указывающее номер столбца, на котором заканчивается поиск. Если найден текст, совпадающий с образцом поиска, то аргументу *столбец2* присваивается номер столбца, в котором найден последний символ этого текста.
словоЦеликом	Необязательный аргумент. Переменная типа **Boolean**. Если данный аргумент имеет значение **True** (–1), то проводится поиск только целых слов. По умолчанию для этого аргумента устанавливается значение **False** (0), при этом найденный текст может быть частью другого слова.
учетРегистра	Необязательный аргумент. Переменная типа **Boolean**, значение **True** которой указывает, что поиск проводится с учетом буквенного регистра; используемое по умолчанию значение **False** задает режим поиска без учета регистра.
Шаблон	Необязательный аргумент. Переменная типа **Boolean**, значение **True** которой указывает, что аргумент *образец* может содержать подстановочные знаки, такие как звездочка (*) или восклицательный знак (?); используемое по умолчанию значение **False** задает режим поиска без подстановочных знаков.

Дополнительные сведения

Метод **Find** проводит поиск определенной текстовой строки в объекте **Module**. Если текст найден, то метод **Find** возвращает значение **True**; в противном случае возвращается значение **False**.

Для определения местоположения в модуле найденного текста в метод **Find** передаются пустые переменные для аргументов *строка1*, *столбец1*, *строка2* и *столбец2*. При успешном выполнении поиска эти аргументы получают значения, совпадающие с номерами в модуле строки и столбца, которые отвечают первому (*строка1*, *столбец1*) и последнему (*строка2*, *столбец2*) символу этого текста.

Например, если найденный текст находится в строке 5, начинается в столбце 10 и заканчивается в столбце 20, то аргументы получают следующие значения: *строка1* = 5, *столбец1* = 10, *строка2* = 5, *столбец2* = 20.

Методы Find, ReplaceLine, пример

Следующая функция находит в модуле указанный текст и замещает часть строки модуля, совпадающую с этим текстом, другой указанной последовательностью символов, замещая таким образом эту строку модуля новой строкой.

```
Function FindAndReplace(strModuleName As String, strSearchText As String, _
   strNewText As String) As Boolean
   Dim mdl As Module
   Dim lngSLine As Long, lngSCol As Long
   Dim lngELine As Long, lngECol As Long
   Dim strLine As String, strNewLine As String
   Dim intChr As Integer, intBefore As Integer, intAfter As Integer
   Dim strLeft As String, strRight As String

   ' Открывает модуль.
   DoCmd.OpenModule strModuleName
   ' Возвращает ссылку на объект Module.
   Set mdl = Modules(strModuleName)

   ' Поиск текста.
   If mdl.Find(strSearchText, lngSLine, lngSCol, lngELine, lngECol) Then
      ' Сохраняет строку модуля, содержащую найденный текст.
      strLine = mdl.Lines(lngSLine, Abs(lngELine - lngSLine) + 1)
      ' Определяет длину строки.
      intChr = Len(strLine)
      ' Определяет число символов перед найденным текстом.
      intBefore = lngSCol - 1
      ' Определяет число символов после найденного текста.
      intAfter = intChr - CInt(lngECol - 1)
      ' Сохраняет символы слева от найденного текста.

      strLeft = Left$(strLine, intBefore)
      ' Сохраняет символы справа от найденного текста.
      strRight = Right$(strLine, intAfter)
      ' Создает строку, содержащую текст замещения.
      strNewLine = strLeft & strNewText & strRight
      ' Замещает исходную строку модуля.
      mdl.ReplaceLine lngSLine, strNewLine
      FindAndReplace = True
   Else
      MsgBox "Образец поиска не найден."
      FindAndReplace = False
   End If

Exit_FindAndReplace:
   Exit Function

Error_FindAndReplace:

MsgBox Err & ": " & Err.Description
   FindAndReplace = False
   Resume Exit_FindAndReplace
End Function
```

Метод Follow

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthFollowC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthFollowX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthFollowA"}

Метод **Follow** открывает документ или страницу Web, указанные с помощью адреса гиперссылки, содержащегося в элементе управления формы или отчета,

Синтаксис

объект.*гиперссылка*.**Follow**[([*новоеОкно*], [*журнал*], [*сведения*], [*метод*], [*заголовок*])]

Метод **Follow** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Control**, содержащий гиперссылку. Элементы управления подпись, кнопка и рисунок могут содержать гиперссылки.
гиперссылка	Гиперссылка объекта **Control**.
новоеОкно	Переменная типа **Boolean**. Если данный аргумент имеет значение **True** (–1), то документ открывается в новом окне. По умолчанию для этого аргумента устанавливается значение **False** (0), и при этом значении документ открывается в текущем окне.
журнал	Переменная типа **Boolean**. Если этот аргумент имеет значение **True**, то гиперссылка добавляется в папку журнала, а при значении аргумента **False** гиперссылка не добавляется. По умолчанию используется значение **True**.
сведения	Строка или массив данных типа **Byte**, которые содержат дополнительную информацию для перемещения к гиперссылке. Этот аргумент используется, например, для указания параметра поиска в файле .asp или .idc. При работе со средством просмотра Web значение аргумента *сведения* выводится после адреса гиперссылки и отделяется от адреса знаком вопроса (?). При задании значения для аргумента *сведения* знак вопроса не указывается.
метод	Значение типа **Integer**, определяющее способ присоединения аргумента *сведения* к адресу. В качестве значения аргумента *метод* используется одна из следующих встроенных констант.

Константа	Описание
msoMethodGet	Аргумент *сведения* добавляется непосредственно к адресу гиперссылки; в этом случае он представляет собой текстовую строку. Это значение используется по умолчанию.
msoMethodPost	Аргумент *сведения* пересылается на сервер; в этом случае он может быть как строкой, так и массивом типа **Byte**.

заголовок	Строка, содержащая заголовок документа. По умолчанию аргумент *заголовок* представляет собой пустую строку ("").

Дополнительные сведения

Метод **Follow** приводит к выполнению тех же действий, что и выбор гиперссылки с помощью мыши.

Если требуется, чтобы гиперссылка открывалась в ответ на действия пользователя, удобно включить метод **Follow** в процедуру обработки событий. Например, при загрузке пользователем определенной формы будет открываться страница Web, содержащая справочную информацию.

В методе **Follow** не требуется указывать адрес, который является значением свойства **Адрес гиперссылки (HyperlinkAddress)** элемента управления. Необходимо указать только имя элемента управления, содержащего эту гиперссылку. В отличие от этого, при использовании метода **FollowHyperlink** необходимо указывать адрес гиперссылки, по которой выполняется переход.

Метод Follow, пример

В следующем примере задается значение свойства **Адрес гиперссылки** (**HyperlinkAddress**) для кнопки. При загрузке формы пользователем открывается гиперссылка.

Для проверки данного примера создайте новую форму и добавьте в нее кнопку с именем «Command0». Включите следующую процедуру в модуль формы и перейдите в режим формы.

```
Private Sub Form_Load()
   Dim ctl As CommandButton

   Set ctl = Me!Command0
   With ctl
      .Visible = False
      .HyperlinkAddress = "http://www.microsoft.com/"
      .Hyperlink.Follow
   End With
End Sub
```

Метод FollowHyperlink

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthFollowHyperlinkC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthFollowHyperlinkX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acmthFollowHyperlinkA"}

Метод **FollowHyperlink** открывает документ или страницу Web, указанные с помощью адреса гиперссылки.

Синтаксис

[*приложение*.]**FollowHyperlink** *адрес*, [*подадрес*], [*новоеОкно*], [*журнал*], [*сведения*], [*метод*], [*заголовок*]

Метод **FollowHyperlink** использует следующие аргументы.

Аргумент	Описание
приложение	Объект Microsoft Access **Application**. .
адрес	Строковое выражение, значением которого является допустимый адрес гиперссылки.
подадрес	Строковое выражение, значением которого является указанное место документа, определяемого с помощью аргумента *адрес*. По умолчанию используется значение пустая строка ("").
новоеОкно	Переменная типа **Boolean**. Если данный аргумент имеет значение **True** (–1), то документ открывается в новом окне. По умолчанию для этого аргумента устанавливается значение **False** (0), и при этом значении документ открывается в текущем окне.
журнал	Переменная типа **Boolean**. Если этот аргумент имеет значение **True**, то гиперссылка добавляется в папку журнала, а при значении аргумента **False** гиперссылка не добавляется. По умолчанию используется значение **True**.
сведения	Строка или массив данных типа **Byte**, которые содержат дополнительную информацию для перемещения к гиперссылке. Этот аргумент используется, например, для указания параметра поиска в файле .asp или .idc. При работе со средством просмотра Web значение аргумента *сведения* выводится после адреса гиперссылки и отделяется от адреса знаком вопроса (?). При задании значения для аргумента *сведения* знак вопроса не указывается.
метод	Значение типа **Integer**, определяющее способ присоединения аргумента *сведения* к адресу. В качестве значения аргумента *метод* используется одна из следующих встроенных констант.

Константа	Описание
msoMethodGet	Аргумент *сведения* добавляется непосредственно к адресу гиперссылки; в этом случае он представляет собой текстовую строку. Это значение используется по умолчанию.
MsoMethodPost	Аргумент *сведения* пересылается на сервер; в этом случае он может быть как строкой, так и массивом типа **Byte**.

заголовок	Строка, содержащая заголовок документа. По умолчанию аргумент *заголовок* представляет собой пустую строку.

Дополнительные сведения

Метод **FollowHyperlink** позволяет выполнить переход по гиперссылке, которая не содержится в элементе управления. Эта гиперссылка указывается разработчиком или пользователем. Например, разработчик может вывести пользователю приглашение ввести адрес гиперссылки в диалоговом окне, а затем использовать метод **FollowHyperlink** для перехода по этой гиперссылке.

Аргументы *сведения* и *метод* предоставляют дополнительную информацию при перемещении к гиперссылке. Этот аргумент используется, например, для передачи параметров средству поиска узлов. Например, при обращении за сведениями о Microsoft на узел сервера Yahoo, можно вызвать метод **FollowHyperlink** следующим образом:

```
Application.FollowHyperlink "http://search.yahoo.com/bin/search", _
   , , , "p=Microsoft", msoMethodGet
```

Для перехода по гиперссылке, связанной с элементом управления, используется метод **Follow**.

Метод FollowHyperlink, пример

Следующая функция запрашивает у пользователя адрес гиперссылки, а затем выполняет переход по этой гиперссылке:

```
Function GetUserAddress() As Boolean
   Dim strInput As String

   On Error GoTo Error_GetUserAddress
   strInput = InputBox("Введите адрес")
   Application.FollowHyperlink strInput, , True
   GetUserAddress = True

Exit_GetUserAddress:
   Exit Function

Error_GetUserAddress:
   MsgBox Err & ": " & Err.Description
   GetUserAddress = False
   Resume Exit_GetUserAddress
End Function
```

Примером вызова данной функции может служить следующая процедура:

```
Sub CallGetUserAddress()
   If GetUserAddress = True Then
      MsgBox "Выполнен переход по гиперссылке."
   Else
      MsgBox "Переход по гиперссылке невозможен."
   End If
End Sub
```

Метод InsertLines

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthInsertLinesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthInsertLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acmthInsertLinesA"}

Метод **InsertLines** вставляет одну или несколько строк программы в стандартный модуль или в модуль класса.

Синтаксис

объект.**InsertLines** *строка*, *текст*

Метод **InsertLines** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Module**.
строка	Номер строки модуля, перед которой вставляется текст.
Текст	Текст, вставляемый в модуль.

Дополнительные сведения

При использовании метода **InsertLines** содержимое строки модуля, указанной в аргументе *строка*, перемещается вниз.

Если требуется добавить в модуль несколько программных строк, то в текстовой строке, которая передается в аргументе *Текст*, следует отметить концы будущих строк модуля с помощью встроенной константы **vbCrLf**. Наличие в тексте этой константы вызывает возврат каретки и перевод строки.

Нумерация строк в модуле начинается с 1. Для определения числа строк в модуле используется свойство **CountOfLines**.

Метод InsertText

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthInsertTextC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthInsertTextX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthInsertTextA"}

Метод **InsertText** вставляет указанную текстовую строку в стандартный модуль или в модуль класса.

Синтаксис

объект.**InsertText** *текст*

Метод **InsertText** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Module**.
текст	Текст, вставляемый в модуль.

Дополнительные сведения

Если текст вставляется в модуль с помощью метода **InsertText**, Microsoft Access помещает вставленный текст в конце модуля после всех процедур.

Если требуется добавить в модуль несколько программных строк, то в тексте, который передается в аргументе *Текст*, следует отметить концы будущих строк модуля с помощью встроенной константы **vbCrLf**. Наличие в тексте этой константы вызывает возврат каретки и перевод строки.

Для указания строки модуля, перед которой вставляется текст, используется метод **InsertLines**. Если требуется вставить программные строки в раздел описаний модуля, используется метод **InsertLines**, а не **InsertText**.

Примечание. В предыдущих версиях Microsoft Access метод **InsertText** использовался с объектом **Application**. Использование метода **InsertText** объекта **Application** в данной версии также допускается, однако рекомендуется применять вместо него метод **InsertText** объекта **Module**.

Метод InsertText, пример

В следующем примере в стандартный модуль вставляется текст:

```
Function InsertProc(strModuleName) As Boolean
   Dim mdl As Module, strText As String

   On Error GoTo Error_InsertProc
   ' Открывает модуль.
   DoCmd.OpenModule strModuleName
   ' Возвращает ссылку на объект Module.
   Set mdl = Modules(strModuleName)
   ' Присваивает значение строковой переменной.
   strText = "Sub DisplayMessage()" & vbCrLf _
      & vbTab & "MsgBox ""Чушь!""" & vbCrLf _
      & "End Sub"
   ' Вставляет текст в модуль.
   mdl.InsertText strText
   InsertProc = True

Exit_InsertProc:
   Exit Function

Error_InsertProc:
   MsgBox Err & ": " & Err.Description
   InsertProc = False
   Resume Exit_InsertProc
End Function
```

Метод RefreshDatabaseWindow

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthRefreshDatabaseWindowC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthRefreshDatabaseWindowX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthRefreshDatabaseWindowA"}

Метод **RefreshDatabaseWindow** обновляет окно базы данных после создания, удаления или переименования таблицы, запроса, формы, отчета, макроса или модуля.

Синтаксис

[*приложение*.]**RefreshDatabaseWindow**

Метод **RefreshDatabaseWindow** использует следующий аргумент.

Аргумент	Описание
приложение	Объект Microsoft Access **Application**.

Дополнительные сведения

Метод **RefreshDatabaseWindow** используется для немедленного отображения в окне базы данных любых изменений объектов Microsoft Access. Например, если из Visual Basic добавляется новая форма, которая затем сохраняется, то метод **RefreshDatabaseWindow** позволяет вывести имя новой формы на вкладке **Формы** окна базы данных сразу после сохранения формы.

Метод RefreshDatabaseWindow, пример

В следующем примере создается новая форма, затем она сохраняется, и окно базы данных обновляется:

```
Sub CreateFormAndRefresh()
   Dim frm As Form

   Set frm = CreateForm
   DoCmd.Save , "НоваяФорма"
   RefreshDatabaseWindow
End Sub
```

Метод ReplaceLine

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthReplaceLineC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acmthReplaceLineX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS":"acmthReplaceLineA"}

Метод **ReplaceLine** замещает указанную строку стандартного модуля или модуля класса.

Синтаксис

объект.**ReplaceLine** *строка*, *текст*

Метод **ReplaceLine** использует следующие аргументы.

Аргумент	Описание
объект	Объект **Module**.
Строка	Значение типа **Long**, указывающее номер строки модуля, которая будет замещена.
Текст	Текст, замещающий существующую строку модуля.

Дополнительные сведения

Нумерация строк в модуле начинается с 1. Для определения числа строк в модуле используется свойство **CountOfLines**.

Метод RunCommand

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthRunCommandC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acmthRunCommandX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acmthRunCommandA"}

Метод **RunCommand** выполняет команду встроенного меню или встроенной панели инструментов.

Синтаксис

[*объект*.]**RunCommand** *команда*

Метод **RunCommand** использует следующие аргументы.

Аргумент	Описание
объект	Необязательный аргумент. Объект **Application** или объект **DoCmd**.
Команда	Встроенная константа, указывающая, какая именно команда встроенного меню или панели инструментов выполняется.

Дополнительные сведения

Каждой команде меню или панели инструментов Microsoft Access соответствует константа, которая используется в методе **RunCommand** для выполнения этой команды в программе Visual Basic.

Перечень всех констант для аргумента *команда* содержится в разделе Константы метода **RunCommand**. Для того, чтобы вывести константы в окне просмотра объектов, выберите **Access** в списке **Проекты/библиотеки**, затем выберите **AcCommand** в списке **Классы**. Константы выводятся в списке **Компонент**.

Не допускается применение метода **RunCommand** для выполнения команды специального меню или специальной панели инструментов. Он используется только для встроенных меню или панелей инструментов.

Метод **RunCommand** заменяет метод **DoMenuItem** объекта **DoCmd**.

Метод RunCommand, пример

В следующем примере метод **RunCommand** используется для открытия диалогового окна **Параметры** (окно выводится при выборе команды **Параметры** в меню **Сервис**):

```
Function OpenOptionsDialog() As Boolean
   On Error GoTo Error_OpenOptionsDialog
   DoCmd.RunCommand acCmdOptions
   OpenOptionsDialog = True

Exit_OpenOptionsDialog:
   Exit Function

Error_OpenOptionsDialog:
   MsgBox Err & ": " & Err.Description
   OpenOptionsDialog = False
   Resume Exit_OpenOptionsDialog
End Function
```

Функция GUIDFromString

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctGUIDFromStringC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acfctGUIDFromStringX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acfctGUIDFromStringA"}

Функция **GUIDFromString** преобразует текстовую строку в код GUID, который является массивом типа **Byte**.

Синтаксис

GUIDFromString(*выражение***)**

Функция **GUIDFromString** использует следующий аргумент.

Аргумент	Описание
выражение	Строковое выражение, значением которого является код GUID в форме строки.

Дополнительные сведения

В ядре базы данных Microsoft Jet коды GUID хранятся в виде массивов типа **Byte**. Однако Microsoft Access не возвращает данные типа **Byte** из элементов управления в формах или отчетах. Для возвращения некоторого значения GUID из элемента управления его необходимо преобразовать в строку. Для преобразования кода GUID в строку используется функция **StringFromGUID**. Для обратного преобразования используется функция **GUIDFromString**.

Функция GUIDFromString, пример

В следующем примере функция **GUIDFromString** используется для преобразования строки в код GUID. Эта строка представляет собой код GUID, сохраняемый в виде последовательности символов в реплицируемой таблице «Сотрудники». Поле s_GUID является скрытым полем, которое добавляется к любой реплицируемой таблице в реплицируемой базе данных.

```
Sub CheckGUIDType()
   Dim dbs As Database, rst As Recordset

   Set dbs = CurrentDb
   Set rst = dbs.OpenRecordset("Сотрудники")
   Debug.Print rst!s_GUID
   Debug.Print TypeName(rst!s_GUID)
   Debug.Print TypeName(GuidFromString(rst!s_GUID))
End Sub
```

Функция StringFromGUID

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctStringFromGUIDC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acfctStringFromGUIDX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acfctStringFromGUIDA"}

Функция **StringFromGUID** преобразует код GUID, который представляет собой массив типа **Byte**, в текстовую строку.

Синтаксис

StringFromGUID(*GUID***)**

Функция **StringFromGUID** использует следующий аргумент.

Аргумент	Описание
GUID	Массив типа **Byte**, используемый для однозначной идентификации приложения, компонента или элемента данных в операционной системе.

Дополнительные сведения

В ядре базы данных Microsoft Jet коды GUID хранятся в виде массивов типа **Byte**. Однако Microsoft Access не возвращает данные типа **Byte** из элементов управления в формах или отчетах.. Для возвращения некоторого значения GUID из элемента управления его необходимо преобразовать в строку. Для преобразования кода GUID в строку используется функция **StringFromGUID**. Для обратного преобразования используется функция **GUIDFromString**.

Например, иногда необходимо обратиться к полю, содержащему код GUID, при репликации базы данных. Для возвращения в форме значения из элемента управления, присоединенного к полю кода GUID, используется функция **StringFromGUID**, преобразующая код GUID в строку.

Для того, чтобы присоединить элемент управления к полю s_GUID реплицируемой таблицы, следует выбрать команду **Параметры** в меню **Сервис** и установить флажок **Системные объекты** на вкладке **Вид** диалогового окна **Параметры**.

Функция StringFromGUID, пример

В следующем примере из элемента управления s_GUID в форме «Сотрудники» возвращается код GUID в виде строки символов, которая присваивается строковой переменной. Элемент управления s_GUID присоединен к полю s_GUID, одному из полей операционной системы, которое добавляется к любой реплицированной таблице в реплицированной базе данных.

```
Sub StringValueOfGUID()
   Dim ctl As Control, strGUID As String

   Set ctl = Forms!Сотрудники!s_GUID
   Debug.Print TypeName(ctl.Value)
   strGUID = StringFromGUID(ctl.Value)
   Debug.Print TypeName(strGUID)
End Sub
```

Метод Add (семейство Pages)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthAddPagesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthAddPagesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthAddPagesA"}

Метод **Add** добавляет новый объект **Page** в семейство **Pages** элемента управления «Набор вкладок».

Синтаксис

объект.**Add** [*номер*]

Метод **Add** использует следующие аргументы.

Аргумент	Описание
объект	Семейство **Pages** элемента управления «Набор вкладок».
номер	Целое значение, указывающее индекс объекта **Page**, перед которым добавляется новый объект **Page**. Индекс объекта **Page** соответствует значению свойства этого объекта **PageIndex**. Если данный аргумент опущен, новый объект **Page** добавляется в конце семейства.

Дополнительные сведения

Первый объект **Page** в семействе **Pages** отвечает крайней левой странице в элементе управления «Набор вкладок» и имеет индекс 0. Второй объект **Page** находится непосредственно справа от первой страницы и имеет индекс 1. Индексация продолжается в этом порядке для всех объектов **Page** элемента управления «Набор вкладок».

Если для аргумента *индекс* указано значение 0, то новый объект **Page** добавляется перед первым объектом семейства **Pages**. В этом случае новый объект **Page** становится первым объектом в семействе и приобретает индекс 0.

Добавление объекта **Page** в семейство **Pages** элемента управления «Набор вкладок» возможно только в том случае, если форма открыта в режиме конструктора.

Методы Add, Remove (семейство Pages), пример

В следующем примере добавляется страница в элемент управления «Набор вкладок» в форме, открытой в режиме конструктора. Для проверки данного примера создайте новую форму с именем «Form1» и элемент управления «Набор вкладок» с именем TabCtl0. Вставьте следующую функцию в стандартный модуль и выполните модуль.

```
Function AddPage() As Boolean
   Dim frm As Form
   Dim tbc As TabControl, pge As Page

   On Error GoTo Error_AddPage
   Set frm = Forms!Form1
   Set tbc = frm!TabCtl0
   tbc.Pages.Add
   AddPage = True

Exit_AddPage:
   Exit Function

Error_AddPage:
   MsgBox Err & ": " & Err.Description
   AddPage = False
   Resume Exit_AddPage
End Function
```

В следующем примере удаляются страницы из элемента управления «Набор вкладок».

```
Function RemovePage() As Boolean
   Dim frm As Form
   Dim tbc As TabControl, pge As Page

   On Error GoTo Error_RemovePage
   Set frm = Forms!Form1
   Set tbc = frm!TabCtl0
   tbc.Pages.Remove
   RemovePage = True

Exit_RemovePage:
   Exit Function

Error_RemovePage:
   MsgBox Err & ": " & Err.Description
   RemovePage = False
   Resume Exit_RemovePage
End Function
```

Метод Remove (семейство Pages)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acmthRemovePagesC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acmthRemovePagesX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acmthRemovePagesA"}

Метод **Remove** удаляет объект **Page** из семейства **Pages** элемента управления «Набор вкладок».

Синтаксис

объект.**Remove** [*элемент*]

Метод **Remove** использует следующие аргументы.

Аргумент	Описание
объект	Строковое выражение, значением которого является семейство **Pages** элемента управления «Набор вкладок».
Элемент	Целое значение, указывающее индекс удаляемого объекта **Page**. Индекс объекта **Page** соответствует значению свойства этого объекта **PageIndex**. Если данный аргумент опущен, удаляется последний объект семейства.

Дополнительные сведения

Индексация в семействе **Pages** начинается с 0. Крайняя левая страница в элементе управления «Набор вкладок» имеет индекс 0. Страница, расположенная непосредственно справа от крайней левой страницы имеет индекс 1, и т.д.

Удаление объекта **Page** из семейства **Pages** элемента управления «Набор вкладок» возможно только в том случае, если форма открыта в режиме конструктора.

Метод Remove (семейство References)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acmthRemoveReferencesC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acmthRemoveReferencesX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acmthRemoveReferencesA"}

Метод **Remove** удаляет объект **Reference** из семейства **References**.

Синтаксис

объект.**Remove** *ссылка*

Метод **Remove** использует следующие аргументы.

Аргумент	**Описание**
объект	Строковое выражение, значением которого является семейство **References**.
Ссылка	Объект **Reference**, представляющий удаляемую ссылку.

Дополнительные сведения

Для определения имени удаляемого объекта **Reference** проверьте список **Проекты/библиотеки** в окне просмотра объектов. В этом окне выводятся имена всех ссылок, установленных в данный момент. Эти имена соответствуют значениям свойства **Имя (Name)** объекта **Reference**.

Метод Remove (семейство References), пример

Первая из двух следующих функций добавляет ссылку на элемент управления-календарь в семейство **References**. Вторая функция удаляет ссылку на элемент управления-календарь.

```
Function AddReference() As Boolean
   Dim ref As Reference, strFile As String

   On Error GoTo Error_AddReference
   strFile = "C:\Windows\System\Mscal.ocx"
   ' Создает ссылку на элемент управления-календарь.
   Set ref = References.AddFromFile(strFile)
   AddReference = True

Exit_AddReference:
   Exit Function

Error_AddReference:
   MsgBox Err & ": " & Err.Description
   AddReference = False
   Resume Exit_AddReference
End Function

Function RemoveReference() As Boolean
   Dim ref As Reference

   On Error GoTo Error_RemoveReference
   Set ref = References!MSACAL
   ' Удаляет ссылку на элемент управления-календарь.
   References.Remove ref
   RemoveReference = True

Exit_RemoveReference:
   Exit Function

Error_RemoveReference:
   MsgBox Err & ": " & Err.Description
   RemoveReference = False
   Resume Exit_RemoveReference
End Function
```

Объект DoCmd

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acobjDoCmdC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjDoCmdX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acobjDoCmdP"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjDoCmdM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjDoCmdE"}

Методы, определенные для объекта **DoCmd** (команда), позволяют запускать макрокоманды Microsoft Access из программ Visual Basic. С помощью макрокоманд выполняют такие действия как закрытие окон, открытие форм и задание значений элементов управления. Например, метод **OpenForm** объекта **DoCmd** позволяет открыть форму, а метод **Hourglass** изменить вид указателя на значок Windows «Занято» (песочные часы).

Примечание Объект **DoCmd** служит для тех же целей, что и инструкция **DoCmd** версий 1.*x* и 2.0 Microsoft Access. Действия, используемые ранее как аргументы инструкции **DoCmd**, стали теперь методами объекта **DoCmd**. Например, для открытия формы из Access Basic в Microsoft Access 2.0 могла быть использована следующая инструкция `DoCmd OpenForm "Заказы"`. В Microsoft Access 97 аналогичное действие производится с помощью следующей команды:

```
DoCmd.OpenForm "Заказы"
```

Синтаксис

[*приложение*.]**DoCmd.***метод* [*арг1, арг2, ...*]

Синтаксис конструкций для объекта **DoCmd** требует указания следующих аргументов.

Аргумент	Описание
приложение	Необязательный аргумент. Определяет объект **Application.**
метод	Имя одного из методов, поддерживаемых объектом.
арг1, арг2, ...	Аргументы указанного метода. Данные аргументы совпадают с аргументами соответствующей макрокоманды.

Дополнительные сведения

Большинство из методов, определенных для объекта **DoCmd**, имеют аргументы, некоторые из которых являются обязательными, а другие необязательными. Если необязательный аргумент опущен, то при выполнении макрокоманды подразумевается значение данного аргумента по умолчанию. Например, метод **OpenForm** использует семь аргументов, но только первый из них, *имяФормы*, является обязательным. Следующий пример демонстрирует конструкцию, открывающую форму «Сотрудники» из текущей базы данных, причем отбираются только записи о сотрудниках, имеющих должность «Торговый агент».

```
DoCmd.OpenForm "Сотрудники", , ,"[Должность] = 'Торговый агент'"
```

Объект **DoCmd** не поддерживает методы, соответствующие следующим макрокомандам:

- **ДобавитьМеню (AddMenu)**
- **Сообщение (MsgBox)** – Используйте функцию **MsgBox.**
- **ЗапускПриложения (RunApp)**. Для запуска другого приложения используйте функцию **Shell.**

- **ЗапускПрограммы (RunCode)** – Вызывайте программу в Visual Basic.
- **КомандыКлавиатуры (SendKeys)** – Используйте инструкцию **SendKeys.**
- **ЗадатьЗначение (SetValue)** – Задавайте значения в конструкциях Visual Basic.
- **ОстановитьВсеМакросы (StopAllMacros)**
- **ОстановитьМакрос (StopMacro)**

Для того чтобы вывести описание макрокоманд Microsoft Access, соответствующих методам объекта **DoCmd**, следует выбрать имя макрокоманды в предметном указателе справочной системы.

Объект DoCmd, пример

Следующая процедура открывает форму в режиме формы и осуществляет переход на новую запись.

```
Sub ShowNewRecord()
   DoCmd.OpenForm "Сотрудники", acNormal
   DoCmd.GoToRecord , , acNewRec
End Sub
```

Метод OpenCurrentDatabase

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅS. пїЅпїЅпїЅпїЅпїЅS":"acmthOpenCurrentDatabaseC;damthOpenDatabase"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"acmthOpenCurrentDatabaseX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acmthOpenCurrentDatabaseA"}

Метод **OpenCurrentDatabase** открывает существующую базу данных и делает ее текущей. Данный метод используется для открытия базы данных Microsoft Access из другого приложения с помощью механизма программирования объектов, ранее называемого механизмом управления объектами OLE. Например, метод **OpenCurrentDatabase** позволяет открыть из Microsoft Excel демонстрационную базу данных «Борей» в окне Microsoft Access.

Синтаксис

приложение.**OpenCurrentDatabase** *базаДанных*[, *монопольный*]

Метод **OpenCurrentDatabase** использует следующие аргументы.

Аргумент	Описание
приложение	Определяет объект **Application.**
базаДанных	Строковое выражение, задающее имя существующего файла базы данных, включая путь и расширение. Если поддерживается сеть, то сетевой маршрут может быть задан в следующем виде: *****Сервер******Ресурс******Папка******ИмяФайла***.mdb**
монопольный	Необязательный аргумент типа **Boolean**. Логическое значение, указывающее, следует ли открыть базу данных в режиме монопольного доступа. По умолчанию, присваивается значение **False**, и база данных открывается в режиме общего доступа.

Дополнительные сведения

Метод **OpenCurrentDatabase** позволяет открывать базы данных Microsoft Access из другого приложения с помощью механизма программирования объектов. После того как экземпляр Microsoft Access открыт из другого приложения, пользователь имеет также возможность создать новую базу данных или открыть существующую. Эта база данных будет открыта в окне Microsoft Access.

Если уже имеется база данных, открытая в окне Microsoft Access, и требуется открыть другу базу данных, следует перед открытием второй базы данных закрыть первую базу данных с помощью метода **CloseCurrentDatabase.**

Для того чтобы открыть базу данных в режиме монопольного доступа, следует задать для аргумента *монопольный* значение **True**. Если данный аргумент опущен, база данных открывается для общего доступа.

Примечание. Не следует путать метод **OpenCurrentDatabase** с методом объектов доступа к данным **OpenDatabase.** Метод **OpenCurrentDatabase** открывает базу данных в окне Microsoft Access. Метод **OpenDatabase** возвращает объектную переменную типа **Database**, представляющую конкретную базу данных, но в действительности не открывает эту базу данных в окне Microsoft Access.

Метод OpenCurrentDatabase, пример

В данном примере база данных Microsoft Access открывается из другого приложения с помощью механизма программирования объектов, а затем открывается форма в этой базе данных.

Следующая программа предназначена для включения в модуль Visual Basic любого приложения, выполняющего роль компонента ActiveX. Например, эту программу можно запустить из Microsoft Excel, из Microsoft Visual Basic или даже из Microsoft Access.

После того как объектная переменная, представляющая объект **Application**, выходит из области определения, экземпляр Microsoft Access, который представляет эта переменная, будет закрыт. Это означает, что данную переменную следует описать на уровне модуля.

```
' Следующую инструкцию необходимо включить в раздел описаний модуля.
Dim appAccess As Access.Application

Sub DisplayForm()
   Const strConPathToSamples = "C:\Program " _
      & "Files\MSOffice\Access\Samples\"

   Dim strDB As String

   ' Инициализирует строку, указывающую путь к базе данных.
   strDB = strConPathToSamples & "Борей.mdb"
   ' Создает новый экземпляр Microsoft Access.
   Set appAccess = _
      CreateObject("Access.Application.8")
   ' Создает новую базу данных в окне Microsoft Access.
   appAccess.OpenCurrentDatabase strDB
   ' Открывает форму "Заказы".
   appAccess.DoCmd.OpenForm "Заказы"
End Sub
```

Примечание. Некоторые приложения, например, Microsoft Visual Basic 4.0, допускают использование ключевого слова **New** при описании объектной переменной типа **Application**. Данное ключевое слово указывает автоматическое создание нового экземпляра Microsoft Access без необходимо дополнительного вызова функции **CreateObject**. Для того чтобы проверить, поддерживает ли используемое приложение данный синтаксис, следует обратиться к документации приложения.

Метод NewCurrentDatabase

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthNewCurrentDatabaseC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthNewCurrentDatabaseX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acmthNewCurrentDatabaseA"}

Метод **NewCurrentDatabase** создает новую базу данных в окне Microsoft Access. Данный метод используется для создания базы данных Microsoft Access из другого приложения с помощью механизма программирования объектов, ранее называемого механизмом управления объектами OLE. Например, можно вызвать метод **NewCurrentDatabase** из Microsoft Excel для создания новой базы данных в окне Microsoft Access.

Синтаксис

*приложение***.NewCurrentDatabase** *базаДанных*

Метод **NewCurrentDatabase** использует следующие аргументы.

Аргумент	Описание
приложение	Определяет объект **Application.**
базаДанных	Строковое выражение, задающее имя нового файла базы данных. Если имя файла имеет расширение, необходимо его указать. Для сетей, допускающих указание сетевого пути, можно указать сетевой путь следующим образом: *****Server******Share******Folder******Filename***.mdb**

Дополнительные сведения

Метод **NewCurrentDatabase** позволяет создать новую базу данных Microsoft Access из другого приложения с помощью механизма программирования объектов. После того как экземпляр Microsoft Access создан из другого приложения, пользователь имеет также возможность создавать новые базы данных или открывать существующие. Эти базы данных будут открываться в окне Microsoft Access.

Указание в аргументе *базаДанных* имени существующего файла приводит к ошибке.

Новая база данных открывается с помощью учетной записи пользователя «Admin».

Метод NewCurrentDatabase, пример

В данном примере новая база данных Microsoft Access создается из другого приложения с помощью механизма программирования объектов, а затем создается новая таблица в этой базе данных.

Следующая программа предназначена для включения в модуль Visual Basic любого приложения, выполняющего роль компонента ActiveX. Например, эту программу можно запустить из Microsoft Excel, из Microsoft Visual Basic или даже из Microsoft Access.

После того как объектная переменная, представляющая объект **Application**, выходит из области определения, экземпляр Microsoft Access, который представляет эта переменная, будет закрыт. Это означает, что данную переменную следует описать на уровне модуля.

```
' Следующую инструкцию необходимо включить в раздел описаний модуля.
Dim appAccess As Access.Application

Sub NewAccessDatabase()
   Dim dbs As Database, tdf As TableDef, fld As Field
   Dim strDB As String

   ' Инициализирует строку, указывающую путь к базе данных.
   strDB = "C:\Мои документы\Newdb.mdb"
   ' Создает новый экземпляр Microsoft Access.
   Set appAccess = _
      CreateObject("Access.Application.8")
   ' Открывает базу данных в окне Microsoft Access.
   appAccess.NewCurrentDatabase strDB
   ' Определяет переменную типа Database.
   Set dbs = appAccess.CurrentDb
   ' Создает новую таблицу.
   Set tdf = dbs.CreateTableDef("Контакты")
   ' Создает поле в новой таблице.
   Set fld = tdf. _
      CreateField("Компания", dbText, 40)
   ' Добавляет объекты Field и TableDef в семейства.
   tdf.Fields.Append fld
   dbs.TableDefs.Append tdf
   Set appAccess = Nothing
End Sub
```

Примечание. Некоторые приложения, например, Microsoft Visual Basic 4.0, допускают использование ключевого слова **New** при описании объектной переменной типа **Application**. Данное ключевое слово указывает автоматическое создание нового экземпляра Microsoft Access без необходимо дополнительного вызова функции **CreateObject**. Для того чтобы проверить, поддерживает ли используемое приложение данный синтаксис, следует обратиться к документации приложения.

Метод Run

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ.
пїЅпїЅпїЅпїЅпїЅS":"acmthRunC;vaconUnderstandingOleAutomation;vaconWorkingAcross;vafctCreateObject;vafctGetObject;vastmSet"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"acmthRunX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acmthRunA"}

Метод **Run** запускает определенную в Microsoft Access или определяемую пользователем процедуру-функцию (**Function**) или процедуру-подпрограмму (**Sub**). Данный метод используется при вызове Microsoft Access из другого приложения с помощью механизма программирования объектов, ранее называемого механизмом управления объектами OLE. Например, метод **Run** вызывают в компоненте ActiveX для запуска процедуры **Sub**, определенной в базе данных Microsoft Access.

Синтаксис

приложение.**Run** *процедура*[*, арг1, арг2, ..., арг30*]

Метод **Run** использует следующие аргументы.

Аргумент	Описание
приложение	Объект **Application**.
процедура	Имя запускаемой процедуры типа **Function** или **Sub**. При вызове процедуры из другой базы данных необходимо указывать имя проекта и имя процедуры, разделяя их точками: "*имяПроекта*.*имяПроцедуры*" При запуске программы Visual Basic, содержащей метод **Run** в библиотечной базе данных поиск процедуры будет проводиться сначала в библиотечной базе, а затем в текущей базе данных.
арг1, *арг2*, ...	Необязательные аргументы. Аргументы для заданной процедуры типа **Function** или **Sub**. Допускается определение до тридцати аргументов.

Дополнительные сведения

В любом компоненте ActiveX можно определить ссылку на библиотеку типов Microsoft Access и использовать определенные в этой библиотеке объекты, методы и свойства в собственных программах. Однако невозможно установить ссылку на конкретную базу данных Microsoft Access из любого другого приложения, не являющегося приложением Microsoft Access. В таких ситуациях метод **Run** позволяет вызвать процедуру, определенную в Microsoft Access, из другого приложения.

Предположим, например, что в базе данных, для которой значением свойства **ProjectName** является «WizCode», определена процедура с именем «НоваяФорма», принимающая строковый аргумент. В Visual Basic процедура «НоваяФорма» вызывается следующим образом:

```
Dim appAccess As New Access.Application
appAccess.OpenCurrentDatabase ("C:\Мои документы\Мастера.mdb")
appAccess.Run "Мастера.НоваяФорма", "АргСтрока"
```

Если возможно существование в другой базе данных процедуры с тем же именем, необходимо указать в аргументе *процедура* имя базы данных, в которой находится нужная процедура.

Метод **Run** позволяет также вызывать из другой базы данных процедуру, находящуюся в адресуемой базе данных Microsoft Access.

Microsoft Access игнорирует любое значение, возвращаемое процедурой, вызванной в методе **Run**.

Метод Run, пример

В данном примере определяемая пользователем процедура **Sub**, находящаяся в модуле базы данных Microsoft Access, запускается из другого приложения, выполняющего роль компонента ActiveX.

Для выполнения данного примера создайте новую базу данных с именем WizCode.mdb и задайте для свойства **ProjectName** значение «WizCode». Откройте в этой базе данных новый модуль. Включите в модуль следующую программу, сохраните модуль и закройте базу данных.

```
Sub Приветствие(strName As String)
   MsgBox("Привет, " & strName)
End Sub
```

После этого запустите следующую программу из Microsoft Excel или Microsoft Visual Basic. Перед запуском необходимо создать ссылку на библиотеку типов Microsoft Access. Для этого выберите в меню **Сервис** команду **Ссылки** и выберите **Microsoft Access 8.0 Object** в диалоговом окне **Ссылки**.

```
' Следующую инструкцию необходимо включить в раздел описаний модуля.
Dim appAccess As Access.Application

Sub RunAccessSub()
   ' Создает экземпляр объекта Application для Microsoft Access.
   Set appAccess = _
      CreateObject("Access.Application.8")
   ' Открывает базу данных WizCode в окне Microsoft Access.
   appAccess.OpenCurrentDatabase "C:\Мои документы\WizCode.mdb", False
   ' Запускает процедуру Sub.
   appAccess.Run "Приветствие", "Герман"
   Set appAccess = Nothing
End Sub
```

Метод Dropdown

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthDropdownC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthDropdownX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthDropdownA"}

Метод **Dropdown** раскрывает список в указанном поле со списком. Например, метод **Dropdown** позволяет раскрыть список в поле со списком кодов поставщиков в тот момент, когда элемент управления получает фокус при вводе данных.

Синтаксис

полеСоСписком.**Dropdown**

Метод **Dropdown** использует следующий аргумент.

Аргумент	Описание
полеСоСписком	Имя элемента управления поля со списком в форме или отчете.

Дополнительные сведения

Если указанный элемент управления не имеет фокуса, возникает ошибка. Вызов данного метода аналогичен нажатию клавиши F4 пользователем в тот момент, когда поле со списком имеет фокус.

Метод Dropdown, пример

В данном примере демонстрируется использование метода **Dropdown** с процедурой обработки события **Получение фокуса (GotFocus)** для раскрытия списка в элементе управления «Сотрудники» при получении фокуса этим элементом.

```
Private Sub Сотрудники_GotFocus()
   Me!Сотрудники.Dropdown
End Sub
```

Метод Undo

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acmthUndoC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthUndoX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthUndoA"}

Метод **Undo** восстанавливает то состояние элемента управления или формы, которое они имели до занесения последних изменений. Например, метод **Undo** позволяет отменить в форме занесение записи, содержащей недопустимые значения.

Синтаксис

объект.**Undo**

Метод **Undo** использует следующий аргумент

Аргумент	Описание
объект	Объект **Form** или объект **Control.**

Дополнительные сведения

Если данный метод применяется к форме, все изменения в текущей записи теряются. Применение метода **Undo** к элементу управления приводит к отмене изменений только в элементе управления.

Применять метод **Undo** следует до обновления формы или элемента управления. Подходящим местом для использования данного метода являются процедуры обработки события формы **До обновления (BeforeUpdate)** или события **Изменение (Change)** элемента управления.

Выполнение метода **Undo** аналогично использованию инструкции **SendKeys** для передачи нажатия клавиши ESC в процедуру обработки события.

Метод Undo, примеры

В данном примере демонстрируется использование метода **Undo** в процедуре обработки события **Изменение (Change)** для отмены изменений в поле «Фамилия», если это поле было изменено.

```
Private Sub Фамилия_Change()
   Me!Фамилия.Undo
End Sub
```

В следующем примере метод **Undo** используется для отмены всех изменений формы до обновления формы.

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
   Me.Undo
End Sub
```

Семейство ItemsSelected

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"accolItemsSelectedC;vastmForEach"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"accolItemsSelectedX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"accolItemsSelectedP;vaproCount"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"accolItemsSelectedM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"accolItemsSelectedE"}

Семейство **ItemsSelected** (выделенные объекты) обеспечивает доступ к данным, содержащихся в выделенных строках списка.

Дополнительные сведения

В отличие от других семейств, семейство **ItemsSelected** содержит не объекты, а значения типа **Variant.** Каждое из значений типа **Variant** представляет целочисленный индекс, указывающий положение выделенной строки в списке или поле со списком.

Для загрузки данных из выделенных строк списка или поля со списком семейство **ItemsSelected** используется вместе со свойством **Column** или свойством **ItemData.** Для перечисления элементов семейства **ItemsSelected** используется конструкция **For Each...Next.**

Например, если в форме имеется список «Сотрудники», то можно с помощью семейства **ItemsSelected** перебрать элементы списка и с помощью свойства **ItemData** возвращать значения присоединенного столбца для каждой выделенной строки списка.

Совет. Для того чтобы разрешить одновременное выделение нескольких строк списка или поля со списком, следует выбрать для свойства **Несвязное выделение** (**MultiSelect**) элемента управления значение «Простой» или «Со связным выбором».

Семейство **ItemsSelected** не имеет методов и имеет одно определенное свойство **Count.**

Семейство ItemsSelected, примеры

В данном примере для каждой выделенной строки в списке «Название» в форме «Контакты» выводится значение из присоединенного столбца. Для выполнения этого примера создайте список, определите для него свойство **Присоединенный столбец (BoundColumn)** и укажите для свойства **Несвязное выделение (MultiSelect)** значение «Простой» или «Со связным выбором». Переключитесь в режим формы, выделите строки списка и запустите следующую программу.

```
Sub BoundData()
   Dim frm As Form, ctl As Control
   Dim varItm As Variant

   Set frm = Forms!Контакты
   Set ctl = frm!Название
   For Each varItm In ctl.ItemsSelected
      Debug.Print ctl.ItemData(varItm)
   Next varItm
End Sub
```

В следующем примере используется тот же список, но выводятся значения всех столбцов для каждой выделенной строки списка.

```
Sub AllSelectedData()
   Dim frm As Form, ctl As Control
   Dim varItm As Variant, intI As Integer

   Set frm = Forms!Контакты
   Set ctl = frm!Название
   For Each varItm In ctl.ItemsSelected
      For intI = 0 To ctl.ColumnCount - 1
         Debug.Print ctl.Column(intI, varItm)
      Next intI
      Debug.Print
   Next varItm
End Sub
```

Объект Application

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acobjApplicationC;daobjDBEngine;vaconCreatingObjVar;vaconUnderstandingOleAutomation;vafctCreateObject;vafctGetObject"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjApplicationX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acobjApplicationP;ofproCommandBar"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjApplicationM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjApplicationE"}

Объект **Application** представляет активное приложение Microsoft Access.

Дополнительные сведения

Объект **Application** (приложение) содержит все объекты Microsoft Access и семейства, в том числе семейство **Forms**, семейство **Reports**, семейство **Modules,** семейство **References**, объект **Screen** и объект **DoCmd**.

Объект **Application** используется для вызова методов или определения свойств, относящихся ко всему приложению Microsoft Access. Например, метод **SetOption** объекта **Application** позволяет определить из программы Visual Basic общие параметры базы данных. В следующем примере продемонстрирован способ установки флажка **Строка состояния** в группе **Отображение на экране** на вкладке **Вид** диалогового окна **Параметры**.

```
Application.SetOption "Строка состояния", True
```

Microsoft Access является компонентом ActiveX, поддерживающим программирование объектов, ранее называемое программированием OLE. Допускается управление объектами Microsoft Access из другого приложения, также поддерживающего программирование объектов. Такие действия осуществляются с помощью объекта **Application**.

Например, Microsoft Visual Basic является компонентом ActiveX. Пользователь Microsoft Visual Basic имеет возможность открыть базу данных Microsoft Access и выполнять действия с ее объектами. Для этого необходимо сначала создать в Microsoft Visual Basic ссылку на библиотеку объектов Microsoft Access 8.0. Затем следует создать новую копию класса **Application** и связать с ним объектную переменную, как показано ниже:

```
Dim appAccess As New Access.Application
```

В приложениях, не поддерживающих ключевое слово **New**, можно создать новую копию класса **Application** с помощью функции **CreateObject**:

```
Dim appAccess As Object
Set appAccess = CreateObject("Access.Application.8")
```

После создания новой копии класса **Application** становится возможным открытие базы данных или создание новой базы данных с помощью методов **OpenCurrentDatabase** или **NewCurrentDatabase**. Далее следует определить свойства объекта **Application** и вызывать его методы. После возвращения ссылки на объект **DBEngine** с помощью свойства **DBEngine** объекта **Application** разрешается доступ ко всем объектам доступа к данным DAO (Data Access Objects) и семействам с помощью этой ссылки.

Объект **Application** позволяет управлять другими объектами Microsoft Access. Например, с помощью метода **OpenForm** объекта Microsoft Access **DoCmd** можно открыть форму Microsoft

Access из Microsoft Excel:

```
appAccess.DoCmd.OpenForm "Заказы"
```

Для получения дополнительных сведений о создании ссылок и работе с объектами с помощью программирования объектов следует обратиться к документации приложения, которое является компонентом ActiveX.

Объект Application, пример

В данном примере выводятся текущие значения некоторых свойств объекта **Application**, задается значение параметра и осуществляется выход из приложения с сохранением всех объектов:

```
Sub ApplicationInformation()
   ' Выводит имя и тип текущего объекта.
   Debug.Print Application.CurrentObjectName
   Debug.Print Application.CurrentObjectType
   ' Включает параметр "Скрытые объекты" в группе "Отображение на экране".
   Application.SetOption "Скрытые объекты", True
   ' Выход из Microsoft Access с сохранением всех объектов.
   Application.Quit acSaveYes
End Sub
```

В следующем примере демонстрируется использование Microsoft Access в качестве компонента ActiveX. В Microsoft Excel, Visual Basic или другом приложении, являющемся компонентом ActiveX, создайте ссылку на Microsoft Access. Для этого в меню **Сервис** окна модуля следует выбрать команду **Ссылки** и установить флажок **Библиотека объектов Microsoft Access 8.0**. Затем введите следующую программу в модуль Visual Basic этого приложения и вызовите процедуру GetAccessData.

В данном примере имя базы данных и имя отчета передаются в процедуру, которая создает новую копию класса **Application**, открывает базу данных и печатает указанный отчет.

```
' Описывает объектную переменную в разделе описаний модуля.
   Dim appAccess As Access.Application

Sub GetAccessData()
   Dim strDB As String
   Dim strReportName As String

   ' Строка содержит путь базы данных.
   strDB = "C:\Program Files\Microsoft Office\Office\Samples\Борей.mdb"
   ' Строка содержит имя отчета.
   strReportName = "Прейскурант"
   PrintAccessReport strDB, strReportName
End Sub

Sub PrintAccessReport(strDB As String, strReportName As String)
   ' Возвращает ссылку на объект Application.
   Set appAccess = New Access.Application
   ' Открывает базу данных в окне Microsoft Access.
   appAccess.OpenCurrentDatabase strDB
   ' Выводит отчет на печать.
   appAccess.DoCmd.OpenReport strReportName
   MsgBox "По окончании печати " & strReportName & _
      " нажмите OK"
   appAccess.CloseCurrentDatabase
   Set appAccess = Nothing
End Sub
```

Объект Control

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acobjControlC;vaconCreatingObjVar;vakeyMe"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjControlX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acobjControlP"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acobjControlM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjControlE"}

Объект **Control** представляет элемент управления в форме, отчете или в разделе, который присоединен к другому элементу управления или находится внутри него. На следующем рисунке схематически представлена связь объекта **Control** с объектом **Form** или **Report**.

Дополнительные сведения

Все элементы управления в форме или отчете принадлежат к семейству **Controls** соответствующего объекта **Form** или **Report**. Элементы управления определенного раздела принадлежат к семейству **Controls** этого раздела. Элементы управления, находящиеся внутри таких элементов управления, как элемент управления «Набор вкладок» или группа переключателей принадлежат к соответствующим семействам **Controls** этих элементов управления. Элемент управления типа подпись, присоединенный к другому элементу управления, принадлежит к семейству **Controls** этого элемента управления.

При ссылках на конкретные объекты **Control** в семействе **Controls** допускаются как явные, так и неявные ссылки на семейство **Controls**.

```
' Неявная ссылка на элемент НовыеДанные семейства Controls.
Me!НовыеДанные

' Если имя элемента управления содержит пробел:
Me![Новые данные]

' Выполняется несколько медленнее.
Me("НовыеДанные")

' Ссылка на элемент управления по его индексу в семействе.
Me(0)

' Ссылка с использованием семейства подчиненной формы.
Me.ctlSubForm.Controls!НовыеДанные

' Явные ссылки на элемент НовыеДанные семейства Controls.
Me.Controls!НовыеДанные

Me.Controls("НовыеДанные")

Me.Controls(0)
```

Примечание. Представление объекта **Form** или **Report** в программе с помощью ключевого слова **Me** допускается только при ссылках на форму или отчет в программах, находящихся в модуле класса. При ссылках на форму или отчет из стандартного модуля или из модулей других форм и отчетов необходимо указывать полную ссылку.

Каждый объект **Control** обозначается конкретной встроенной константой. Например, константа **acTextBox** описывает элемент управления типа поля, а константа **acCommandButton** —

кнопку. Константа для конкретного элемента управления Microsoft Access задается с помощью свойства **ControlType** этого элемента управления.

Определить тип существующего элемента управления позволяет свойство **ControlType**. Однако для использования элемента управления в программе нет необходимости знать его тип. Достаточно просто представить этот элемент управления с помощью переменной типа **Control**.

Если элемент управления, на который осуществляется ссылка, является встроенным в Microsoft Access и его тип известен, то для его представления используется переменная специального типа. Например, для элемента управления типа поля описывается переменная типа **TextBox**, как показано в следующем примере.

```
Dim txt As TextBox
Set txt = Forms!Сотрудники!Фамилия
```

Примечание. Если элемент управления является элементом управления ActiveX, то для его обозначения следует описать переменную типа **Control**; в этом случае использование переменной специального типа не допускается. Переменная, предназначенная для представления элемента управления неизвестного заранее типа, также описывается с типом **Control**.

Семейство **Controls** элемента управления типа группы может содержать другие элементы управления, такие как переключатель, флажок, выключатель и подпись.

Элемент управления «Набор вкладок» содержит семейство **Pages**, которое представляет собой особый тип семейства **Controls**. Семейство **Pages** содержит объекты **Page** (страница), являющиеся элементами управления. В свою очередь, каждый объект **Page** также содержит семейство **Controls**, включающее все элементы управления на соответствующей странице.

Другие семейства **Controls** содержат объекты **Control**, к которым может быть присоединена подпись. К таким элементам управления относятся поле, группа переключателей, переключатель, флажок, выключатель, поле со списком, список, кнопка, присоединенная рамка объекта и свободная рамка объекта.

Объект Control, семейство Controls, пример

В следующей процедуре перебираются все элементы управления, входящие в семейство **Controls** формы. Процедура вызывается в модуле формы, и в нее передается объект **Form** с помощью ключевого слова **Me**. Если элемент управления является полем, то в процедуре задаются значения его свойств.

```
' Вызов процедуры SetTextBoxProperties.
SetTextBoxProperties Me

Sub SetTextBoxProperties(frm As Form)
   Dim ctl As Control

   ' Перебор всех компонентов семейства Controls.
   For Each ctl In frm.Controls
      ' Проверяет, является ли элемент управления полем.
      If ctl.ControlType = acTextBox Then
         ' Задает свойства элемента управления.
         With ctl
            .SetFocus
            .Enabled = True
            .Height = 400
            .SpecialEffect = 0
         End With
      End If
   Next ctl
End Sub
```

Объект Form

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acobjFormC;vaconCreatingObjVar;vakeyMe"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjFormX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acobjFormP"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acobjFormM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjFormE"}

Объект **Form** представляет конкретную форму Microsoft Access.

Дополнительные сведения

Объект **Form** является компонентом семейства **Forms**, в которое входят все текущие открытые формы. В рамках семейства **Forms** отдельные формы нумеруются с помощью индекса, начинающегося с нуля. Допускаются ссылки на объект **Form** в семействе **Forms** либо по имени соответствующей формы, либо по ее индексу в семействе. При ссылках на определенную форму семейства рекомендуется применять ссылки по имени, поскольку индекс формы в семействе может измениться. Имена, содержащие пробелы, следует заключать в квадратные скобки ([]).

Синтаксис	Пример
Forms!*имяФормы*	Forms!ФормаЗаказы
Forms![*имя формы***]**	Forms![Форма Заказы]
Forms("*имяФормы***")**	Forms("ФормаЗаказы")
Forms(*индекс***)**	Forms(0)

Каждый объект **Form** содержит семейство **Controls**, включающее все элементы управления в форме. При ссылках на конкретные элементы управления в форме допускаются как явные, так и неявные ссылки на семейство **Controls**. Программа с неявными ссылками на объекты семейства **Controls** выполняется быстрее. Ниже демонстрируются два способа ссылки на элемент управления «НовыеДанные» в форме «Заказы»:

```
' Неявная ссылка.
Forms!Заказы!НовыеДанные

' Явная ссылка.
Forms!Заказы.Controls!НовыеДанные
```

В следующем примере демонстрируются два способа ссылки на элемент управления «НовыеДанные» в подчиненной форме `ctlSubForm`, содержащейся в форме «Заказы»:

```
Forms!Заказы.ctlSubForm.Form!Controls.НовыеДанные

Forms!Заказы.ctlSubForm!НовыеДанные
```

Семейство Forms

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ.
пїЅпїЅпїЅпїЅпїЅS":"accolFormsC;dacolDocument;daobjContainer;daobjDocument;vastmForEach"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"accolFormsX":1} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"accolFormsP"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"accolFormsM"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"accolFormsE"}

Семейство **Forms** содержит все формы, открытые в данный момент в базе данных Microsoft Access.

Дополнительные сведения

Семейство **Forms** используется в программах Visual Basic или в выражениях для ссылок на формы, открытые в данный момент. Например, для задания или возвращения значений свойств конкретных форм бывает удобно организовать перебор всех компонентов семейства **Forms**.

Совет. Компоненты семейства обычно перебираются с помощью инструкции **For Each...Next**.

Допускаются ссылки на конкретный объект **Form** в семействе **Forms** либо по имени соответствующей формы, либо по ее индексу в семействе. При ссылках на определенную форму семейства рекомендуется применять ссылки по имени, поскольку индекс формы в семействе может измениться.

Индексация компонентов семейства **Forms** начинается с нуля. Ссылки на форму по индексу имеют вид Forms(0) для первой открытой формы в семействе, Forms(1) для второй и т.д. Если сначала открыта форма Form1, а после нее Form2, то ссылка по индексу на Form2 в семействе **Forms** выглядит как Forms(1). Если затем форма Form1 будет закрыта, то ссылка по индексу на Form2 в семействе **Forms** будет иметь вид Forms(0).

Примечание. Для того чтобы составить перечень всех форм базы данных, как открытых, так и закрытых, следует выполнить перебор компонентов семейства **Documents** объекта **Container** типа Forms. При этом свойство **Name** каждого конкретного объекта **Document** возвращает имя формы.

Не допускается удаление или добавление пользователем объектов **Form** в семействе **Forms**.

Объект Form, семейство Forms, пример

В данном примере создается новая форма и определяются ее свойства.

```
Sub NewForm()
   Dim frm As Form

   ' Создает новую форму.
   Set frm = CreateForm
   ' Задает значения свойств формы.
   With frm
      .RecordSource = "Товары"
      .Caption = "Форма Товары"
      .ScrollBars = 0
      .NavigationButtons = True
   End With
   ' Восстанавливает окно формы.
   DoCmd.Restore
End Sub
```

В следующем примере перебираются компоненты семейства **Forms** и выводятся имена всех форм в этом семействе. Для каждой формы перебираются компоненты семейства **Controls** и выводятся имена всех элементов управления в этой форме.

```
Sub AllOpenForms()
   Dim frm As Form, ctl As Control

   ' Перебирает компоненты семейства Forms.
   For Each frm In Forms
      ' Выводит имя формы.
      Debug.Print frm.Name
      ' Перебирает компоненты семейства Controls каждой формы.
      For Each ctl In frm.Controls
         ' Выводит имя каждого элемента управления.
         Debug.Print ">>>"; ctl.Name
      Next ctl
   Next frm
End Sub
```

В следующем примере семейство **Forms** используется в выражении. Пусть имеется форма с именем «Заказы», содержащая элемент управления «СтранаПолучателя». В примере применяется функция **IIf** для возвращения значения, вычисляемого на основе текущего значения этого элемента управления. Если элемент управления «СтранаПолучателя» содержит пустое значение (**Null**), выражение возвращает пустую строку; в других случаях оно возвращает содержимое этого элемента. Введите следующее выражение в ячейку свойства **Данные (ControlSource)** поля в форме:

```
= IIf(IsNull(Forms!Заказы! СтранаПолучателя), "", Forms!Заказы!
СтранаПолучателя)
```

Объект Report

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acobjReportC;vaconCreatingObjVar;vakeyMe"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjReportX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acobjReportP"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjReportM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjReportE"}

Объект **Report** представляет конкретный отчет Microsoft Access.

Дополнительные сведения

Объект **Report** является компонентом семейства **Reports**, в которое входят все текущие открытые отчеты. В рамках семейства **Reports** отдельные отчеты нумеруются с помощью индекса, начинающегося с нуля. Допускаются ссылки на объект **Report** в семействе **Reports** либо по имени отчета, либо по его индексу в семействе. Имена отчетов, содержащие пробелы, следует заключать в квадратные скобки ([]).

Синтаксис	Пример
Reports!*имяОтчета*	Reports!ОтчетЗаказы
Reports![*имя отчета***]**	Reports![Отчет Заказы]
Reports("*имяОтчета***")**	Reports("ОтчетЗаказы")
Reports(*индекс***)**	Reports(0)

Каждый объект **Report** содержит семейство **Controls**, включающее все элементы управления в отчете. При ссылках на конкретные элементы управления в отчете допускаются как явные, так и неявные ссылки на семейство **Controls**. При неявных ссылках на объекты из семейства **Controls** программы выполняются быстрее. В следующем примере демонстрируются два способа ссылки на элемент управления «НовыеДанные» в отчете «Заказы».

```
' Неявная ссылка.
Reports!Заказы!НовыеДанные

' Явная ссылка.
Reports!Заказы.Controls!НовыеДанные
```

Семейство Reports

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS.
nїSnїSnїSnїSnїS":"accolReportsC;dacolDocument;daobjContainer;daobjDocument;vastmForEach"} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"accolReportsX":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS":"accolReportsP"} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"accolReportsM"} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"accolReportsE"}

Семейство **Reports** содержит все отчеты, открытые в данный момент в базе данных Microsoft Access.

Дополнительные сведения

Семейство **Reports** используется в программах Visual Basic или в выражениях для ссылок на отчеты, открытые в данный момент. Например, для задания или возвращения значений свойств конкретных отчетов бывает удобно организовать перебор всех компонентов семейства.

Совет. Компоненты семейства обычно перебираются с помощью инструкции **For Each...Next**.

Допускаются ссылки на объект **Report** в семействе **Reports** либо по имени отчета, либо по его индексу в семействе.

Индексация компонентов семейства **Reports** начинается с нуля. Ссылки на отчет по индексу имеют вид Reports(0) для первого открытого отчета в семействе, Reports(1) для второго и т.д. Если сначала открыт отчет Report1, а после него Report2, то ссылка по индексу на Report2 в семействе **Reports** выглядит как Reports(1). Если затем отчет Report1 будет закрыт, то ссылка по индексу на Report2 в семействе **Reports** будет иметь вид Reports(0).

Примечание. Для того чтобы составить перечень всех отчетов базы данных, как открытых, так и закрытых, следует выполнить перебор компонентов семейства **Documents** объекта **Container** типа Reports. При этом свойство **Name** каждого конкретного объекта **Document** возвращает имя отчета.

Не допускается удаление или добавление пользователем объектов **Report** в семействе **Reports**.

Объект Report, семейство Reports, пример

В данном примере создается новый отчет и определяются его свойства.

```
Sub NewReport()
   Dim rpt As Report

   ' Возвращает объектную переменную, указывающую на новый объект Report.
   Set rpt = CreateReport
   ' Задает значения свойств нового отчета.
   With rpt
      .RecordSource = "Товары"
      .Caption = "Отчет Товары"
   End With
   ' Восстанавливает окно отчета.
   DoCmd.Restore
End Sub
```

В следующем примере перебираются компоненты семейства **Reports** и выводятся имена всех отчетов в этом семействе. Для каждого отчета перебираются компоненты семейства **Controls** и выводятся имена всех элементов управления в этом отчете.

```
Sub AllOpenReports()
   Dim rpt As Report, ctl As Control

   ' Перебирает компоненты семейства Reports.
   For Each rpt In Reports
      ' Выводит имя каждого отчета.
      Debug.Print rpt.Name
      ' Перебирает компоненты семейства Controls каждой формы.
      For Each ctl In rpt.Controls
         ' Выводит имя каждого элемента управления.
         Debug.Print ">>>"; ctl.Name
      Next ctl
   Next rpt
End Sub
```

Объект Screen

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acobjScreenC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acobjScreenX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjScreenP"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acobjScreenM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acobjScreenE"}

Объект **Screen** (экран) представляет отдельную форму, отчет или элемент управления, которые в текущий момент имеют фокус.

Дополнительные сведения

Объект **Screen** и его свойства используются в программах Visual Basic или в выражениях для ссылок на форму, отчет или элемент управления, которые имеют фокус. Например, объект **Screen** вместе со свойством **ActiveForm** позволяет сослаться на форму, находящуюся в активном окне, когда имя формы неизвестно. Следующая инструкция позволяет вывести имя формы, находящейся в активном окне:

```
MsgBox Screen.ActiveForm.Name
```

Ссылка на объект **Screen** не делает форму, отчет или элемент управления активными. Для того, чтобы активизировать форму, отчет или элемент управления, следует вызвать метод **SelectObject** объекта **DoCmd**.

Если ссылка на объект **Screen** делается в тот момент, когда нет активной формы, отчета или элемента управления, Microsoft Access возвращает ошибку выполнения.Например, если активным является окно стандартного модуля, то пример программы, приведенный выше в данном разделе, возвращает ошибку.

Объект Screen, пример

В данном примере с помощью объекта **Screen** выводится имя формы, находящейся в активном окне, и имя активного элемента управления в этой форме.

```
Sub ActiveObjects()
   Dim frm As Form, ctl As Control

   ' Возвращает объект Form, указывающий на активную форму.
   Set frm = Screen.ActiveForm
   MsgBox frm.Name & " имя активной формы."
   ' Возвращает объект Control, указывающий на
   ' активный элемент управления.
   Set ctl = Screen.ActiveControl
   MsgBox ctl.Name & " имя активного элемента " _
      & "управления."
End Sub
```

Объект Module

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acobjModuleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acobjModuleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjModuleP"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acobjModuleM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acobjModuleE"}

Объект **Module** укаывает на стандартный модуль или на модуль класса.

Дополнительные сведения

В Microsoft Access существуют модули классов, не связанные с объектами, а также модули форм и модули отчетов, связанные с формами и отчетами.

Для определения типа объекта **Module** (стандартный или модуль класса) служит его свойство **Type**.

Семейство **Modules** содержит в себе все открытые объекты **Module** не зависимо от их типа, что дает возможность их группировки и разделения.

Для получения ссылки на отдельный стандартный объект **Module** или модуль класса в семействе **Modules** используются следующие синтаксические конструкции.

Синтаксис	Описание
Modules!*имяМодуля*	Аргумент *имяМодуля* является названием объекта **Module**.
Modules("*имяМодуля***")**	Аргумент *имяМодуля* является названием объекта **Module**.
Modules(*индекс***)**	Аргумент *индекс* является позицией объекта в семействе.

Следующий пример возвращает ссылку на стандартный объект **Module** и присваивает ее переменному объекту:

```
Dim mdl As Module
Set mdl = Modules![Utility Functions]
```

Необходимо заметить, что скобки, обрамляющие имя объекта **Module** необходимы только в том случае, когда это имя включает в себя пробелы.

Следующий пример возвращает ссылку на объект **Module** формы и присваивает ее переменному объекту:

```
Dim mdl As Module
Set mdl = Modules!Form_Employees
```

Для указания на особый модуль формы или отчета также возможно использование свойства **Module** объектов **Form** или **Report**:

Forms!*имяФорым***.Module**

Следующий пример возвращает ссылку на объект **Module**, связанный со служащими, и присваивает ее переменному объекту:

```
Dim mdl As Module
Set mdl = Forms!Служащие.Module
```

После получения ссылки на объект **Module** возможно задание и считывание его свойств и вызов его методов.

Объект Module, семейство Modules, пример

Следующий пример возвращает ссылку на объект **Module** семейства **Modules** и возвращает число строк программы в модуле:

```
Function LinesInModule(strModuleName As String) As Long
   Dim mdl As Module

   On Error GoTo Error_LinesInModule
   ' Открывает модуль.
   DoCmd.OpenModule strModuleName
   ' Возвращает указатель на объект Module.
   Set mdl = Modules(strModuleName)
   ' Возвращает число строк в модуле.
   LinesInModule = mdl.CountOfLines

Exit_LinesInModule:
   Exit Function

Error_LinesInModule:
   MsgBox Err & ": " & Err.Description
   ' В случае ошибки возвращает -1.
   LinesInModule = -1
   Resume Exit_LinesInModule
End Function
```

Семейство Modules

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"accolModulesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"accolModulesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"accolModulesP"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"accolModulesM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"accolModulesE"}

Семейство **Modules** содержит в себе все открытые в базе данных Microsoft Access стандартные модули и модули классов.

Дополнительные сведения

Для нумерации объектов семейства **Modules** используется конструкция **For Each...Next**. Для определения наличия отдельного объекта **Module** в стандартном модуле или в модуле класса служит свойство **Type** объекта **Module**.

Семейство **Modules** содержит в себе все открытые модули не зависимо от того, откомпилированы они или нет, приостановлено их выполнение, или они выполняются.

Семейство **Modules** принадлежит к объекту **Application** Microsoft Access.

Объекты **Module** семейства **Modules** пронумерованы начиная с 0.

Объект Page

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acobjPageC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acobjPageX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS":"acobjPageP"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acobjPageM"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acobjPageE"}

Объект **Page** соответствует вкладке в наборе вкладок.

Дополнительные сведения

Объект **Page** входит в семейство **Pages** элементов управления **Набор вкладок**.

Для получения ссылки на объект **Page** семейства **Pages** используются следующие синтаксические конструкции.

Синтаксис	Описание
Pages!*имяВкладки*	Аргумент *имяВкладки* является названием объекта **Page**.
Pages("*имяВкладки***")**	Аргумент *имяВкладки* является названием объекта **Page**.
Pages(*индекс***)**	Аргумент *индекс* является позицией объекта в семействе.

Создание, перемещение и удаление объектов **Page** и наборов их свойств допустимо в программе Visual Basic и в режиме конструктора формы. Для создания нового объекта **Page** в программе Visual Basic служит метод **Add** семейства **Pages**. Для удаления объекта **Page** следует использовать метод **Remove** семейства **Pages**.

Для создания нового объекта **Page** в режиме конструктора формы необходимо щелкнуть правой кнопкой элемент управления **Набор вкладок** и выбрать из появившегося контекстного меню команду **Вставить вкладку**. Также допустимы копирование и вставка существующих вкладок. В этом случае для задания свойств нового объекта **Page** необходимо использовать окно свойств.

Свойство **PageIndex** возвращает номер объекта **Page** в семействе **Pages**. Свойство **Value** набора вкладок содержит значение, равное значению свойства **PageIndex** текущей вкладки. Данные свойства используются для определения текущей вкладки после перехода на другую, или для изменения последовательности появления вкладок.

Объект **Page** является также одним из типов объекта **Control**. Константой свойства **ControlType** для объекта **Page** является константа **acPage**. Хотя данный объект является элементом управления, объект **Page** принадлежит семейству **Pages**, а не семейству **Controls**. Семейство **Pages** элемента управления набором вкладок является специальным типом семейства **Controls**.

Любой объект **Page** может содержать один или нескольких элементов управления. Элементы управления объекта **Page** принадлежат семейству **Controls** этого объекта. Для работы с элементом управления объекта **Page** необходимо использование ссылки на этот элемент в семействе **Controls** объекта **Page**.

Объект Page, семейство Pages, пример

В следующей процедуре при выборе пользователем вкладки производится перечисление всех элементов управления каждой вкладки набора.

Для выполнения данного примера создайте новую форму с элементом управления **Набор вкладок** TabCtl0 и для его свойства **Изменение (OnChange)** задайте значение [Процедура обработки событий]. Вставьте в модуль формы следующий текст. Перейдите в режим формы и щелкните другую вкладку для перечисления ее элементов управления.

```
Private Sub TabCtl0_Change()
   Dim tbc As Control, pge As Page
   Dim ctl As Control

   ' Возвращает указатель на набор вкладок.
   Set tbc = Me!TabCtl0
   ' Возвращает указатель на текущую страницу.
   Set pge = tbc.Pages(tbc.Value)
   ' Перечисляет элементы управления текущей страницы.
   Debug.Print pge.Name & "Элементы управления:"
   For Each ctl In pge.Controls
      Debug.Print ctl.Name
   Next ctl
   Debug.Print
End Sub
```

Семейство Pages

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"accolPagesC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"accolPagesX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"accolPagesP"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"accolPagesM"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"accolPagesE"}

Семейство **Pages** содержит в себе все объекты **Page** в наборе вкладок.

Дополнительные сведения

Семейство **Pages** является специальным типом семейства **Controls**, принадлежащим элементу управления набора вкладок. Оно содержит объекты **Page**, являющиеся элементами управления. Отличием семейства **Pages** от обычных семейств **Controls** является возможность добавления и удаления объектов **Page** с помощью методов семейства **Pages**.

Для добавления в семейство **Pages** нового объекта **Page** из программы Visual Basic служит метод **Add** этого семейства. Для удаления объекта **Page** – метод **Remove**. Для подсчета числа объектов **Page** – свойство **Count**.

Можно также использовать метод **CreateControl** для добавления объекта **Page** в семейство **Pages** набора вкладок. Для этого необходимо указать имя набора вкладок в аргументе *главный* функции **CreateControl**. Константой свойства **ControlType** для объекта **Page** является константа **acPage**.

Для перечисления объектов семейства **Pages** используется конструкция **For Each...Next**.

Объекты **Page** семейства **Pages** пронумерованы начиная с 0.

Объект Reference

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅS":"acobjReferenceC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acobjReferenceX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acobjReferenceP"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acobjReferenceM"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acobjReferenceE"}

Объект **Reference** указывает на библиотеку типов другого приложения или проекта.

Дополнительные сведения

Создание объекта **Reference** сопровождается динамической установкой указателя из программы Visual Basic.

Объект **Reference** входит в семейство **References**. Для получения ссылки на объект **Reference** семейства **References** используются следующие синтаксические конструкции.

Синтаксис	Описание
References!*имяСсылки*	Аргумент *имяСсылки* является названием объекта **Reference**.
References("*имяСсылки***")**	Аргумент *имяСсылки* является названием объекта **Reference**.
References(*индекс***)**	Аргумент *индекс* является позицией объекта в семействе.

Следующий пример возвращает ссылку на объект **Reference**, представляющую указатель на библиотеку типов Microsoft Access:

```
Dim ref As Reference
Set ref = References!Access
```

Объект Reference, семейство References, пример

Следующий пример создает ссылку на указанную библиотеку типов:

```
Function ReferenceFromFile(strFileName As String) As Boolean
   Dim ref As Reference

   On Error GoTo Error_ReferenceFromFile
   Set ref = References.AddFromFile(strFileName)
   ReferenceFromFile = True

Exit_ReferenceFromFile:
   Exit Function

Error_ReferenceFromFile:
   MsgBox Err & ": " & Err.Description
   ReferenceFromFile = False
   Resume Exit_ReferenceFromFile
End Function
```

Для вызова данной функции допустимо использование процедуры, сходной с нижеследующей, создающей ссылку на элемент управления **Календарь**:

```
Sub CreateCalendarReference()
   If ReferenceFromFile("C:\Windows\System\Mscal.ocx") = True Then
      MsgBox "Указатель установлен успешно."
   Else
      MsgBox "Указатель не был успешно установлен."
   End If
End Sub
```

Семейство References

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"accolReferencesC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"accolReferencesX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"accolReferencesP"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"accolReferencesM"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"accolReferencesE"}

Семейство **References** содержит в себе все объекты **Reference,** представляющие собой установленные в текущий момент ссылки.

Дополнительные сведения

Для просмотра диалогового окна **Ссылки**, содержащего список ссылок, непосредственно соответствующих объектам **Reference** семейства **References**, необходимо выбрать в меню **Сервис** команду **Ссылки**. Каждому установленному флажку в списке соответствует один объект **Reference**. Не установленные флажки не соответствуют объектам, входящим в семейство **References**.

Для перечисления объектов семейства **References** используется конструкция **For Each...Next**.

Семейство **References** принадлежит к объекту **Application** Microsoft Access.

Объекты **Reference** семейства **References** пронумерованы начиная с 1.

Свойство hWndAccessApp

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acprohWndAccessAppC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acprohWndAccessAppX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acprohWndAccessAppA"}

Свойство **hWndAccessApp** используется для возвращения дескриптора, присваиваемого в Microsoft Windows 95 главному окну Microsoft Access.

Значения

Задаваемое в Microsoft Access значение свойства **hWndAccessApp** имеет тип **Long Integer** и является доступным только для чтения.

Дополнительные сведения

Значение данного свойства используется в Visual Basic для вызовов функций интерфейса программирования приложений (API) Windows или других внешних процедур, требующих указания дескриптора окна в качестве аргумента.

Для помещения дескриптора в окно объекта Microsoft Access, например объектов Form или Report, следует использовать свойство **hWnd.**

Свойство «Тип рисунка» (PictureType)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPictureTypeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPictureTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPictureTypeA"}

Свойство **Тип рисунка (PictureType)** определяет сохранение помещенного в объект рисунка в виде связанного или внедренного объекта.

Значения

Свойство **Тип рисунка (PictureType)** может иметь следующие значения.

Значение	Описание	Visual Basic
Внедренный	(Значение по умолчанию) Рисунок внедряется в объект и становится частью файла базы данных.	0
Связанный	Рисунок связывается с объектом. В Microsoft Access сохраняется указатель на файл рисунка на диске.	1

Значение свойства **Тип рисунка (PictureType)** задается в окне свойств объекта, в макросе или в программе Visual Basic.

Значение данного свойства может быть задано только в режиме конструктора формы или отчета.

Для элементов управления допускается задание значения по умолчанию для данного свойства при помощи окна стандартных свойств элемента управления или метода Visual Basic **DefaultControl.**

Дополнительные сведения

Если для данного свойства выбрано значение «Внедренный», размер базы данных увеличивается на размер файла рисунка, а для некоторых метафайлов (.wmf) увеличение размера базы данных может превышать размер файла рисунка более чем вдвое. При заданном для данного свойства значении «Связанный» размер базы данных не увеличивается, поскольку в Microsoft Access сохраняется только указатель на файл рисунка на диске.

Примечание. Если связанный файл перемещается в другой каталог, необходимо обновить связь с помощью свойства объекта **Рисунок (Picture).**

Значения отдельных битов, образующих изображение на внедренном рисунке, сохраняются в свойстве объекта **PictureData.** Для связанного рисунка в данном свойстве записан путь к файлу на диске.

Для изменения связанного рисунка может быть использовано отдельное приложение. Изменения связанного рисунка будут отражены при следующем открытии объекта базы данных, содержащего рисунок.

Свойство Assistant

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅS. пїЅпїЅпїЅпїЅпїЅS":"acproAssistantC;ofobjAssistant;ofproItem"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"acproAssistantX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproAssistantA"}

Свойство **Assistant** возвращает указатель на объект **Assistant.**

Значения

[*приложение*.]**Assistant**[.*метод или свойство*]

Свойство **Assistant** может иметь следующие значения.

Значение	Описание
приложение	Необязательное. Объект **Application.**
Assistant	Обязательное. При использовании без указания метода или свойства возвращает свойство **Элемент (Item)** объекта **Assistant.**
метод или свойство	Метод или свойство объекта **Assistant**.

Свойство **Assistant** доступно только из программ Visual Basic.

Дополнительные сведения

Доступ ко всем свойствам и методам объекта **Assistant** открывается после установления на него указателя. Для его установления выберите из меню **Сервис** модуля, находящегося в режиме конструктора, команду **Ссылки.** После этого в диалоговом окне **Ссылки** создайте связи с библиотекой объектов Microsoft Office 8.0, установив соответствующий флажок. В случае использование констант библиотеки объектов Microsoft Office 8.0 для задания свойств объекта **Assistant** или в качестве аргумента метода этого объекта возможно автоматическое создание ссылок.

Свойство Assistant, пример

В следующем примере свойство **Assistant** объекта **Application** используется для управления различными свойствами и методами объекта **Assistant.** Для его выполнения вставьте следующий текст в раздел описаний стандартного модуля. При помощи диалогового окна **Ссылки,** вызываемого командой **Ссылки** из меню **Сервис,** создайте указатель на библиотеку объектов Microsoft Office 8.0. После этого щелкните мышкой в любом месте процедуры и запустите ее нажатием клавиши F5.

```
Sub AnimateAssistant()
   Dim blnState As Boolean
   With Assistant
      ' Сохраняет состояние помощника на экране.
      blnState = .Visible
      ' Выводит помощника на экран.
      If blnState = False Then .Visible = True
      ' Запускает анимацию помощника.
      .Animation = msoAnimationAppear
      ' Выводит свойства Item и FileName объекта Assistant.
      MsgBox "Привет, меня зовут " & .Item & ". Я живу в " & .FileName, ,
"Сведения о помощнике:"
      ' Восстанавливает состояние помощника на экране.
      .Visible = blnState
   End With
End Sub
```

Свойство CommandBars

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproCommandBarsC;ofobjCommandBars;ofproCount"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproCommandBarsX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproCommandBarsA"}

Свойство **CommandBars** возвращает указатель на семейство **CommandBars.**

Значения

[*приложение*.]**CommandBars**[.*метод или свойство*]

Свойство **CommandBars** может иметь следующие значения.

Значение	Описание
приложение	Необязательное. Объект **Application.**
CommandBars	Обязательное. При использовании без указания метода или свойства возвращает семейство **CommandBars.**
метод или свойство	Метод или свойство семейства **CommandBars.** Свойство **Item** (семейства) является свойством по умолчанию для семейства **CommandBars.**

Свойство **CommandBars** доступно только из программ Visual Basic.

Дополнительные сведения

Семейство **CommandBars** является семейством встроенных и специальных панелей команд в приложении. Для указания на конкретный элемент семейства используются объектные индексы – строковые выражения, содержащие имя объекта-члена. Индекс первого такого объекта равен 1. Их общее число содержится в свойстве **Count** семейства **CommandBars.**

Доступ ко всем свойствам и методам объекта **CommandBars** открывается после установления на него указателя. Для его установления выберите из меню **Сервис** модуля, находящегося в режиме конструктора, команду **Ссылки.** После этого в диалоговом окне **Ссылки** создайте связи с библиотекой объектов Microsoft Office 8.0, установив соответствующий флажок.

Свойство CommandBars, пример

В следующем примере свойство **CommandBars** объекта **Application** используется для управления различными свойствами и методами семейства **CommandBars.** Для его выполнения вставьте следующий текст в раздел описаний стандартного модуля. При помощи диалогового окна **Ссылки,** вызываемого одноименной командой из меню **Сервис,** создайте указатель на библиотеку объектов Microsoft Office 8.0. После этого щелкните мышкой в любом месте процедуры `GetCommandBarInfo` и запустите ее нажатием клавиши F5.

```
Sub GetCommandBarInfo()
   Dim strCBBarName As String

   strCBBarName = InputBox("Введите имя панели команд", "Получение сведений
о панели команд", "Строка меню")
   If Len(strCBBarName) > 0 Then
      CommandBarInfo strCBBarName
   End If
End Sub

Sub CommandBarInfo(strCBBarName As String)
   Dim cbr As CommandBar

   On Error Resume Next
   Set cbr = CommandBars(strCBBarName)
   If Err = 0 then
      With cbr
         MsgBox "Панель команд '" & .Name & "' содержит " & _
            .Controls.Count & IIf(.Controls.Count = 1, " элемент управления
и ", _
            " элементы управления ") & IIf(.BuiltIn, "да", "нет") & _
            " встроенная панель инструментов", vbOKOnly, "'" & _
            .Name & "' Сведения:"
      End With
   Else
      MsgBox "Неправильное имя панели команд "
   End If
End Sub
```

Свойство Item (Семейства)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproItemCollectionsC;dacolProperty;daproCount"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproItemCollectionsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproItemCollectionsA;dacolProperty"}

Свойство **Item** возвращает элемент семейства по заданному положению или по индексу.

Значения

объект.**Item(***индекс***)**

Свойство **Item** может иметь следующие значения.

Значение	Описание
объект	Одно из следующих семейств: **Controls, Forms, Modules, Pages, Properties,**, **Reports.**
индекс	Выражение, задающее положение члена семейства, указанного аргументом *объект.* Если это выражение является числовым, то *индекс* должен быть числом от 0 до значения свойства **Count** без единицы. Если это выражение является строковым, то *индекс* должен быть именем члена семейства.

Свойство **Item** доступно только для чтения и только в программе Visual Basic.

Дополнительные сведения

Если значение аргумента *индекс* не соответствует ни одному из существующих компонентов семейства, возникает ошибка.

Свойство **Item** является свойством по умолчанию для семейств. Это означает, что его точного задания не требуется. Например, следующие конструкции эквивалентны:

```
Debug.Print Modules(0)

Debug.Print Modules.Item(0)
```

Свойство OLEData

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproOLEDataC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproOLEDataX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproOLEDataA"}

Свойство **OLEData** копирует данные из одной свободной рамки объекта в другую или из одного элемента ActiveX в другой.

Значения

имяЭлемента.**OLEData**

Свойство **OLEData** может иметь следующие значения.

Значение	Описание
имяЭлемента	Обязательное. Имя элемента управления типа свободной рамки объекта.
OLEData	Обязательное. Данные, содержащиеся в свободной рамке объекта или в элементе ActiveX.

Свойство **OLEData** доступно только в программе Visual Basic. Для задания свойства **OLEData** элемента ActiveX возможно использование в режиме конструктора этого же свойства другого элемента ActiveX.

Дополнительные сведения

Данное свойство позволяет выводить в свободной рамке объекта данные из другой свободной рамки объекта.

При присвоении в качестве значения свойства **OLEData** одного элемента ActiveX значения этого же свойства другого элемента ActiveX первый из них становится копией второго. Например, в результате следующего присвоения значения свойства элемент управления TreeView становится элементом управления Календарь:

```
Me!TreeView.OLEData = Me!МойКалендарь.OLEData
```

Свойство Properties

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPropertiesC;dacolProperties;daproCount"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPropertiesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPropertiesA"}

Свойство **Properties** возвращает указатель на семейство **Properties** элемента управления.

Значения

имяЭлемента.**Properties**[.*метод или свойство*]

Свойство **Properties** может иметь следующие значения.

Значение	Описание
имяЭлемента	Обязательное. Имя элемента управления, на семейство **Properties** которого необходима ссылка.
Properties	Обязательное. При использовании без указания метода или свойства возвращает семейство **Properties.**
метод или свойство	Метод или свойство объекта семейства **Properties.** Свойство **Item** (семейства) является свойством по умолчанию для семейства **Properties.**

Свойство **Properties** доступно только для чтения и только в программе Visual Basic.

Дополнительные сведения

Семейство **Properties** является семейством всех свойств, относящихся к элементу управления. Для указания на конкретные компоненты семейства используются их порядковые номера в семействе или строковые выражения, являющиеся именами этих компонентов. Порядковый номер первого объекта семейства равен 0. Общее число таких объектов равно значению свойства **Count** семейства **Properties** без единицы.

Свойство Properties, пример

В следующей процедуре свойство **Properties** используется для вывода в окно отладки всех свойств, связанных с элементом управления в форме. Для выполнения этой программы поместите в форму кнопку `СвойстваСписка` и вставьте следующий текст в раздел описаний формы. Для вывода списка свойств в окне отладки нажмите эту кнопку.

```
Sub СвойстваСписка_Click()
   ListControlProps Me
End Sub

Sub ListControlProps(frm As Form)
   Dim ctl As Control, prp As Property

   On Error GoTo props_err
   For Each ctl In frm.Controls
      Debug.Print ctl.Properties("Имя")
      For Each prp In ctl.Properties
         Debug.Print vbTab & prp.Name & " = " & prp.Value
      Next prp
   Next ctl

props_bye:
   Exit Sub

props_err:
   If Err = 2187 Then
      Debug.Print vbTab & prp.Name & " = Доступно только во время
разработки."
      Resume Next
   Else
      Debug.Print vbTab & prp.Name & " = Ошибка: " & Err
      Resume Next
   End If
End Sub
```

Свойство «Цвет границы» (BorderColor)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproBorderColorC;vafctQBColor;vafctRGB"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproBorderColorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproBorderColorA"}

Свойство **Цвет границы (BorderColor)** позволяет указать цвет границы элемента управления.

Значения

Свойство **Цвет границы (BorderColor)** задается с помощью числового выражения, значение которого определяет цвет границы элемента управления.

Значение данного свойства задается с помощью построителя палитры, который вызывается нажатием кнопки построителя ... справа от ячейки в окне свойств. Построитель палитры позволяет создавать специальные цвета для границ элементов управления.

Кроме того, значение данного свойства можно задать с помощью кнопки **Цвет линии/границы** ... на панели инструментов **Форматирование (форма/отчет),** в макросе или в программе Visual Basic.

Для элементов управления можно указать используемый по умолчанию цвет границы в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

Цвет границы элемента управления становится видимым только при заданном для свойства **Оформление (SpecialEffect)** значении «Обычное» или «С тенью». При других значениях этого свойства указание цвета границы с помощью свойства **Цвет границы (BorderColor)** приводит к тому, что свойство **Оформление (SpecialEffect)** получает значение «Обычное».

Свойства «Цвет границы» (BorderColor), «Цвет фона» (BackColor), «Цвет текста» (ForeColor), пример

В следующем примере функция **RGB** используется для задания свойств **Цвет границы (BorderColor)**, **Цвет фона (BackColor)** и **Цвет текста (ForeColor),** в зависимости от значения поля «Задолженность». Ту же задачу можно решить с помощью функции **QBColor**. Для установки характеристик отображения элемента управления сразу при открытии формы или при переходе к другой записи вставьте следующий текст в процедуру обработки события `Form_Current( )`.

```
Sub Form_Current()
   Dim curAmntDue As Currency, lngBlack As Long
   Dim lngRed As Long, lngYellow As Long, lngWhite As Long

   If Not IsNull(Me!Задолженность.Value) Then
      curAmntDue = Me!Задолженность.Value
   Else
      Exit Sub
   End If
   lngRed = RGB(255, 0, 0)
   lngBlack = RGB(0, 0, 0)
   lngYellow = RGB(255, 255, 0)
   lngWhite = RGB(255, 255, 255)
   If curAmntDue > 100 Then
      Me!Задолженность.BorderColor = lngRed
      Me!Задолженность.ForeColor = lngRed
      Me!Задолженность.BackColor = lngYellow
   Else
      Me!Задолженность.BorderColor = lngBlack
      Me!Задолженность.ForeColor = lngBlack
      Me!Задолженность.BackColor = lngWhite
   End If
End Sub
```

Свойство «Тип границы» (BorderStyle)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproBorderStyleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproBorderStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproBorderStyleA"}

- Формы. Свойство **Тип границы (BorderStyle)** позволяет указать тип и элементы границы (строка заголовка, **оконное** меню, кнопки свертывания и развертывания окна или кнопка **Закрыть**), используемые в форме. Как правило, используются разные типы границ для обычных форм, всплывающих форм и специальных диалоговых окон.
- Элементы управления. Свойство **Тип границы (BorderStyle)** определяет вид границы элемента управления.

Значения

Для форм свойство **Тип границы (BorderStyle)** может иметь следующие значения.

Значение	Описание	Visual Basic
Отсутствует	Форма не имеет границ и связанных с границей элементов. Изменение ее размеров невозможно.	0
Тонкая	Форма выводится с тонкой границей и может иметь любые элементы границы. Изменение размеров формы невозможно (команда **Размер** в оконном меню является недоступной). Такую настройку часто используют для всплывающих форм. (Для того, чтобы форма всегда выводилась поверх всех остальных окон Microsoft Access, можно также задать для свойства **Всплывающее окно (PopUp)** значение «Да».)	1
Изменяемая	(Значение по умолчанию). Форма имеет стандартную границу форм Microsoft Access, может иметь любые из связанных с границей элементов и допускает изменение размеров. Эта настройка является стандартной для обычных форм Microsoft Access.	2
Окно диалога	Форма имеет жирную (двойную) границу и может включать строку заголовка, кнопку **Закрыть** и оконное меню. Свертывание или развертывание формы, а также изменение ее размеров недопустимо (команды **Развернуть**, **Свернуть** и **Размер** в оконном меню недоступны). Эта настройка часто используется для специальных диалоговых окон. Для того, чтобы сделать форму модальной, следует задать для свойства **Модальное окно (Modal)** значение «Да». Для того, чтобы определить модальную всплывающую форму, чем обычно являются окна диалога, следует одновременно задать значение «Да» для свойств **Всплывающее окно (PopUp)** и **Модальное окно (Modal)**.	3

Задание значения свойства **Тип границы (BorderStyle)** допускается только в режиме конструктора формы с помощью окна свойств, в макросе или в программе Visual Basic.

Для элементов управления свойство **Тип границы (BorderStyle)** может иметь следующие значения.

Значение	Описание	Visual Basic
Отсутствует	(Значение по умолчанию только для надписей, диаграмм и подчиненных отчетов) Граница является невидимой	0
Сплошная	(Значение по умолчанию). Сплошная линия границы.	1
Штриховая	Штриховая линия границы.	2
Пунктирная	Пунктирная линия границы.	3
Точечная	Точечная линия границы.	4
Редкоточечная	Линия границы, образуемая редкими точками.	5
Штрих-пунктирная	Линия границы, образуемая комбинацией штрих-точка.	6
Штрих-точечная	Линия границы, образуемая комбинацией штрих-точка-точка.	7

Значение свойства **Тип границы (BorderStyle)** элемента управления задается в его окне свойств, в макросе или в программе Visual Basic.

Для элементов управления можно указать используемое по умолчанию значение этого свойства в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

Тип границы элемента управления является видимым только если свойство **Оформление (SpecialEffect)** имеет значение «Обычное» или «С тенью». При других значениях данного свойства указание цвета границы с помощью свойства **Цвет границы (BorderStyle)** приводит к тому, что свойство **Оформление (SpecialEffect)** получает значение «Обычное».

Для форм свойство **Тип границы (BorderStyle)** задает характеристики границы, позволяющие на глаз различать нормальные формы, всплывающие формы и специальные диалоговые окна. Другие характеристики формы определяются свойствами **Модальное окно (Modal)** и **Всплывающее окно (PopUp).**

Кроме того для форм определены свойства **Кнопка оконного меню (ControlBox), Кнопка закрытия (CloseButton), Кнопки размеров окна (MinMaxButtons), Полосы прокрутки (ScrollBars), Поле номера записи (NavigationButtons)** и **Область выделения (RecordSelectors).** Значения этих свойств комбинируются следующими способами.

- Если для свойства **Тип границы (BorderStyle)** задано значение «Отсутствует» или «Диалоговое окно», форма не имеет кнопок свертывания и развертывания окна, вне зависимости от значения свойства **Кнопки размеров окна (MinMaxButtons)**.
- При значении свойства **Тип границы (BorderStyle)** «Отсутствует», форма не имеет оконного меню, вне зависимости от значения свойства **Кнопка оконного меню (ControlBox)**;
- Значение свойства **Тип границы (BorderStyle)** не влияет на вывод полос прокрутки, кнопок перехода, поля номера записи или области выделения.

Свойство **Тип границы (BorderStyle)** действует только в режиме формы. В режиме конструктора значения данного свойства игнорируются.

При значении «Отсутствует», заданном для свойства **Тип границы (BorderStyle)** всплывающей формы, единственным способом закрыть форму является помещение в нее кнопки **Закрыть,**

которая запускает макрос, содержащий макрокоманду **Закрыть (Close)** или процедуру обработки события Visual Basic, в которой вызывается метод **Close.**

Всплывающие формы обычно создаются с фиксированными размерами, однако, можно разрешить изменение размеров всплывающей формы, задав для ее свойства **Всплывающее окно (PopUp)** значение «Да», а для свойства **Тип границы (BorderStyle)** — значение «Изменяемая».

Для того, чтобы задать для свойств **Модальное окно (Modal)** и **Всплывающее окно (PopUp)** значение «Да», можно также использовать макрокоманду **Открыть форму (OpenForm)** со значением «Диалоговое окно» в аргументе «Режим окна».

Свойство «Цвет текста» (ForeColor)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproForeColorC;vafctQBColor;vafctRGB;vamthCircle"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproForeColorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproForeColorA"}

Свойство **Цвет текста (ForeColor)** позволяет указать цвет текста, выводящегося в элементе управления. Использование данного свойства для элементов управления в формах или в отчетах облегчает считывание и перенос специальных значений. Например, можно изменить цвет текста элемента управления «На складе», если значение падает ниже уровня, при котором требуется разместить новый заказ.

В отчетах это свойство используют для создания специальных эффектов при подготовке к печати на цветном принтере. В этом случае данное свойство указывает цвета печати текста и рисунков для методов **Print, Line** и **Circle.**

Значения

Свойство **Цвет текста (ForeColor)** задается с помощью числового выражения, значение которого определяет цвет текста, выводящегося в элементе управления.

Значение данного свойства задается с помощью построителя палитры, который вызывается нажатием кнопки построителя ... справа от ячейки в окне свойств. Построитель палитры позволяет создавать специальные цвета для текста элементов управления.

Для элементов управления значение данного свойства можно задать с помощью кнопки **Цвет текста**  на панели инструментов **Форматирование (форма/отчет),** в окне свойств элемента управления, в макросе или в программе Visual Basic.

Для элементов управления можно указать используемый по умолчанию цвет текста в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

В отчетах значение свойства **Цвет текста (ForeColor)** можно задать только в макросе или в процедуре обработки события Visual Basic, указанной в свойстве события раздела **Печать (OnPrint).**

Для установки этого свойства для объектов **TableDef** и **QueryDef** также используется кнопка **Цвет текста**  на панели инструментов **Форматирование (режим таблицы)** или в программе Visual Basic с помощью свойства **DatasheetForeColor.**

Дополнительные сведения

Для полей, полей со списком, надписей и кнопок, содержащих гиперссылку, свойство **Цвет текста (ForeColor)** автоматически получает значение, указанное в поле **Цвет просмотренной гиперссылки** или **Цвет гиперссылки** на вкладке **Гиперссылки/HTML** диалогового окна **Параметры**, вызываемого командой **Параметры** из меню **Сервис.** При снятии гиперссылки с элемента управления (например, путем задания в качестве значения свойства присоединенного поля **Данные (ControlSource)** источника, не являющегося полем гиперссылки), автоматически восстанавливается стандартное значение свойства **Цвет текста (ForeColor)**. Свойство **Цвет текста (ForeColor)** применимо только для кнопок, содержащих надпись, а не рисунок.

Свойство «Рисунок» (Picture)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproPictureC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproPictureX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproPictureA"}

Свойство **Рисунок (Picture)** позволяет указать рисунок или другой вид графической информации, выводящийся на кнопке, в элементе управления-рисунке, на переключателе, на вкладке элемента управления типа набора вкладок или в фоновом рисунке формы или отчета. На кнопке можно разместить или рисунок или надпись.

Значения

В свойстве **Рисунок (Picture)** содержится строка «(рисунок)», либо путь и имя файла выводящегося точечного рисунка или рисунка в другом графическом формате.

Для задания значения данного свойства используются:

- окно свойств. Нажмите кнопку построителя ... справа от ячейки в окне свойств (для кнопок и переключателей). После выбора из списка **Выбор рисунка** файла, содержащего рисунок, свойство принимает значение «(рисунок)»;
- макрос;
- программа Visual Basic. Допустимо использование строкового выражения, включающего путь и имя файла рисунка, например:

```
btnShowLogo.Picture = "c:\Windows\Winlogo.bmp"
```

- команда **Рисунок** из меню **Вставка** (для элементов управления-рисунков и фоновых рисунков в формах и отчетах).

По умолчанию для данного свойства задается значение «(отсутствует)». После размещения рисунка в объекте данное свойство получает значение «(рисунок)» или путь и имя файла рисунка. Удаление значения «(рисунок)» или пути и имени файла рисунка из ячейки свойства приводит к удалению рисунка из объекта. В ячейке свойства при этом снова выводится «(отсутствует)».

Если для свойства **Тип рисунка (PictureType)** выбрано значение «Внедренный», то рисунок, помещенный в объект сохраняется вместе с объектом.

Дополнительные сведения

Допускается создание специальных рисунков в Microsoft Paintbrush или другом приложении, создающем растровые рисунки. Файл рисунка должен иметь расширение .bmp, .ico или .dib. Также допустимо использование файлов в форматах .wmf и .emf и любых других графических форматов, для которых имеется графический фильтр. В формах, отчетах и в элементах управления «Рисунок» применимы все рисунки. В кнопках и переключателях поддерживаются только файлы с расширением .bmp.

На кнопке можно разместить или рисунок или надпись. Если для кнопки одновременно определены рисунок и надпись, то выводится только рисунок. Надпись появляется на кнопке после удаления рисунка. Microsoft Access помещает рисунок на кнопке с выравниванием по центру и обрезает края рисунка, если его размеры превышают размеры кнопки.

Совет. Чтобы создать кнопку или переключатель, на которых одновременно выводится рисунок и надпись, добавьте надпись в рисунок и укажите составной рисунок в свойстве **Рисунок (Picture)** элемента управления.

Свойство «Вывод на экран» (Visible)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅS. пїЅпїЅпїЅпїЅпїЅS":"acproVisibleC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"acproVisibleX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproVisibleA"}

Свойство **Вывод на экран (Visible)** позволяет делать скрытыми или видимыми формы, отчеты, разделы форм и отчетов, а также элементы управления. Это удобно в тех случаях, когда требуется обеспечить доступ к содержащимся в форме данным без необходимости выводить эту форму на экран. Например, таким способом значение элемента управления из скрытой формы используют в условиях отбора в запросе.

Значения

Свойство **Вывод на экран (Visible)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Объект является видимым.	**True** (–1)
Нет	Объект не является видимым.	**False** (0)

Значение данного свойства задается в окне свойств объекта (для разделов и всех элементов управления, за исключением конца страницы), в макросе или в программе Visual Basic.

Для форм и отчетов значение данного свойства следует задавать в макросе или в программе Visual Basic.

Для элементов управления значение данного свойства можно указать в окне стандартных значений или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

Свойство Вывод на экран (**Visible)** не влияет на вывод столбца в режиме таблицы. Для того, чтобы указать, является ли столбец видимым в режиме таблицы, следует использовать свойство **ColumnHidden.**

Если требуется скрыть объект при печати, используется свойство **DisplayWhen.**

Свойство **Вывод на экран (Visible)** в макросе или в процедуре обработки события, выполняющихся в ответ на событие **Текущая запись (Current),** позволяет сделать видимым или скрытым элемент управления в форме или в отчете. Например, так можно выводить поздравления сотрудникам, чья выработка в отчете «Продажи» превышает определенную величину.

Примечание Существует также свойство **Visible** объекта **Application,** позволяющее сделать невидимым целое приложение. Значение этого свойства задается только средствами Visual Basic.

Свойство «Цвет фона» (BackColor)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproBackColorC;vafctQBColor;vafctRGB"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproBackColorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproBackColorA"}

Свойство **Цвет фона (BackColor)** позволяет указать цвет внутренней области элемента управления или раздела.

Значения

Свойство **Цвет фона (BackColor)** задается с помощью числового выражения, значение которого определяет цвет фона элемента управления или раздела.

Значение данного свойства задается с помощью построителя палитры, который вызывается нажатием кнопки построителя справа от ячейки в окне свойств. Построитель палитры позволяет создавать специальные цвета заливки для элемента управления.

Кроме того, можно задать значение этого свойства с помощью кнопки **Цвет заливки/фона** на панели инструментов **Форматирование (форма/отчет),** в окне свойств элемента управления или раздела, в макросе или в программе Visual Basic.

В программе Visual Basic данное свойство задается с помощью числового выражения, значение которого имеет тип **Long.**

Для элементов управления используемое по умолчанию значение этого свойства можно указать в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Для объектов **TableDef** и **QueryDef** задание данного свойства возможно с помощью кнопки **Цвет заливки/фона** на панели инструментов **Форматирование (режим таблицы)** или в программе Visual Basic с помощью свойства **DatasheetBackColor.**

Дополнительные сведения

Для того, чтобы использовать свойство **Цвет фона (BackColor)** необходимо выбрать для свойства **Тип фона (BackStyle)** значение «Обычный», если последнее свойство является доступным.

Свойство «Тип фона» (BackStyle)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproBackStyleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproBackStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproBackStyleA"}

Свойство **Тип фона (BackStyle)** позволяет указать, является ли указанный цвет элемента управления видимым.

Значения

Свойство **Тип фона (BackStyle)** может иметь следующие значения.

Значение	Описание	Visual Basic
Обычный	(Значение по умолчанию для всех элементов управления, кроме групп). Элемент управления выводится с цветом фона, заданным в свойстве **Цвет фона (BackColor).**	1
Прозрачный	(Значение по умолчанию для групп). Цвет фона не выводится. Видимым является цвет формы или отчета позади элемента управления.	0

Значение данного свойства задается с помощью кнопки **Цвет заливки/фона** ... на панели инструментов **Форматирование (форма/отчет),** в окне свойств элемента управления, в макросе или в программе Visual Basic.

Для элементов управления используемое значение по умолчанию данного свойства можно указать в окне свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

При выборе кнопки **Прозрачный** на палитре **Цвет фона** свойство **Тип фона (BackStyle)** получает значение «Прозрачный»; в остальных случаях свойство **Тип фона (BackStyle)** получает значение «Обычный».

Совет. Для того, чтобы сделать невидимой кнопку, задайте значение «Да» для ее свойства **Прозрачный (Transparent).**

Свойство «Ширина границы» (BorderWidth)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproBorderWidthC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproBorderWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproBorderWidthA"}

Свойство **Ширина границы (BorderWidth)** позволяет указать ширину границы элемента управления.

Значения

Свойство **Ширина границы (BorderWidth)** может иметь следующие значения.

Значение	Описание	Visual Basic
Сверхтонкая	(Значение по умолчанию). Самая узкая граница, которую можно создать на системе пользователя.	0
От 1 до 6 пунктов	Значение ширины границы в пунктах.	От 1 до 6

Значение данного свойства задается с помощью кнопки **Толщина линии/границы** ... на панели инструментов **Форматирование (форма/отчет),** в окне свойств элемента управления, в макросе или в программе Visual Basic.

Для элементов управления используемое по умолчанию значение этого свойства можно указать в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

Для того, чтобы использовать свойство **Ширина границы (BorderWidth),** необходимо указать для свойства **Оформление (SpecialEffect)** значение «Обычное» или «С тенью», а для свойства **Тип границы (BorderStyle)** - любое значение, кроме «Отсутствует». Если при установке значения свойства **Ширина границы (BorderWidth)** для свойства **Оформление (SpecialEffect)** установлено любое другое значение и/или для свойства **Тип границы (BorderStyle)** задано значение «Отсутствует», свойство **Оформление (SpecialEffect)** автоматически получает значение «Обычное», а свойство **Тип границы (BorderStyle)** - значение «Сплошная».

Точное значение ширины границы определяется параметрами компьютера и принтера. На некоторых системах границы «Сверхтонкая» и с шириной 1 пункт выглядят одинаково.

Свойство FillColor

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFillColorC;vafctQBColor;vafctRGB"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFillColorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproFillColorA"}

Свойство **FillColor** позволяет указать цвет заливки прямоугольников и кругов, созданных в отчете с помощью методов **Line** и **Circle.** Данное свойство также используют в программах Visual Basic для оформления специальных отчетов, предназначенных для печати на цветном принтере или для просмотра на цветном мониторе.

Значения

Свойство **FillColor** задается с помощью числового выражения, значения которого определяет цвет заливки для всех прямоугольников и кругов.

Значение данного свойства задается только в макросе или в процедуре обработки события Visual Basic, определенной для свойства события раздела **Печать (OnPrint).**

Для задания значения данного свойства допускается применение функций **RGB** или **QBColor.** Значение свойства **FillColor** принадлежит к типу данных **Long.**

Свойство «Наклон линии» (LineSlant)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproLineSlantC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproLineSlantX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproLineSlantA"}

Свойство **Наклон линии (LineSlant)** позволяет указать проведение линии от верхнего левого угла к нижнему правому или от верхнего правого угла к нижнему левому.

Значения

Свойство **Наклон линии (LineSlant)** может иметь следующие значения.

Значение	Описание	Visual Basic
****	(Значение по умолчанию). От верхнего левого угла к нижнему правому.	**False** (0)
/	От верхнего правого угла к нижнему левому.	**True** (–1)

Значение данного свойства задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Дополнительные сведения

Свойство **Наклон линии (LineSlant)** используется для изменения направления линии. Позиции и размеры объекта-линии в форме или отчете задаются с помощью мыши.

Свойство «Оформление» (SpecialEffect)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproSpecialEffectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproSpecialEffectX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproSpecialEffectA"}

Свойство **Оформление (SpecialEffect)** позволяет указать способ объемного представления раздела или элемента управления.

Значения

Свойство **Оформление (SpecialEffect)** может иметь следующие значения.

Значение	Пример	Описание	Visual Basic
Обычное		Объект оформляется как плоский с использованием стандартных системных цветов или специальных цветов, заданных в режиме конструктора.	0
Приподнятое		Объект выводится с подсветкой слева и сверху и с тенью снизу и справа.	1
Утопленное		Объект выводится с тенью слева и сверху и с подсветкой снизу и справа.	2
Вдавленное		Объект выводится с ограничивающей вдавленной линией.	3
С тенью		Объект выводится с тенью снизу и справа.	4
Рельефное		Нижняя граница объекта представляется вдавленной линией.	5

Значение данного свойства задается с помощью кнопки **Оформление** на панели инструментов **Форматирование (форма/отчет),** в окне свойств объекта, в макросе или в программе Visual Basic.

Для элементов управления используемое по умолчанию значение данного свойства можно указать в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

Значение свойства **Оформление (SpecialEffect)** оказывает влияние на настройки, задаваемые свойствами **Тип границы (BorderStyle), Цвет границы (BorderColor)** и **Ширина границы (BorderWidth).** Например, если для свойства **Оформление (SpecialEffect)** задано значение «Приподнятое», то значения свойств **Тип границы (BorderStyle), Цвет границы (BorderColor)** и **Ширина границы (BorderWidth)** игнорируются. Кроме того, указание или изменение значения свойств **Тип границы (BorderStyle), Цвет границы (BorderColor)** и **Ширина границы (BorderWidth)** может привести к установке для свойства **Оформление (SpecialEffect)** значения «Обычное».

Примечание. После задания для свойства **Оформление (SpecialEffect)** поля значения С тенью происходит уменьшение высоты области текста. Для ее увеличения измените значение свойства **Высота (Height)** поля.

Свойство «Прозрачный» (Transparent)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproTransparentC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproTransparentX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproTransparentA"}

Свойство **Прозрачный (Transparent)** позволяет указать, является ли кнопка видимой.

Совет. Для того, чтобы сделать другие элементы управления видимыми или прозрачными, используйте свойство **Тип фона (BackStyle).**

Значения

Свойство **Прозрачный (Transparent)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Кнопка является невидимой.	**True** (–1)
Нет	(Значение по умолчанию). Кнопка является видимой.	**False** (0)

Значение данного свойства задается в окне свойств кнопки, в макросе или в программе Visual Basic.

Дополнительные сведения

Данное свойство используют для размещения невидимых кнопок поверх других элементов управления. Например, можно разместить поверх рисунка в рамке рисунка несколько невидимых кнопок, запускающих различные макросы или процедуры обработки события, когда пользователь выбирает различные области рисунка.

Примечание. Для того, чтобы скрыть или отключить кнопку, используйте свойство **Вывод на экран (Visible).** Отключить кнопку без ее скрытия позволяет свойство **Доступ (Enabled).** Для того, чтобы скрыть кнопку при выводе формы или отчета на печать, используйте свойство **Режим вывода (DisplayWhen).**

Свойства FillColor, FillStyle, пример

В следующем примере метод **Circle** используется для рисования окружности и для создания в ней вырезанного сектора. Для его заполнения красным цветом используются свойства **FillColor** и **FillStyle.** Также из верхнего левого угла в центр окружности проводится линия.

Для выполнения данного примера в Microsoft Access создайте новый отчет, задайте для свойства **Печать (OnPrint)** области данных значение [Процедура обработки событий]. Введите в модуль отчета следующий текст и переключитесь в режим предварительного просмотра.

```
Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
   Const conPI = 3.14159265359

   Dim sngHCtr As Single, sngVCtr As Single, sngRadius As Single
   Dim sngStart As Single, sngEnd As Single

   sngHCtr = Me.ScaleWidth / 2                 ' Центр по горизонтали.
   sngVCtr = Me.ScaleHeight / 2                ' Центр по вертикали.
   sngRadius = Me.ScaleHeight / 3              ' Радиус окружности.
   Me.Circle(sngHCtr, sngVCtr), sngRadius      ' Рисует окружность.
   sngStart = -0.00000001                      ' Начало вырезанного
сектора.

sngEnd = -2 * conPI / 3                        ' Конец вырезанного сектора.
   Me.FillColor = RGB(255,0,0)                 ' Красный цвет для
вырезанного сектора.
   Me.FillStyle = 0                            ' Заполняет вырезанный
сектор.
   ' Рисует вырезанный сектор в окружности.
   Me.Circle(sngHCtr, sngVCtr), sngRadius, , sngStart, sngEnd

   ' Проводит линию к центру окружности.
   Dim intColor As Integer
   Dim sngTop As Single, sngLeft As Single
   Dim sngWidth As Single, sngHeight As Single

   Me.ScaleMode = 3                            ' Задает масштаб в точках.
   sngTop = Me.ScaleTop                        ' Верхняя внутренняя
сторона.
   sngLeft = Me.ScaleLeft                      ' Левая внутренняя сторона.
   sngWidth = Me.ScaleWidth / 2                ' Ширина внутренней стороны.
   sngHeight = Me.ScaleHeight / 2              ' Высота внутренней стороны.
   intColor = RGB(255, 0, 0)                   ' Устанавливает красный
цвет.

   ' Проводит линию.
   Me.Line (sngTop, sngLeft)-(sngWidth, sngHeight), intColor
End Sub
```

Свойство ActiveDatasheet

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproActiveDatasheetC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproActiveDatasheetX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproActiveDatasheetA"}

Свойство **ActiveDatasheet** вместе с объектом **Screen** используется для идентификации объекта или для создания ссылки на имеющий фокус объект в режиме таблицы.

Значения

Значением свойства **ActiveDatasheet** является активный объект в режиме таблицы, имеющий фокус во время выполнения.

Данное свойство доступно только в макросе или в программе Visual Basic и допускает только чтение во всех режимах.

Дополнительные сведения

Свойство **ActiveDatasheet** используется для ссылок на активный объект в режиме таблицы и его свойства и методы. Например, в следующей конструкции свойство **ActiveDatasheet** используется для ссылки на верхнюю строку активного объекта в режиме таблицы.

```
TopRow = Screen.ActiveDatasheet.SelTop
```

Свойство ActiveDatasheet, пример

В следующем примере свойство **ActiveDatasheet** используется для определения ячейки, имеющей фокус, или, если выделено несколько ячеек, для определения первой выделенной строки и столбца выделенного блока.

```
Sub GetSelection()
   Dim objDatasheet As Object
   Dim lngFirstRow As Long, lngFirstColumn As Long
   Const conNoActiveDatasheet = 2484
   On Error GoTo GetSelection_Err
   Set objDatasheet = Screen.ActiveDatasheet
   lngFirstRow = objDatasheet.SelTop
   lngFirstColumn = objDatasheet.SelLeft
   MsgBox "Координаты первой ячейки выделенного блока"
      & "Строка" & lngFirstRow & ", Столбец" & lngFirstColumn
GetSelection_Bye:
   Exit Sub
GetSelection_Err:
   If Err = conNoActiveDatasheet Then
      Resume GetSelection_Bye
   End If
End Sub
```

Свойство «Присоединенный столбец» (BoundColumn)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproBoundColumnC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproBoundColumnX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproBoundColumnA"}

Когда пользователь выбирает элемент в списке или в поле со списком, свойство **Присоединенный столбец (BoundColumn)** определяет значение какого столбца становится значением элемента управления. Если элемент управления присоединен к полю, то значение, находящееся в столбце, указанном в свойстве **Присоединенный столбец (BoundColumn),** заносится в поле, указанное в значение свойства **Данные (ControlSource).**

Значения

Свойство **Присоединенный столбец (BoundColumn)** может иметь следующие значения.

Значение	Описание
0	В текущую запись вместо значения элемента столбца заносится значение свойства **ListIndex.** Для первой строки списка свойство **ListIndex** имеет значение 0, для второй строки 1 и т.д. Microsoft Access Задает значение свойства **ListIndex,** когда пользователь выбирает элемент в списке или в раскрывающемся списке поля со списком. Задание значения 0 для свойства **Присоединенный столбец (BoundColumn)** и использование значений свойства **ListIndex** элемента управления может быть полезным в ситуациях, когда удобнее работать только с последовательностью номеров элементов списка, а не с их значениями.
1 или более высокое	(По умолчанию 1). Значением элемента управления становится значение, находящееся в указанном столбце. Если элемент управления присоединен к полю, то значение, полученное элементом управления, заносится в текущую запись. Значение свойства **Присоединенный столбец (BoundColumn)** не может превышать значение свойства **Число столбцов (ColumnCount).**

Значение свойства **Присоединенный столбец (BoundColumn)** можно задать в окне свойств элемента управления, в макросе или в программе Visual Basic.

Для полей таблиц, для которых в свойстве **Тип элемента управления (DisplayControl)** задан тип отображения «Поле со списком» или «Список», значение данного свойства задается на вкладке **Подстановка** раздела свойств поля в режиме конструктора таблицы.

Примечание. Microsoft Access задает значение этого свойства автоматически, если в режиме конструктора таблицы для поля в столбце «Тип данных» выбирается «Мастер подстановок».

В программе Visual Basic в качестве значения свойства **Присоединенный столбец (BoundColumn)** следует задать число или числовое выражение со значением от 0 до значения свойства **Число столбцов (ColumnCount).**

Дополнительные сведения

Крайний левый видимый столбец поля со списком (крайний левый столбец для которого значение свойства **Ширина столбцов (ColumnWidths)** поля со списком не равно 0) содержит данные, выводящиеся в поле поля со списком в режиме формы или в режиме отчета. Свойство **Присоединенный столбец (BoundColumn)** определяет содержимое какого столбца будет сохранено при выборе строки в списке. Это позволяет сохранять значение элемента управления, отличное от значения, выводящегося в поле со списком.

Примечание. Если присоединенный столбец не является крайним левым из видимых столбцов в элементе управления (или если свойство **Присоединенный столбец**

(BoundColumn) имеет значение 0), свойство **Ограничиться списком (LimitToList)** получает значение «Да».

В Microsoft Access нумерация столбцов в свойстве **Column** начинается с нуля. Это означает, что для ссылки на значение первого столбца следует использовать выражение `Column(0)`, для ссылки на значение первого столбца – выражение `Column(1)` и т.д. Однако нумерация столбцов в свойстве **Присоединенный столбец (BoundColumn)** начинается с единицы. Это означает, что если для свойства **Присоединенный столбец (BoundColumn)** задано значение 1, то для ссылки на значение этого столбца следует использовать выражение `Column(0)`.

При заданном для свойства **Автоподстановка (AutoExpand)** значении «Да», Microsoft Access автоматически вводит значение, дополняющее введенные в поле поля со списком символы до значения из списка.

Свойство Column

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproColumnC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproColumnX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproColumnA"}

Свойство **Column** используется для ссылок на конкретный столбец, на комбинацию столбца и строки поля со списком с несколькими столбцами, или на список. Для ссылки на первый столбец используется значение 0, для ссылки на второй столбец – значение 1 и т.д. Для ссылки на первую строку используется значение 0, на вторую – 1 и т.д. Например, содержащего коды и названия клиентов, ссылка название клиента во втором столбце пятой строки имеет вид:

```
Forms!Контакты!Клиенты.Column(1, 4)
```

Значения

имяЭлементаУправления.**Column(***столбец***,** *строка***)**

Свойство **Column** использует следующие значения.

Значение	Описание
имяЭлемента Управления	Объект **Control,** представляющий собой активное поле или поле со списком.
столбец	Целое число из диапазона от 0 до значения свойства **Число столбцов (ColumnCount)** минус единица.
строка	Целое число из диапазона от 0 до значения свойства **ListCount** минус единица.

Значение данного свойства можно задать только в макросе или в программе Visual Basic. В режиме конструктора данное свойство недоступно, а во всех остальных режимах доступно только для чтения.

Дополнительные сведения

Свойство **Column** используется для присвоения элемента списка другому элементу управления, обычно полю. Например, чтобы указать в свойстве поля **Данные (ControlSource)** значение элемента из второго столбца выбранной строки списка, следует использовать такое выражение:

```
=Forms!Клиенты!Название.Column(1)
```

Если пользователь не выбрал элемент в списке или в поле со списком, свойство **Column** будет иметь пустое **(Null)** значение. Для определения произведения выбора допустимо использование функции **IsNull,** как показано в примере:

```
If IsNull(Forms!Клиенты!Страна)
   Then MsgBox "Выбор не сделан."
End If
```

Примечание. Для того чтобы определить число столбцов в списке, проверьте значение свойства **Число столбцов (ColumnCount).**

Свойства Column, «Число столбцов» (ColumnCount), пример

В следующем примере свойства **Column** и **Число столбцов (ColumnCount)** используются для печати элементов в выбранной строке списка.

```
Sub Просмотр_списка()
   Dim intNumColumns As Integer, inti As Integer
   Dim frmCust As Form

   Set frmCust = Forms!frmCustomers
   If frmCust!lstCustomerNames.ItemsSelected.Count > 0 Then
      ' Выбор сделан?
      intNumColumns = frmCust!lstCustomerNames.ColumnCount
      Debug.Print "Список содержит "; intNumColumns; _
         IIf(intNumColumns = 1, " столбец", " столбца(ов)"); " данных."
      Debug.Print "Выделено:"
      For inti = 0 To intNumColumns - 1
         Debug.Print frmCust!lstCustomerNames.Column(inti)    ' Печатает
данные из столбца.
      Next inti
   Else
      Debug.Print "Выбор в списке не сделан."
   End If
End Sub
```

Свойство «Число столбцов» (ColumnCount)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproColumnCountC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproColumnCountX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproColumnCountA"}

Свойство **Число столбцов (ColumnCount)** определяет число столбцов, выводящихся в списке или в раскрывающемся списке поля со списком, или число столбцов, передаваемых в объекты OLE в диаграмме или в рамке свободного объекта. Например, если задать для свойства **Число столбцов (ColumnCount)** списка в форме «Сотрудники» значение 3, то можно вывести в одном столбце имена сотрудников, в другом их фамилии, а в третьем коды.

Значения

Значением свойства **Число столбцов (ColumnCount)** может быть целое число в диапазоне от 1 до полного числа полей в таблице, запросе или инструкции SQL, или до максимального числа значений из списка, указанных в свойстве **Источник строк (RowSource)** элемента управления.

Значение свойства **Число столбцов (ColumnCount)** задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Для полей таблиц, для которых в свойстве **Тип элемента управления (DisplayControl)** задан тип отображения «Поле со списком» или «Список», значение данного свойства задается на вкладке **Подстановка** раздела свойств поля в режиме конструктора таблицы.

Примечание. Microsoft Access задает значение этого свойства автоматически, если в режиме конструктора таблицы для поля в столбце «Тип данных» выбирается «Мастер подстановок».

Дополнительные сведения

Поле со списком или список могут содержать несколько столбцов. Если в свойстве **Источник строк (RowSource)** элемента управления указано имя таблицы, запроса или инструкции SQL, то в поле со списком или в списке выводятся слева направо поля из источника вплоть до номера, указанного в свойстве **Число столбцов (ColumnCount).**

Для того чтобы вывести другой набор полей, следует создать либо новый запрос, либо новую инструкцию SQL, задающие другой набор или порядок полей, и указать их в свойстве **Источник строк (RowSource).**

Если свойство **Источник строк (RowSource)** содержит список значений (для свойства **Тип источника строк (RowSourceType)** установлено значение «Список значений»), то эти значения помещаются в строки и столбцы поля со списком или списка в том же порядке, в котором они представлены в свойстве **Источник строк (RowSource).** Например, если свойство **Источник строк (RowSource)** содержит список «Красный; Зеленый; Синий; Желтый» и для свойства **Число столбцов (ColumnCount)** задано значение 2, то первая строка поля со списком или списка будет в первом столбце содержать «Красный», а во втором – «Зеленый». В первом столбце второй строки будет помещено значение «Синий», а во втором – «Желтый».

Свойство **Ширина столбцов (ColumnWidths)** позволяет указать ширину столбцов, выводящихся в элементе управления, или скрыть эти столбцы.

Свойство «Заглавия столбцов» (ColumnHeads)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproColumnHeadsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproColumnHeadsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproColumnHeadsA"}

Свойство **Заглавия столбцов (ColumnHeads)** используют для вывода строки заголовков солбцов в списках, в полях со списком или в объекта OLE, принимающих заголовки столбцов. Кроме того, данное свойство позволяет создать подпись для каждого значения, отложенного на диаграмме. Конкретный вид заголовков, выводящихся в первой строке списка, определяется свойством объекта **Тип источника строк (RowSourceType).**

Значения

Свойство **Заглавия столбцов (ColumnHeads)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Включает вывод в качестве заголовков столбцов или подписей на диаграмме либо имен или подписей полей, либо данных, содержащихся в первой строке списка.	**True** (–1)
Нет	(Значение по умолчанию). Отключение вывода заголовков столбцов.	**False** (0)

Значение свойства **Заглавия столбцов (ColumnHeads)** задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Для полей таблиц, для которых в свойстве **Тип элемента управления (DisplayControl)** задан тип отображения «Поле со списком» или «Список», значение данного свойства задается на вкладке **Подстановка** раздела свойств поля в режиме конструктора таблицы.

Примечание. Microsoft Access задает значение этого свойства автоматически, если в режиме конструктора таблицы для поля в столбце «Тип данных» выбирается «Мастер подстановок».

Дополнительные сведения

Свойство **Тип источника строк (RowSourceType)** определяет использование в качестве заголовков столбцов либо имен полей, либо содержимого первой строки списка. Значение «Таблица/запрос» задает вывод в заголовках столбцов имен полей. Если поле имеет подпись, то в заголовке столбца выводится подпись поля. Например, если список содержит три столбца (свойство **Число столбцов (ColumnCount)** содержит значение 3), а в свойстве **Тип источника строк (RowSourceType)** указано значение «Таблица/запрос», то в заголовках выводятся имена первых трех полей (или подписей).

Если для свойства **Тип источника строк (RowSourceType)** задано значение «Список значений», то в качестве заголовков столбцов будут использованы элементы первой строки этого списка, как значения свойства **Источник строк (RowSource).**) Например, если список содержит три столбца, а в свойстве **Тип источника строк (RowSourceType)** выбрано значение «Список значений», то заголовками столбцов будут имена первых трех элементов значения свойства **Источник строк (RowSource).**

Совет. При невозможности выделения в режиме формы первой строки списка или поля со списком, проверьте, что для свойства **Заглавия столбцов (ColumnHeads)** установлено значение «Да».

В поле со списком заголовки столбцов выводятся только при раскрытии списка.

Свойство «Ширина столбцов» (ColumnWidths)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅS. пїЅпїЅпїЅпїЅпїЅS":"acproColumnWidthsC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acproColumnWidthsX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproColumnWidthsA"}

Свойство **Ширина столбцов (ColumnWidths)** определяет ширину столбца в поле со списком или в списке с несколькими столбцами. Его использование также позволяет скрывать столбцы.

Значения

Свойство **Ширина столбцов (ColumnWidths)** содержит значение, указывающее ширину каждого столбца в дюймах или в сантиметрах, в зависимости от системы измерений (американской или метрической), указанной в поле **Система единиц** вкладки **Числа** диалогового окна **Язык и стандарты** панели управления Windows. По умолчанию устанавливается либо 1 дюйм, либо 2,54 сантиметра. Значение свойства **Ширина столбцов (ColumnWidths)** должно лежать в диапазоне от 0 до 22 дюймов (55,87 см) для каждого столбца списка или поля со списком.

Разделителем значений ширины отдельных столбцов служит точка с запятой (;) (или другой символ, выбранный в поле **Разделитель элементов списка** диалогового окна **Язык и стандарты).**

Для того чтобы сделать столбец скрытым, следует задать для него ширину 0. Для любого или для всех столбцов можно задать в свойстве **Ширина столбцов (ColumnWidths)** пустое значение. Пустое значение ширины столбца задается, если перед разделителем списка в соответствующей позиции не указано числовое значение. В этом случае Microsoft Access автоматически задает стандартное значение ширины столбца, зависящее от числа столбцов и ширины поля со списком или списка.

Примечание. В элементе управления «поле со списком» выводится элемент из первого видимого столбца списка.

Значение свойства **Ширина столбцов (ColumnWidths)** можно задать в окне свойств элемента управления, в макросе или в программе Visual Basic.

Для полей таблиц, для которых в свойстве **Тип элемента управления (DisplayControl)** задан тип отображения «Поле со списком» или «Список», значение данного свойства задается на вкладке **Подстановка** раздела свойств поля в режиме конструктора таблицы.

Примечание. Microsoft Access задает значение этого свойства автоматически, если в режиме конструктора таблицы для поля в столбце «Тип данных» выбирается «Мастер подстановок».

В программе Visual Basic строковое выражение задает ширину столбцов в единицах твип. Разделителем значений ширины является точка с запятой. Другие единицы измерения следует указывать в явном виде (см или "). Например, следующее строковое выражение задает ширину трех столбцов в сантиметрах.

```
"6 см;0;6 см"
```

Дополнительные сведения

Если значение свойства **Ширина столбцов (ColumnWidths)** оставлено пустым, Microsoft Access устанавливает ширину всех столбцов равной ширине списка или поля со списком, деленной на число столбцов в списке.

Если суммарная ширина столбцов с указанной шириной превышает ширину элемента управления, то столбцы, находящиеся на правом краю, становятся скрытыми, а в элементе управления выводится горизонтальная полоса прокрутки.

Можно указать значения ширины отдельных столбцов, а для остальных оставить значения ширины пустыми. В этом случае Microsoft Access разделит остаточную ширину элемента

управления на число остающихся столбцов. При этом минимальным рассчитываемым значением ширины столбца является ширина 1440 твип (1 дюйм).

Например, для списка с шириной 8 см и тремя столбцами допустимыми являются следующие настройки.

Значение	Описание
1.5 см;0;4.5 см	Ширина первого столбца 1,5 см, второй столбец скрыт, ширина третьего столбца 4,5 см.
2 см;;3 см	Ширина первого столбца 2 см, ширина второго столбца задается по умолчанию, ширина третьего столбца 3 см. Поскольку видимой является только часть третьего столбца, выводится горизонтальная полоса прокрутки.
(Пустое)	Все три столбца имеют одинаковую ширину (2,67 см).

Примечание. Не следует путать данное свойство со свойством **ColumnWidth,** задающим ширину конкретного столбца в объекте в режиме таблицы.

Свойство «Ограничиться списком» (LimitToList)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproLimitToListC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproLimitToListX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproLimitToListA"}

Свойство **Ограничиться списком (LimitToList)** позволяет указать вывод в поле со списком только значений, принадлежащих списку.

Значения

Свойство **Ограничиться списком (LimitToList)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Принимается значение, выбранное в списке или в раскрывающемся списке поля со списком, а также введенное пользователем значение, совпадающее с элементом списка. Введенное значение, не совпадающее с элементом списка, не принимается. В этом случае пользователь должен ввести другое значение, выбрать элемент в списке, нажать клавишу ESC, или выбрать в меню **Правка** команду **Отменить.**	**True** (–1)
Нет	(Значение по умолчанию). Допускается ввод любого текста, удовлетворяющего свойству **Условие на значение (ValidationRule.**	**False** (0)

Значение свойства **Ограничиться списком (LimitToList)** задается в окне свойств поля со списком, в макросе или в программе Visual Basic.

Для полей таблиц, для которых в свойстве **Тип элемента управления (DisplayControl)** задан тип отображения «Поле со списком», значение данного свойства задается на вкладке **Подстановка** раздела свойств поля в режиме конструктора таблицы..

Примечание. Microsoft Access задает значение этого свойства автоматически, если в режиме конструктора таблицы для поля в столбце «Тип данных» выбирается «Мастер подстановок».

Дополнительные сведения

Если для свойства **Ограничиться списком (LimitToList)** присоединенного поля со списком задано значение «Нет», пользователь имеет возможность ввести в поле значение, не принадлежащее списку. Microsoft Access сохраняет новое значение в базовой таблице или запросе формы, в поле, указанном в свойстве поля со списком **Данные (ControlSource),** а не в таблице или запросе, указанных в свойстве поля со списком **Источник строк (RowSource).** Для появления введенных значений в списке, Новое значение в таблице или запросе, являющимися источником списка для поля со списком, следует задавать с помощью макроса или процедуры обработки события Visual Basic, выполняющейся при возникновении события **Отсутствие в списке (NotInList).**

Примечание. Если в качестве значения свойства **Присоединенный столбец (BoundColumn)** поля со списком выбран не первый из видимых столбов (или если значением этого свойства является 0), свойство **Ограничиться списком (LimitToList)** автоматически принимает значение «Да».

Если одновременно заданы значения «Да» для свойств **Ограничиться списком (LimitToList)** и **Автоподстановка (AutoExpand),** то по мере ввода пользователем символов в поле Microsoft

Access проводит поиск в списке подходящих значений и ограничивает ими список.

Если для свойства **Ограничиться списком (LimitToList)** задано значении «Да», и пользователь щелкнет мышкой на стрелке поля со списком, то по мере ввода пользователем символов в поле Microsoft Access выделяет подходящие значения, даже если для свойства **Автоподстановка (AutoExpand)** задано значение «Нет». Выделенные значения появляются в списке поля со списком после нажатия пользователем клавиши ENTER, при переходе к другому элементу управления или к другой записи.

В Microsoft Access 97 если для свойства **Ограничиться списком (LimitToList)** задано значение «Да» или **True,** вне зависимости от того, содержит ли список пустые значения, поле со списком принимает пустое значение **(Null).** Для того, чтоб запретить ввод пустых значений в поле со списком, задайте для свойства **Обязательное поле (Required)** поля таблицы, к которой присоединено поле со списком, значение «Да».

Свойство «Автоподстановка» (AutoExpand)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅS":"acproAutoExpandC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acproAutoExpandX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproAutoExpandA"}

Свойство **Автоподстановка (AutoExpand)** позволяет указать при вводе в поле со списком автоматическое дополнение введенных символов до значения из списка. Это ускоряет процесс ввода существующих значений, при котором раскрытие списка становится необязательным.

Значения

Свойство **Автоподстановка (AutoExpand)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Введенные в поле символы автоматически дополняются до первого соответствующего им значения из списка. По мере вода новых символов выводящееся в поле значение изменяется в соответствии с введенными символами.	**True** (–1)
Нет	Пользователь должен ввести полное значение или выбрать значение из списка.	**False** (0)

Значение свойства **Автоподстановка (AutoExpand)** задается в окне свойств поля со списком, в макросе или в программе Visual Basic.

Дополнительные сведения

При вводе пользователем значения в поле со списком Microsoft Access проводит в списке поиск значений, первые символы которых совпадают с символами, уже введенными в поле. Если для свойства **Автоподстановка (AutoExpand)** задано значение «Да», то в поле выводится первое из найденных в списке значений, соответствующее введенным символам.

Если для свойства **Ограничиться списком (LimitToList)** задано значение «Да» и поле со списком раскрыто, то по мере ввода пользователем символов в поле Microsoft Access выделяет в списке соответствующие значения, даже в том случае, если для свойства **Автоподстановка (AutoExpand)** установлено значение «Нет». В том случае, если пользователь нажмет клавишу ENTER, или перейдет к следующему элементу управления или записи, выбранное значение заносится в поле со списком.

Свойство ListCount

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproListCountC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproListCountX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproListCountA"}

Свойство **ListCount** определяет число строк в списке или в поле со списком.

Значения

Свойство **ListCount** возвращает число строк в списке или в поле со списком. Значение данного свойства является доступным только для чтения и не может быть изменено пользователем.

Данное свойство используется только в макросах и в программах Visual Basic. Считать его значение можно только в режиме формы или в режиме таблицы.

Дополнительные сведения

Свойство **ListCount** содержит общее число строк в поле или в поле со списком, определяемое в свойствах **Источник строк (RowSource)** и **Тип источника строк (RowSourceType)** элемента управления. Если в основе элемента управления лежит таблица или запрос (для свойства **Тип источника строк (RowSourceType)** задано значение «Таблица/запрос», а в качестве значения свойства **Источник строк (RowSource)** указана конкретная таблицы или запрос), то значением свойства **ListCount** является число строк в результирующем наборе записей таблицы или запроса. Если в качестве значения свойства **Тип источника строк (RowSourceType)** установлен «Список значений», то значением свойства **ListCount** является число строк в списке значений, указанном свойством **Источник строк (RowSource),** зависящее от списка значений и числа столбцов в списке или в поле со списком, заданного свойством **Число столбцов (ColumnCount).**

Если свойство **Заглавия столбцов (ColumnHeads)** имеет значение «Да», то при подсчете числа строк, возвращаемого свойством **ListCount,** учитывается строка заголовков. Для полей со списками и списков, основанных на таблице или запросе, добавление заголовков столбцов приводит к добавлению еще одной строки. Для полей со списками и списков, основанных на списке значений, добавление заголовков столбцов не влияет на число строк (первая строка значений определяет заглавия столбцов).

Свойство **ListCount** вместе со свойством **Число строк списка (ListRows)** позволяет указать, сколько строк будет выводиться при раскрытии списка в поле со списком.

Свойства ListCount, «Число строк списка» (ListRows), пример

В данном примере с помощью свойства **ListCount** определяется число строк в раскрывающемся списке поля со списком «Список клиентов» в форме «Клиенты». Возвращаемое число используется при задании значения свойства **Число строк списка (ListRows),** определяющего число строк, выводящихся при раскрытии списка.

```
Sub РазмерСпискаКлиентов()
   Dim ListControl As Control

   Set ListControl = Forms!Клиенты!СписокКлиентов
   With ListControl
      If .ListCount < 8 Then
         .ListRows = .ListCount
      Else
         .ListRows = 8
      End If
   End With
End Sub
```

Свойство ListIndex

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproListIndexC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproListIndexX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproListIndexA"}

Свойство **ListIndex** определяет элемент, выбранный в списке или в поле со списком.

Значения

Значением свойства **ListIndex** является целое число в диапазоне от 0 до полного числа элементов в списке минус единица. Microsoft Access задает значение данного свойства при выборе элемента в списке или в раскрывающемся списке поля со списком. При выборе первого элемента списка свойство **ListIndex** получает значение 0, при выборе второго элемента значение 1 и т.д.

Данное свойство используется в макросах или в программах Visual Basic. Считать его значение можно в режиме формы и в режиме таблицы. Свойство **ListIndex** является доступным только для чтения и в других режимах оно недоступно.

Дополнительные сведения

Для того, чтобы сделать доступным значение свойства **ListIndex,** можно также задать значение 0 для свойства **Присоединенный столбец (BoundColumn)** списка или поля со списком. Если для свойства **Присоединенный столбец (BoundColumn)** задано значение 0, то в поле базовой таблицы, к которому присоединяется список или поле со списком, содержится значение, совпадающее со значением свойства **ListIndex.**

Для списков также определено свойство **Несвязное выделение (MultiSelect),** обеспечивающее возможность выделения нескольких элементов списка. Для того, чтобы определить, какие из элементов списка являются выделенными, следует использовать свойство списка **Selected.** Его значением является массив целых чисел, с индексами от 0 до значения свойства **ListCount** минус единица. Для каждого выделенного элемента списка соответствующий элемент массива имеет значение **True** (–1), а для невыделенных значений **False** (0).

Для доступа к данным выделенных строк списка или поля со списком также используется семейство **ItemsSelected.**

Свойство «Число строк списка» (ListRows)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproListRowsC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproListRowsX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproListRowsA"}

Свойство **Число строк списка (ListRows)** задает максимальное число строк, выводящихся в раскрывающемся списке поля со списком.

Значения

Значением свойства **Число строк списка (ListRows)** является целое число, определяющее максимальное число выводящихся строк. По умолчанию задается значение 8. Диапазон значений свойства **Число строк списка (ListRows)** от 1 до 255.

Значение данного свойства задается в окне свойств поля со списком, в макросе или в программе Visual Basic.

Для полей таблиц, свойство **Тип элемента управления (DisplayControl)** которых имеет значение «Поле со списком», значение данного свойства задается в режиме конструктора таблицы на вкладке **Подстановка** раздела «Свойства поля».

Совет. При выборе в качестве типа данных поля в режиме конструктора таблицы мастера подстановок, свойство **ListRows** задается автоматически.

В программе Visual Basic значение данного свойства задается с помощью числового выражения.

Для поля со списком используемое по умолчанию значение этого свойства можно указать в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

Если полное число строк списка превышает указанное в свойстве **Число строк списка (ListRows),** раскрывающийся список выводится с вертикальной полосой прокрутки.

Свойство «Ширина списка» (ListWidth)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproListWidthC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproListWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproListWidthA"}

Свойство **Ширина списка (ListWidth)** задает ширину раскрывающегося списка в поле со списком.

Значения

Свойство **Ширина списка (ListWidth)** задает ширину раскрывающегося списка в поле со списком в дюймах или сантиметрах, в зависимости от системы измерений (американской или метрической), указанной в поле **Система единиц** на вкладке **Числа** диалогового окна **Языки и стандарты** панели управления Windows. Для того, чтобы использовать единицы измерения, отличные от стандартных, следует указать символ единицы измерения, например «см». Значение по умолчанию («Авто») задает ширину раскрывающегося списка, равную ширине поля.

Значение свойства **Ширина списка (ListWidth)** задается в окне свойств поля со списком, в макросе или в программе Visual Basic.

Для полей таблиц, свойство **Тип элемента управления (DisplayControl)** которых имеет значение «Поле со списком», значение данного свойства задается в режиме конструктора таблицы на вкладке **Подстановка** раздела «Свойства поля».

Совет. При выборе в качестве типа данных поля в режиме конструктора таблицы мастера подстановок, свойство **Ширина списка (ListWidth)** задается автоматически.

В программе Visual Basic значение данного свойства задается с помощью числового выражения. Стандартными единицами измерения Visual Basic являются единицы твип.

Для поля со списком используемое по умолчанию значение этого свойства можно указать в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

Ширина раскрывающегося списка в поле со списком может быть больше, чем ширина поля, но меньше ее быть не может.

Если необходимо вывести список, содержащий несколько столбцов, следует задать значение ширины, достаточное для вывода требуемого числа столбцов.

Совет. При создании поля со списком не забудьте выделить достаточно места для данных для автоматически добавляемой вертикальной полосы прокрутки.

Свойство «Текст» (Text)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproTextC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproTextX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproTextA"}

Свойство **Текст (Text)** задает или возвращает текст, содержащийся в поле или в поле со списком.

Значения

Значением свойства **Текст (Text)** является текст, выводящийся в элементе управления. Данное свойство также можно использовать для обращения к тексту, находящемуся в элементе управления в текущий момент.

Значение свойства **Текст (Text)** можно считать или установить только в макросе или в программе Visual Basic.

Примечание. Для того, чтобы было можно считать или установить значение свойства **Текст (Text),** элемент управления должен иметь фокус. В противном случае возникнет ошибка. Для переноса фокуса на элемент управления используется метод **SetFocus** или макрокоманду **КЭлементуУправления (GoToControl).**

Дополнительные сведения

Пока элемент управления имеет фокус свойство **Текст (Text)** содержит текст, который находится в элементе управления в текущий момент; свойство **Значение (Value)** содержит последние сохраненные данные. При переносе фокуса на другой элемент управления происходит обновление данных и в качестве значения свойства **Значение (Value)** устанавливается новое значение. После этого свойство **Текст (Text)** недоступно до тех пор, пока элемент управления не получит фокус вновь. Если для сохранения данных элемента управления использовалась команда **Сохранить запись** из меню **Записи** без переноса фокуса, то значения свойств **Текст (Text)** и **Значение (Value)** будут одинаковыми.

Свойство «Текст» Text, пример

В следующем примере свойство **Текст (Text)** используется для включения кнопки `Следующий` после ввода текста в поле `Имя`. Пока поле пусто, кнопка отключена.

```
Sub Имя_Change()
   Следующий.Enabled = Len(Me!Имя.Text & "")<>0
End Sub
```

Свойство «Автопереход по Tab» (AutoTab)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproAutoTabC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproAutoTabX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproAutoTabA"}

Свойство **Автопереход по Tab (AutoTab)** определяет режим автоматического перехода на следующий объект при вводе в поле последнего символа, отвечающего заданной для поля маске ввода. Фокус автоматически переводится на следующий элемент управления, соответствующий установленной для формы последовательности перехода.

Значения

Свойство **Автопереход по Tab (AutoTab)** может иметь следующие значения.

Значения	Описание	Visual Basic
Да	Осуществляет автопереход при вводе в поле последнего разрешенного символа.	**True** (–1)
Нет	(Значение по умолчанию). при вводе в поле последнего разрешенного символа автопереход не происходит.	**False** (0)

Значение данного свойства задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Значение этого свойства может быть также задано в окне стандартных свойств элемента управления или с помощью метода **DefaultControl** из программы на Visual Basic.

Свойство **Автопереход по Tab (AutoTab)** определяет порядок переходов как в режиме формы, так и в режиме таблицы.

Дополнительные сведения

Маска ввод для элемента управления задается в свойстве **Маска ввода** (**InputMask**).

Кроме того, для элемента управления, присоединенного к полю, можно определить маску ввода, задавая значение свойства **Маска ввода (InputMask)** для поля базовой таблицы или запроса. Если поле в форме создается путем переноса имени поля с помощью мыши из списка полей, то элемент управления-поле наследует маску ввода, определенную для поля таблицы.

Свойство **Автопереход по Tab (AutoTab)** удобно использовать в том случае, если в каждой записи в форме в конкретное поле обычно вводится максимальное допустимое число символов. Например, такой режим подходит для поля «КодТипа», содержащего коды из пяти символов.

Свойства «Доступ» (Enabled), «Блокировка» (Locked)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproEnabledC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproEnabledX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproEnabledLockedA":1}

- **Доступ (Enabled)** – Данное свойство определяет, может ли элемент управления получить фокус в режиме формы.
- **Блокировка (Locked)** – Данное свойство определяет возможность изменения данных в элементе управления в режиме формы.

Значения

Свойство **Доступ (Enabled)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию для всех элементов управления, кроме свободной рамки объекта). Элемент управления может получить фокус. Для свободной рамки объекта двойное нажатие кнопки мыши вызывает основную команду объекта. Например, при этом может быть воспроизведен внедренный файл звукозаписи.	**True** (–1)
Нет	(Значение по умолчанию для свободной рамки объекта). Элемент управления не может получить фокус и выводится как отключенный. Значение «Нет» для свойства группы делает невозможным перевод фокуса ни на саму группу, ни на один из содержащихся в группе параметров.	**False** (0)

Свойство **Блокировка (Locked)** может принимать следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию для свободной рамки объекта). Функции элемента управления запрещают редактирование, добавление или удаление данных.	**True**
Нет	(Значение по умолчанию для всех элементов управления, кроме свободной рамки объекта). Функции элемента управления позволяют редактировать, добавлять или удалять данные.	**False**

Значение данного свойства задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Дополнительные сведения

Свойство **Доступ (Enabled)** используют для включения и отключения элементов управления. Например, в режиме формы можно отключить кнопку до тех пор, пока не будет изменено значение в поле формы. После этого кнопка включается с помощью процедуры обработки события элемента управления **После обновления (AfterUpdate)** или с помощью макроса.

Свойство **Блокировка (Locked)** позволяет защитить содержимое поля, сделав его доступным только для чтения. Например, можно заблокировать элемент управления, предназначенный

для отображения данных, либо постоянно, либо до выполнения определенного условия.

Комбинация настроек свойств **Доступ (Enabled)** и **Блокировка (Locked)** дает следующие результаты.

Доступ	**Блокировка**	**Результат**
Да	Да	Элемент управления может иметь фокус. Допускается отображение данных и их копирования; изменение данных невозможно.
Да	Нет	Элемент управления может иметь фокус. Допускается отображение данных, их копирование и изменение.
Нет	Да	Элемент управления не может иметь фокус. Допускается отображение данных; копирование и изменение данных невозможно.
Нет	Нет	Элемент управления не может иметь фокус. Сам элемент управления и находящиеся в нем данные недоступны (отключены).

Свойство **Переход по Tab** (**TabStop**) вместе со свойством **Доступ (Enabled)** позволяет запретить выделение кнопки с помощью клавиши TAB, сохранив при этом возможность выполнения действий при нажатии кнопки. Значение «Нет» свойства **Переход по Tab (TabStop)** означает, что кнопка исключена из заданной последовательности переходов между элементами управления. Однако, если при этом свойство **Доступ (Enabled)** имеет значение «Да», то при нажатии кнопки по-прежнему будут выполняться определенные для кнопки действия.

Для того чтобы сделать возможным изменение внедренного или связанного объекта в свободной рамке объекта, необходимо задать значение «Да» для свойства свободной рамки объекта **Доступ (Enabled)** и значение «Нет» для свойства **Блокировка (Locked)**.

Свойства «Доступ» (Enabled), «Блокировка» (Locked), пример

В следующем примере значения свойства кнопки **Доступ (Enabled)** и свойств элемента управления **Доступ (Enabled)** и **Блокировка (Locked)** изменятся в зависимости от значения поля «Должность» в текущей записи. Если сотрудник является директором, активизируется кнопка «СведенияОЗарплате», а также активизируется и разблокируется элемент управления «ЛичныеДанные».

```
Sub Form_Current()
   If Me!Должность = "Директор" Then
      Me!СведенияОЗарплате.Enabled = True
      Me!ЛичныеДанные.Enabled = True
      Me!ЛичныеДанные.Locked = False
   Else
      Me!СведенияОЗарплате.Enabled = False
      Me!ЛичныеДанные.Enabled = False
      Me!ЛичныеДанные.Locked = True
   End If
End Sub
```

Свойства «Доступ» (Enabled), «Блокировка» (Locked) - Применение

Свойство «Доступ» (Enabled)

Свойство «Блокировка» (Locked)

Свойство «Поведение по Enter» (EnterKeyBehavior)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproEnterKeyBehaviorC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproEnterKeyBehaviorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproEnterKeyBehaviorA"}

Свойство **Поведение по Enter (EnterKeyBehavior)** определяет действие, выполняющееся при нажатии клавиши ENTER в поле в режиме формы или в режиме таблицы. Например, значение «Перевод строки» свойства **Поведение по Enter (EnterKeyBehavior)** позволяет облегчить ввод многострочного текста в поле, присоединенное к полю MEMO таблицы. Если для данного свойства не задано значение «Перевод строки», то для создания в поле новой строки необходимо нажимать клавиши CTRL+ENTER.

Значения

Свойство **Поведение по Enter (EnterKeyBehavior)** может иметь следующие значения.

Значение	Описание	Visual Basic
По умолчанию	(Значение по умолчанию). Microsoft Access использует значения, заданные в разделе **Переход по Enter** вкладки **Клавиатура** диалогового окна **Параметры**. Это диалоговое окно выводится при выборе элемента **Параметры** из меню **Сервис**. Подробнее см. в разделе «Дополнительные сведения».	**False** (0)
Перевод строки	При нажатии клавиши ENTER в поле создается новая строка, позволяющая вводить дополнительный текст.	**True** (–1)

Значение данного свойства задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Можно также задать значение этого свойства в окне стандартных свойств элемента управления или с помощью метода **DefaultControl** из программы на Visual Basic.

Дополнительные сведения

Группа **Переход при нажатии Enter** на вкладке **Клавиатура** диалогового окна **Параметры** содержит следующие настройки.

Параметр	Описание
Нет	Нажатие клавиши ENTER не имеет последствий.
Следующее поле	Нажатие ENTER приводит к переводу курсора на следующий элемент управления или поле в форме или таблице согласно последовательности перехода.
Следующая запись	Нажатие ENTER приводит к переводу курсора на первый элемент управления или поле в следующей записи форы или таблицы.

Свойство «Блокировка записей» (RecordLocks)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproRecordLocksC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproRecordLocksX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproRecordLocksA"}

Свойство **Блокировка записей (RecordLocks)** определяет способы блокировки записей и их реализацию при попытке двух пользователей одновременно изменить одну и ту же запись. Когда один пользователь изменяет запись, Microsoft Access может автоматически заблокировать эту запись, чтобы запретить другим пользователям изменять эту запись до завершения работы с ней первого пользователя.

- Формы – Данное свойство определяет способы блокировки записей в базовой таблице или запросе при обновлении содержимого сетевой базы данных.
- Отчеты – Данное свойство определяет, блокируются ли записи в базовой таблице или запросе при печати или просмотре отчета.
- Запросы – Данное свойство определяет, блокируются ли записи в запросе (обычно в запросе на изменение в сетевой базе данных) при выполнении запроса.

Значения

Свойство **Блокировка записей (RecordLocks)** может иметь следующие значения.

Значение	Описание	Visual Basic
Отсутствует	(Значение по умолчанию). Применяется в формах, которые могут изменяться одновременно несколькими пользователями. Такую блокировку называют также «нежесткой». Если два пользователя пытаются сохранить изменения одной и той же записи, то Microsoft Access выведет соответствующее сообщение пользователю, который предпринял такую попытку вторым. Ему предоставляется следующие возможности: отменить внесенные изменения, скопировать запись в буфер обмена или переписать запись, измененную другим пользователем. Данное значение обычно применяется для форм, которые доступны только для чтения, или для форм в базах данных, рассчитанных на одного пользователя. Этот режим используется также в сетевых базах данных, чтобы разрешить нескольким пользователям одновременно вносить изменения в одну и ту же запись. В отчете записи не блокируются на время печати или предварительного просмотра. В запросах записи не блокируются на время выполнения.	0
Всех записей	Блокируются все записи в базовой таблице или запросе при открытии формы в режиме формы или в режиме таблицы, во время просмотра или печати	1

	отчета и при выполнении запроса. Другим пользователям разрешается просматривать записи, но они не могут изменять, добавлять или удалять любые записи до закрытия формы, до завершения печати отчета или до завершения запроса.	
Изменяемой записи	(Только в формах и запросах). Страница записей блокируется, как только любой из пользователей начинает вносить изменения в любое поле одной из этих записей, и остается заблокированной до тех пор, пока запись не будет сохранена. Одна запись может изменяться одновременно только одним пользователем. Такую блокировку иногда называют «жесткой».	2

Значение данного свойства задается в окне свойств объекта, в макросе или в программе Visual Basic.

Дополнительные сведения

Пользуйтесь значением «Отсутствует» для форм в том случае, если только один пользователь работает с базовыми таблицами или запросами и вносит все изменения в базу данных.

В сетевой базе данных значение «Отсутствует» можно использовать в случае, когда требуется предоставить возможность нескольким пользователям изменять одну запись и достаточно ограничиться предупреждением, что другой пользователь также изменил запись. Значение «Изменяемой записи» следует использовать, если требуется запретить одновременное редактирование записи разными пользователями.

Настройка «Всех записей» обеспечивает полный запрет любых изменений данных после начала просмотра или печати отчета или после запуска запроса на добавление, на удаление, на создание таблицы или на обновление.

В режиме формы и режиме таблицы в области выделения каждой заблокированной записи изображается символ блокировки  .

Примечание. Для того чтобы изменить стандартные настройки свойства **Блокировка записей (RecordLocks)** выберите в меню **Сервис** команду **Параметры.** В диалоговом окне **Параметры** выберите вкладку **Другие** и выберите нужный параметр в группе **Блокировка по умолчанию**.

Данные в форме, отчете или запросе базы данных, связь с которыми осуществляется по протоколу ODBC, обрабатываются таким образом, как будто задана настройка блокировок «Отсутствует», вне зависимости от текущего значения свойства **Блокировка записей (RecordLocks)**.

Свойство «Тип набора записей» (RecordsetType)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproRecordsetTypeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproRecordsetTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproRecordsetTypeA"}

Свойство **Тип набора записей (RecordsetType)** определяет тип набора записей, выводящегося в форме. Например, если требуется запретить изменение данных в присоединенном элементе управления в форме в режиме формы или в режиме таблицы, следует выбрать для свойства **Тип набора записей (RecordsetType)** значение «Статический набор».

Значения

Свойство **Тип набора записей (RecordsetType)** может иметь следующие значения.

Значения	Описание	Visual Basic
Динамический набор	(Значение по умолчанию). Допускается изменение присоединенных элементов управления в единственной таблице или в нескольких таблицах со связями типа «один-к-одному». Не допускается изменение данных для элементов управления присоединенных к полям таблиц со связями типа «один-ко-многим» со стороны «один», если не разрешено каскадное изменение между таблицами. Дополнительные сведения см. в разделе, посвященном возможности изменения записей с помощью запроса.	0
Динамический набор (Несогл.)	Допускается редактирование всех таблиц и элементов управления, присоединенных к их полям .	1
Статический набор	Не допускаются изменения данных в таблицах и в присоединенных к их полям элементах управления.	2

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Дополнительные сведения

Пользователь имеет возможность создавать формы, базирующиеся на нескольких таблицах, и присоединять элементы управления к полям этих таблиц. Значение свойства **Тип набора записей (RecordsetType)** позволяет ограничить элементы управления, в которых допускается изменение данных.

Наряду со свойством **Тип набора записей (RecordsetType)** каждый элемент управления в форме имеет свойство **Блокировка (Locked),** которое также позволяет запретить изменение данных в элементе управления и базовых таблицах. Если для свойства **Блокировка (Locked)** задано значение «Да», изменение данных становится невозможным.

Свойство «Тип набора записей» (RecordsetType), пример

В следующем примере обновление данных разрешается только для пользователя с именем «ADMIN». Если имя пользователя, передаваемое в общей переменной `gstrUserID` не совпадает с именем «ADMIN», для свойства **Тип набора записей (RecordsetType)** задается значение «Статический набор».

```
Sub Form_Open(Cancel As Integer)
   Const conSnapshot = 2
   If gstrUserID <> "ADMIN" Then
      Forms!Сотрудники.RecordsetType = conSnapshot
   End If
End Sub
```

Свойство «Значение» (Value)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproValueC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproValueX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproValueA"}

Свойство **Значение (Value)** позволяет для выделенного элемента управления определить или указать выбранное значение или параметр в элементе управления или текст, содержащийся в поле.

- Флажки, переключатели, выключатели – Данное свойство позволяет определить или указать, является ли данный элемент управления выделенным.
- Поле со списком, список группа – Данное свойство позволяет определить или указать, какое значение или какой параметр является выбранным.
- Поле – Данное свойство позволяет определить или указать текст, содержащийся в поле.
- Набор вкладок – Данное свойство позволяет определить или указать объект **Page.**

Значения

Свойство **Значение (Value)** может иметь следующие значения, зависящие от типа элемента управления.

Элемент	Содержимое	Описание	Visual Basic
Флажок	**True**	Флажок установлен.	**True** (–1)
	False	(Значение по умолчанию). Флажок снят.	**False** (0)
Поле со списком	[Текст, выводящийся в поле].	Данное значение может как совпадать, так и не совпадать со значением свойства поля со списком **Text.** Текущим значением свойства **Text** является значение, выводящееся в верхней части (в поле) поля со списком; свойству **Значение (Value)** присваивается значение свойства **Text** только после того, как этот текст будет сохранен.	
Список	[Значение элемента списка]	Значение, содержащееся в присоединенном столбце .	
Переключатель	**True**	Переключатель установлен.	**True**
	False	(Значение по умолчанию). Переключатель снят.	**False**
Группа	[Значение свойства **Значение параметра** (**OptionValue**)]	Значение свойства **Значение параметра (OptionValue)** для выбранного элемента группы.	
Поле	[Значение свойства **Text** элемента управления]	Свойство **Text** возвращает отформатированную строку. Значение свойства **Text** может отличаться от значения	

		свойства **Значение (Value)** для поля. Значение свойства **Text** задает текущее содержимое элемента управления, а свойство **Значение (Value)** – сохраненное значение поля. Свойство **Text** всегда является текущим, если фокус принадлежит данному элементу управления.	
Выключатель	**True**	Выключатель нажат.	**True**
	False	Выключатель не нажат.	**False**
Набор вкладок	[Значение типа **Integer,** представляющее индексный номер текущего выбранного объекта **Page**]	Свойство **Значение (Value)** набора вкладок равно индексному номеру текущего выбранного объекта **Page**. Для каждой вкладки набора существует объект **Page**. Первому объекту **Page** всегда присваивается номер 0, второму – 1, и т. д.	
Присоединенная рамка объекта или диаграмма		Свойство **Значение (Value)** для присоединенной рамки объекта или диаграммы приравнивается значению поля, к которому присоединен объект. Так как эти поля обычно содержат объекты OLE или объекты-диаграммы, то значение свойства не представляет никакого интереса.	
Элементы ActiveX		Некоторые элементы ActiveX поддерживают свойство **Значение (Value)**. Например, это свойство устанавливается для элемента управления «Календарь». Дополнительные сведения по данному вопросу см. в документации по конкретному элементу ActiveX.	

Значение данного свойства задается только в макросе или в программе Visual Basic.

Дополнительные сведения

В свойстве **Значение (Value)** возвращает или устанавливает значение стандартного свойства элемента управления, т.е. свойства, для ссылки на которое не обязательно явно указывать его название. Например, для элемента управления-поля стандартным является свойство **Text**, поэтому допускается ссылка на значение этого свойства без явного указания его названия:

```
Forms!Клиенты!Фамилия = "Иванов"
```

Это означает, что следующие две инструкции являются эквивалентными:

```
Forms!Клиенты!Кредит.Value = True
```

```
Forms!Клиенты!Кредит = True
```

Примечание. Не следует путать свойство **Value** со свойством **Значение по умолчанию** (**DefaultValue**), в котором задается значение, присваиваемое элементу управления по умолчанию при создании новой записи.

Свойство «Значение» (Value), пример

В данном примере демонстрируется вызов одной из двух процедур в зависимости от состояния флажка **Кредит** в форме "Клиенты".

```
Sub СпособОплаты()
   If Forms!Клиенты!Кредит.Value = False Then
      ОплатаНаличными
   ElseIf Forms!Клиенты!Кредит.Value = True Then
      ПредоставлениеКредита
   End If
End Sub
```

Свойство OldValue

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproOldValueC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproOldValueX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproOldValueA"}

Свойство **OldValue** определяет исходное (неизменное) значение присоединенного элемента управления.

Значения

Свойство **OldValue** содержит данные, которые присоединенный элемент управления имел до начала редактирования. Это свойство во всех режимах доступно только для чтения.

Доступ к данному свойству возможен только в макросе или в программе Visual Basic.

Значение свойства **OldValue** присваивается переменной в следующей конструкции:

```
OriginalValue = Forms!Клиенты!Уплачено.OldValue
```

Дополнительные сведения

Microsoft Access использует свойство **OldValue** для загрузки значения присоединенного элемента управления из сохраненной записи. При редактировании присоединенного элемента в форме внесенные изменения не заносятся в базу данных до перехода к следующей записи. Свойство **OldValue** содержит прежние (неизменные) основные данные.

Использование свойства **OldValue** дает дополнительную возможность отмены изменений, внесенных в элемент управления. В следующем примере показано, как можно восстановить прежнее значение элемента управления-поля формы:

```
Sub Отмена_Click()
   Dim ctlTextbox As Control
   For Each ctlTextbox in Me.Controls
   If ctlTextbox.ControlType = acTextBox Then
      ctlTextbox.Value = ctl.OldValue
   End If
   Next ctl
End Sub
```

Если значение элемента управления не было изменено, эта конструкция не будет иметь последствий. После перемещения на следующую запись происходит обновление источника записей, после чего текущее значение поля и значение свойства **OldValue** становятся одинаковыми.

Значение свойства **OldValue** имеет тот же тип данных, что и поле, с которым связан элемент управления.

Свойство OldValue, пример

В данном примере проверяется, не превышает ли разность между введенным в поле и исходным значениями 10 процентов от исходного значения. Если изменение слишком велико, то с помощью свойства **OldValue** восстанавливается исходное значение. Для вызова этой процедуры может использоваться событие **До обновления (BeforeUpdate)** элемента управления, содержащего проверяемые данные.

```
Sub Validate_Field()
   Dim curNewValue As Currency
   Dim curOriginalValue As Currency
   Dim curChange As Currency, strMsg As String

   curNewValue = Forms!Товары!Цена
   curOriginalValue = Forms!Товары!Цена.OldValue
   curChange = Abs(curNewValue - curOriginalValue)
   If curChange > (curOriginalValue * .1) Then
      strMsg = "Изменение цены превышает 10 процентов от исходного
значения. " _
      & "Восстанавливается исходная цена."
      MsgBox strMsg, vbExclamation, "Неверные изменения"
      Forms!Товары!Цена = curOriginalValue
   End If
End Sub
```

Свойства SelLength, SelStart, SelText

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproSelLengthC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproSelLengthX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproSelLengthA"}

Данные свойства используются при определении позиции курсора, установлении диапазона выделения, выделении части строки элементов управления поле и поле со списком и при удалении текста.

- Свойство **SelLength** задает или возвращает число символов, выделенных в поле или в поле со списком.
- Свойство **SelStart** задает или возвращает позицию начала выделенного блока текста или позицию курсора, если выделенного текста нет.
- Свойство **SelText** возвращает строку, содержащую выделенный текст, или **пустую строку,** если выделенного текста нет.

Значения

- В свойствах **SelLength** и **SelStart** используются данные типа **Long Integer** в диапазоне от 0 до полного числа символов в поле или в поле со списком.
- В свойстве **SelText** используется строковое выражение, содержащее выделенный в элементе управления текст. Выделенный в элементе управления текст замещается значением свойства **SelText.**

Значения свойств **SelLength** и **SelStart** задаются в макросе или в программе Visual Basic. Свойство **SelText** является доступным только для чтения.

Дополнительные сведения

Для того, чтобы задать или получить значения этих свойств, элемент управления должен иметь фокус. Для перемещения фокуса на элемент управления используется метод **SetFocus.**

Отрицательные значения свойства **SelLength** приводят к ошибке выполнения.

При попытке задать для свойства **SelStart** значение, превышающее число содержащихся в элементе управления символов, значение свойства автоматически устанавливается равным числу символов в поле.

При изменении значения свойства **SelStart** снимается выделение, курсор помещается в текст и свойство **SelLength** получает значение 0.

Свойства SelLength, SelStart, SelText, пример

В данном примере приведены две процедуры обработки событий, которые используются для поиска текста, указанного пользователем. Текст, в котором должен быть выполнен поиск, задается в процедуре обработки события формы **Загрузка (Load).** Процедура обработки события **Нажатие кнопки (Click),** определенная для кнопки «Поиск» (нажатие которой запускает операцию поиска), запрашивает у пользователя искомый текст и выделяет его в случае успешного поиска.

```
Sub Form_Load()
   Dim ctlTextToSearch As Control
   Set ctlTextToSearch = Forms!Форма1!Поле1
   ctlTextToSearch.SetFocus    ' Переводит фокус на поле.
   ctlTextToSearch.Text = "Эта компания дважды в год размещает огромные
заказы " _
      & "на чеснок, петрушку и укроп."
End Sub

Sub Поиск_Click()
   Dim strSearch As String, intWhere As Integer
   Dim ctlTextToSearch As Control
   ' Принимает от пользователя образец поиска.
   With Me!Поле1
      strSearch = InputBox("Введите образец поиска:")
      ' Поиск строки в тексте.
      intWhere = InStr(.Value, strSearch)
      If intWhere Then
         ' Строка найдена.
         .SetFocus
         .SelStart = intWhere - 1
         .SelLength = Len(strSearch)
      Else
         ' Уведомление пользователя.
         MsgBox "Строка не найдена."
      End If
   End With
End Sub
```

Свойство «Значение по умолчанию» (DefaultValue)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDefaultValueC;daproDefaultValue"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDefaultValueX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDefaultValueA"}

Свойство **Значение по умолчанию (DefaultValue)** позволяет указать значение, автоматически вводящееся в поле при создании новой записи. Например, в таблице «Адреса» может оказаться удобным указать автоматический ввод значения «Москва» в поле «Город». При заполнении таблицы пользователи смогут оставить в этом поле стандартное значение или, при необходимости, указать другой город.

Примечания

- Свойство **Значение по умолчанию (DefaultValue)** не определено для следующих элементов управления: флажки, переключатели и выключатели, входящие в группу параметров. Оно определено только для самой группы.
- Свойство **Значение по умолчанию (DefaultValue)** определено для всех типов полей в таблицах за исключением полей с типом данных «Счетчик» или «Поле объекта OLE».

Значения

В свойстве **Значение по умолчанию (DefaultValue)** задается текст или выражение, значение которого автоматически вводится в поле при создании новой записи. Например, если в свойстве поля **Значение по умолчанию (DefaultValue)** задается выражение `=Now()`,то в поле автоматически выводятся текущие значения даты и времени. Максимальная длина текста или выражения, задающего значение этого свойства, составляет 255 символов.

Значение данного свойства для элемента управления задается в окне его стандартных свойств. Для поля значение данного свойства можно установить в режиме конструктора таблицы (в области свойств поля), в макросе или в программе Visual Basic.

В программе Visual Basic значение данного свойства задается с помощью строкового выражения. Например, следующая инструкция задает для свойства **Значение по умолчанию (DefaultValue)** поля «СпособОплаты» значение «Наличные»:

```
Forms!Счет!СпособОплаты.DefaultValue = """Наличные"""
```

Примечание. Для того чтобы определить значение данного свойства для поля в программе Visual Basic, следует использовать свойство **DefaultValue** объектов доступа к данным.

Дополнительные сведения

Свйство **Значение по умолчанию (DefaultValue)** используется только при создании новой записи. Изменение значения данного свойства не влияет на существующие записи.

Если задать значение свойства **Значение по умолчанию (DefaultValue)** для элемента управления, связанного с полем таблицы, для которого также определено свойство **Значение по умолчанию (DefaultValue),** то настройка элемента управления имеет приоритет над настройкой таблицы.

Если элемент управления создается путем переноса поля с помощью мыши из списка полей, то для элемента управления используется значение свойства **Значение по умолчанию (DefaultValue),** определенное для поля базовой таблицы, хотя свойство **Значение по умолчанию (DefaultValue)** этого элемента управления остается пустым.

В качестве значения по умолчанию для элемента управления можно указать значение другого элемента управления. Например, если задать в свойстве вычисляемого поля **Значение по умолчанию (DefaultValue)** следующее выражение, то значением по умолчанию для данного поля станет значение, указанное в свойстве **Значение по умолчанию (DefaultValue)** поля `«Образец»`.

```
=Forms!Форма1!Образец
```

Если оба элемента управления находятся в одной форме, то необходимо, чтобы элемент управления, служащий источником значения по умолчанию, находился в определенной для формы последовательности перехода перед элементом управления, принимающим это значение.

Свойство «Формат поля» (Format)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFormatC;vafctFormat"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproFormatX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproFormatA"}

Свойство **Формат поля (Format)** позволяет указать форматы вывода текста, чисел, дат и значений времени на экран и на печать. Например, для поля «Цена» разумно указать в свойстве **Формат поля (Format)** формат «Денежный» и установить для его свойства **Число десятичных знаков (DecimalPlaces)** значение 2 или «Авто». В этом случае введенное в поле значение 4321,678 будет отображаться как 4 321,68р.

Допустимо как использование встроенных, так и специальных форматов, созданных при помощи символов форматирования.

Значения

В свойстве **Формат поля (Format)** задаются разные настройки для различных типов данных. Более подробное описание конкретных настроек см. в одном из следующих разделов.

- Поля даты/времени
- Числовые и денежные поля
- Текстовые и MEMO-поля
- Логические поля

Для элемента управления значение данного свойства задается в окне свойств. Для поля в таблице или запросе значение данного свойства задается в режиме конструктора таблицы (в разделе свойств поля) или в окне запроса (в окне свойств поля). Кроме того, значение данного свойства можно задать в макросе или в программе Visual Basic.

Примечание. В программе Visual Basic следует ввести строковое выражение, соответствующее одному из стандартных форматов или задающее специальный формат.

Дополнительные сведения

Свойство **Формат поля (Format)** определяет только способ отображения данных. Оно не влияет на способ их сохранения.

В Microsoft Access определены стандартные форматы для полей с типами данных «Числовой», «Дата/время», «Логический», «Текстовый» и «Поле MEMO». В качестве стандартных используются национальные форматы, выбираемые в окне **Язык и стандарты** панели управления Windows. Набор форматов определяется настройками для конкретной страны. Например, если на вкладке **Язык и стандарты** указать **Английский (США),** то число 1234.56 в денежном формате будет выглядеть как $1,234.56. Но если указать на этой вкладке **Русский,** то это число будет выглядеть так: 1 234,56р.

Настройка **Формат поля (Format),** заданная в режиме конструктора таблицы, используется для отображения данных в режиме таблицы. Эта же настройка применяется при создании связанных с этим полем новых элементов управления в форме или отчете.

Ниже перечислены символы, используемые при определении специальных форматов для любого типа данных.

Символ	Значение
(Пробел)	Выводит пробел как символьную константу.
"ABC"	Все символы внутри кавычек считаются символьными константами.
!	Выравнивает символы по левому, а не по правому краю.
*	Заполняет доступное пустое пространство следующим

	символом.
\	Выводит следующий символ как символьную константу. Для этой же цели можно использовать кавычки.
[*цвет*]	Задает цвет, название которого указано в скобках. Допустимые имена цветов: Черный (Black), Синий (Blue), Зеленый (Green), Бирюзовый (Cyan), Красный (Red), Лиловый (Magenta), Желтый (Yellow), Белый (White).

Не разрешается смешивать в одном формате специальные символы, предназначенные для определения числовых форматов, форматов даты/времени и текстовых форматов.

Если для поля определена маска ввода, а в свойстве **Формат поля (Format)** задается другое форматирование тех же данных, то приоритет имеют настройки, задаваемые в свойстве **Формат поля (Format)** и маска ввода игнорируется. Например, если в режиме таблицы определяется маска ввода пароля одновременно с указанием значения свойства **Формат поля (Format)** либо для таблицы, либо для элемента управления в форме, то маска ввода пароля будет игнорироваться, а данные отображаться в соответствие со значением свойства **Формат поля (Format).**

Свойство «Формат поля» (Format), пример

В следующих трех инструкциях значение свойства **Формат поля (Format)** задается с помощью стандартных форматов:

```
Me!Дата.Format = "Medium Date"

Me!Время.Format = "Long Time"

Me!Зарегистрировано.Format = "Yes/No"
```

В следующей инструкции значение свойства **Формат поля (Format)** задается с помощью специального формата. Дата будет выведена в формате: янв 1995.

```
Forms!Сотрудники!ДатаНайма.Format = "mmm yyyy"
```

В следующей программе демонстрируется функция Visual Basic, задающая вывод числовых данных в денежном формате и вывод текстовых значений прописными буквами. Функция вызывается в свободном элементе управления с именем «Налог» при возникновении события **Потеря фокуса (LostFocus).**

```
Function FormatValue() As Integer
   Dim varEnteredValue As Variant

   varEnteredValue = Forms!Сводка!Налог.Value
   If IsNumeric(varEnteredValue) = True Then
      Forms!Сводка!Налог.Format = "Currency"
   Else
      Forms!Сводка!Налог.Format = ">"
   End If
End Function
```

Свойство «Формат поля» (Format) - Числовые и денежные поля

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFormatNumberC;vafctFormat"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproFormatNumberX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproFormatNumberA"}

Свойство **Формат поля (Format)** позволяет указать использование встроенных числовых форматов для числовых и денежных типов данных.

Значения

Встроенные форматы

В следующей таблице приводятся встроенные значения свойства **Формат поля (Format)** для числовых полей.

Значение	Описание
Основной	(Значение по умолчанию). Числа отображаются так, как они были введены.
Денежный	Используются разделители групп разрядов; отрицательные числа выводятся в круглых скобках; свойство **Число десятичных знаков (DecimalPlaces)** по умолчанию получает значение 2.
Фиксированный	Выводится по крайней мере один разряд; свойство **Число десятичных знаков (DecimalPlaces)** по умолчанию получает значение 2.
С разделителями разрядов	Числа выводятся с разделителями групп разрядов; свойство **Число десятичных знаков (DecimalPlaces)** по умолчанию получает значение 2.
Процентный	Значение умножается на 100; добавляется символ процентов (%);свойство **Число десятичных знаков (DecimalPlaces)** по умолчанию получает значение 2.
Экспоненциальный	Числа выводятся в экспоненциальной нотации.

Специальные форматы

Специальные числовые форматы могут включать в себя от одного до четырех разделов, отделенных друг от друга точкой с запятой (;). Каждый формат содержит спецификацию для различных типов числовых данных.

Раздел	Описание
Первый	Формат положительных чисел.
Второй	Формат отрицательных чисел.
Третий	Формат нулевых значений.
Четвертый	Формат пустых **(Null)** значений.

Например, возможно использование следующего специального денежного формата:

```
# ##0,00 р.;-# ##0,00 р. [Красный]
```

Этот формат определяется двумя компонентами, разделяемыми точкой с запятой, которые определяют вывод положительных и отрицательных значений.

Если описано несколько разделов, но формат каждого из них не указан, в результате либо не будет никакого форматирования, либо будет использовано форматирование первого из них.

Специальные числовые форматы определяются с помощью следующих символов.

Символ	Описание
. (точка)	Десятичный разделитель. Символ десятичного разделителя выбирается в окне **Язык и стандарты** панели управления Windows.
, (запятая)	Разделитель групп разрядов.
0	Прототип разряда. Выводится цифра или 0.
#	Прототип разряда. Выводится цифра или ничего не выводится.
$	Выводится символ "$".
%	Процентный формат. Число умножается на 100 и к нему добавляется символ процентов.
E– или e–	Экспоненциальная нотация (перед отрицательными показателями степени изображается знак минус, а перед положительными ничего). Этот символ используется только вместе с другими символами (например, 0.00E-00).
E+ or e+	Экспоненциальная нотация (перед отрицательными показателями степени изображается знак минус, а перед положительными знак плюс). Этот символ используется только вместе с другими символами (например, 0.00E+00).

Дополнительные сведения

Свойство **Число десятичных знаков (DecimalPlaces)** позволяет вывести число с количеством знаков в дробной части, отличным от задающегося в свойстве **Формат поля (Format).**

Свойство «Формат поля» (Format) - Числовые и денежные поля, пример

Ниже приводятся примеры встроенных числовых форматов.

Формат	Значение	Выводится как
Основной	3456,789 –3456,789 213,21 р.	3456,789 –3456,789 213,21
Денежный	3456,789 –3456,789	3,457 р. -3,457 р.
Фиксированный	3456,789 –3456,789 3,56645	3456,79 –3456,79 3.57
С разделителями разрядов	3456,789	3,456,79
Процентный	3 0,45	300,00% 45,00%
Экспоненциальный	3456,789 –3456,789	3,46E+03 –3,46E+03

Ниже приводятся примеры специальных числовых форматов.

Формат	Описание
0;(0);;"Пусто"	Стандартный вывод положительных чисел; вывод отрицательных чисел в скобках; вместо нулевых значений не выводится ничего; пустые значения отображаются словом «Пусто».
+0.0;–0.0;0.0	Положительные и отрицательные значения выводятся со знаками (+) или (-); нулевые и пустые значения отображаются как 0.0.

Свойство «Формат поля» (Format) - Поля даты/времени

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFormatDateC;vafctFormat"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFormatDateX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproFormatDateA"}

Свойство **Формат поля (Format)** позволяет указать использование встроенных или специальных числовых форматов для полей даты/времени.

Значения

Встроенные форматы

В следующей таблице приводятся встроенные значения свойства **Формат поля (Format)** для полей даты/времени.

Значение	Описание
Полный формат даты	(Значение по умолчанию). Если значение содержит только дату, то время не отображается; если значение содержит только время, то дата не отображается. Данный формат является комбинацией двух: «Краткий формат даты» и «Длинный формат времени». Примеры: 01.11.95 1:07:19 и 23.01.96 23:01:04.
Длинный формат даты	Совпадает с настройкой «Полный формат», задающейся в окне **Язык и стандарты** панели управления Windows. Пример: 1 Июнь 1995 г.
Средний формат даты	Пример: 03-апр-95.
Краткий формат даты	Совпадает с настройкой «Краткий формат даты», задающейся в окне **Язык и стандарты** панели управления Windows. Пример: 11.06.95. **Внимание.** Значения краткого формата даты предполагают, что даты из диапазона 01.01.00 и 31.12.29 относятся к двадцать первому веку (то есть предполагаются года от 2000 до 2029). Также предполагают, что даты из промежутка 01.01.30 и 31.12.99 к двадцатому веку (то есть предполагаются года от 1930 до 1999).
Длинный формат времени	Совпадает с форматом времени, задающемся в окне **Время Язык и стандарты** на вкладке **Время** панели управления Windows. Пример: 20:58:10.
Средний формат времени	Пример: 05:34 PM.
Краткий формат времени	Пример: 17:34.

Специальные форматы

Специальные форматы даты и времени определяются с помощью следующих символов.

Символ	Описание
: (двоеточие)	Разделитель компонентов времени. Символ разделителя

	выбирается в окне **Язык и стандарты** панели управления Windows.
/	Разделитель компонентов даты.
c	Задает встроенный «Полный формат даты».
d	Номер дня месяца, состоящий из 1 или 2 цифр (1-31).
dd	Номер дня месяца, состоящий из 2 цифр (01-31).
ddd	Сокращенное название дня недели (Пн-Вс).
dddd	Полное название дня недели (понедельник-воскресенье).
ddddd	Задает встроенный «Краткий формат даты».
dddddd	Задает встроенный «Длинный формат даты».
w	Номер дня недели (1-7).
ww	Номер недели в году (1-53).
m	Номер месяца, состоящий из 1 или 2 цифр (1-12).
mm	Номер месяца, состоящий из 2 цифр (01-12).
mmm	Первые три буквы названия месяца (янв-дек).
mmmm	Полное название месяца (Январь-Декабрь).
q	Номер квартала в году (1-4).
y	Номер дня в году (1-366).
yy	Последние две цифры номера года (01-99).
yyyy	Полный номер года (0100-9999).
h	Число часов, состоящее из 1 или 2 цифр (0-23).
hh	Число часов, состоящее из 2 цифр (00-23).
n	Число минут, состоящее из 1 или 2 цифр (0-59).
nn	Число минут, состоящее из 2 цифр (00-59).
s	Число секунд, состоящее из 1 или 2 цифр (0-59).
ss	Число секунд, состоящее из 2 цифр (00-59).
ttttt	Задает встроенный «Длинный формат времени».
AM/PM	12-часовой формат времени с добавлением прописных букв «AM» или «PM».
am/pm	12-часовой формат времени с добавлением строчных букв «am» или «pm».
A/P	12-часовой формат времени с добавлением прописных букв «A» или «P».
a/p	12-часовой формат времени с добавлением строчных букв «a» или «p».
AMPM	12-часовой формат времени; используется индикатор «утро/день», выбранный в окне **Язык и стандарты** панели управления Windows.

Специальные форматы выводятся в соответствии со значениями, установленными в окне **Язык и стандарты** панели управления Windows. Специальные форматы, противоречащие настройкам окна **Язык и стандарты** игнорируются.

Примечание. Для добавления в специальный формат запятой или другого символа разделителя следует ввести этот символ в кавычках: mmm d", "yyyy.

Свойство «Формат поля» (Format) - Поля даты/времени, пример

Ниже приводятся примеры специальных форматов даты/времени.

Формат	Выводится как
ddd", "mmm d", "yyyy	Пн, июн 2, 1996
mmmm dd", "yyyy	Июнь 02, 1996
"Неделя с номером "ww	Неделя с номером 22
"Сегодня "dddd	Сегодня среда

Для вывода символов «A.D.» перед годом или символов «B.C.» после года, в зависимости от знака введенного числа, допустимо использование специального формата. Для того, чтобы посмотреть результат применения данного формата, создайте в таблице новое поле, задайте для него числовой тип данных и введите следующий формат:

"A.D. " #;# " B.C."

Положительные числа выводятся как года с предшествующими символами «A.D.». Отрицательные числа выводятся как года с последующими символами «B.C.».

Свойство «Формат поля» (Format) - Логические поля

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFormatYesC;vafctFormat"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproFormatYesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproFormatYesA"}

Свойство **Формат поля (Format)** позволяет указать использование встроенных форматов для логических полей: Истина/Ложь, Да/Нет или Вкл/Выкл.

Значения

В Microsoft Access элементами управления, используемыми по умолчанию для логических полей, являются флажки. При их использовании игнорируются встроенные и специальные форматы. Таким образом, эти форматы применимы только к данным, содержащимся в полях.

Встроенные форматы

Свойство **Формат поля (Format)** позволяет выбрать встроенный или специальный формат отображения логических значений: «Истина/Ложь», «Да/Нет» или «Вкл/Выкл». Значения «Истина», «Да» и «Вкл» эквивалентны логическому значению **True,** а значения «Ложь», «Нет» и «Выкл» эквивалентны логическому значению **False.** Если пользователь выбирает встроенный формат, а затем вводит эквивалентное логическое значение, то введенное значение отображается в выбранном формате. Например, если значение **True** или «Вкл» вводится в элемент управления-поле, для которого в свойстве **Формат поля (Format)** указан формат «Да/Нет», то введенное значение автоматически преобразуется в «Да».

Специальные форматы

Специальный логический формат может сдержать от одного до трех разделов, разделяемых точкой с запятой (;).

Раздел	Описание
Первый	Первый раздел не влияет на формат логических значений. Однако символ точки с запятой (;) указать необходимо.
Второй	Задает строковое значение, заменяющее «Да», «Истина» или «Вкл».
Третий	Задает строковое значение, заменяющее «Нет», «Ложь» или «Выкл».

Свойство «Формат поля» (Format) - Логические поля, пример

В следующем примере демонстрируется специальный логический формат, в котором логическая истина представлена словом «Всегда», выводящимся синим цветом, а логическая ложь словом «Никогда», выводящимся красным цветом.

```
;"Всегда"[Синий];"Никогда"[Красный]
```

Свойство «Формат поля» (Format) - Текстовые и MEMO-поля

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFormatTextC;vafctFormat"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFormatTextX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproFormatTextA"}

Свойство **Формат поля (Format)** позволяет создавать специальные форматы для текстовых и MEMO-полей с помощью специальных символов.

Значения

Для создания специальных форматов текстовых и MEMO-полей используются следующие символы.

Символ	Описание
@	Обязательный текстовый символ или пробел.
&	Необязательный текстовый символ.
<	Преобразует все символы в строчные.
>	Преобразует все символы в прописные.

Специальные форматы для текстовых полей и полей MEMO могут включать один или два раздела, разделяемых точкой с запятой (;). Эти разделы описывают спецификации формата различных данных в поле.

Раздел	Описание
Первый	Формат отображения текста.
Второй	Формат отображения строк нулевой длины и пустых **(Null)** значений.

Например, для помещения в элемент управления- поле слова «Пусто», когда в поле отсутствует строка, в качестве значения свойства **Формат поля (Format)** допустимо применение специального формата **@;"Пусто".** Символ «@» определяет отображение текста, введенного в поле, а слово «Пусто» - строку нулевой длины или пустое (Null) значение поля.

Примечание. В Microsoft Access версии 2.0 для возвращения одного значения для строки нулевой длины и другого для пустой **(Null)** строки допустимо использование функции **Format,** а для автоматического форматирования полей таблицы в режиме таблицы или элементов управления в формах или в отчетах – использование свойства **Формат поля (Format).**

В Microsoft Access 97 необходима отдельная проверка на пустые **(Null)** значения и возвращаемое значение зависит от ее результата. Например, в выражениях можно было одновременно использовать функции **IIf** и **Format,** как в следующем примере:

```
var = IIf(IsNull(strX),"Пусто", Format(strX, "@;СНД"))
```

Это изменение применимо только в случае использования для форматирования строк функции **Format,** в зависимости от того, является ли эта строка строкой нулевой длины или пустым **(Null)** значением. Прочие форматирующие выражения, работающие с функцией **Format,** работают также, как и в предыдущих версиях.

Установить свойство **Формат поля (Format)** элемента управления таким образом, чтобы выводить различные результаты для пустых **(Null)** значений и строк нулевой длины невозможно. Для их отличия в элементе управления формы задайте в качестве его свойства **Данные (ControlSource)** выражение, проверяющее пустоту строки. Например, для вывода в элементе управления слов «Пусто» или «СНД» в качестве значения его свойства **Данные (ControlSource)** задайте следующее выражение:

=IIf(IsNull([MyControl]), "Пусто", Format([MyControl], "@;СНД"))

Свойство «Формат поля» (Format) - Текстовые и MEMO-поля, пример

Ниже приведены примеры специальных форматов для полей с типами «Текстовый» и «Поле MEMO».

Формат	Значение	Выводится как
@@@-@@-@@@@	465043799	465-04-3799
@@@@@@@@@@	465-04-3799 465043799	465-04-3799 465043799
>	иванов ИВАНОВ Иванов	ИВАНОВ ИВАНОВ ИВАНОВ
<	иванов ИВАНОВ Иванов	иванов иванов иванов
@;"Не заполнено"	Пустое **(Null)** значение	Не заполнено
	Пустая строка	Не заполнено
	Любой текст	*Отображается введенный текст.*

Свойство «Маска ввода» (InputMask)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproInputMaskC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproInputMaskX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproInputMaskA"}

Свойство **Маска ввода (InputMask)** задает маску ввода, облегчающую ввод данных в элемент управления- поле. Например, удобно создать следующую маску ввода для поля «Телефон», позволяющую вводить только цифры и автоматически добавляющую промежуточные символы: (___) ___-____. Значение данного свойства определяется автоматически при использовании мастера по созданию масок ввода.

Значения

Значение свойства **Маска ввода (InputMask)** может содержать до трех разделов, разделяемых точкой с запятой (;).

Раздел	Описание
Первый	Представляет саму маску ввода (например, !(999) 000-0000). Перечень символов, используемых для определения масок ввода приводится ниже в таблице.
Второй	Определяет режим занесения в таблицу строковых констант, добавляемых к символам, вводящимся пользователем. Введенный в данный компонент символ 0 указывает, что постоянные символы (например, скобки и дефисы в маске ввода телефонных номеров) сохраняются вместе с введенными пользователем символами; значение 1 или пустое значение данного компонента указывает, что сохраняются только символы, введенные пользователем.
Третий	Определяет символ, используемый для изображения пустых позиций в маске ввода, в которые помещаются вводящиеся пользователем символы. В этом компоненте можно указать любой символ ANSI; пробел необходимо заключить в кавычки (" ").

В программах Visual Basic значение данного свойства задается при помощи строкового выражения. Например, следующая инструкция для поля определяет маску ввода для телефонных номеров:

```
Forms!Клиенты!Телефон.InputMask = "(###) ###-####"
```

При создании маски ввода пользователь имеет возможность указать, что часть данных следует вводить обязательно (например, региональный код для телефонных номеров), а другие данные являются необязательными (например, добавочный номер телефона). Эти символы определяют тип данных, например номер символа, который необходимо ввести для каждого символа маски ввода.

Символы, которые следует вводить в маску ввода определяются следующими специальными символами.

Символ	Описание
0	Цифра (обязательный символ; знаки (+) и (-) не разрешены).
9	Цифра или пробел (необязательный символ; знаки (+) и (-) не разрешены).
#	Цифра или пробел (необязательный символ; незаполненные позиции выводятся как пробелы в режиме редактирования, но удаляются при сохранении данных; знаки (+) и (-) не разрешены).

L	Буква (обязательный символ).
?	Буква (необязательный символ).
A	Буква или цифра (обязательный символ).
a	Буква или цифра (необязательный символ).
&	Любой символ или пробел (обязательный символ).
C	Любой символ или пробел (необязательный символ).
. , : ; - /	Десятичный разделитель, разделители групп разрядов, времени или даты. (Используемые символы разделителей определяются настройками, выбранными в окне **Язык и стандарты** панели управления Windows).
<	Преобразует все символы к нижнему регистру.
>	Преобразует все символы к верхнему регистру.
!	Указывает, что маска ввода заполняется справа налево; этот символ следует использовать, если в левой части маски находятся позиции, заполнять которые не обязательно. Маски ввода обычно заполняются слева направо. Символ восклицательного знака можно помещать в произвольную позицию в маске ввода.
\	Указывает, что следующий символ следует воспринимать как постоянный (а не специальный) символ (например, \A представляет символ «A»).

Примечание. Значение «Пароль» свойства **Маска ввода (InputMask)** определяет элемент управления, предназначенный для ввода пароля. Любые символы, вводящиеся в этот элемент управления будут отображаться на экране звездочками (*). Подобная маска ввода используется для предотвращения вывода на экран печатаемых символов.

Для элемента управления значение данного свойства задается в окне свойств. Для поля в таблице или запросе значение данного свойства задается в режиме конструктора таблицы (в окне свойств поля) или в режиме конструктора окна запроса (в окне свойств поля).

Кроме того, значение свойства **Маска ввода (InputMask)** можно задать в макросе или в программе Visual Basic.

Дополнительные сведения

При вводе данных в поле, для которого определена маска ввода, всегда используется режим замены. При удалении символа путем нажатия клавиши BACKSPACE символ заменяется на пробел.

При копировании или перемещении содержимого поля, для которого определена маска ввода, в буфер обмена постоянные символы маски копируются вне зависимости от режима их сохранения.

Примечание. Маска ввода используется только при вводе символов в поле или в поле со списком с клавиатуры и игнорируется при всех остальных операциях, например, при импорте данных, при выполнении запроса на изменение, а также при вводе символов в элемент управления с помощью конструкции Visual Basic, в которой задается значение свойства **Text** элемента управления, или при выполнении макрокоманды **«Задать значение» (SetValue)** в макросе.

Если пользователь определяет маску ввода и одновременно задает для того же поля значение свойства **Формат поля (Format),** то при выводе данных приоритет имеет свойство **Формат поля (Format).** Это означает, что при форматировании данных сохраненная маска ввода игнорируется. Данные в базовой таблице при этом не изменяются; свойство **Формат поля (Format)** определяет только режим отображения данных.

Свойство «Маска ввода» (InputMask), примеры

В следующей таблице приводятся примеры часто используемых масок ввода и образцы значений, соответствующих этим маскам.

Маска ввода	Образцы значений
(000) 000-0000	(206) 555-0248
(999) 999-9999	(206) 555-0248
	( ) 555-0248
(000) AAA-AAAA	(206) 555-TELE
#999	–20
	2000
>L????L?000L0	GREENGR339M3
	MAY R 452B7
>L0L 0L0	T2F 8M4
00000-9999	98115-
	98115-3007
>L<??????????????	Мария
	Изаура
SSN 000-00-0000	SSN 555-55-5555
>LL00000-0000	DB51392-0493

Свойства «Условие на значение» (ValidationRule), «Сообщение об ошибке» (ValidationText)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acproValidationRuleC;daproValidationRule;daproValidationText"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproValidationRuleX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproValidationRuleA"}

Свойство **Условие на значение (ValidationRule)** определяет требования к данным, вводящимся в запись, в поле или в элемент управления. Свойство **Сообщение об ошибке (ValidationText)** позволяет указать текст сообщения, выводящегося на экран, если введенные данные нарушают условие, определенное в свойстве **Условие на значение (ValidationRule).**

Примечание. Свойства **Условие на значение (ValidationRule)** и **Сообщение об ошибке (ValidationText)** неприменимы к таким элементам управления как отдельные флажки, переключатели или выключатели, входящие в группу. Они применяются только для самой группы.

Значения

Значение свойства **Условие на значение (ValidationRule)** определяется с помощью выражения, а значение свойства **Сообщение об ошибке (ValidationText)** – с помощью строкового значения. Максимальная длина значения свойства **Условие на значение (ValidationRule)** составляет 2048 символов. Максимальная длина значения свойства **Сообщение об ошибке (ValidationText)** составляет 255 символов.

Для элементов управления в качестве значения свойства **Условие на значение (ValidationRule)** может быть указано любое правильное выражение. Выражения, определяющие условия на значения для полей или записей, не должны содержать функции, определяемые пользователем, статистические функции или функции по подмножеству, функции **CurrentUser** или **Eval,** а также ссылки на формы, запросы и таблицы. Кроме того, выражение, указанное в качестве условия для поля, не должно содержать ссылки на другие поля. Выражение, указанное в качестве условия на значение для записи, может содержать ссылки на поля той же таблицы.

Значения свойств **Условие на значение (ValidationRule)** и **Сообщение об ошибке (ValidationText)** задаются:

- в разделе свойств поля в режиме конструктора таблицы (условие на значение поля);
- в окне свойств таблицы, вызываемом командой **Свойства** меню **Вид** в режиме конструктора таблицы (условие на значение записи);
- в окне свойств элемента управления формы;
- в макросе или в программе Visual Basic. В программе Visual Basic значения данных свойств задаются при помощи строковых выражений.

Для того чтобы задать значения данных свойств для полей таблиц и записей в программе Visual Basic следует использовать свойство**ValidationRule** объектов доступа к данным.

Дополнительные сведения

Microsoft Access автоматически накладывает условия на значение, определяемые типом данных поля; например, не допускается ввод текста в числовые поля. Свойство **Условие на значение (ValidationRule)** позволяет указать дополнительные условия.

Если пользователь задает значение свойства **Условие на значение (ValidationRule),** но не определяет свойство **Сообщение об ошибке (ValidationText),** то при нарушении условия на значение Microsoft Access выводит стандартное сообщение об ошибке. Если значение свойства **Сообщение об ошибке (ValidationText)** задано, то в сообщении об ошибке выводится текст, указанный в качестве значения этого свойства.

Например, при вводе записи о сотруднике можно потребовать в свойстве **Условие на значение (ValidationRule),** чтобы значение поля «Дата найма» лежало в интервале между датой основания фирмы и текущей датой. Если введенная дата лежит вне данного диапазона, можно указать в свойстве **Сообщение об ошибке (ValidationText)** текст сообщения «Неверная дата найма».

При создании элемента управления путем переноса поля с помощью мыши из списка полей, условие, наложенное на данное поле, остается в действии, хотя и не отображается в ячейке свойства **Условие на значение (ValidationRule)** этого элемента управления. Условия на значение поля всегда наследуются элементами управления, присоединенными к этому полю.

Проверка условий на значение, определенных для элементов управления, полей и записей, выполняется в следующих случаях:

- условия на значение поля и элемента управления, проверяются после изменения данных при переводе фокуса из этого поля или элемента управления;
- условия на значение записи проверяются при переходе на другую запись;
- если условия определены и для поля, и для присоединенного к этому полю элемента управления, то проверка обоих условий выполняется при потере фокуса этим полем или элементом управления, если его значение было изменено.

В следующей таблице приводятся примеры выражений, определяющих свойства **Условие на значение (ValidationRule)** и **Сообщение об ошибке (ValidationText).**

Условие на значение	**Сообщение об ошибке**
<> 0	Требуется ненулевое значение.
> 1000 Or Is Null	Допускаются пустые значения или значения, превышающие 1000.
Like "A????"	Значение должно содержать 5 символов и начинаться с «А».
>= #1/1/96# And <#1/1/97#	Требуется дата, относящаяся к 1996 г.
DLookup("КодКлиента", "Клиенты", "КодКлиента = Forms!Клиенты!КодКлиента") Is Null	Требуется уникальное значение в поле «КодКлиента» (статистические функции допустимы только для условий на значение на уровне формы).

По умолчанию, поле, для которого определено условие на значение, не может содержать пустые **(Null)** значения. Для того чтобы сделать пустые значения допустимыми, следует в выражение, определяющее условие на значение, добавить оператор `Is Null`, например, <> 8 `Or Is Null`, и убедиться, что для свойства **«Обязательное поле» (Required)** задано значение «Нет».

Невозможно определить условия на значение для полей или записей таблиц, созданных вне Microsoft Access (например, для таблиц dBASE, Paradox или SQL Server). Таблицы этих типов допускают определение условий на значение только для элементов управления.

Свойства «Условие на значение» (ValidationRule), «Сообщение об ошибке» (ValidationText), пример

В данном примере поля определяется условие, требующее ввода значения, большего или равного 65. При нарушении этого условия выводится сообщение об ошибке. Значения свойств задаются с помощью функции `SetFieldValidation`.

```
Dim strTblName As String, strFldName As String
Dim strValidRule As String
Dim strValidText As String, intX As Integer

strTblName = "Клиенты"
strFldName = "Возраст"
strValidRule = ">= 65"
strValidText = "Введите число, большее или равное 65."
intX = SetFieldValidation(strTblName, strFldName, _
   strValidRule, strValidText)

Function SetFieldValidation(strTblName As String, _
   strFldName As String, strValidRule As String, _
   strValidText As String) As Integer

   Dim dbs As Database, tdf As TableDef, fld As Field

   Set dbs = CurrentDb
   Set tdf = dbs.TableDefs(strTblName)
   Set fld = tdf.Fields(strFldName)
   fld.ValidationRule = strValidRule
   fld.ValidationText = strValidText
End Function
```

В следующем примере с помощью функции `SetTableValidation` задается условие на значение записи, требующее чтобы значение в поле «КонечнаяДата» шло после значения в поле «НачальнаяДата».

```
Dim strTblName As String, strValidRule As String
Dim strValidText As String
Dim intX As Integer

strTblName = "Сотрудники"
strValidRule = "КонечнаяДата > НачальнаяДата"
strValidText = "Конечная дата должна идти после начальной даты."
intX = SetTableValidation(strTblName, strValidRule, strValidText)

Function SetTableValidation(strTblName As String, _
   strValidRule As String, strValidText As String) _
   As Integer

   Dim dbs As Database, tdf As TableDef

   Set dbs = CurrentDb
   Set tdf = dbs.TableDefs(strTblName)
   tdf.ValidationRule = strValidRule
   tdf.ValidationText = strValidText
End Function
```

Свойства событий

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅS. пїЅпїЅпїЅпїЅпїЅS":"acproEventPropertiesC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"acproEventPropertiesX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproEvents":1}

Свойства событий вызывают запуск макроса или соответствующей процедуры обработки событий Visual Basic при возникновении конкретного события. Например, если указать в свойстве кнопки **Нажатие кнопки (OnClick)** имя макроса, этот макрос будет запускаться при нажатии кнопки. Полный список свойств событий приводится в разделе Перечень событий и свойств событий.

Значения

Для запуска макроса необходимо указать в значении свойства имя макроса. Имя существующего макроса можно выбрать из списка. Если макрос входит в группу макросов, его имя включается в список под именем группы в формате *имяГруппыМакросов.имяМакроса*.

Для запуска процедуры обработки событий, связанной с событием, выберите в списке элемент **[Процедура обработки событий]**.

Примечание. Использование процедуры обработки событий является стандартным методом вызова программы Visual Basic в ответ на событие, однако, допускается определение пользователем функции, выполняемой в ответ на событие. Для этого следует ввести в ячейку события знак равенства (=), за которым следуют имя функции и скобки в формате =*имяФункции***( )**.

Значения свойств событий могут быть заданы в окне свойств объекта, а также в макросе или в программе Visual Basic. Отметим, что не допускается задание значений свойств событий при форматировании и при печати формы или отчета.

Совет. Значения свойств событий удобно задавать с помощью построителей. Для вызова построителя следует нажать кнопку построителя ... рядом с ячейкой свойства или установить указатель в ячейке свойства, нажать праву кнопку мыши и выбрать команду **Построить** в контекстном меню. В окне диалога **Построитель** предлагается следующий выбор:

- построитель макросов позволяет создать макрос и связать его с данным событием или изменить макрос, уже связанный с событием;
- построитель программ позволяет создать и связать с данным событием процедуру обработки событий, а также изменить уже определенную для события процедуру обработки;
- построитель выражений позволяет выбрать и указать для данного события определяемую пользователем функцию.

В программе Visual Basic значение свойства задается с помощью строкового выражения.

Запускается	Синтаксис	Пример
Макрос	"*имяМакроса*"	`Кнопка1.OnClick = "МойМакрос"`
Процедура обработки событий	**"[Процедура обработки событий]"**	`Кнопка1.OnClick = "[Процедура обработки событий]"`
Функция пользователя	**"=*имяФункции*( )"**	`Кнопка1.OnClick = "=МояФункция()"`

Свойства событий. Пример программы Visual Basic.

В данном примере демонстрируется использование значения, введенного в элемент управления «Страна», для определения макроса, запускаемого нажатием кнопки «Печать отчета для страны».

```
Private Sub Страна_AfterUpdate()
   If Страна = "Россия" Then
      [Печать отчета для страны].OnClick = "ПечатьОтчетаДляРоссии"
   ElseIf Страна = "Украина" Then
      [Печать отчета для страны].OnClick = "ПечатьОтчетаДляУкраины"
   End If
End Sub
```

Свойства событий и объекты, к которым они применяются

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproEventsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproEventsX":1}

Свойства событий	Применение
После подтверждения Del (AfterDelConfirm)	Формы.
После вставки (AfterInsert)	Формы.
После обновления (AfterUpdate)*	Формы. Элементы управления (присоединенная рамка объекта, флажок, поле со списком, список, переключатель, группа, поле, выключатель) в форме.
До подтверждения Del (BeforeDelConfirm)	Формы.
До вставки (BeforeInsert)	Формы.
До обновления (BeforeUpdate)*	Формы. Элементы управления (присоединенная рамка объекта, флажок, поле со списком, список, переключатель, группа, поле, выключатель) в форме.
Включение (OnActivate)	Формы и отчеты.
Применение фильтра (OnApplyFilter)	Формы.
Изменение (OnChange)	Элементы управления (поле со списком, набор вкладок, поле) в форме.
Нажатие кнопки (OnClick)*	Формы. Разделы формы. Элементы управления (присоединенная рамка объекта, диаграмма, флажок, поле со списком, кнопка, рисунок, надпись, список, переключатель, группа, набор вкладок, прямоугольник, поле, выключатель, свободная рамка объекта) в форме.
Закрытие (OnClose)	Формы и отчеты.
Текущая запись (OnCurrent)	Формы.
Двойное нажатие кнопки (OnDblClick)*	Формы. Разделы формы. Элементы управления (присоединенная рамка объекта, диаграмма, флажок, поле со списком, кнопка, рисунок, надпись, список, переключатель, группа, прямоугольник, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Отключение (OnDeactivate)	Формы и отчеты.
Удаление (OnDelete)	Формы.
Вход (OnEnter)*	Элементы управления (компоненты ActiveX, присоединенная рамка объекта, диаграмма, флажок, поле со списком, кнопка, список, переключатель, группа, подчиненная форма/отчет, поле, набор вкладок, выключатель, свободная рамка объекта) в

	форме.
Ошибка (OnError)	Формы и отчеты.
Выход (OnExit)*	Элементы управления (компоненты ActiveX, присоединенная рамка объекта, диаграмма, флажок, поле со списком, кнопка, список, переключатель, группа, подчиненная форма/отчет, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Фильтрация (OnFilter)	Формы.
Форматирование (OnFormat)	Разделы отчета.
Получение фокуса (OnGotFocus)	Формы. Элементы управления (компоненты ActiveX, присоединенная рамка объекта, флажок, поле со списком, кнопка, список, переключатель, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Клавиша вниз (OnKeyDown)	Формы. Элементы управления (присоединенная рамка объекта, флажок, поле со списком, кнопка, список, переключатель, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Нажатие клавиши (OnKeyPress)	Формы. Элементы управления (присоединенная рамка объекта, флажок, поле со списком, кнопка, список, переключатель, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Клавиша вверх (OnKeyUp)	Формы. Элементы управления (присоединенная рамка объекта, флажок, поле со списком, кнопка, список, переключатель, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Загрузка (OnLoad)	Формы.
Потеря фокуса (OnLostFocus)	Элементы управления (компоненты ActiveX, присоединенная рамка объекта, флажок, поле со списком, кнопка, список, переключатель, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Кнопка вниз (OnMouseDown)	Формы. Разделы форм. Формы. Элементы управления (присоединенная рамка объекта, диаграмма, флажок, поле со списком, кнопка, рисунок, надпись, список, переключатель, группа, прямоугольник, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Перемещение указателя (OnMouseMove)	Формы. Разделы форм. Формы. Элементы управления (присоединенная рамка объекта, диаграмма, флажок, поле со списком, кнопка, группа, прямоугольник, рисунок, надпись, список, переключатель, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Кнопка вверх (OnMouseUp)	Формы. Разделы форм. Формы. Элементы управления (присоединенная рамка объекта, диаграмма, флажок, поле со списком, кнопка, группа, прямоугольник, надпись, список, переключатель, поле, набор вкладок, выключатель, свободная рамка объекта) в форме.
Отсутствие данных	Отчеты.

(OnNoData)	
Отсутствие в списке (OnNotInList)	Элементы управления (поле со списком) в форме.
Открытие (OnOpen)	Формы и отчеты.
Страница (OnPage)	Отчеты.
Печать (OnPrint)	Разделы отчета.
Изменение размера (OnResize)	Формы.
Возврат (OnRetreat)	Разделы отчета.
Таймер (OnTimer)	Формы.
Выгрузка (OnUnload)	Формы.
При обновлении (**OnUpdated**)	Элементы управления (компоненты ActiveX, присоединенная рамка объекта, свободная рамка объекта) в форме.

***** Данные свойства неприменимы к флажкам, переключателям или выключателям, содержащимся в группе параметров. Они применимы только самой группе.

Свойство «Фильтр» (Filter)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproFilterC;daproFilter"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproFilterX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproFilterA"}

Свойство **Фильтр (Filter)** определяет подмножество записей, выводящихся после применения фильтра к форме, запросу или таблице.

Значения

Значение свойства **Фильтр (Filter)** определяется строковым выражением, содержащим предложение WHERE без ключевого слова WHERE. Например, следующие инструкции Visual Basic определят фильтр, отбирающий клиентов из России:

```
Me.Filter = "Страна  = 'Россия'"
Me.FilterOn = True
```

Значение данного свойства задается в окне свойств формы или отчета, в макросе или в программе Visual Basic.

Кроме того, можно задать значение данного свойства в режиме формы или в режиме таблицы. Для этого следует выбрать в меню **Записи** команду **Фильтр** и одну из команд в подменю.

Примечание. Значения свойства **Фильтр (Filter)** не оказывают влияния на свойство **Filter** объектов доступа к данным.

Дополнительные сведения

Свойство **Фильтр (Filter)** используется для сохранения фильтра, который предполагается применять в дальнейшем. Фильтры сохранятся вместе с объектами, в которых они созданы. Сохраненные фильтры автоматически загружаются вместе с объектами, но при этом не применяются автоматически.

При создании нового объекта он наследует значения свойств **Источник записей** (**RecordSource**), **Фильтр (Filter)**, **Порядок сортировки** (**OrderBy**) и **Сортировка включена** (**OrderByOn**) таблицы или запроса, на базе которых создается объект.

Для применения сохраненного фильтра к форме, запросу или таблице следует нажать кнопку **Применить фильтр** ... на панели инструментов, выбрать в меню **Записи** команду **Применить фильтр** или задать в макросе или в программе Visual Basic для свойства **Фильтр включен** (**FilterOn**) значение **True** (-1). Для того чтобы применить фильтр в отчете, следует задать значение «Да» для свойства **Фильтр включен** в окне свойств отчета.

Кнопка **Применить фильтр** служит индикатором состояния свойств **Фильтр (Filter)** и **Фильтр включен (FilterOn)**. Пока отсутствует фильтр, который можно применить, эта кнопка является недоступной. Если существующий фильтр уже применен, кнопка **Применить фильтр** выводится нажатой.

Для автоматического применения фильтра при открытии формы следует указать в свойстве события формы **Открытие (OnOpen)** либо макрос, содержащий макрокоманду **ПрименитьФильтр (ApplyFilter),** либо процедуру обработки события, в которой вызывается метод **ApplyFilter** объекта **DoCmd**.

Для снятия фильтра следует еще раз нажать нажатую кнопку **Применить фильтр**, выбрать в меню **Записи** команду **Удалить фильтр** или задать в программе Visual Basic для свойства **Фильтр включен (FilterOn)** значение **False** (0).

Примечание. Для сохранения фильтра как запроса выберите в меню **Файл** команду **Сохранить как запрос** в окне фильтра для формы или в окне расширенного фильтра.

Если значение свойства **Фильтр (Filter)** задается в режиме конструктора формы, Microsoft Access не проверяет правильность выражения SQL. Если введено неверное выражение SQL,

ошибка возникает при применении фильтра.

Свойство «Применение автофильтра» (FilterLookup)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFilterLookupC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproFilterLookupX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproFilterLookupA"}

Свойство **Применение автофильтра (FilterLookup)** определяет, будут ли выводиться значения в присоединенном поле при работе в окне фильтра для формы.

Значения

Свойство **Применение автофильтра (FilterLookup)** может иметь следующие значения.

Значение	Описание	Visual Basic
Никогда	Значения поля не выводятся. Пользователь имеет возможность указать, могут ли отбираемые записи содержать пустые **Null**) значения.	0
Параметр базы данных	(Значение по умолчанию). Вывод значений поля определяется в группе **Параметры фильтра по умолчанию**, выводящейся на вкладке **Правка/поиск** в окне диалога Параметры, которое открывается командой **Параметры** из меню **Сервис**.	1
Всегда	Значения поля выводятся всегда.	2

Значение свойства **Применение автофильтра (FilterLookup)** задается в окне свойств поля, в макросе или в программе Visual Basic.

Допускается также определение свойства **Применение автофильтра (FilterLookup)** для поля в окне стандартных свойств элемента управления. или с помощью метода **DefaultControl** из программы на Visual Basic.

Дополнительные сведения

Если окажется, что вывод списков значений полей занимает слишком много времени, пользователь имеет возможность ограничить число полей, в которых такие списки выводятся. Для этого следует снять соответствующий флажок в группе **Параметры фильтра по умолчанию**, выводящейся на вкладке **Правка/поиск** в окне диалога **Параметры**, которое открывается командой параметры из меню **Сервис**. Для того чтобы обеспечить вывод всех значений в длинных списках, следует ввести в поле **Не отображать списки, содержащие более заданного числа строк** значение, превышающее максимальное число записей в любом неиндексированном поле в базовом источнике записей.

Свойство «Фильтр включен» (FilterOn)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFilterOnC;daproFilter"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFilterOnX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproFilterOnA"}

Свойство **Фильтр включен (FilterOn)** позволяет указать или проверить, применяется ли свойство **Фильтр** (**Filter**) для формы или отчета.

Значения

Свойство **Фильтр включен (FilterOn)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Свойство объекта **Фильтр (Filter)** применяется.	**True** (–1)
Нет	(Значение по умолчанию). Свойство объекта **Фильтр (Filter)** не применяется.	**False** (0)

Для отчетов значение свойства **Фильтр включен (FilterOn)** задается в окне свойств или в программе Visual Basic.

Для форм значение свойства **Фильтр включен (FilterOn)** задается в макросе или в программе Visual Basic. Кроме того, можно задать значение данного свойства, нажатием кнопки **Применить фильтр** ... на панелях инструментов **Режим формы** или **Фильтр/сортировка**.

Дополнительные сведения

Для того чтобы применить сохраненный фильтр следует для формы нажать кнопку **Применить фильтр** или для форм и отчетов применить фильтр, указав в макросе или в программе Visual Basic значение **True** (-1) для свойства **Фильтр включен (FilterOn)**. Для отчетов можно также задать для свойства **Фильтр включен** значение «Да» в окне свойств отчета.

Кнопка **Применить фильтр** служит индикатором состояния свойств **Фильтр (Filter)** и **Фильтр включен (FilterOn)**. Пока отсутствует фильтр, который можно применить, эта кнопка является недоступной. Если существующий фильтр уже применен, кнопка **Применить фильтр** выводится нажатой. Для автоматического применения фильтра при открытии формы следует указать в свойстве события формы **Открытие (OnOpen)** либо макрос, содержащий макрокоманду **ПрименитьФильтр (ApplyFilter)**, либо процедуру обработки события, в которой вызывается метод **ApplyFilter** объекта **DoCmd**.

Для снятия фильтра следует еще раз нажать нажатую кнопку **Применить фильтр**, выбрать в меню **Записи** команду **Удалить фильтр** или задать в программе Visual Basic для свойства **Фильтр включен (FilterOn)** значение **False** (0). Для отчетов можно снять фильтр, указав для свойства **Фильтр включен (FilterOn)** значение "Нет" в окне свойств отчета.

Примечание. При создании нового объекта он наследует значения свойств **Источник записей** (**RecordSource**), **Фильтр (Filter)**, **Порядок сортировки** (**OrderBy**) и **Сортировка включена** (**OrderByOn**) базовой таблицы или запроса. Для форм и отчетов наследуемые фильтры не применяются автоматически при открытии объекта.

Свойство «Порядок сортировки» (OrderBy)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproOrderByC;daproFilter"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproOrderByX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproOrderByA"}

Свойство **Порядок сортировки (OrderBy)** определяет порядок сортировки записей в форме, запросе, отчете или таблице.

Значения

Значение свойства **Порядок сортировки (OrderBy)** задается с помощью строкового выражения, содержащего имена полей, по которым проводится сортировка записей. Если указано несколько имен, необходимо разделять их запятой (,).

Если в значении свойства **Порядок сортировки (OrderBy)** заданы имена одного или нескольких полей, будет выполнена сортировка по возрастанию. Аналогично, в программах Visual Basic также по умолчанию выполняется сортировка полей по возрастанию.

Для сортировки записей по убыванию следует ввести ключевое слово **DESC** в конце строкового выражения. Например, для сортировки записей о клиентах по убыванию укажите для свойства **Порядок сортировки (OrderBy)** значение «Название DESC».

Значение свойства **Порядок сортировки (OrderBy)** задается в окне свойств объекта, в макросе или в программе Visual Basic.

Дополнительные сведения

В отчетах для применения порядка сортировки, указанного в свойстве объекта **Порядок сортировки (OrderBy)**, следует задать для свойства **Сортировка включена** (**OrderByOn**) значение «Да». В формах следует выбрать поле, по которому требуется выполнить сортировку, и либо нажать соответствующую кнопку **Сортировка** на панели инструментов, либо выбрать в меню **Записи** команду **Сортировка** и соответствующую команду в подменю. Кроме того, и для форм, и для отчетов можно задать значение свойства **Сортировка включена (OrderByOn)** в программе Visual Basic.

Установка значения свойства **Порядок сортировки (OrderBy)** для открытого отчета приводит к запуску его процедур обработки событий **Закрытие (Close)** и **Открытие (Open)**.

Примечание. При создании нового объекта он наследует значения свойств **Источник записей** (**RecordSource**), **Фильтр (Filter)**, **Порядок сортировки (OrderBy)** и **Сортировка включена (OrderByOn)** таблицы или запроса, на базе которых создается объект. Для форм и отчетов наследуемые фильтры не применяются автоматически при открытии объекта.

Свойство «Сортировка включена» (OrderByOn)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproOrderByOnC;daproFilter"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproOrderByOnX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproOrderByOnA"}

Свойство **Сортировка включена (OrderByOn)** определяет, применяется ли текущее значение свойства объекта **Порядок сортировки** (**OrderBy**).

Значения

Свойство **Сортировка включена (OrderByOn)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). При открытии объекта применяется текущее значение свойства **Порядок сортировки (OrderBy)**.	**True** (–1)
Нет	Текущее значение свойства **Порядок сортировки (OrderBy)** не применяется при открытии объекта.	**False** (0)

Для отчетов значение свойства **Сортировка включена (OrderByOn)** задается в окне свойств отчета, в макросе или в программе Visual Basic.

Для всех остальных объектов значение свойства **Сортировка включена (OrderByOn)** задается либо нажатием кнопки **Сортировка** на панели инструментов, либо в программе Visual Basic.

Дополнительные сведения

При создании нового объекта он наследует значения свойств **Источник записей** (**RecordSource**), **Фильтр** (**Filter**), **Порядок сортировки (OrderBy)**, **Сортировка включена (OrderByOn)** и **Фильтр включен** (**FilterOn**) базовой таблицы или запроса.

Свойство «Применение фильтров» (AllowFilters)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproAllowFilteringC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproAllowFilteringX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproAllowFilteringA"}

Свойство **Применение фильтров (AllowFilters)** позволяет указать, допускается ли применение фильтров для отбора записей в форме.

Значения

Свойство **Применение фильтров (AllowFilters)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Допускается применение фильтров.	**True** (–1)
Нет	Применение фильтров не допускается.	**False** (0)

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Дополнительные сведения

Фильтры используются для вывода на экран временных наборов записей, отбираемых в базе данных. Фильтры отбирают записи, удовлетворяющие определенным условиям отбора. Например, с помощью фильтра можно отобрать в таблице «Сотрудники» записи, относящиеся к сотрудникам, имеющим более 5 лет стажа. Кроме того, фильтры позволяют ограничить доступ к записям, содержащим конфиденциальную информацию, например, финансовые или медицинские данные.

При заданном для свойства **Применение фильтров (AllowFilters)** значении «Нет» отключаются все относящиеся к фильтрам команды в меню **Записи,** в контекстном меню формы и на панели инструментов **Форматирование (форма/отчет).** Сюда относятся следующие команды и кнопки:

- команды **Фильтр**, **Применить фильтр** и **Удалить фильтр**;
- команды **Фильтр по выделенному**, **Применить фильтр** и **Удалить фильтр** в контекстном меню формы;
- соответствующие кнопки **Фильтр по выделенному** ..., **Изменить фильтр** ... и **Применить/Удалить фильтр** ....

Примечание. Задание для свойства **Применение фильтров (AllowFilters)** значения «Нет» не влияет на свойства **Фильтр (Filter)** и **Фильтр включен (FilterOn).** Поэтому их использование для установки и удаления фильтров допустимо. Для этой же цели возможно применение следующих макрокоманд и методов:

Макрокоманды	Методы
ПрименитьФильтр (ApplyFilter)	**ApplyFilter**
ОткрытьФорму (OpenForm)	**OpenForm**
ПоказатьВсеЗаписи (ShowAllRecords)	**ShowAllRecords**

Свойство «Применение фильтров» (AllowFilters), пример

В данном примере демонстрируется функция, переключающая значение свойства формы **Применение фильтров (AllowFilters):**

```
Function ToggleAllowFilters() As Integer
   Forms!Клиенты.AllowFilters = Not Forms!Клиенты.AllowFilters
End Function
```

Свойство «Кнопка оконного меню» (ControlBox)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproControlBoxC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproControlBoxX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproControlBoxA"}

Свойство **Кнопка оконного меню (ControlBox)** определяет вывод для формы оконного меню в режиме формы и в режиме таблицы.

Значения

Свойство **Кнопка оконного меню (ControlBox)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). В режиме формы и в режиме таблицы выводится оконное меню формы.	**True** (–1)
Нет	В режиме формы и в режиме таблицы форма выводится без оконного меню.	**False** (0)

Примечание. Если для свойства **Кнопка оконного меню (ControlBox)** задано значение «Нет», форма выводится без кнопок свертывания, развертывания и закрытия окна.

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Значение данного свойства можно задать только в режиме конструктора формы.

Дополнительные сведения

Для того чтобы вывести форму с оконным меню, необходимо задать для свойства **Кнопка оконного меню (ControlBox)** значение «Да» и выбрать для свойства **Тип границы (BorderStyle)** значение «Тонкая», «Изменяемая» или «Окна диалога».

Даже при указанном в свойстве **Кнопка оконного меню (ControlBox)** значении «Нет» форма, открывающаяся в режиме конструктора, всегда имеет оконное меню.

Если для свойства **Кнопка оконного меню (ControlBox)** задано значение «Нет» вывод оконного меню подавляется:

- при открытии формы в режиме формы из окна базы данных;
- при открытии формы из макроса;
- при открытии формы из программы Visual Basic;
- при открытии формы в режиме таблицы;
- при переключении из режима конструктора в режим формы или в режим таблицы.

Свойство «Кнопка оконного меню» (ControlBox), пример

В данном примере для свойства **Кнопка оконного меню (ControlBox)** формы «ОкноСообщения» задается значение **False** (0):

```
Forms!ОкноСообщения.ControlBox = False
```

Свойство «Режим вывода» (DisplayWhen)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDisplayWhenC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDisplayWhenX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDisplayWhenA"}

Свойство **Режим вывода (DisplayWhen)** позволяет указать, какие из разделов или элементов управления формы выводятся на экран или на печать.

Примечание. Свойство **Режим вывода (DisplayWhen)** применимо только к следующим разделам формы: к области данны, к заголовку формы и к примечанию формы. Также оно применимо ко всем элементам управления в форме (за исключением конца страницы).

Значения

Свойство **Режим вывода (DisplayWhen)** может иметь следующие значения.

Значение	Описание	Visual Basic
Всегда	(Значение по умолчанию). Объект выводится в режиме формы и при печати.	0
Только при печати	Объект скрыт в режиме формы, но выводится при печати.	1
Только на экран	Объект выводится в режиме формы, но не печатается.	2

Значение данного свойства задается в окне свойств объекта, в макросе или в программе Visual Basic.

Для элементов управления можно указать используемое по умолчанию значение этого свойства в окне стандартных свойств элемента управления или при помощи метода Visual Basic **DefaultControl.**

Дополнительные сведения

Некоторые элементы управления полезны только в режиме формы. Для того чтобы не печатать эти элементы управления, следует выбрать для свойства **Режим вывода (DisplayWhen)** значение «Только на экран». Например, форма может содержать кнопки или инструкции, которые не следует печатать, или заголовок и примечание формы, которые не следует изображать на экране, однако, необходимо напечатать. В последнем случае следует выбрать для свойства **Режим вывода (DisplayWhen)** значение «Только при печати».

Совет. В отчетах для указания элементов управления, которые не следует печатать, используют события **Форматирование (Format)** и **Возврат (Retreat),** для которых определяют процедуру обработки события или макроса, задающий значение свойства **Вывод на экран (Visible).** Для предотвращения печати раздела отчета также можно отменить свойства **Форматирование (Format)** или **Печать (Print).**

Свойство «Формат для печати» (LayoutForPrint)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproLayoutForPrintC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproLayoutForPrintX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproLayoutForPrintA"}

Свойство **Формат для печати (LayoutForPrint)** определяет использование в форме или отчете принтерных или экранных шрифтов.

Значения

Свойство **Формат для печати (LayoutForPrint)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию для отчетов). Использование шрифтов принтера.	**True** (–1)
Нет	(Значение по умолчанию для форм). Использование экранных шрифтов.	**False** (0)

Значение данного свойства задается в окне свойств формы или отчета, в макросе или в программе Visual Basic.

Значение данного свойства задается только в режиме конструктора формы или в режиме конструктора отчета.

Дополнительные сведения

При выборе шрифта в Microsoft Access выбирается либо экранный шрифт, либо шрифт принтера, в зависимости от установленного значения свойства **Формат для печати (LayoutForPrint).** Необходимо помнить, что символы одного и того же шрифта на экране и на бумаге могут выглядеть по-разному, поскольку одноименные экранные шрифты и шрифты принтера могут отличаться.

Совет. При использовании масштабируемых шрифтов, например, шрифтов TrueType, символы на экране и на бумаге выглядят практически одинаково.

Экранные шрифты -- это установленные в системе буквы, цифры и прочие символы, изображаемые на экране. При установке принтера дополнительные экранные шрифты могут быть установлены автоматически.

Шрифты принтера -- это буквы, цифры и прочие символы, которые печатаются на бумаге при печати формы или отчета. В этом случае набор используемых шрифтов определяется конкретным установленным принтером.

Если свойство **Формат для печати (LayoutForPrint)** имеет значение «Да», то в списке на панели инструментов **Форматирование (форма/отчет)** выводятся имена шрифтов и размеры в пуктах, доступные для выбранного принтера.

Если форма или отчет были разработаны в системе, использующей принтер, отличный от того, на котором они должны быть напечатаны, Microsoft Access выводит на экран сообщение о том, что данная форма (или отчет) разработана для принтера другого типа. Если пользователь подтверждает печать формы или отчета, то для текущего принтера выполняется подстановка шрифтов взамен недоступных для этого принтера. Аналогичная подстановка шрифтов возможна после изменения значения свойства **Формат для печати (LayoutForPrint).** Например, допустимо создание формы или отчета при установленном для свойства **Формат для печати (LayoutForPrint)** значении «Нет» и после этого поменять его на значение «Да». В этом случае можно указать другой шрифт для любого элемента управления, выводящегося в форме или в отчете.

Свойство «Кнопки размеров окна» (MinMaxButtons)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproMinMaxButtonsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproMinMaxButtonsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproMinMaxButtonsA"}

Свойство **Кнопки размеров окна (MinMaxButtons)** определяет вывод в форме кнопок свертывания и развертывания окна.

Значения

Свойство **Кнопки размеров окна (MinMaxButtons)** может иметь следующие значения.

Значение	Описание	Visual Basic
Отсутствуют	Форма выводится без кнопок свертывания и развертывания окна.	0
Свертывания	Выводится только кнопка свертывания окна.	1
Развертывания	Выводится только кнопка развертывания окна.	2
Все	(Значение по умолчанию). Выводятся обе кнопки свертывания и развертывания окна.	3

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Значение свойства **Кнопки размеров окна (MinMaxButtons)** можно задать только в режиме конструктора формы.

Дополнительные сведения

Нажатие кнопки развертывания окна увеличивает размеры окна формы до размеров окна Microsoft Access. Нажатие кнопки свертывания окна превращает окно формы в значок внизу окна Microsoft Access.

Для того чтобы вывести в форме кнопку развертывания или свертывания окна, необходимо, чтобы свойство формы **Тип границы (BorderStyle)** имело значение «Тонкая» или «Изменяемая», а свойство **Кнопка оконного меню (ControlBox)** имело значение «Да». Если для свойства формы **Тип границы (BorderStyle)** задано значение «Отсутствует» или «Окна диалога» или если свойство **Кнопка оконного меню (ControlBox)** имеет значение «Нет», то форма выводится без кнопок развертывания или свертывания окна, вне зависимости от значения свойства **Кнопки размеров окна (MinMaxButtons).**

Даже при заданном в свойстве **Кнопки размеров окна (MinMaxButtons)** значении «Отсутствуют» форма в режиме конструктора всегда выводится с кнопками свертывания и развертывания окна.

При значении свойства **Кнопки размеров окна (MinMaxButtons)** «Отсутствуют» становятся недоступными команды **Свернуть** и **Развернуть** в оконном меню формы.

Свойство «Строка меню» (MenuBar)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproMenuBarC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproMenuBarX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproMenuBarA"}

Свойство **Строка меню (MenuBar)** позволяет созда строку меню, используемую при работе с базой данных, формой или отчетом. Для создания этой строки меню используется подкоманда **Настройка** команды **Панели инструментов** из меню **Вид**. Для получения дополнительных сведений о создании специальных панелей инструментов см. раздел Создание специальных строк меню для текущей базы данных.

Данное свойство также позволяет указать макрос создания меню, с помощью которого выводится специальная строка меню для базы данных, формы или отчета.

Примечание. В предыдущих версиях Microsoft Access для создания специальной строки меню необходимо было задать для свойства **Строка меню (MenuBar)** имя соответствующего макроса. После этого создавалась группа макросов, содержащая команды этих строк меню. В Microsoft Access 97 этот механизм также поддерживается. Однако для создания специальных строк меню настоятельно рекомендуется использовать диалоговое окно **Настройка,** для вызова которого необходимо выбрать из меню **Вид** команду **Панели инструментов.**

Значения

Введите имя вызываемой строки меню. Если свойство **Строка меню (MenuBar)** остается пустым, Microsoft Access выводит на экран встроенную строку меню (меню по умолчанию) или главное меню приложения. При установки для свойства **Строка меню (MenuBar)** значения, не являющегося именем существующей строки меню или макроса меню, строка меню в форме или отчете не выводится вовсе (в этом случае не выводится даже встроенная стока меню).

Значение данного свойства задается в окне свойств объекта в макросе или в программе Visual Basic.

В программе Visual Basic данное свойство задается с помощью строкового выражения, значением которого является имя вызываемого меню.

Для того чтобы вывести с помощью макроса или программы Visual Basic встроенную или главную строку меню в базе данных, форме или отчете, необходимо задать для данного свойства значение пустой строки ("").

Дополнительные сведения

Если для формы или отчета определено свойство **Строка меню (MenuBar),** Microsoft Access выводит указанную строку меню на экран при открытии формы или отчета. Это меню выводится, когда форма или отчет получает фокус.

Если определить свойство **Строка меню (MenuBar)** для объекта **Application,** то специальная строка меню выводится во всей базе данных. Однако при задании значения свойства **Строка меню (MenuBar)** для формы или отчета, каждый раз, когда эта форма или отчет получит фокус, вместо строки меню базы данных на экран будет выводится строка меню этой формы или этого отчета. После потери фокуса будет восстановлено общее меню базы данных.

Примечание. Для переключения между специальным и встроенным меню базы данных нажмите клавиши CTRL+F11.

Свойство «Строка меню» (MenuBar), пример

В данном примере в свойстве **Строка меню (MenuBar)** указывается строка меню с именем «МенюПользователя»:

```
Forms!Клиенты.MenuBar = "МенюПользователя"
```

Для того чтобы вывести встроенную строку меню формы или общее меню приложения, необходимо задать для свойства **Строка меню (MenuBar)** значение пустой строки ("").

```
Forms!Клиенты.MenuBar = ""
```

Свойство «Модальное окно» (Modal)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproModalC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproModalBasicX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproModalA"}

Свойство **Модальное окно (Modal)** позволяет указать открытие формы в режиме модального окна, при котором для перевода фокуса на другой объект необходимо предварительно закрыть форму.

Значения

Свойство **Модальное окно (Modal)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Форма в режиме формы открывается как модальная.	**True** (–1)
Нет	(Значение по умолчанию). Форма не является модальной.	**False** (0)

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Дополнительные сведения

После открытия модальной формы все остальные окна становятся недоступными до тех пор, пока она не будет закрыта (хотя допустимо переключение в окна других приложений). Для того чтобы дополнительно отключить меню и панель инструментов, установите значение «Да» для свойств **Модальное окно (Modal)** и **Всплывающее окно (PopUp)** формы.

Различные типы границ формы задаются в свойстве **Тип границы (BorderStyle).** Обычно, для модальных форм выбирают **Тип границы (BorderStyle)** «Окно диалога».

Совет. Свойства **Модальное окно (Modal), Всплывающее окно (PopUp)** и **Тип границы (BorderStyle)** используются для создания специальных диалоговых окон. При этом для свойств **Модальное окно (Modal)** и **Всплывающее окно (PopUp)** обычно задают значение «Да», а для свойства **Тип границы (BorderStyle)** значение «Окно диалога».

Значение «Да» свойства **Модальное окно (Modal)** делает окно формы модальным только:

- при открытии в режиме формы из окна базы данных;
- при открытии в режиме формы из макроса или из программы Visual Basic;
- при переключении из режима конструктора в режим формы.

Модальная форма не допускает переключения в режим таблицы из режима формы. Для этого необходимо переключиться в режим конструктора, а затем – в режим таблицы.

Форма не является модальной в режиме конструктора, в режиме таблицы, а также при переключении в режим формы из режима конструктора.

Примечание. Открыть форму с автоматической установкой значения «Да» для свойств **Модальное окно (Modal)** и **Всплывающее окно (PopUp)** позволяет макрокоманда **ОткрытьФорму (OpenForm)** со значением «Окно диалога» в аргументе «Режим окна».

Свойство «Поле номера записи» (NavigationButtons)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproNavigationButtonsC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproNavigationButtonsX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproNavigationButtonsA"}

Свойство **Поле номера записи (NavigationButtons)** определяет вывод в форме кнопок перехода и поля номера записи.

Значения

Свойство **Поле номера записи (NavigationButtons)** может иметь следующие свойства.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Форма выводится с кнопками перехода и полем номера записи.	**True** (–1)
Нет	Форма выводится без кнопок перехода и поля номера записи.	**False** (0)

Значение данного свойства устанавливается в окне свойств формы, в макросе или в программе Visual Basic.

Дополнительные сведения

Кнопки перехода являются удобным средством перехода к первой, предыдущей, следующей, последней или пустой (новой) записи. В поле номера записи выводится номер текущей записи. Полное число записей выводится непосредственно за кнопками перехода. Для перехода к конкретной записи введите ее номер в поле номера записи.

После удаления из формы кнопок перехода для создания в ней специальных средств передвижения допустимо создание специальных кнопок перехода и добавление их в форму. Для получения дополнительных сведений о создании специальных кнопок перехода см. раздел Переключение между элементами управления, записями и страницами формы при помощи Visual Basic.

Свойство ObjectPalette

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproObjectPaletteC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproObjectPaletteX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproObjectPaletteA"}

Свойство **ObjectPalette** определяет, какая палитра используется в приложении при создании объектов следующих типов:

- объект OLE, помещаемый в присоединенную рамку объекта, в диаграмму или в свободную рамку объекта;
- точечный рисунок, или другое изображение, помещаемое на кнопку, рисунок или переключатель с помощью свойства **Рисунок** (**Picture**).

Значения

В Microsoft Access значение свойства **ObjectPalette** используется для определения свойства **PaintPalette** формы или отчета. Значение этого свойства имеет тип **String.**

Для следующих объектов, режимов и элементов управления значение свойства **ObjectPalette** является доступным только для чтения.

Объект	Режим	Элемент управления
Форма	Режим конструктора и режим формы	Кнопка, диаграмма, рисунок, переключатель, свободная рамка объекта.
	Режим формы	Присоединенная рамка объекта
Отчет	Режим конструктора	Кнопка, диаграмма, рисунок, переключатель, свободная рамка объекта. Данное значение свойства недоступно для присоединенной рамки объекта во всех режимах отчета.

Значение данного свойства используется только в макросах или в программах Visual Basic.

Если приложение, связанное с объектом OLE, точечным рисунком или другим графическим изображением, не имеет назначенной палитры, то свойство **ObjectPalette** принимает значение строки нулевой длины.

Дополнительные Сведения

Определение свойства **ObjectPalette** делает палитру приложения, с которым связан объект OLE, доступной для указания в свойстве **PaintPalette** формы или отчета. Например, чтобы сделать палитру, используемую в Microsoft Graph, доступной при разработке формы Microsoft Access, следует указать в качестве значения свойства формы **PaintPalette** значение свойства **ObjectPalette** существующего элемента управления-диаграммы.

Примечание. Windows позволяет иметь каждый раз только одну активизированную палитру. Microsoft Access позволяет иметь в форме несколько графических объектов, в каждом из которых используется собственная цветовая палитра. Свойство формы **Источник палитры** (**PaletteSource**) определяет, какая палитра будет использована для отображения графических объектов.

Свойства ObjectPalette, PaintPalette, пример

В данном примере свойству **PaintPalette** формы «Пейзаж» присваивается значение свойства **ObjectPalette** элемента управления «Океан» из формы «Картины». (Элемент управления "Океан" может быть кнопкой, диаграммой, присоединенной рамкой объекта, переключателем или свободной рамкой объекта).

```
Forms!Пейзаж.PaintPalette = Forms!Картины!Океан.ObjectPalette
```

Свойства **ObjectPalette** и **PaintPalette** позволят запрограммировать изменение используемой цветовой палитры при выполнении формы. Обычно, с помощью этих свойств в свойстве текущей формы **PaintPalette** указывают палитру элемента управления, имеющего фокус.

Например, в форме может выводиться океанский пейзаж с большим количеством оттенков синего и картина заката с широким набором оттенков красного цвета. Так как Windows позволяет иметь каждый раз только одну активизированную палитру, только один из рисунков может быть представлен с помощью собственной палитры. В следующей программе процедура обработки события **Вход (Enter)** указывает в свойстве формы **PaintPalette** палитру, заданную в свойстве **ObjectPalette** элемента управления. При этом элемент управления, имеющий фокус, будет отображен с собственной палитрой.

```
Sub Океан_Enter()
   Me.PaintPalette = Me!Океан.ObjectPalette
End Sub

Sub Закат_Enter()
   Me.PaintPalette = Me!Закат.ObjectPalette
End Sub
```

Свойство PaintPalette

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPaintPaletteC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPaintPaletteX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPaintPaletteA"}

Свойство **PaintPalette** позволяет указать палитру, которая будет задана для формы или отчета.

Значения

Свойство **PaintPalette** задается в макросе или в программе Visual Basic. Значение этого свойства имеет тип **String.**

Значение свойства **PaintPalette** может быть установлено путем присвоения ему значения свойства **ObjectPalette** в макросе или в программе на Visual Basic, путем присвоения значения свойства **PaletteSource** (в этом случае Microsoft Access автоматически установит значение **PaletteSource** для свойства **PaintPalette**), или с помощью назначения свойству **PaintPalette** одной формы или отчета значения свойства **PaintPalette** другой формы или отчета.

Для формы значение свойства **PaintPalette** можно задать в режиме конструктора и в режиме формы.

Для отчета значение свойства **PaintPalette** можно задать только в режиме конструктора.

Дополнительные Сведения

При определении свойства **PaintPalette** Microsoft Access создает копию указанной палитры. Эта копия сохраняется вместе с макетом формы или отчета и будет доступна при изменении этой формы или отчета.

Изменения, внесенные в палитру, указанную в значении свойства **PaintPalette**, не влияют на копию палитры, сохраняемую вместе с макетом формы или отчета. Для обновления копии палитры необходимо повторно выполнить программу или макрос, определяющий свойство **PaintPalette**, или переопределить свойство **Источник палитры (PaletteSource)** при открытии формы или отчета.

При указании значения свойства формы или отчета **PaintPalette** Microsoft Access автоматически обновляет значение свойства **Источник палитры (PaletteSource)**. Аналогично, при определении свойства формы или отчета **Источник палитры (PaletteSource)** Microsoft Access автоматически обновляет значение свойства **PaintPalette**. Например, если в свойстве **PaintPalette** будет указана специальная палитра, значение свойства **Источник палитры (PaletteSource)** изменится на «(Специальный)». Свойство **PaintPalette** (доступное только в макросах или из программы Visual Basic) используется для установки палитры формы или отчета. Свойство **Источник палитры** позволяет установить палитру для формы или отчета в окне свойств, задав существующий графический файл.

Примечание. Windows позволяет иметь каждый раз только одну активизированную палитру. Microsoft Access позволяет иметь в форме несколько графических объектов, в каждом из которых используется собственная цветовая палитра. Свойство формы **Источник палитры (PaletteSource)** определяет, какая палитра будет использована для отображения графических объектов.

Свойство **ObjectPalette** позволяет связывать палитру приложения с объектом OLE, рисунком или другим графическим объектом, содержащемся в элементе управления формы или отчета, доступного в свойстве **PaintPalette**. Например, для того чтобы использовать палитру, созданную в Microsoft Graph, при разработке формы в Microsoft Access, следует присвоить свойству формы **PaintPalette** значение свойства **ObjectPalette** существующего объекта диаграммы.

Свойство «Источник палитры» (PaletteSource)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproPaletteSourceC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproPaletteSourceX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproPaletteSourceA"}

Свойство **Источник палитры (PaletteSource)** определяет палитру, используемую в форме или отчете.

Значения

Введите путь и имя файла одного из следующих типов:

- .dib (device-independent bitmap)
- .pal (палитра Windows)
- .ico (значок Windows)
- .bmp (bitmap, растровый рисунок Windows)
- .wmf, .emf или другие файлы графических объектов, для которых существуют графические фильтры

По умолчанию задается значение «(Стандартный)», указывающее использование встроенной палитры Microsoft Access. Если изменить эту настройку и ввести путь и имя файла, то в ячейке свойства будет выведено сообщение «(Специальный)».

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Для формы значение свойства **Источник палитры (PaletteSource)** можно задать в режиме конструктора или в режиме формы..

Для отчета значение свойства **Источник палитры (PaletteSource)** можно задать только в режиме конструктора. Установка этого свойства в других режимах не допускается.

Дополнительные сведения

Windows позволяет иметь каждый раз только одну активизированную палитру. Microsoft Access позволяет иметь в форме несколько графических объектов, в каждом из которых используется собственная цветовая палитра. Свойства формы **Источник палитры (PaletteSource)** и **PaintPalette** определяют, какая палитра будет использована для отображения графических объектов.

При указании значения свойства формы или отчета **Источник палитры (PaletteSource)** Microsoft Access автоматически обновляет значение свойства **PaintPalette**. Аналогично, при определении свойства формы или отчета **PaintPalette** Microsoft Access автоматически обновляет значение свойства **Источник палитры (PaletteSource)**. Например, если в свойстве **PaintPalette** будет указана специальная палитра, значение свойства **Источник палитры** изменится на «(Специальный)». Свойство **PaintPalette** (доступное только в макросах или из программ Visual Basic) используется для установки палитры формы или отчета. Свойство **Источник палитры** позволяет установить палитру для формы или отчета в окне свойств, задав существующий графический файл.

Свойство «Всплывающее окно» (PopUp)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproPopUpC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproPopUpX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproPopupA"}

Свойство **Всплывающее окно (PopUp)** позволяет указать открытие формы в виде всплывающего окна. Например, с помощью значений «Да» для свойств **Всплывающее окно (PopUp)** и **Модальное окно (Modal)** и значения «Окно диалога» свойства **Тип границы (BorderStyle)** создают специальные диалоговые окна.

Значения

Свойство **Всплывающее окно (PopUp)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Форма открывается в режиме формы как всплывающая. Она отображается над всеми остальными окнами Microsoft Access.	**True** (–1)
Нет	(Значение по умолчанию). Форма не является всплывающей формой.	**False** (0)

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Значение свойства **Всплывающее окно (PopUp)** можно задать только в режиме конструктора формы.

Дополнительные сведения

Для указания типа границы всплывающей формы используют свойство **Тип границы (BorderStyle)**. Обычно, для всплывающих форм выбирают значение свойства **Тип границы (BorderStyle)** «Тонкая».

Совет. Свойства **Всплывающее окно (PopUp), Модальное окно (Modal)** и **Тип границы (BorderStyle)** используют при создании специальных диалоговых окон. Для свойства **Модальное окно (Modal)** при этом задают значение «Да», для свойства **Всплывающее окно** значение «Да», а для свойства **Тип границы (BorderStyle)** выбирают значение «Окна диалога».

Установка значения «Да» свойства **Всплывающее окно (PopUp)** делает форму всплывающей только если выполнено одно из следующих условий:

- форма открыта в режиме формы из окна базы данных;
- форма открыта в режиме формы с помощью макроса или из программы Visual Basic;
- произошло переключение из режима конструктора в режим формы.

Значение «Да» свойства **Всплывающее окно (PopUp)** запрещает переключение из режима формы в другие режимы, панель инструментов при этом недоступна. (Нельзя переключиться из режима формы в режим таблицы даже в макросе или в программе Visual Basic). Для того чтобы открыть такую форму в другом режиме, следует закрыть ее и вновь открыть в режиме конструктора или в режиме таблицы.

В режиме конструктора и в режиме таблицы форма не может быть всплывающей. Форма не будет всплывающей и при переключении из режима таблицы в режим формы.

Примечание. Макрокоманда **ОткрытьФорму (OpenForm)** со значением «Окно диалога» в аргументе «Режим окна» автоматически открывает форму с установленными для свойств **Всплывающее окно (PopUp)** и **Модальное окно (Modal)** значениями «Да».

Совет Если максимизировать какое-либо окно в Microsoft Access, то все остальные окна

также будут максимизированы при открытии или переключении на них. Однако всплывающие окна не максимизируются. Чтобы форма изменяла свой размер при максимизации других окон следует установить значение «Да» для свойства **Всплывающее окно (PopUp)**.

Свойство «Контекстные меню» (ShortcutMenu)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproShortcutMenusC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproShortcutMenusX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproShortcutMenusA"}

Свойство **Контекстные меню (ShortcutMenu)** определяет открытие контекстного меню объекта при нажатии правой кнопки мыши, когда указатель находится на объекте в форме. Разработчик имеет возможность отключить вывод контекстного меню, чтобы запретить пользователю изменять базовый источник записей формы.

Значения

Свойство **Контекстные меню (ShortcutMenu)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Контекстные меню отображаются.	**True** (–1)
Нет	Контекстные меню не отображаются.	**False** (0)

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Дополнительные сведения

Данное свойство управляет отображением контекстных меню для формы и для всех ее элементов управления. Если свойство **Контекстные меню (ShortcutMenu)** принимает значение «Нет», то при нажатии правой кнопки мыши контекстные меню не выводятся для формы и для всех ее элементов управления.

Разработчикам мастеров часто приходится отключать контекстные меню, чтобы запретить пользователям просмотр и использование их команд. Это особенно важно для форм, предназначенных для выбора выведенных на экран параметров. Например, значение «Нет» свойства **Контекстные меню (ShortcutMenu)** формы «Окно запуска» в базе данных «Борей» запрещает пользователям использовать контекстные меню формы и размещенных в ней элементов управления.

Свойство «Контекстные меню» (ShortcutMenu), пример

В данном примере отключается вывод контекстных меню в форме «Счет».

```
Forms!Счет.ShortcutMenu = False
```

Свойство «Индекс перехода по Tab» (TabIndex)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproTabIndexC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproTabIndexX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproTabIndexA"}

Свойство **Индекс перехода по Tab (TabIndex)** задает положение элемента управления в установленной последовательности перехода в форме.

Примечание. Свойство **Индекс перехода по Tab (TabIndex)** применяется только к элементам управления в формах, а не в отчетах.

Значения

Свойство **Индекс перехода по Tab (TabIndex)** имеет целое значение, представляющее положение элемента управления в определенной для формы последовательности перехода. Допустимыми являются значения от 0 для первого элемента управления до полного числа элементов управления минус 1 для последнего элемента. Например, в форме, содержащей 3 элемента управления, для каждого из которых определено свойство **Индекс перехода по Tab (TabIndex)**, допустимыми значениями этого свойства будут 0, 1 и 2.

Отрицательные значения свойства **Индекс перехода по Tab (TabIndex)** приводят к ошибке при выполнении.

Значение данного свойства задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Примечание Последовательность перехода для элементов управления формы также может быть задана с помощью команды **Последовательность перехода** в меню **Вид**. Эта команда устанавливает значение свойства **TabOrder** для элементов управления.

Дополнительные сведения

При размещении каждого элемента управления в форме ему отводится определенное место в последовательности перехода. Каждый новый элемент управления после размещения в форме автоматически помещается в конец последовательности перехода. При изменении пользователем значения свойства **Индекс перехода по Tab (TabIndex)** для одного элемента управления Microsoft Access автоматически выполняет необходимую перенумерацию остальных элементов управления.

В режиме формы скрытые и отключенные элементы управления сохраняют свое положение в последовательности перехода, однако, пропускаются при перемещении по элементам управления с помощью клавиши TAB.

Изменение положения других элементов управления в последовательности перехода не влияет на доступ к элементу управления с помощью назначенной клавиши. Например, если клавиша назначена для подписи поля, то при нажатии этой клавиши фокус будет переведен на поле, даже если значение свойства поля **Индекс перехода по Tab (TabIndex)** было изменено.

При нажатии назначенной клавиши, соответствующей элементу управления, для которого не определено свойство **Индекс перехода по Tab (TabIndex)** (не включенного в последовательность перехода), фокус переводится на следующий в последовательности перехода элемент, который может иметь фокус.

Свойство «Индекс перехода по Tab» (TabIndex), пример

В данном примере изменяется последовательность перехода между кнопкой и полем. Подразумевается, что «Поле1» было создано раньше и имеет значение свойства **Индекс перехода по Tab (TabIndex)** 0, а «Кнопка1» значение 1.

```
Sub Form_Click()
   Me!Кнопка1.TabIndex = 0
   Me!Поле1.TabIndex = 1
End Sub
```

Свойство «Переход по Tab» (TabStop)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproTabStopC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproTabStopX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproTabStopA"}

Свойство **Переход по Tab (TabStop)** указывает, может ли элемент управления получить фокус в режиме формы.

Примечание. Свойство **Переход по Tab (TabStop)** применяется только к элементам управления в формах, а не в отчетах.

Данное свойство неприменимо к флажкам, переключателям и выключателям, входящим в группу. Оно применяется только к самой группе.

Значения

Свойство **Переход по Tab (TabStop)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Можно перевести фокус на элемент управления с помощью клавиши TAB.	**True** (–1)
Нет	Нельзя перевести фокус на элемент управления с помощью клавиши TAB.	**False** (0)

Значение данного свойства задается окне свойств элемента управления, в макросе или в программе Visual Basic.

Дополнительные сведения

При размещении каждого элемента управления в форме ему отводится определенное место в последовательности перехода в форме. Каждый новый элемент управления автоматически помещается в конец последовательности перехода. Если требуется исключить элемент управления из последовательности перехода, следует задать для его свойства **Переход по Tab (TabStop)** значение «Нет».

В режиме формы скрытые и отключенные элементы управления сохраняют свое положение в последовательности перехода, однако, пропускаются при перемещении по элементам управления с помощью клавиши Tab, даже если для них в свойстве **Переход по Tab (TabStop)** задано значение «Да».

Если элемент управления включен (свойство **Доступ** (**Enabled**) имеет значение «Да»), то допускается его выделение с помощью мыши или назначенной клавиши, вне зависимости от значения его свойства **Переход по Tab (TabStop)**. Например, можно задать значение «Нет» для свойства **Переход по Tab (TabStop)** кнопки, чтобы запретить пользователям выделение кнопки с помощью клавиши TAB. Однако пользователи при этом сохранят возможность нажимать кнопку.

Свойство «Дополнительные сведения» (Tag)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproTagC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproTagX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproTagA"}

Свойство **Дополнительные сведения (Tag)** позволяет сохранить произвольные дополнительные сведения о форме, отчете, разделе или элементе управления, используемые в приложении.

Значения

Введите строковое выражение длиной до 2048 символов. Значением по умолчанию является пустая строка ("").

Значение данного свойства задается в окне свойств объекта, в макросе или в программе Visual Basic.

Дополнительные сведения

В отличие от других свойств, значение свойства **Дополнительные сведения (Tag)** не влияет на атрибуты объекта.

Свойство **Дополнительные сведения (Tag)** позволяет определить для объекта кодовую строку без возникновения каких-либо побочных эффектов. Такая строка может быть использована для проверки правильности передачи объекта (формы, отчета, раздела или элемента управления) в процедуру в качестве аргумента.

Свойство «Дополнительные сведения» (Tag), пример

В следующем примере свойство **Дополнительные сведения (Tag)** используется для определения специальных сообщений об элементах управления, размещенных в форме. При получении фокуса элементом управления соответствующий ему текст выводится в надписи с именем «Надпись1». Текст сообщения задается в свойстве **Дополнительные сведения (Tag)** для каждого элемента управления. При получении фокуса элементом управления значение его свойства **Дополнительные сведения (Tag)** присваивается свойству **Подпись (Caption)** элемента управления-надписи.

```
Sub Form_Load()
   Dim frmMessageForm As Form
   Set frmMessageForm = Forms!Форма1
   frmMessageForm!Надпись1.Caption = ""          ' Очистка.
   frmMessageForm!Поле1.Tag = "Это поле с именем 'Поле1'."
   frmMessageForm!Кнопка1.Tag = "Это кнопка с именем 'Кнопка1'"
End Sub

Sub txtDescription_GotFocus()
   Me!Надпись1.Caption = Me!Поле1.Tag     ' Значение свойства выводится в
надписи.
End Sub

Sub Поле1_LostFocus()
   Me!Надпись1.Caption = ""
End Sub

Sub Кнопка1_GotFocus()
   Me!Надпись1.Caption = Me!Кнопка1.Tag         ' Значение свойства
выводится в надписи.
End Sub

Sub Кнопка1_LostFocus()
   Me.Надпись1.Caption = " "
End Sub
```

Свойство «Автоматический размер» (AutoResize)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproAutoResizeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproAutoResizeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproAutoresizeA"}

Свойство **Автоматический размер (AutoResize)** определяет автоматический выбор при открытии окна формы его размеров, достаточных для полного отображения записей.

Значения

Свойство **Автоматический размер (AutoResize)** может иметь следующие значения.

Значения	Описание	Visual Basic
Да	(Значение по умолчанию). Окно формы автоматически открывается с размерами, достаточными для полного отображения записей.	**True** (–1)
Нет	При открытии окна формы восстанавливаются последние сохраненные размеры. Для того чтобы сохранить размеры формы, следует открыть форму, установить нужные размеры, сохранить ее с помощью команды **Сохранить** из меню **Файл** и закрыть форму. В следующий раз форма будет открыта с теми же размерами.	**False** (0)

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Это свойство допускает изменение значения только в режиме конструктора формы.

Дополнительные сведения

Автоматическое изменение размеров окна формы имеет место только при открытии формы в режиме формы. Если форма была открыта в режиме конструктора или режиме таблицы, а затем переведена в режим формы, то ее размеры не изменятся автоматически.

Если свойство формы **Автоматический размер (AutoResize)** имеет значение «Нет», а свойство **Выравнивание по центру** (**AutoCenter**) значение «Да», и пользователь, изменив форму в режиме конструктора, не переключился в режим формы перед сохранением формы, то при следующем открытии формы в режиме формы она будет обрезана по правому и нижнему краям.

Если свойство **Выравнивание по центру (AutoCenter)** имеет значение «Нет», то при открытии окна формы ее верхний левый угол имеет такое же положение, как и в момент закрытия формы.

Свойство «Отмена» (Cancel)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCancelC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproCancelX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproCancelA"}

Свойство **Отмена (Cancel)** позволяет определить кнопку в форме как кнопку **Отмена**.

Значения

Свойство **Отмена (Cancel)** может иметь следующие значения.

Значения	Описание	Visual Basic
Да	Данная кнопка является кнопкой **Отмена**.	**True** (–1)
Нет	(Значение по умолчанию). Данная кнопка не является кнопкой **Отмена**.	**False** (0)

Значение данного свойства задается в окне свойств кнопки, в макросе или в программе Visual Basic.

Дополнительные сведения

Если для свойства кнопки **Отмена (Cancel)** задано значение «Да», то эта кнопка становится кнопкой **Отмена** в форме. Однако следует написать макрос или процедуру обработки события на Visual Basic, определяющую действия, производимые при нажатии кнопки **Отмена** (например, закрытие формы без сохранения внесенных изменений). Установите для свойства события **Нажатие кнопки (OnClick)** значение имени макроса или процедуры обработки события.

Если для свойства кнопки **Отмена (Cancel)** задано значение «Да» и форма является активной формой, то действие кнопки выполняется при нажатии кнопки, при нажатии клавиши ESC или при переводе фокуса на кнопку и нажатии клавиши ENTER.

Примечание. Если в момент нажатия клавиши ESC фокус находится в поле, то будет восстановлено исходное содержимое поля, а все внесенные в него изменения потеряны.

Только одна кнопка в форме может быть кнопкой **Отмена**. Если для кнопки устанавливается значение «Да» свойства **Отмена (Cancel)**, то для всех остальных кнопок в этой форме автоматически устанавливается значение «Нет».

Совет. Для форм, в которых выполнятся необратимые операции, например, удаление, рекомендуется назначить кнопку **Отмена** кнопкой, используемой по умолчанию. Для этого следует одновременно задать для свойств кнопки **Отмена (Cancel)** и **По умолчанию (Default)** значение «Да».

Свойство «По умолчанию» (Default)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDefaultC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDefaultX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDefaultA"}

Свойство **По умолчанию (Default)** определяет, получает ли данная кнопка в форме фокус по умолчанию.

Значения

Свойство **По умолчанию (Default)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Кнопка получает фокус по умолчанию.	**True** (–1)
Нет	(Значение по умолчанию) Кнопка не получает фокус по умолчанию.	**False** (0)

Значение данного свойства задается в окне свойств кнопки, в макросе или в программе Visual Basic.

Дополнительные сведения

Если свойство кнопки **По умолчанию (Default)** имеет значение «Да» и форма является активной формой, то пользователь может нажать эту кнопку с помощью мыши или путем нажатия клавиши ENTER (в последнем случае фокус не должен находиться на другой кнопке).

Только одна кнопка в форме может быть кнопкой, используемой по умолчанию. Если для кнопки устанавливается значение «Да» свойства **По умолчанию (Default)**, то для всех остальных кнопок в этой форме автоматически устанавливается значение «Нет».

Совет. Для форм, в которых выполнятся необратимые операции, например, удаление, рекомендуется назначить кнопку **Отмена** кнопкой, используемой по умолчанию. Для этого следует одновременно задать для свойств кнопки **Отмена** (**Cancel**) и **По умолчанию (Default)** значение «Да».

Свойства «Режим по умолчанию» (DefaultView), «Допустимые режимы» (ViewsAllowed)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproDefaultViewC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproDefaultViewX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproDefaultViewA"}

- **Режим по умолчанию (DefaultView)** – определяет режим открытия формы.
- **Допустимые режимы (ViewsAllowed)** – указывает, могут ли пользователи переходить из режима таблицы в режим формы и обратно с помощью команд **Форма** и **Режим таблицы** из меню **Вид** или при нажатии стрелки, которая находится за кнопкой **Вид,** и выборе **Режим формы** или **Режим таблицы**.

Значения

Свойство **Режим по умолчанию (DefaultView)** может иметь следующие значения.

Значение	Описание	Visual Basic
Простая форма	(Значение по умолчанию). В форме выводится одна запись.	0
Ленточная форма	В форме одновременно выводятся несколько записей (сколько уместится в текущем окне). Каждая запись выводится в отдельной области данных формы.	1
Таблица	Поля формы выводятся в табличном формате (по строкам и столбцам).	2

Свойство **Допустимые режимы (ViewsAllowed)** может иметь следующие значения.

Значения	Описание	Visual Basic
Все	(Значение по умолчанию). Пользователи могут переключаться между режимами формы и таблицы.	0
Форма	Не допускается переход в режим таблицы из режима формы.	1
Таблица	Не допускается переход в режим формы из режима таблицы.	2

Режим конструктора всегда остается доступным (при наличии соответствующих разрешений на доступ).

Значения данных свойств задаются в окне свойств, в макросе или в программе Visual Basic.

Дополнительные сведения

Режимы, отображаемые в списке при нажатии кнопки **Вид** и в меню **Вид** зависят от значения свойства **Допустимые режимы (ViewsAllowed)**. Например, если это свойство принимает значение «Таблица», то элемент **Режим формы** будет недоступен в списке, выводимом при нажатии кнопки **Вид**, и в меню **Вид**.

Ниже представлены результаты возможных комбинаций значений этих свойств.

Режим по умолчанию	Допустимые режимы	Описание
Простая форма, Ленточная форма или Таблица	Все	Пользователи могут переключаться между режимами формы и таблицы.

Простая форма или Ленточная форма	Форма	Не допускается переход из режима формы в режим таблицы.
Простая форма или Ленточная форма	Таблица	Пользователь может перейти из режима формы в режим таблицы, но не может вернуться обратно.
Таблица	Форма	Пользователь может перейти из режима таблицы в режим формы, но не может вернуться обратно.
Таблица	Таблица	Не допускается переход из режима таблицы в режим формы.

Свойства «Число делений по X» (GridX), «Число делений по Y» (GridY)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproGridXYC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproGridXYX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproGridXYA"}

Свойства **Число делений по X (GridX)** и **Число делений по Y (GridY)** задают ширину и высоту ячеек сетки в режиме конструктора формы или в режиме конструктора отчета.

Значения

Целое значение, определяющее число делений (от 1 до 64), приходящихся на одну единицу измерения (обычно, это сантиметры или дюймы). Если на вкладке **Числа** диалогового окна **Языки и стандарты** контрольной панели Windows определена американская **Система измерений**, то по умолчанию свойство **Число делений по X (GridX)** имеет значение 24, а свойство **Число делений по Y (GridY)** значение 24.

Значения данных свойств задаются в окне свойств, в макросе или в программе Visual Basic.

В программе Visual Basic значения данных свойств определяется с помощью числовых выражений.

Дополнительные сведения

Свойства **Число делений по X (GridX)** и **Число делений по Y (GridY)** позволяют управлять размещением и выравниванием объектов в форме или отчете, варьируя густоту сетки. Для вывода сетки на экран выберите команду **Сетка** в меню **Вид**. Если значение свойства **Число делений по X (GridX)** или **Число делений по Y (GridY)** превышает 24, то точки сетки исчезают с экрана (хотя линии сетки остаются).

Свойство Painting

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproPaintingC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproPaintingX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproPaintingA"}

Свойство **Painting** определяет возможность обновления изображения формы или отчета.

Значения

Свойство **Painting** может иметь следующие значения.

Значение	Описание
True (–1)	(Значение по умолчанию) Обновление формы или отчета разрешено.
False (0)	Обновление формы или отчета запрещено.

Значение данного свойства можно задать только в макросе или в программе Visual Basic.

Данное свойство применимо только в режиме формы и недоступно во всех остальных режимах.

Дополнительные сведения

Свойство **Painting** аналогично макрокоманде **ВыводНаЭкран (Echo).** Однако с помощью свойства **Painting** можно запретить обновление изображения отдельной формы или отчета, а с помощью макрокоманды **ВыводНаЭкран (Echo)** - всех окон, открытых в текущем приложении.

Значение **False** свойства **Painting** для формы или отчета приводит также к запрету обновления изображения всех содержащихся в них элементов управления (за исключением подчиненной формы или подчиненного отчета). Для того чтобы запретить обновление изображения элемента управления-подчиненной формы/отчета, необходимо задать значение **False** для свойства **Painting** подчиненной формы или подчиненного отчета. (Обратите внимание на то, что свойство **Painting** следует устанавливать для самой подчиненной формы или подчиненного отчета, а не для элемента управления-подчиненной формы/отчета).

При потере или получении фокуса формой или отчетом для свойства **Painting** автоматически устанавливается значение **True**. Установка значения **False** для формы или отчета приводит к тому, что изменения в форме, отчете или их элементах управления не отображаются. Например, если несколько элементов формы автоматически изменяют размер при изменении размера формы, и требуется скрыть от пользователя перемещения каждого элемента управления, то следует установить для свойства **Painting** значение **False**, затем переместить все элементы управления и установить свойству значение **True**.

Свойство Painting, пример

В данном примере свойство **Painting** используется для разрешения или запрещения обновления изображения формы, в зависимости от значения переменной `SetPainting` **True** или **False**. Если обновление изображения формы разрешено, на экран во время обновления формы выводится значок песочных часов.

```
Function EnablePaint(frmName As Form, ByVal SetPainting As Integer) As
Integer
   frmName.Painting = SetPainting
   If SetPainting = False Then       ' Если обновление запрещено,
      DoCmd.Hourglass True           ' выводится значок.
   Else
      DoCmd.Hourglass False          ' Значок не выводится.
   End If
End Function
```

Свойство «Область выделения» (RecordSelectors)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acproRecordSelectorC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproRecordSelectorX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproRecordSelectorA"}

Свойство **Область выделения (RecordSelectors)** определяет вывод для формы области выделения записей в режиме формы.

Значения

Свойство **Область выделения (RecordSelectors)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Каждая запись имеет область выделения.	**True** (–1)
Нет	Ни одна запись не имеет области выделения.	**False** (0)

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Дополнительные сведения

Данное свойство позволяет удалить область выделения записей при создании специального окна диалога или палитры. Это свойство также применимо для форм, у которых свойство **Режим по умолчанию (DefaultView)** имеет значение «Простая форма».

При редактировании записи в области выделения выводится значок  . Если для свойства **Область выделения (RecordSelectors)** задано значение «Нет», а для свойства **Блокировка записей** (**RecordLocks**) выбрано значение «Изменяемой записи» (жесткая блокировка), видимых признаков блокировки записи нет.

Свойство «Полосы прокрутки» (ScrollBars)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproAllowScrollingC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowScrollingX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowScrollingA"}

Свойство **Полосы прокрутки (ScrollBars)** определяет вывод полос прокрутки в форме или в поле.

Значения

Свойство **Полосы прокрутки (ScrollBars)** может иметь следующие значения.

Значения	Описание	Visual Basic
Отсутствуют	(Значение по умолчанию для поля). Полосы прокрутки в форме или в поле не выводятся.	0
Только по горизонтали	В форме выводится горизонтальная полоса прокрутки. Не применимо к полям.	1
Только по вертикали	В форме выводится вертикальная полоса прокрутки.	2
Все	(Значение по умолчанию для формы). В форме выводится вертикальная и горизонтальная полосы прокрутки. Не применимо к полям.	3

Значение данного свойства задается в окне свойств формы или элемента управления-поля, а также в макросе или в программе Visual Basic.

Для поля можно указать используемое по умолчанию значение этого свойства в окне стандартных свойств элемента управления или с помощью метода **DefaultControl** из Visual Basic.

Дополнительные сведения

Если размеры формы превышают доступные размеры окна, свойство **Полосы прокрутки (ScrollBars)** обеспечивает возможность просмотра всей формы.

Свойство **Поле номера записи** (**NavigationButtons**) определяет вывод на экран кнопок, обеспечивающих переходы по записям.

Свойство «Текст строки состояния» (StatusBarText)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproStatusBarTextC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproStatusBarTextX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproStatusBarTextA"}

Свойство **Текст строки состояния (StatusBarText)** определяет текст, выводящийся в строке состояния, когда элемент управления является выделенным.

Примечание. Свойство **Текст строки состояния (StatusBarText)** применяется только к элементам управления в формах, а не в отчетах.

Значения

Значение свойства **Текст строки состояния (StatusBarText)** задается с помощью строкового выражения длиной до 255 символов.

Значение данного свойства задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Примечание. Длина строки, умещающейся в строке состояния, определяется параметрами компьютера и монитора.

Дополнительные сведения

Свойство **Текст строки состояния (StatusBarText)** используют для вывода инструкций и других сведений об элементе управления. Например, при переводе фокуса на поле можно вывести в строке состояния краткое описание значений, которые следует ввести в это поле.

Примечание. Для вывода всплывающей подсказки используется также свойство **Всплывающая подсказка (ControlTipText).**

Если элемент управления создан путем переноса поля из списка полей, то по умолчанию в свойство **Текст строки состояния (StatusBarText)** копируется значение свойства **Описание** (**Description**) этого поля.

Для элементов управления, содержащих гиперссылку, при перемещении на них указателя мыши в строке состояния выводится адрес гиперссылки (если установлен флажок **Вывод адреса гиперссылки в строку состояния**). Этот флажок находится на вкладке **Гиперссылки/HTML** диалогового окна **Параметры**, которое выводится по команде **Параметры** из меню **Сервис**. При редактировании элемента управления в строке состояния выводится текст, заданный в свойстве **Текст строки состояния (StatusBarText)**.

Свойство «Разрешить добавление» (AllowAdditions)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproAllowAdditionsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproAllowAdditionsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproAllowAdditionsA"}

Свойство **Разрешить добавление (AllowAdditions)** определяет, имеет ли пользователь возможность добавлять записи через форму.

Значения

Свойство **Разрешить добавление (AllowAdditions)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Пользователь имеет возможность добавлять новые записи.	**True** (–1)
Нет	Пользователь не имеет возможности добавлять новые записи.	**False** (0)

Значение свойства **Разрешить добавление (AllowAdditions)** задается в окне свойств формы, в макросе или в программе Visual Basic.

Дополнительные сведения

При заданном для свойства **Разрешить добавление (AllowAdditions)** значении «Нет» пользователям разрешается просмотр и изменение существующих записей, но не разрешается добавление новых записей.

Для того чтобы запретить изменение существующих записей (сделать форму доступной только для чтения), следует задать для свойств **Разрешить добавление (AllowAdditions)**, **Разрешить удаление (AllowDeletions)** и **Разрешить изменение (AllowEdits)** значения «Нет». Можно также сделать записи доступными только для чтения, указав значение «Статический набор» для свойства **Тип набора записей (RecordsetType).**

Если требуется открыть форму только для ввода данных, следует задать для свойства формы **Ввод данных (DataEntry)** значение «Да».

При значении «Нет» свойства **Разрешить добавление (AllowAdditions)** команда **Ввод данных** в меню **Записи** становится недоступной.

Замечание. При использовании аргумента «Режим данных» макрокоманды **ОткрытьФорму** Microsoft Access изменит значения некоторых установленных свойств. Если задать для этого аргумента значение «Изменение», то при открытии формы будут использованы следующие значения свойств:

- **РазрешитьИзменение (AllowEdits)** – Да
- **AllowDeletions** – Да
- **AllowAdditions** – Да
- **DataEntry** – Нет

Для предотвращения изменения значений этих свойств макрокоманды **ОткрытьФорму** необходимо настроить аргумент «Режим данных» так, чтобы использовались значения по умолчанию для данной формы.

Свойство «Разрешить автозамену» (AllowAutoCorrect)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowAutoCorrectC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowAutoCorrectX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowAutoCorrectA"}

Свойство **Разрешить автозамену (AllowAutoCorrect)** определяет, производится ли автоматическая замена значений, введенных пользователем в элементах управления поле или поле со списком в форме.

Значения

Свойство **Разрешить автозамену (AllowAutoCorrect)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Выполняется автоматическая замена значений, перечисленных в списке в окне диалога **Автозамена**.	**True** (–1)
Нет	Автоматическая замена значений не производится.	**False** (0)

Значение свойства **Разрешить автозамену (AllowAutoCorrect)** задается в окне свойств элемента управления, в макросе или в программе Visual Basic. Кроме того, можно определить значение данного свойства в окне стандартных свойств элемента управления или при помощи метода Visual Basic **DefaultControl**.

Дополнительные сведения

Окно диалога **Автозамена** открывается командой **Автозамена** в меню **Сервис**.

Свойство «Разрешить удаление» (AllowDeletions)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproAllowDeletionsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproAllowDeletionsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproAllowDeletionsA"}

Свойство **Разрешить удаление (AllowDeletions)** определяет, имеет ли пользователь возможность удалять записи через форму.

Значения

Свойство **Разрешить удаление (AllowDeletions)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Пользователь имеет возможность удалять записи.	**True** (–1)
Нет	Пользователь не имеет возможности удалять записи.	**False** (0)

Значение свойства **Разрешить удаление (AllowDeletions)** задается в окне свойств формы, в макросе или в программе Visual Basic.

Дополнительные сведения

При заданном для данного свойства значении «Нет» пользователям разрешается просмотр и изменение существующих записей, но не разрешается их удаление. При заданном для свойства **Разрешить удаление (AllowDeletions)** значении «Да» записи могут быть удалены, если это не противоречит существующим условиям целостности данных.

Для того чтобы запретить изменение существующих записей (сделать форму доступной только для чтения), следует задать для свойств **Разрешить добавление (AllowAdditions)**, **Разрешить удаление (AllowDeletions)** и **Разрешить изменение (AllowEdits)** значения «Нет». Можно также сделать записи доступными только для чтения, указав значение «Статический набор» для свойства **Тип набора записей (RecordsetType)**.

При заданном для свойства **Разрешить удаление (AllowDeletions)** значении «Нет» команда **Удалить запись** в меню **Правка** становится недоступной.

Замечание. При использовании аргумента «Режим данных» макрокоманды **ОткрытьФорму (OpenForm)** Microsoft Access изменит значения некоторых установленных свойств. Если задать для этого аргумента значение «Изменение», то при открытии формы будут использованы следующие значения свойств:

- **РазрешитьИзменение (AllowEdits)** – Да
- **AllowDeletions** – Да
- **AllowAdditions** – Да
- **DataEntry** – Нет

Для предотвращения изменения значений этих свойств в макрокоманде **ОткрытьФорму** необходимо настроить аргумент «Режим данных» так, чтобы использовались значения по умолчанию для данной формы.

Свойство «Разрешить изменение» (AllowEdits)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproAllowEditsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproAllowEditsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproAllowEditsA"}

Свойство **Разрешить изменение (AllowEdits)** определяет, имеет ли пользователь возможность изменять записи через форму.

Значения

Свойство **Разрешить изменение (AllowEdits)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Пользователь имеет возможность изменять записи.	**True** (–1)
Нет	Пользователь не имеет возможности изменять записи.	**False** (0)

Значение свойства **Разрешить изменение (AllowEdits)** задается в окне свойств формы, в макросе или в программе Visual Basic.

Дополнительные сведения

Свойство **Разрешить изменение (AllowEdits)** позволяет запретить изменение выводящихся в форме данных. Для того чтобы запретить изменение данных в конкретном элементе управления, следует использовать свойства **Доступ (Enabled)** или **Блокировка (Locked)**.

Для того чтобы запретить изменение существующих записей (сделать форму доступной только для чтения), следует задать для свойств **Разрешить добавление (AllowAdditions)**, **Разрешить удаление (AllowDeletions)** и **Разрешить изменение** значения «Нет». Можно также сделать записи доступными только для чтения, указав значение «Статический набор» для свойства **Тип набора записей (RecordsetType)**.

При значении «Нет» свойства **Разрешить изменение (AllowEdits)** команда **Ввод данных** в меню **Записи** становится недоступной.

Замечание. При использовании аргумента «Режим данных» макрокоманды **ОткрытьФорму (OpenForm)** Microsoft Access изменит значения некоторых установленных свойств. Если задать для этого аргумента значение «Изменение», то при открытии формы будут использованы следующие значения свойств:

- **РазрешитьИзменение (AllowEdits)** – Да
- **AllowDeletions** – Да
- **AllowAdditions** – Да
- **DataEntry** – Нет

Для предотвращения изменения значений этих свойств в макрокоманде **ОткрытьФорму** необходимо настроить аргумент «Режим данных» так, чтобы использовались значения по умолчанию для данной формы.

Свойство «Кнопка закрытия» (CloseButton)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproCloseButtonC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproCloseButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproCloseButtonA"}

Свойство **Кнопка закрытия (CloseButton)** определяет активизацию кнопки закрытия формы.

Значения

Свойство **Кнопка закрытия (CloseButton)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Кнопка закрытия активизирована.	**True** (–1)
Нет	Кнопка закрытия отключена, а команда **Закрыть оконного меню** становится недоступной.	**False** (0)

Значение свойства **Кнопка закрытия (CloseButton)** задается окне свойств формы или отчета, в макросе или в программе Visual Basic.

Значение свойства **Кнопка закрытия (CloseButton)** доступно только в режиме конструктора формы.

Дополнительные сведения

Если для свойства **Кнопка закрытия (CloseButton)** задано значение «Нет», кнопка закрытия остается видимой, но является отключенной. При этом необходимо создать специальное средство закрытия формы, например, кнопку в форме или специальную команду меню, запускающую макрос или процедуру обработки события, приводящую к закрытию формы.

Совет. Для закрытия формы также можно использовать сочетание клавиш ALT+F4.

Свойство ControlType

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproControlTypeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproControlTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproControlTypeA"}

Свойство **ControlType** используется в конструкциях Visual Basic для определения типа элемента управления в форме или отчете.

Значения

Значения свойства **ControlType** задаются с помощью констант, которые определяют тип элемента управления.

Константа	Элемент управления
acLabel	Надпись
acRectangle	Прямоугольник
acLine	Линия
acImage	Рисунок
acCommandButton	Кнопка
acOptionButton	Переключатель
acCheckBox	Флажок
acOptionGroup	Группа параметров
acBoundObjectFrame	Присоединенная рамка объекта
acTextBox	Поле
acListBox	Список
acComboBox	Поле со списком
acSubform	Подчиненная форма/отчет
acObjectFrame	Свободная рамка объекта или диаграммы
acPageBreak	Конец страницы
acPage	Вкладка
acCustomControl	Элемент ActiveX
acToggleButton	Выключатель
acTabCtl	Набор вкладок

Значения свойства **ControlType** могут быть заданы только в программе Visual Basic в режиме конструктора формы или отчета и доступны для чтения во всех режимах.

Дополнительные сведения

Свойство **ControlType** используется не только для проверки типа элемента управления в программе, но и для программного изменения типа элемента управления. Например, можно преобразовать поле в поле со списком, указав для свойства **ControlType** поля значение **acComboBox** в режиме конструктора формы.

Свойство **ControlType** позволяет изменять характеристики близких по типам элементов управления в форме при выполнении определенных условий. Например, если пользователям запрещается изменение данных в полях, то можно задать для свойства полей **Оформление (SpecialEffect)** значения «Обычное», а для свойства формы **Разрешить изменение (AllowEdits)** значение «Нет». Свойство **Оформление (SpecialEffect)** не влияет на режим изменения данных; в этом случае оно просто используется для указания пользователям, что режим редактирования данных изменен.

Свойство **ControlType** используется также для указания типа элемента управления, создаваемого с помощью функции **CreateControl**.

Свойства ControlType, «Разрешить добавление» (AllowAdditions), «Разрешить удаление» (AllowDeletions), «Разрешить изменение» (AllowEdits), пример

В следующем примере значения свойства **ControlType** проверяются для всех элементов управления в форме. Для всех надписей и полей данная процедура изменяет значение свойства **Оформление (SpecialEffect)**. Когда свойство **Оформление (SpecialEffect)** надписи получает значение «С тенью» (Shadowed), свойство **SpecialEffect** поля значение «Обычное» (Normal), а свойства **Разрешить добавление (AllowAdditions)**, **Разрешить удаление (AllowDeletions)** и **Разрешить изменение (AllowEdits)** получают значение «Да» (**True**), Значение переменной `intCanEdit` изменяется для разрешения изменения следующих данных.

```
Sub ToggleControl(frm As Form)
   Dim ctl As Control
   Dim intI As Integer, intCanEdit As Integer
   Const conTransparent = 0
   Const conWhite = 16777215
   For Each ctl in frm.Controls
      With ctl
         Select Case .ControlType
            Case acLabel
               If .SpecialEffect = acEffectShadow Then
                  .SpecialEffect = acEffectNormal
                  .BorderStyle = conTransparent
                  intCanEdit = True
               Else
                  .SpecialEffect = acEffectShadow
                  intCanEdit = False
               End If
            Case acTextBox
               If .SpecialEffect = acEffectNormal Then
                  .SpecialEffect = acEffectSunken
                  .BackColor = conWhite
               Else
                  .SpecialEffect = acEffectNormal
                  .BackColor = frm.Detail.BackColor
               End If
         End Select
      End With
   Next ctl
   If intCanEdit = False Then
      With frm
         .AllowAdditions = False
         .AllowDeletions = False
         .AllowEdits = False
      End With
   Else
      With frm
         .AllowAdditions = True
         .AllowDeletions = True
         .AllowEdits = True
      End With
   End If
End Sub
```

Свойство «Ввод данных» (DataEntry)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDataEntryC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDataEntryX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDataEntryA"}

Свойство **Ввод данных (DataEntry)** определяет режим открытия формы, присоединенной к источнику данных, только для ввода данных. Данное свойство не определяет возможность добавления записи; оно служит для определения выводимости существующих записей.

Значения

Свойство **Ввод данных (DataEntry)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	При открытии формы выводится только пустая запись.	**True** (–1)
Нет	(Значение по умолчанию). В форме выводятся существующие записи.	**False** (0)

Значение свойства **Ввод данных (DataEntry)** задается в окне свойств формы, в макросе или в программе Visual Basic.

Значение данного свойства может быть задано во всех режимах.

Дополнительные сведения

Значение данного свойства может быть задано во всех режимах при значении «Да» свойства **Разрешить добавление (AllowAdditions)**.

Задание значения «Да» для свойства **Ввод данных (DataEntry)** в программе Visual Basic эквивалентно выбору команды **Ввод данных** в меню **Записи**. Значение «Нет», заданное в программе Visual Basic, эквивалентно команде **Удалить фильтр** в меню **Записи**.

Замечание. При использовании аргумента «Режим данных» макрокоманды **ОткрытьФорму** Microsoft Access изменит значения некоторых установленных свойств. Если задать для этого аргумента значение «Изменение», то при открытии формы будут использованы следующие значения свойств:

- **РазрешитьИзменение (AllowEdits)** – Да
- **AllowDeletions** – Да
- **AllowAdditions** – Да
- **DataEntry** – Нет

Для предотвращения изменения значений этих свойств макрокоманды **ОткрытьФорму (OpenForm)** необходимо настроить аргумент «Режим данных» так, чтобы использовались значения по умолчанию для данной формы.

Свойство «Несвязное выделение» (MultiSelect)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproMultiSelectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproMultiSelectX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproMultiSelectA"}

Свойство **Несвязное выделение (MultiSelect)** определяет режим одновременного выделения нескольких элементов списка в форме.

Значения

Свойство **Несвязное выделение (MultiSelect)** может иметь следующие значения.

Значение	Описание	Visual Basic
Отсутствует	(Значение по умолчанию). Выделение нескольких элементов не допускается.	0
Простой	Выделение нескольких элементов или отмена выделения производится при выборе элементов с помощью мыши или путем нажатия клавиши ПРОБЕЛ.	1
Со связным выбором	Допускается расширение области выделения с помощью нажатия кнопки мыши или клавиши перемещения курсора при нажатой клавише SHIFT. Элемент списка выделяется или его выделение снимается нажатием кнопки мыши при нажатой клавише CTRL.	2

Значение свойства **Несвязное выделение (MultiSelect)** задается в окне свойств списка, в макросе или в программе Visual Basic.

Значение данного свойства может быть задано только в режиме конструктора формы.

Дополнительные сведения

Свойство **ListIndex** возвращает индекс выделенного элемента. При значениях свойства **Несвязное выделение (MultiSelect)** «Со связным выбором» или «Простой» выделенные элементы определяются с помощью свойства списка **Selected** или с помощью семейства **ItemsSelected**. При одновременном выделении нескольких элементов списка это значение всегда устанавливается в **Null**.

Если свойство **Несвязное выделение (MultiSelect)** имеет значение «Со связным выбором», то при обновлении списка сбрасывается выделение всех элементов, сделанное пользователем.

Свойство «Разделители записей» (DividingLines)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDividingLinesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDividingLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproDividingLinesA"}

Свойство **Разделители записей (DividingLines)** определяет вывод разделительных линий между разделами формы или между записями в ленточной форме.

Значения

Свойство **Разделители записей (DividingLines)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Вывод разделительных линий между разделами формы или между записями в ленточной форме.	**True** (–1)
Нет	Разделительные линии не выводятся.	**False** (0)

Значение свойства **Разделители записей (DividingLines)** задается в окне свойств формы, в макросе или в программе Visual Basic.

Значение данного свойства может быть задано во всех режимах.

Свойство Selected

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproSelectedC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproSelectedX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproSelectedA"}

Свойство **Selected** используется в программах Visual Basic для проверки, выделен ли конкретный элемент списка.

Значения

Значением свойства **Selected** является массив с индексами, отсчитываемыми от 0, в котором фиксируется состояние выделения для каждого элемента списка.

Значение	Описание
True (–1)	Элемент списка выделен.
False (0)	Элемент списка не выделен.

Значение свойства **Selected** задается или считывается в программах Visual Basic.

Данное свойство доступно только во время выполнения.

Дополнительные сведения

Если свойство списка **Несвязное выделение (MultiSelect)** имеет значение «Простой», значение **True** свойства **Selected** может иметь только один элемент списка. Если для свойства списка **Несвязное выделение (MultiSelect)** задано значение «С несвязным выбором» или «Со связным выбором», значение **True** свойства **Selected** могут иметь любые или все элементы списка. Значение свойства **Value** списка несвязного выделения, привязанного к полю, всегда равно **Null**. Выделенные элементы определяются с помощью свойства списка **Selected** или с помощью семейства **ItemsSelected**.

Свойство **Selected** позволяет выделять элементы списка в программе Visual Basic. Например, следующее выражение выделяет пятый элемент списка:

```
Me!ИмяСписка.Selected(4) = True
```

Свойство Selected, пример

В данном примере свойство **Selected** используется для перемещения выделенных элементов из списка с именем `список1` в список-получатель с именем `список2`. Для списка `список2` в свойстве **Тип источника строк (RowSourceType)** указывается «Список значений», а значение свойства элемента управления **Источник строк (RowSource)** собирается из всех элементов, выделенных в списке `список1`. Для свойства списка `список1` **Несвязное выделение (MultiSelect)** задается значение «Со связным выбором» (Extended). Функция CopySelected( ) вызывается нажатием кнопки **КопироватьВыделенные**.

```
Sub КопироватьВыделенные_Click()
   CopySelected Me
End Sub

Function CopySelected(frm As Form) As Integer
   Dim ctlSource As Control
   Dim ctlDest As Control
   Dim strItems As String
   Dim intCurrentRow As Integer
   Set ctlSource = frm!список1
   Set ctlDest = frm!список2
   For intCurrentRow = 0 To ctlSource.Listcount - 1
      If ctlSource.Selected(intCurrentRow) Then
         strItems = strItems & ctlSource.Column(0, intCurrentRow) & ";"
      End If
   Next intCurrentRow
   ' Сброс значения свойства списка-получателя «Источник строк».
   ctlDest.RowSource = ""
   ctlDest.RowSource = strItems
End Function
```

Свойство «Тройное состояние» (TripleState)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproTripleStateC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproTripleStateX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproTripleStateA"}

Свойство **Тройное состояние (TripleState)** определяет способ отображения пустых (**Null**) значений флажка, выключателя или переключателя.

Примечание. Данное свойство неприменимо к конкретному элементу управления, включенному в группу параметров.

Значения

Свойство **Тройное состояние (TripleState)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Элемент управления при переключении проходит три состояния «Да» (выбран), «Нет» (не выбран) и **Null** (неопределенное). При свойстве **значение Null** этот элемент недоступен.	**True** (–1)
Нет	(Значение по умолчанию). Элемент управления имеет значения «Да» или «Нет». Пустые (**Null**) значения рассматриваются как значения «Нет».	**False** (0)

Значение свойства **Тройное состояние (TripleState)** задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Значение данного свойства может быть задано во всех режимах.

Свойство CurrentX, CurrentY

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCurrentXYC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproCurrentXYX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproCurrentXYA"}

Свойства **CurrentX** и **CurrentY** задают горизонтальную и вертикальную координаты позиции, с которой начинаются печать или рисование, выполняемые с помощью метода в отчете. Например, эти свойства позволяют указать положение центра круга, создаваемого в разделе отчета.

Значения

- Свойство **CurrentX** задает значение типа **Single,** используемое в числовом выражении для указания горизонтальной координаты для последующего метода, выполняющего операции печати или рисования.
- Свойство **CurrentY** задает значение типа **Single,** используемое в числовом выражении для указания вертикальной координаты для последующего метода, выполняющего операции печати или рисования.

Эти координаты отсчитываются от верхнего левого угла раздела отчета, в котором содержится ссылка на свойство **CurrentX** или **CurrentY.** Левому краю раздела отчета соответствует значение **CurrentX,** равное 0, верхнему краю раздела отчета соответствует значение 0 свойства **CurrentY.**

Значения свойств **CurrentX** и **CurrentY** задаются в макросе или в процедуре обработки события Visual Basic, указанной в значении свойства **Печать (OnPrint)** раздела отчета.

Дополнительные сведения

Свойство **ScaleMode** позволяет определить используемые для указания координат единицы измерения, такие как пункты, пикселы, символы, дюймы, миллиметры или сантиметры.

При выполнении перечисленных ниже графических методов значения свойств **CurrentX** и **CurrentY** представляют следующие величины.

Метод	Значения свойств CurrentX и CurrentY
Circle	Координаты центра объекта.
Line	Координата конца линии (*x2*, *y2*).
Print	Координаты следующей позиции печати.

Свойство DrawMode

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDrawModeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDrawModeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproDrawModeA"}

Свойство **DrawMode** определяет взаимодействие пера (цвет, используемый для рисования) с существующими цветами фона при рисовании с помощью методов **Line,** **Circle** или **PSet** в отчете, выводящемся для просмотра или на печать.

Значения

Свойство **DrawMode** может иметь следующие значения.

Значение	Описание
1	Черное перо.
2	Инверсия настройки 15 (NotMergePen).
3	Сочетание компонентов цвета, общих для фона и инвертированного пера (MaskNotPen).
4	Инверсия настройки 13 (NotCopyPen).
5	Сочетание компонентов цвета, общих для пера и инвертированного фона (MaskPenNot).
6	Инверсия цвета фона (Invert).
7	Сочетание компонентов цвета пера и фона за вычетом общих компонентов (XorPen).
8	Инверсия настройки 9 (NotMaskPen).
9	Сочетание компонентов цвета, общих для пера и фона (MaskPen).
10	Инверсия настройки 7 (NotXorPen).
11	Перо поднято, поэтому изображение не меняется. Фактически данная настройка задает отключение рисования (Nop).
12	Сочетание цвета фона и инверсированного цвета пера (MergeNotPen).
13	(Значение по умолчанию). Цвет, указанный в значении свойства **Цвет текста (ForeColor)** (CopyPen).
14	Сочетание цвета пера и инвертированного цвета фона (MergePenNot).
15	Сочетание цвета пера и цвета фона (MergePen).
16	Белое перо.

Значение свойства **DrawMode** принадлежит к типу данных **Integer.**

Значение свойства **DrawMode** задается в макросе или в процедуре обработки события Visual Basic, указанной в свойстве **Печать (OnPrint)** раздела отчета.

Дополнительные сведения

Данное свойство позволяет получить дополнительные визуальные эффекты при рисовании в отчете. Microsoft Access сравнивает цвет каждого пиксела рисунка с цветом соответствующего пиксела существующего фонового изображения и определяет итоговый цвет этого пиксела в отчете. Например, значение 7 задает использование оператора **Xor** для сравнения нового пиксела с пикселем фона.

Свойство DrawStyle

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDrawStyleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDrawStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproDrawStyleA"}

Свойство **DrawStyle** задает тип линии при использовании методов **Line** и **Circle** для печати линий в отчете.

Значения

Свойство **DrawStyle** может иметь следующие значения.

Значение	Тип линии
0	(Значение по умолчанию). Сплошная, без заливки
1	Пунктирная, без заливки
2	Точечная, без заливки
3	Штрих-пунктирная, без заливки
4	Штрих-точечная, без заливки
5	Невидимая, без заливки
6	Невидимая, с заливкой

Значение свойства **DrawStyle** принадлежит типу данных **Integer.**

Значение данного свойства задается в макросе или в процедуре обработки события Visual Basic, указанной в свойстве **Печать (OnPrint)** раздела отчета.

Дополнительные сведения

Установка значения свойства **DrawStyle** дает результаты, перечисленные в таблице, только в том случае, когда для свойства **DrawWidth** задано значение 1. Если значение свойства **DrawWidth** превышает 3, то значения свойства **DrawStyle** от 1 до 4 дают сплошную линию (само свойство **DrawStyle** при этом не изменяется).

Свойство DrawWidth

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDrawWidthC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDrawWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDrawWidthA"}

Свойство **DrawWidth** задает ширину линии при использовании методов **Line,** **Circle** и **PSet** при печати или предварительном просмотре отчета.

Значения

Свойство **DrawWidth** может иметь значение типа **Integer** в диапазоне от 1 до 32767, задающее ширину линии в пикселах. По умолчанию задается значение 1, т.е. ширина в 1 пиксел.

Значение данного свойства задается в макросе или в процедуре обработки события Visual Basic, указанной в свойстве **Печать (OnPrint)** раздела отчета.

Дополнительные сведения

Для увеличения ширины линии необходимо увеличить значение данного свойства. Если значение свойства **DrawWidth** превышает 3, то значения свойства **DrawStyle** от 1 до 4 дают сплошную линию (само значение свойства **DrawStyle** при этом не изменяется). Значение 1, заданное для свойства **DrawWidth,** позволяет получить с помощью свойства **DrawStyle** результаты, перечисленные в таблице в разделе описания этого свойства.

Свойство FillStyle

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFillStyleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproFillStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproFillStyleA"}

Свойство **FillStyle** задает тип заливки (прозрачная, сплошная, заполнение по шаблону) кругов и прямоугольников, созданных в отчете с помощью методов **Circle** и **Line.**

Значения

Свойство **FillStyle** может иметь следующие значения.

Значение	Тип заливки
0	Сплошная
1	(Значение по умолчанию). Прозрачная
2	Горизонтальная штриховка
3	Вертикальная штриховка
4	Косая штриховка (снизу вверх)
5	Косая штриховка (сверху вниз)
6	Двойная штриховка (прямая)
7	Двойная штриховка (косая)

Значение данного свойства задается в макросе или в процедуре обработки события Visual Basic, указанной в свойстве **Печать (OnPrint)** раздела отчета.

Дополнительные сведения

Если для свойства **FillStyle** задано значение 0, то круг или прямоугольник имеют сплошную заливку, цвет которой определяется значением свойства **FillColor.** Если данное свойство имеет значение 1, то указанная заливка становится невидимой и прозрачной для цвета фонового объекта.

Для использования свойства **FillStyle** необходимо задать для свойства **Оформление (SpecialEffect)** значение Обычное.

Свойства ScaleHeight, ScaleWidth

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproScaleHeightWidthC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproScaleHeightWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproScaleHeightWidthA"}

Свойства **ScaleHeight** и **ScaleWidth** задают число условных единиц, соответственно, в ширине и высоте страницы для методов **Circle,** **Line,** **PSet** и **Print,** используемых при выводе отчета на просмотр или на печать или при записи его результатов в файл.

Значения

- Свойство **ScaleHeight** содержит значение типа **Single,** используемое в числовом выражении для указания высоты страницы отчета в условных единицах измерения;
- свойство **ScaleWidth** содержит значение типа **Single,** используемое в числовом выражении для указания ширины страницы отчета в условных единицах измерения.

По умолчанию, внутренняя ширина и высота страницы отчета измеряются в единицах твип.

Значения свойств **ScaleHeight** и **ScaleWidth** задаются в макросе или в процедуре обработки события Visual Basic, указанной в свойстве **Печать (OnPrint)** раздела отчета.

Дополнительные сведения

С помощью свойств **ScaleHeight** и **ScaleWidth** можно создать специальную систему координат для рисования или печати. Например, инструкция `ScaleHeight = 100` устанавливает внутреннюю высоту раздела, равную 100 единицам (т.е. шаг координатной сетки по вертикали будет равняться 1/100 высоты раздела).

Для определения системы координат, основанной на стандартных единицах измерения, таких как пункты, пикселы, символы, дюймы, миллиметры или сантиметры, необходимо использовать свойство **ScaleMode.**

При положительных значениях свойств **ScaleHeight** и **ScaleWidth** координаты отсчитываются сверху вниз и слева направо, а при отрицательных снизу вверх и справа налево.

С помощью этих свойств и связанных с ними свойств **ScaleLeft** и **ScaleTop** можно создать специальную систему координат с положительными и отрицательными координатами. Все эти четыре свойства взаимодействуют со свойством **ScaleMode** следующими способами.

- Определение любого из этих четырех свойств автоматически задает для свойства **ScaleMode** значение 0.
- Положительное значение свойства **ScaleMode** автоматически устанавливает значения свойств **ScaleHeight** и **ScaleWidth** соответствующими новой единице измерения и задает для свойств **ScaleLeft** и **ScaleTop** значения 0. Кроме того, в соответствии с новыми координатами изменяются значения свойств текущей точки **CurrentX** и **CurrentY.**

Для определения свойств **ScaleHeight**, **ScaleWidth**, **ScaleLeft** и **ScaleTop** в одной инструкции можно также использовать метод **Scale.**

Примечание. Не путайте свойства **ScaleHeight** и **ScaleWidth** со свойствами **Высота (Height)** и **Ширина (Width).**

Свойства ScaleLeft, ScaleTop

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproScaleLeftTopC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproScaleLeftTopX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproScaleLeftTopA"}

Свойства **ScaleLeft** и **ScaleTop** определяют горизонтальную и вертикальную координаты верхнего левого угла страницы при использовании методов **Circle, Line, PSet** и **Print** в отчете, выводящемся на просмотр или на печать и при сохранении его результатов в файл.

Значения

- Свойство **ScaleLeft** содержит значение типа **Single,** используемое в числовом выражении для указания горизонтальной координаты левого края страницы в установленных единицах измерения. Значение по умолчанию 0.
- Свойство **ScaleTop** содержит значение типа **Single,** используемое в числовом выражении для указания вертикальной координаты верхнего края страницы в установленных единицах измерения. Значение по умолчанию 0.

Значения свойств **ScaleLeft** и **ScaleTop** задаются в макросе или в процедуре обработки события Visual Basic, указанной в свойстве **Печать (OnPrint)** раздела отчета.

Дополнительные сведения

С помощью этих свойств и связанных с ними свойств **ScaleHeight** и **ScaleWidth** можно создать специальную систему координат с положительными и отрицательными координатами. Все эти четыре свойства взаимодействуют со свойством **ScaleMode** следующими способами:

- определение любого из этих четырех свойств автоматически задает для свойства **ScaleMode** значение 0;
- положительное значение свойства **ScaleMode** автоматически устанавливает значения свойств **ScaleHeight** и **ScaleWidth** соответствующими новой единице измерения и задает для свойств **ScaleLeft** и **ScaleTop** значения 0. Кроме того, в соответствии с новыми координатами изменяются значения свойств текущей точки **CurrentX** и **CurrentY.**

Для определения свойств **ScaleHeight**, **ScaleWidth**, **ScaleLeft** и **ScaleTop** в одной инструкции можно также использовать метод **Scale.**

Примечание. Не путайте свойства **ScaleLeft** и **ScaleTop** со свойствами **От левого края (Left)** и **От верхнего края (Top).**

Свойство ScaleMode

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproScaleModeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproScaleModeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproScaleModeA"}

Свойство **ScaleMode** используется в программах Visual Basic для указания единиц измерения для координат на странице при использовании методов **Circle, Line, PSet** и **Print** в отчете, выводящемся на просмотр или на печать и при записи его результатов в файл.

Значения

Свойство **ScaleMode** может иметь следующие значения.

Значение	Описание
0	Указывает, что определено хотя бы одно из свойств **ScaleHeight, ScaleWidth, ScaleLeft** и **ScaleTop.**
1	(Значение по умолчанию). Твип
2	Пункты
3	Пикселы
4	Символы (по горизонтали: 120 твип на символ; по вертикали: 240 твип на символ)
5	Дюймы
6	Миллиметры
7	Сантиметры

Значение этого свойства имеет тип данных **Integer.**

Значение свойства **ScaleMode** задается в макросе или в процедуре обработки события Visual Basic, указанной в свойстве **Печать (OnPrint)** раздела отчета.

Дополнительные сведения

С помощью взаимосвязанных свойств **ScaleHeight, ScaleWidth, ScaleLeft** и **ScaleTop** можно создать специальную систему координат с положительными и отрицательными координатами. Все эти четыре свойства взаимодействуют со свойством **ScaleMode** следующими способами.

- Определение любого из этих четырех свойств автоматически задает для свойства **ScaleMode** значение 0.
- Положительное значение свойства **ScaleMode** автоматически устанавливает значения свойств **ScaleHeight** и **ScaleWidth** соответствующими новой единице измерения и задает для свойств **ScaleLeft** и **ScaleTop** значения 0. Кроме того, в соответствии с новыми координатами изменяются значения свойств текущей точки **CurrentX** и **CurrentY.**

Свойство «Группировка по датам» (DateGrouping)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDateGroupingC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDateGroupingX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDateGroupingA"}

Свойство **Группировка по датам (DateGrouping)** определяет порядок группировки дат в отчете. Например, в настройках для России первым днем недели считается понедельник. Если для свойства **Группировка (GroupOn)** поля типа даты установить значение «По неделям», данные в отчете будут группироваться от понедельника до воскресенья.

Примечание. Свойство **Группировка по датам (DateGrouping)** применяется ко всему отчету, а не только к конкретной группе.

Значения

Свойство **Группировка по датам (DateGrouping)** может иметь следующие значения.

Значение	Описание	Visual Basic
Американский стандарт	Microsoft Access использует стандартные американские значения параметров «Первый день недели» («Воскресенье») и «Первая неделя года» («С 1 января»).	0
Параметры настройки	(Значение по умолчанию). Microsoft Access использует значения параметров «Первый день недели» и «Первая неделя года», указанные в окне **Язык и стандарты** панели управления Windows.	1

Значение данного свойства задается в окне свойств отчета, в макросе или в программе Visual Basic.

Значение свойства **Группировка по датам (DateGrouping)** можно задать только в режиме конструктора отчета или в процедуре обработки события **Открытие (Open)** отчета.

Дополнительные сведения

Свойство **Группировка по датам (DateGrouping)** не влияет на порядок сортировки в отчете.

Свойства «Заголовок группы» (GroupHeader), «Примечание группы» (GroupFooter)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproGroupHeaderFooterC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproGroupHeaderFooterX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproGroupHeaderFooterA"}

Свойства **Заголовок группы (GroupHeader)** и **Примечание группы (GroupFooter)** позволяют создать в отчете области заголовка группы и примечания группы для выбранного поля или выражения. Заголовки и примечания группы используются для обозначения или объединения данных в группы записей. Например, если для свойства **Заголовок группы (GroupHeader)** поля «Типы» задать значение «Да», название каждой группы товаров будет начинаться с названия ее типа.

Значения

Свойства **Заголовок группы (GroupHeader)** и **Примечание группы (GroupFooter)** могут иметь следующие значения.

Значение	Описание
Да	Создает заголовок или примечание группы.
Нет	(Значение по умолчанию). Не создает заголовок или примечание группы.

Значения данных свойств задаются в окне **Сортировка и группировка.**

Значения этих свойств можно задать только в режиме конструктора отчета.

Примечание. Непосредственно задать значения этих свойств в программе Visual Basic невозможно. Для создания области заголовка или примечаний группы по полю или по выражению в программе Visual Basic используйте функцию **CreateGroupLevel.**

Дополнительные сведения

Для того, чтобы указать для следующих свойств группировки — **Группировка (GroupOn), Интервал (GroupInterval)** и **Не разрывать (KeepTogether)** — значения, отличные от задающихся по умолчанию, необходимо сначала задать значение «Да» для одного или обоих свойств **Заголовок группы (GroupHeader)** или **Примечание группы (GroupFooter)** для выбранного поля или выражения.

Свойство «Интервал» (GroupInterval)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproGroupIntervalC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproGroupIntervalX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproGroupIntervalA"}

Свойство **Интервал (GroupInterval)** вместе со свойством **Группировка (GroupOn)** определяет способ группировки записей в отчете. Данное свойство указывает величину интервалов, на которые должны быть разбиты значения в группируемых записях. Допустимые интервалы группировки определяются типом данных и значением свойства **Группировка (GroupOn)** поля или выражения, по которому проводится группировка. Например, значение 1 свойства **Интервал (GroupInterval)** может определять группировка записей по первому символу в названии товара.

Значения

Значения свойства **Интервал (GroupInterval)** принадлежат к типу данных **Integer** и зависят от типа данных поля и установленного значения свойства **Группировка (GroupOn).** По умолчанию для свойства **Интервал (GroupInterval)** задается значение 1.

Значение данного свойства задается в окне **Сортировка и группировка,** в макросе или в программе Visual Basic.

Значение свойства **Интервал (GroupInterval)** можно задать только в режиме конструктора отчета или в процедуре обработки события **Открытие (Open)** отчета.

Дополнительные сведения

Ниже приведены примеры значений свойства **Интервал (GroupInterval)** для полей различных типов.

Тип поля	Группировка	Интервал
Все	По полному значению	(Значение по умолчанию). Автоматически задается значение 1.
Текстовое	По первым символам	Значение 3 задает группировку по первым трем символам значения поля (например, «пароход» и «парусник» попадут в одну группу).
Дата/время	По неделям	Значение 3 задает группировку по двухнедельным интервалам.
Дата/время	По часам	Значение 12 задает группировку по двенадцатичасовым интервалам.

Для того, чтобы присвоить свойству **Интервал (GroupInterval)** значение, отличное от используемого по умолчанию (1), необходимо предварительно задать значение «Да» для одного или для обоих свойств **Заголовок группы (GroupHeader)** или **Примечание группы (GroupFooter)** для выбранного поля или выражения.

Свойство «Уровень группировки» (GroupLevel)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproGroupLevelC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproGroupLevelX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproGroupLevelA"}

Свойство **Уровень группировки (GroupLevel)** используется в программах Visual Basic для ссылки на уровень группировки, в котором выполняется группировка или сортировка отчета.

Значения

Значением свойства **Уровень группировки (GroupLevel)** является массив, в котором каждое значение представляет уровень группировки. Для ссылки на уровень группировки используется следующий синтаксис:

GroupLevel(*n***)**

Число *n* представляет уровень группировки. Нумерация уровней начинается с 0. Первому полю или выражению присваивается уровень 0, второму – уровень 1 и т.д. Всего допустимо использование до 10 уровней группировки (от 0 до 9).

В следующем примере показано, как использовать свойство **Уровень группировки (GroupLevel)** для ссылки на конкретный уровень группировки.

Уровень группировки	Указывает на
GroupLevel(0)	Первое поле или выражение, по которому проводится сортировка или группировка.
GroupLevel(1)	Второе поле или выражение, по которому проводится сортировка или группировка.
GroupLevel(2)	Третье поле или выражение, по которому проводится сортировка или группировка.

Данное свойство используется только в программах Visual Basic при определении свойств **Сортировка (SortOrder), Группировка (GroupOn), Интервал (GroupInterval), Не разрывать (KeepTogether)** и **Данные (ControlSource).** Для задания значений этих свойств используется процедура обработки события **Открытие (Open)** отчета.

Дополнительные сведения

В отчетах допускается группировка и сортировка данных по нескольким полям или выражениям. Каждое поле или выражение, по которым проводится группировка и сортировка, образуют отдельный уровень группировки.

Для указания полей и выражений, по которым проводится группировка или сортировка, используется функция **CreateGroupLevel.**

Если в отчете уже определена группа (для свойства **Уровень группировки (GroupLevel)** задано значение 0), то свойство **Данные (ControlSource)** позволяет переопределить уровень группировки в процедуре обработки события **Открытие (Open)** отчета. Например, в следующей процедуре при открытии формы `Сортировка` значение свойства **Данные (ControlSource)** заменяется на значение, содержащееся в поле `Приглашение`:

```
Private Sub Report_Open(Cancel As Integer)
   Me.GroupLevel(0).ControlSource _
      = Forms!Сортировка!Приглашение
End Sub
```

Свойства группировки, пример

В данном примере с помощью свойства **Сортировка (SortOrder)** и свойств группировки для первого уровня группировки в отчете «Товары по типам» создается отсортированный в алфавитном порядке список товаров.

```
Private Sub Report_Open(Cancel As Integer)
    ' Задает сортировку по убыванию.
    Me.GroupLevel(0).SortOrder = False
    ' Задает свойство «Группировка» (GroupOn).
    Me.GroupLevel(0).GroupOn = 1
    ' Задает для свойства «Интервал» (GroupInterval) значение 1.
    Me.GroupLevel(0).GroupInterval = 1
    ' Задает для свойства «Не разрывать» (KeepTogether) значение «Первую
область данных».
    Me.GroupLevel(0).KeepTogether = 2
End Sub
```

Свойство «Группировка» (GroupOn)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproGroupOnC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproGroupOnX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproGroupOnA"}

Свойство **Группировка (GroupOn)** определяет способ группировки данных по полю или выражению в зависимости от типа данных. Например, данное свойство позволяет указать для поля даты группировку «По месяцам».

Значения

Набор значений свойства **Группировка (GroupOn)** для поля определяется его типом данных, как показано в следующей таблице. Для выражений допустимо использование всех типов данных. По умолчанию для всех типов данных задается группировка «По полному значению».

Тип поля	Значение	Группа	Visual Basic
Текстовое	(Значение по умолчанию). По полному значению	Записи, имеющие одинаковые значения поля или выражения.	0
	По первым символам	Записи, имеющие одинаковые *n* первых символов в значении поля или выражения.	1
Дата/время	(Значение по умолчанию). По полному значению	Записи, имеющие одинаковые значения поля или выражения.	0
	По годам	Даты, относящиеся к одному календарному году.	2
	По кварталам	Даты, относящиеся к одному календарному кварталу.	3
	По месяцам	Даты, относящиеся к одному месяцу.	4
	По неделям	Даты, относящиеся к одной неделе.	5
	По дням	Совпадающие даты.	6
	По часам	Значения времени, относящиеся к одному часу.	7
	По минутам	Значения времени, относящиеся к одной минуте.	8
Счетчик, денежное, числовое	(Значение по умолчанию). По полному значению	Записи, имеющие одинаковые значения поля или выражения.	0
	Интервал	Значения, попадающие в указанный интервал.	9

Значение свойства **Группировка (GroupOn)** задается в окне **Сортировка и группировка,** в макросе или в программе Visual Basic.

В программах Visual Basic значение данного свойства задается в процедуре обработки события **Открытие (Open)** отчета.

Дополнительные сведения

Для того, чтобы присвоить свойству **Группировка (GroupOn)** значение, отличное от стандартного «По полному значению», необходимо предварительно установить значение «Да» для одного или обоих свойств выбранного поля или выражения **Заголовок группы (GroupHeader)** или **Примечание группы (GroupFooter).**

Свойство «Не разрывать» (KeepTogether) - Группы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproKeepTogetherGroupsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproKeepTogetherGroupsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproKeepTogetherGroupsA"}

Свойство **Не разрывать (KeepTogether)** указывает, следует ли печатать разные элементы данной группы — в том числе заголовок группы, область данных и примечание группы — на одной странице отчета. Например, можно указать, что заголовок группы всегда должен печататься на одной странице с первой областью данных.

Значения

Свойство группы **Не разрывать (KeepTogether)** может иметь следующие значения.

Значение	Описание	Visual Basic
Нет	(Значение по умолчанию). Группа печатается без обязательного удерживания разных областей на одной странице.	0
Полную группу	Вся группа печатается на одной странице.	1
Первую область данных	Заголовок группы печатается на текущей странице только в том случае, если вместе с ним помещается первая запись области данных.	2

Значение свойства **Не разрывать (KeepTogether)** задается в окне **Сортировка и группировка,** в макросе или в программе Visual Basic.

В программе Visual Basic значение данного свойства можно задать в режиме конструктора отчета или в процедуре обработки события **Открытие (Open)** отчета с помощью свойства **Уровень группировки (GroupLevel).**

Дополнительные сведения

Для того, чтобы присвоить свойству **Не разрывать (KeepTogether)** значение, отличное от «Нет», необходимо предварительно задать значение «Да» для одного или для обоих свойств выбранного поля или выражения **Заголовок группы (GroupHeader)** или **Примечание группы (GroupFooter).**

Группа состоит из заголовка группы, области данных и примечания группы. Значение «Полную группу» свойства **Не разрывать (KeepTogether)** игнорируется, если группа слишком велика, чтобы уместиться на одной странице. Аналогично, значение «Первую область данных» игнорируется, если заголовок группы или первая запись области данных слишком велики, чтобы уместиться на одной странице.

Если для свойства **Не разрывать (KeepTogether)** раздела задано значение «Нет», а то же свойство группы имеет значение «Полную группу» или «Первую область данных», свойство **Не разрывать (KeepTogether)** для раздела игнорируется.

Свойство «Сортировка» (SortOrder)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproSortOrderC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproSortOrderX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproSortOrderA"}

Свойство **Сортировка (SortOrder)** определяет порядок сортировки для полей и выражений в отчете. Например, при печати списка поставщиков можно расположить записи по названиям организаций.

Значения

Свойство **Сортировка (SortOrder)** может иметь следующие значения.

Значение	Описание	Visual Basic
По возрастанию	(Значение по умолчанию). Сортировка значений по возрастанию («А»-«Я», 0-9).	**False** (0)
По убыванию	Сортировка значений по убыванию («Я»-«А», 9-0).	**True** (–1)

Значение свойства **Сортировка (SortOrder)** задается в окне **Сортировка и группировка,** в макросе или в программе Visual Basic.

В программе Visual Basic значение свойства **Сортировка (SortOrder)** можно задать в режиме конструктора отчета или в процедуре обработки события **Открытие (Open)** отчета с помощью свойства **Уровень группировки (GroupLevel).**

Свойство ImageHeight

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproImageHeightC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproImageHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproImageHeightA"}

Свойство **ImageHeight** используется в программах Visual Basic для определения в единицах твип высоты рисунка в элементе управления-рисунке.

Значения

Значение свойства **ImageHeight** принадлежит к типу **Integer** и определяет высоту рисунка в единицах твип.

Данное свойство доступно только для чтения во всех режимах.

Дополнительные сведения

Свойство **ImageHeight** вместе со свойством **ImageWidth** позволяет определить размеры рисунка в элементе управления-рисунке. Значения этих свойств позволят согласовать размеры выводящегося рисунка с размерами элемента управления, которые определятся свойствами **Высота (Height) и Ширина (Width).**

Свойства ImageHeight, ImageWidth, пример

В данном примере выводится приглашение пользователю указать имя файла рисунка, которое задается в качестве значения свойства **Рисунок (Picture)** элемента управления «Рисунок1». С помощью свойств **ImageHeight** и **ImageWidth** размеры элемента управления изменятся по размерам рисунка.

```
Sub GetNewPicture(frm As Form)
   Dim ctlImage As Control
   Set ctlImage = frm!Рисунок1
   ctlImage.Picture = InputBox("Введите полное имя файла рисунка")
   ctlImage.Height = ctlImage.ImageHeight
   ctlImage.Width = ctlImage.ImageWidth
End Sub
```

Свойство ImageWidth

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproImageWidthC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproImageWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproImageWidthA"}

Свойство **ImageWidth** используется в программах Visual Basic для определения в единицах твип ширины рисунка в элементе управления-рисунке.

Значения

Значение свойства **ImageWidth** принадлежит к типу **Integer** и определяет ширину рисунка в единицах твип.

Данное свойство доступно только для чтения во всех режимах.

Дополнительные сведения

Свойство **ImageWidth** вместе со свойством **ImageHeight** позволяет определить размеры рисунка в элементе управления-рисунке. Значения этих свойств позволят согласовать размеры выводящегося рисунка с размерами элемента управления, которые определяются свойствами **Высота (Height) и Ширина (Width).**

Свойство EventProcPrefix

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproEventProcPrefixC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproEventProcPrefixX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproEventProcPrefixA"}

Свойство **EventProcPrefix** возвращает префиксную часть имени процедуры обработки события. Например, если для кнопки определена процедура обработки события с именем Адрес_Click, то свойство **EventProcPrefix** возвращает строку «Адрес».

Значения

Значением свойства **EventProcPrefix** является строковое выражение.

Данное свойство доступно только в макросе или в программе Visual Basic и во всех режимах допускает только чтение.

Дополнительные сведения

Имя процедуры обработки событий Microsoft Access состоит из префиксной части, символа подчеркивания (_) и имени события.

Свойство «Интервал таймера» (TimerInterval)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproTimerIntervalC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproTimerIntervalX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproTimerIntervalA"}

Свойство **Интервал таймера (TimerInterval)** позволяет указать измеряемый в миллисекундах промежуток времени между событиями **Таймер (Timer)** для формы.

Значения

Значением свойства **Интервал таймера (TimerInterval)** является целое число типа **Long Integer** в интервале от 0 до 2 147 483 647.

Значение данного свойства задается в окне свойств формы, с помощью макроса или в программе Visual Basic.

Примечание. Visual Basic позволяет задать значение свойства **Интервал таймера (TimerInterval)** в процедуре обработки события формы **Загрузка (Load)**.

Дополнительные сведения

Программу Visual Basic, которая должна выполняться через определенные промежутки времени, указанные в свойстве **Интервал таймера (TimerInterval)**, следует поместить в процедуру обработки события формы **Таймер (Timer)**. Например, для выполнения повторного запроса по записям формы через каждые 30 секунд, следует поместить в процедуру обработки события формы **Таймер (Timer)** программу, выполняющую повторный запрос, и задать для свойства **Интервал таймера (TimerInterval)** значение 30 000.

Свойство «Интервал таймера» (TimerInterval), пример

В данном примере показано, как создать в форме мигающую кнопку, делая значок на кнопке то видимым, то невидимым. В процедуре обработки события формы **Загрузка (Load)** для свойства формы **Интервал таймера (TimerInterval)** задается значение 1000, в результате чего значок появляется или исчезает каждую секунду.

```
Sub Form_Load()
   Me.TimerInterval = 1000
End Sub

Sub Form_Timer()
   Static intShowPicture As Integer
   If intShowPicture Then
      ' Выводится значок.
      Me!btnPicture.Picture = "C:\Icons\Flash.ico"
   Else
      ' Значок не выводится.
      Me!btnPicture.Picture = ""
   End If
   intShowPicture = Not intShowPicture
End Sub
```

Свойство «Автоматический повтор» (AutoRepeat)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproAutoRepeatC;vafctdoevents"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproAutoRepeatX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproAutoRepeatA"}

Свойство **Автоматический повтор (AutoRepeat)** определяет, следует ли повторно выполнять процедуру обработки события или макрос, пока кнопка в форме остается нажатой.

Значения

Свойство **Автоматический повтор (AutoRepeat)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Выполнение макроса или процедуры обработки события **Нажатие кнопки (Click)** повторяется неоднократно до тех пор, пока кнопка остается нажатой.	**True** (–1)
Нет	(Значение по умолчанию). Макрос или процедура обработки события выполняется только один раз.	**False** (0)

Значение данного свойства задается в окне свойств кнопки, с помощью макроса или в программе Visual Basic.

Дополнительные сведения

Первый повторный запуск процедуры обработки события или макроса выполняется спустя 0,5 секунды после первого запуска. Интервал между следующими запусками равняется 0,25 секунды или времени выполнения процедуры или макроса (если это время превышает 0,25 с).

Если программа, связанная с кнопкой, вызывает изменение текущей записи, то значение свойства **Автоматический повтор (AutoRepeat)** игнорируется.

Если программа, связанная с кнопкой, изменяет другой элемент управления в форме, то для корректного выполнения обновления экрана используется функция **DoEvents**.

Свойство Count

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproCountC;daproCount;vaproCount"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproCountX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproCountA"}

Свойство **Count** возвращает число открытых форм или отчетов или число элементов управления в открытой форме или отчете.

Значения

Свойство **Count** имеет значение типа **Integer** и во всех режимах допускает только чтение.

Свойство **Count** объекта определяется в макросе или в программе Visual Basic.

Если открытых форм и отчетов нет, значение свойства **Count** равняется 0.

Свойство Count, пример

В данном примере свойство **Count** используется для управления циклом, в котором выводятся на печать сведения о всех открытых формах и их элементах управления.

```
Sub Print_Form_Controls()
   Dim frm As Form, intI As Integer
   Dim intJ As Integer
   Dim intControls As Integer, intForms As Integer
   intForms = Forms.Count    ' Число открытых форм.
   If intForms > 0 Then
      For intI = 0 To intForms - 1
         Set frm = Forms(intI)
         Debug.Print frm.Name
         intControls = frm.Count
         If intControls > 0 Then
            For intJ = 0 To intControls - 1
               Debug.Print vbTab; frm(intJ).Name
            Next intJ
         Else
            Debug.Print vbTab; "(элементов управления нет)"
         End If
      Next intI
   Else
      MsgBox "Открытых форм нет.", vbExclamation, "Элементы управления"
   End If
End Sub
```

В следующем примере определяются и сохраняются в переменных числа элементов управления в форме и отчете.

```
Dim intFormControls As Integer
Dim intReportControls As Integer
intFormControls = Forms!Сотрудники.Count
intReportControls = Reports!СтоимостьДоставки.Count
```

Свойство CurrentView

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCurrentViewC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproCurrentViewX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acprocurrentviewA"}

Свойство **CurrentView** позволяет определить текущий режим отображения формы. Оно используется, например, когда перед помещением с помощью программы Visual Basic элементов управления в форму, необходимо проверить, находится ли эта форма в режиме формы или в режиме таблицы,

Значения

Свойство **CurrentView** может иметь следующие значения.

Значение	Описание
0	Форма находится в режиме конструктора.
1	Форма находится в режиме формы.
2	Форма находится в режиме таблицы.

Данное свойство доступно только в макросе или в программе Visual Basic и во всех режимах допускает только чтение.

Дополнительные сведения

Данное свойство позволяет выполнять разные задачи в зависимости от текущего режима. Например, процедура обработки события может определять текущий режим, в котором находится форма, и выполнять одну операцию в режиме формы, и другую — в режиме таблицы.

Свойство CurrentView, пример

В следующем примере подпрограмма GetCurrentView используется, чтобы проверить, находится ли форма в режиме формы или в режиме таблицы. В режиме формы сообщение для пользователя выводится в поле в форме; в режиме таблицы то же сообщение выводится в окне сообщений.

```
GetCurrentView Me, "Обратитесь к системному администратору."

Sub GetCurrentView(frm As Form, strDisplayMsg As String)
   Const conFormView = 1
   Const conDataSheet = 2
   Dim intView As Integer
   intView = frm.CurrentView
   Select Case intView
      Case conFormView
         frm!MessageTextBox.SetFocus
         ' Выводит сообщение в поле.
         frm!MessageTextBox = strDisplayMsg
      Case conDataSheet
         ' Выводит сообщение в окне сообщений.
         MsgBox strDisplayMsg
   End Select
End Sub
```

Свойства «Идентификатор справки» (HelpContextID), «Файл справки» (HelpFile)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproHelpContextC;daproHelpContext;vafctMsgBox":1}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproHelpContextX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproHelpContextA":1}

- Свойство **Идентификатор справки (HelpContextID)** определяет контекстный идентификатор раздела специального файла справочной системы, указанного в свойстве **Файл справки (HelpFile)**.
- Свойство **Файл справки (HelpFile)** определяет имя специального файла справочной системы для текущей формы или отчета.

Значения

- Для определения свойства **Идентификатор справки (HelpContextID)** следует ввести значение типа **Long Integer** в диапазоне от 0 до 2 147 483 647, представляющее контекстный идентификатор раздела файла справки, который требуется вывести на экран. По умолчанию задается значение 0.
- Для определения свойства **Файл справки (HelpFile)** следует ввести строковое выражение, содержащее путь и имя специального файла справочной системы, созданного с помощью компилятора справочной системы Windows.

Значения данных свойств задаются в окне свойств, с помощью макроса или в программе Visual Basic. Эти свойства не применяются к элементам управления в отчетах.

Дополнительные сведения

Примечание. Для создания специального файла справки необходим компилятор справочной системы Windows и текстовый редактор, поддерживающий формат RTF, например, Microsoft Word.

Пользователь имеет возможность создавать специальные файлы справки для форм, отчетов и приложений, разработанных в Microsoft Access.

При нажатии клавиши F1 в режиме формы Microsoft Access вызывает приложение, управляющее справочной системой Windows, загружает специальный файл справки, указанный в свойстве **Файл справки (HelpFile)** формы или отчета, и открывает раздел справки, указанный в свойстве **Идентификатор справки (HelpContextID)**.

Если значение свойства элемента управления **Идентификатор справки (HelpContextID)** равняется 0 (значение по умолчанию), то для определения нужного раздела справки используются значения свойств **Идентификатор справки (HelpContextID)** и **Файл справки (HelpFile)** формы. При нажатии клавиши F1 в режиме, отличном от режима формы, а также при нулевом значении свойства **Идентификатор справки (HelpContextID)** и для формы, и для элемента управления, выводится раздел справки Microsoft Access.

Свойства «Идентификатор справки» (HelpContextID), «Файл справки» (HelpFile) — Применение

Свойство **Идентификатор справки (HelpContextID)**

Свойство **Файл справки (HelpFile)**

Свойство hWnd

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS.
nїSnїSnїSnїSnїS":"acprohWndC;vaconUnderstandingNamedOptionalArgs;vastmDeclare":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acprohWndX":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acprohWndA"}

Свойство **hWnd** позволяет определить дескриптор (уникальное значение типа **Long Integer**), который Microsoft Windows присваивает текущему окну.

Значения

Данное свойство допускает только чтение и доступно только в макросе или в программе Visual Basic.

Дополнительные сведения

Данное свойство используется в программах Visual Basic при вызовах функций интерфейса программирования приложений Windows API (Application Programming Interface) или других внешних программ, требующих указания дескриптора **hWnd** в качестве аргумента. Дескриптор текущего окна является одним из аргументов многих функций Windows.

Внимание! Поскольку значение этого свойства может изменяться в процессе выполнения программы, не имеет смысла запоминать значение свойства **hWnd** в общей переменной.

Свойство hWnd, пример

В данном примере значение свойства **hWnd** используется как аргумент функции интерфейса программирования приложений Windows (API) **IsZoomed** для проверки, развернуто ли окно на полный экран.

```
' Введите следующую строку в раздел описаний модуля.
Declare Function IsZoomed Lib "user32" (ByVal hWnd As Long) As Long

Sub Form_Activate()
   Dim intWindowHandle As Long
   intWindowHandle = Screen.ActiveForm.hWnd
   If Not IsZoomed(intWindowHandle) Then
      DoCmd.Maximize
   End If
End Sub
```

Свойство InSelection

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acproInSelectionC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproInSelectionX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproInSelectionA"}

Свойство **InSelection** позволяет определить, выделен ли элемент управления формы в режиме конструктора, или задать выделение элемента управления.

Значения

Свойство **InSelection** может иметь следующие значения.

Значение	Описание
True (–1)	Элемент управления выделен.
False (0)	Элемент управления не выделен.

Данное свойство доступно только в макросе или в программе Visual Basic.

Дополнительные сведения

Вокруг выделенного элемента управления выводятся маркеры изменения размера, позволяющие изменять размеры объекта с помощью мыши. Допускается одновременное выделение нескольких элементов управления.

Свойство InSelection, пример

Следующая функция использует свойство **InSelection** для проверки, выделен ли элемент управления формы `strControlName`.

Для проверки работы данного примера вставьте функцию IsControlSelected в раздел описаний программного модуля в учебной базе данных «Борей», откройте форму «Клиенты» в режиме конструктора и выберите элемент управления «Название». Затем введите следующую строку в окно отладки:

```
? IsControlSelected (Forms!Клиенты, "Название")

Function IsControlSelected(frm As Form, strControlName As String) As
Integer
   Dim intI As Integer, ctl As Control
   If frm.CurrentView <> 0 Then
      ' Форма не находится в режиме конструктора.
      Exit Function
   Else
      For intI = 0 To frm.Count - 1
         Set ctl = frm(intI)
         If ctl.InSelection = True Then
            ' Выделен нужный элемент управления?
            If UCase(ctl.Name) = UCase(strControlName) Then
               IsControlSelected = True
               Exit Function
            End If
         Else
            IsControlSelected = False
         End If
      Next intI
   End If
End Function
```

Свойство OpenArgs

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproOpenArgsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproOpenArgsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproOpenArgsA"}

Свойство **OpenArgs** позволяет определить строковое выражение, указанное в качестве аргумента *аргументыОткрытия* метода **OpenForm**, с помощью которого была открыта форма.

Значения

Данное свойство доступно только в макросе или в программе Visual Basic, в которой вызывается метод **OpenForm** объекта **DoCmd.** Это свойство доступно только для чтения во всех режимах.

Дополнительные сведения

Для того чтобы использовать свойство **OpenArgs**, следует открыть форму с помощью метода **OpenForm** объекта **DoCmd** и задать в качестве значения аргумента *аргументыОткрытия* нужное строковое выражение. После этого значение свойства **OpenArgs** может быть использовано в программе, определенной для этой формы, например, в процедуре обработки события **Открытие (Open)**. Кроме того, к этому свойству можно обратиться в макросе (например, связанном с событием **Открытие**) или в выражении, определяющем значение свойства **Данные (ControlSource)** для элемента управления в форме.

Например, рассмотрим ленточную форму, предназначенную для вывода списка клиентов. Если требуется при открытии этой формы установить фокус на запись о конкретном клиенте, можно указать фамилию этого клиента в свойстве **OpenArgs**, а затем использовать макрокоманду **НайтиЗапись (FindRecord)** в макросе, связанном с событием **Открытие**, для перевода фокуса на запись о клиенте с указанной фамилией.

Свойство OpenArgs, пример

В данном примере свойство **OpenArgs** используется для открытия формы «Сотрудники» и перехода к записи об определенном сотруднике. Демонстрируется использование метода **OpenForm** для определения свойства **OpenArgs**. Эта процедура должна быть выполнена в соответствующий момент — например, при возникновении события **После обновления (AfterUpdate)** в специальном окне диалога, предназначенном для ввода новых сведений о сотруднике.

```
Sub OpenToСидоров()
   DoCmd.OpenForm "Сотрудники", acNormal, , , acReadOnly, , "Сидоров"
End Sub

Sub Form_Open(Cancel As Integer)
   Dim strEmployeeName As String
   ' Если свойство OpenArgs содержит фамилию, находит запись
   ' с этой фамилией и выводит ее в форме. Например, если
   ' в свойстве OpenArgs содержится "Сидоров", переходит к
   ' первой записи "Сидоров".
   strEmployeeName = Forms!Сотрудники.OpenArgs
   If Len(strEmployeeName) > 0 Then
      DoCmd.GoToControl "Фамилия"
      DoCmd.FindRecord strEmployeeName, , True, , True, , True
   End If
End Sub
```

В следующем примере для поиска сотрудника, указанного в свойстве **OpenArgs**, используется метод **FindFirst**.

```
Private Sub Form_Open(Cancel As Integer)
   If Not IsNull(Me.OpenArgs) Then
      Dim strEmployeeName As String
      strEmployeeName = Me.OpenArgs
      Dim RS As Recordset
      Set RS = Me.RecordsetClone
      RS.FindFirst "Фамилия = '" & strEmployeeName & "'"
      If Not RS.NoMatch Then
         Me.Bookmark = RS.Bookmark
      End If
   End If
End Sub
```

Свойство PreviousControl

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproPreviousControlC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproPreviousControlX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproPreviousControlA"}

Свойство **PreviousControl** используется с объектом **Screen** для возвращения ссылки на элемент управления, который последним получил фокус.

Значения

Свойство **PreviousControl** содержит ссылку на элемент управления, который последним имел фокус. После задания ссылки на элемент управления пользователь получает доступ ко всем его свойствам и методам.

Данное свойство допускает только чтение и доступно только в макросе или в программе Visual Basic.

Дополнительные сведения

Свойство **PreviousControl** становится доступным для использования только после передачи фокуса в форме от одного элемента управления другому. До этого при попытке обращения к свойству **PreviousControl** Microsoft Access генерирует ошибку.

Свойство PreviousControl, пример

В данном примере выводится сообщение, если последним элементом управления, получившим фокус, не является поле «Последнее».

```
Function ProcessData() As Integer
   Dim ctlPrevious As Control
   ' Ошибка: предыдущий элемент управления не определен.
   Const conNoPreviousControl = 2483
   On Error GoTo Process_Err
   Set ctlPrevious = Screen.PreviousControl
   If ctlPrevious.Name = "Последнее" Then
      ' Блок обработки данных.
         .
         .
         .
      ProcessData = True
   Else
      ' Переводит фокус на поле «Последнее» и выводит сообщение.
      Me!Последнее.SetFocus
      MsgBox "Введите значение."
      ProcessData = False
   End If
Process_Bye:
   Exit Function
Process_Err:
   If Err = conNoPreviousControl Then
      Me!Последнее.SetFocus
      MsgBox "Введите значение для обработки."
      ProcessData = False
   End If
   Resume Process_Bye
End Function
```

Свойство «Цикл табуляции» (Cycle)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCycleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproCycleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproCycleA"}

Свойство **Цикл табуляции (Cycle)** определяет, что происходит при нажатии клавиши TAB, когда фокус находится на последнем элементе управления в присоединенной форме.

Значения

Свойство **Цикл табуляции (Cycle)** может иметь следующие значения.

Значение	Описание	Visual Basic
Все записи	(Значение по умолчанию). Нажатие клавиши TAB переводит фокус с последнего элемента управления формы на первый элемент управления в следующей записи, в соответствии с последовательностью переходов.	0
Текущая запись	Нажатие клавиши TAB переводит фокус с последнего элемента управления записи на первый элемент управления в той же записи в соответствии с последовательностью переходов.	1
Текущая страница	Нажатие клавиши TAB переводит фокус с последнего элемента управления страницы назад, на первый элемент управления на той же странице в соответствии с последовательностью переходов.	2

Значение свойства **Цикл табуляции (Cycle)** задается в окне свойств формы, с помощью макроса или в программе Visual Basic.

Значение свойства **Цикл табуляции (Cycle)** может быть задано в любом режиме.

Дополнительные сведения

При нажатии клавиши TAB фокус перемещается по всем элементам управления в форме в соответствии с определенной для элементов управления формы последовательностью переходов.

Значение «Все записи» свойства **Цикл табуляции (Cycle)** следует использовать в формах, предназначенных для ввода данных. Это позволит пользователю переходить на новую запись посредством нажатия клавиши TAB.

Примечание. Свойство **Цикл табуляции (Cycle)** определяет действия при нажатии клавиши TAB только в той форме, для который это свойство определено. Если фокус получает элемент управления подчиненной формы, включенный в последовательность переходов, то последовательность действий при нажатии клавиши TAB определяется значением свойства **Цикл табуляции (Cycle)** подчиненной формы.

Для того, чтобы элемент управления подчиненной формы потерял фокус, следует нажать CTRL+TAB.

Свойство «Перехват нажатия клавиш» (KeyPreview)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproKeyPreviewC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproKeyPreviewX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproKeyPreviewA"}

Свойство **Перехват нажатия клавиш (KeyPreview)** определяет, вызываются ли процедуры обработки событий клавиатуры сначала для формы, а уже затем для элемента управления.

Примечание. Событиями клавиатуры являются события **Клавиша вниз (KeyDown)**, **Клавиша вверх (KeyUp)** и **Нажатие клавиши (KeyPress)**.

Значения

Свойство **Перехват нажатия клавиш (KeyPreview)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	События клавиатуры принимаются сначала формой, а затем активным элементом управления.	**True** (–1)
Нет	(Значение по умолчанию). События клавиатуры принимаются только активным элементом управления.	**False** (0)

Значение свойства **Перехват нажатия клавиш (KeyPreview)** задается в окне свойств формы, с помощью макроса или в программе Visual Basic.

Значение свойства **Перехват нажатия клавиш (KeyPreview)** может быть задано в любом режиме.

Дополнительные сведения

Свойство **Перехват нажатия клавиш (KeyPreview)** используется для создания процедур обработки событий клавиатуры в формах. Например, в приложении, использующем функциональные клавиши, заданное для свойства **Перехват нажатия клавиш (KeyPreview)** значение **True** позволяет обрабатывать события клавиатуры на уровне формы и избавляет от необходимости писать программы для каждого элемента управления, который может принимать события клавиатуры.

Для того чтобы обрабатывать события клавиатуры только на уровне формы и предотвратить принятие событий клавиатуры элементами управления, следует задать значение 0 для аргумента «KeyAscii» в процедуре обработки события формы **Нажатие клавиши (KeyPress)** формы, а также значения 0 для аргументов «KeyCode» в процедурах обработки событий формы **Клавиша вниз (KeyDown)** и **Клавиша вверх (KeyUp)**.

Форма, не имеющая видимых или доступных элементов управления, автоматически принимает все события клавиатуры.

Свойство «Перехват нажатия клавиш» (KeyPreview), пример

В данном примере свойство **Перехват нажатия клавиш (KeyPreview)** получает значение **True** (–1) в процедуре обработки события формы **Загрузка (Load)**. В результате события клавиатуры принимаются формой, прежде чем они будут приняты каким-либо элементом управления. В процедуре обработки события формы **Клавиша вниз (KeyDown)** выполняется проверка значения аргумента «KeyCode», чтобы определить, какая клавиша была нажата: F2, F3 или F4.

```
Private Sub Form_Load()
   Me.KeyPreview = True
End Sub

Private Sub Form_KeyDown(KeyCode As Integer, Shift As Integer)
   Select Case KeyCode
      Case vbKeyF2
         ' Обработка событий клавиши F2.
      Case vbKeyF3
         ' Обработка событий клавиши F3.
      Case vbKeyF4
         ' Обработка событий клавиши F4.
      Case Else
   End Select
End Sub
```

Свойство «Контекстное меню» (ShortcutMenuBar)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproShortcutMenuBarC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproShortcutMenuBarX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproShortcutMenuBarA"}

Свойство **Контекстное меню (ShortcutMenuBar)** позволяет определить контекстное меню, которое открывается при нажатии правой кнопки мыши в форме, отчете или на элемента управления формы. Для создания контекстных меню следует указать в меню **Вид** пункт **Панели инструментов**, а затем выбрать команду **Настройка**. Подробные сведения о создании специальных панелей инструментов содержатся в разделе Создание специального контекстного меню для текущей базы данных.

Примечание. Свойство **Контекстное меню (ShortcutMenuBar)** применяется только для элементов управления в формах и не применяется для элементов управления в отчетах.

Свойство **Контекстное меню** применяется также для определения макроса создания меню, который используется для вывода на экран контекстного меню таблицы, отчета, формы или элемента управления в форме.

Примечание. В предыдущих версиях Microsoft Access для создания специального контекстного меню следовало указать в качестве значения свойства **Контекстное меню (ShortcutMenuBar)** имя макроса создания меню, а затем создать группу макросов, содержащую команды для этого меню. Все это допускается и в Microsoft Access 97. Однако теперь для создания специального контекстного меню рекомендуется использовать новое диалоговое окно **Настройка**, которое открывается при выборе команды **Настройка** в подменю **Панели инструментов** меню **Вид**.

Значения

Значением данного свойства является имя контекстного меню, которое будет выводиться на экран. Любая панель команд, свойство **Type** которой имеет значение «Контекстное меню», может использоваться как контекстное меню. Если свойство **Контекстное меню (ShortcutMenuBar)** получает пустое значение, то на экран будет выводиться встроенное (используемое по умолчанию) контекстное меню или глобальное контекстное меню текущего приложения. Если для свойства **Контекстное меню** указано значение, которое не является именем существующего контекстного меню или макроса создания меню, то контекстное меню для отчета, формы или элемента управления в форме не выводится (в этом случае используемое по умолчанию контекстное меню также не будет выводиться).

Значение данного свойства задается в окне свойств объекта, с помощью макроса или в программе Visual Basic.

В программе Visual Basic значение данного свойства определяется с помощью строкового выражения, представляющего собой имя контекстного меню, которое требуется вывести.

Для того, чтобы вывести встроенное контекстное меню базы данных, отчета, формы или элемента управления в форме с помощью макроса или программы Visual Basic, следует задать в качестве значения данного свойства пустую строку ("").

Дополнительные сведения

Для того, чтобы создать специальное контекстное меню, сначала необходимо создать панель инструментов, содержащую все команды, которые предполагается включить в это меню. Затем следует воспользоваться диалоговым окном **Свойства панели инструментов**, которое открывается при выборе нужной панели инструментов в диалоговом окне **Настройка** и нажатии кнопки **Свойства**. В диалоговом окне **Свойства панели инструментов** следует задать для свойства **Type** значение «Контекстное меню». После этого данная панель инструментов будет доступна в списке значений свойства **Контекстное меню (ShortcutMenuBar)** в окне свойств отчета, формы или элемента управления в форме.

Применение свойства **Контекстное меню** вместе с объектом **Application** позволяет использовать специальное контекстное меню как глобальное контекстное меню. Однако если для отчета, формы или элемента управления формы в базе данных определено свойство **Контекстное меню**, то вместо глобального контекстного меню базы данных будет выводиться специальное контекстное меню данного объекта. Пользователь может также определить для конкретного отчета, формы или элемента управления в форме специальное контекстное меню, которое будет выводиться только для одного этого объекта; для этого следует задать специальное имя меню для свойства **Контекстное меню** только данного объекта. Если отчет, форма или элемент управления в форме имеет фокус, то при нажатии пользователем правой кнопки мыши будет выводиться специальное контекстное меню этого объекта; в противном случае будет выводиться глобальное контекстное меню базы данных.

Вывод контекстных меню становится невозможным для любого объекта, если для свойства **AllowShortcutMenus** задано значение **False** (0).

Свойство «Кнопка контекстной справки» (WhatsThisButton)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproWhatsThisButtonC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproWhatsThisButtonX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproWhatsThisButtonA"}

Свойство **Кнопка контекстной справки (WhatsThisButton)** определяет, будет ли в области заголовка формы отображаться кнопка контекстной справки **Что это такое?**

Примечание. Кнопка контекстной справки может быть выведена в области заголовка формы только в том случае, если для свойства **Кнопки размеров окна (MinMaxButtons)** выбрано значение «Отсутствуют».

Значения

Свойство **Кнопка контекстной справки (WhatsThisButton)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Кнопка контекстной справки **Что это такое?** выводится в области заголовка.	**True** (–1)
Нет	(Значение по умолчанию). Кнопка контекстной справки **Что это такое?** не выводится в области заголовка.	**False** (0)

Значение свойства **Кнопка контекстной справки (WhatsThisButton)** задается в окне свойств формы, с помощью макроса или в программе Visual Basic.

Значение данного свойства может быть задано только в режиме конструктора формы.

Дополнительные сведения

При нажатии кнопки **Что это такое?** указатель мыши принимает вид знака вопроса. Когда указатель имеет такой вид, при выборе любого элемента управления выводится специальный раздел справки, определяемый значением свойства этого элемента управления **Идентификатор справки (HelpContextID)**. Если для элемента управления не указан специальный раздел справки, открывается специальный раздел справки для формы. Если специальный раздел справки не был определен ни для элемента управления, ни для формы, то выводится справка Microsoft Access.

Свойство «Всплывающая подсказка» (ControlTipText)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproControlTipTextC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproControlTipTextX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproControlTipTextA"}

Свойство **Всплывающая подсказка (ControlTipText)** определяет текст всплывающей подсказки, которая выводится на экран, когда указатель мыши задерживается на элементе управления.

Значения

Значение свойства **Всплывающая подсказка (ControlTipText)** представляет собой строковое выражение длиной до 255 символов.

Значение свойства **Всплывающая подсказка** задается в окне свойств элемента управления, с помощью макроса или в программе Visual Basic.

Для элементов управления в формах имеется возможность определить значение данного свойства, используемое по умолчанию; оно задается с помощью стандартных свойств элемента управления или метода **DefaultControl** в программе Visual Basic.

Значение свойства **Всплывающая подсказка (ControlTipText)** может быть задано в любом режиме.

Дополнительные сведения

Свойство **Всплывающая подсказка (ControlTipText)** дает удобное средство вывода полезных сведений об элементах управления в формах.

Существуют и другие способы вывода сведений о форме или элементе управления в форме. Свойство **Текст строки состояния (StatusBarText)** позволяет вывести в строке состояния сообщение об элементе управления. Для того чтобы вывести более подробную справку для формы или элемента управления, следует использовать свойства **Файл справки (HelpFile)** и **Идентификатор справки (HelpContextID)**.

Свойства базы данных

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acproDatabaseC;dacolContainer;dacolDocument;dacolProperty;damthAppend;damthCreateProperty;daobj
Document"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDatabaseX":1} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDatabaseA;daobjDatabase"}

Свойства базы данных содержат общие сведения о базе данных. Для просмотра свойств базы данных следует выбрать в меню **Файл** команду **Свойства базы данных**.

Значения

Для того чтобы определить значения свойств базы данных, следует заполнить поля на вкладках **Документ** и **Прочие** в диалоговом окне **Свойства:**<*база данных*> или определить эти свойства в программе Visual Basic.

Если какое-либо свойство базы данных не было определено в диалоговом окне **Свойства:**<*база данных*>, то для его определения в программе Visual Basic необходимо сначала создать это свойство с помощью метода **CreateProperty**, а затем добавить его в семейство **Properties** объекта **Document**. Свойства, выводящиеся на вкладке **Документ**, задаются в программе Visual Basic с помощью объекта **Document** с именем «SummaryInfo», входящего в семейство **Documents**. Свойства, выводящиеся на вкладке **Прочие**, задаются с помощью объекта **Document** с именем «UserDefined», входящего в семейство **Documents**. Прежде чем добавить некоторое свойство в базу данных, следует указать его имя, тип данных и значение по умолчанию. Дополнительные сведения можно вывести с помощью выбора пункта **Особенности** в разделе справки о семействе **Documents**.

Примечание. Пользователь имеет возможность задать или изменить значения только тех свойств базы данных, которые выводятся на вкладках **Документ** и **Прочие**. Остальные свойства базы данных доступны только для чтения.

Дополнительные сведения

Значения свойств базы данных выводятся в диалоговом окне **Свойства:**<*база данных*> на следующих вкладках.

Вкладка	Описание
Общие	Содержит те же сведения, которые выводятся при выделении в проводнике Windows имени файла правой кнопкой мыши и последующем выборе в контекстном меню пункта **Свойства**. Единственное отличие заключается в том, что при просмотре свойств базы данных в Microsoft Access параметры группы **Атрибуты** допускают только чтение.
Документ	Сведения, введенные на этой вкладке, заносятся в объект **Document** с именем «SummaryInfo», входящий в семейство **Documents**. Эти данные аналогичны общим сведениям о документе, которые задаются в других приложениях Microsoft Office. Сведения, заданные на вкладке **Документ**, облегчают идентификацию базы данных как в рамках Microsoft Access, так и во внешних программах, таких как служебная программа поиска файлов Windows. Параметр **База гиперссылки** на вкладке **Документ** используется для создания пути базы гиперссылки, который добавляется в начало относительного значения свойства **HyperlinkAddress**.
Статистика	На этой вкладке выводятся дата и время создания активной базы данных, а также даты и время ее последнего изменения, открытия и печати.

Состав	На этой вкладке выводится список имен объектов, содержащихся в базе данных.
Прочие	Значения свойств, введенные на этой вкладке, заносятся в объект **Document** с именем «UserDefined», входящий в семейство **Documents**.

Свойства базы данных, примеры

В данном примере демонстрируется определение или создание специального определяемого пользователем свойства, которое будет выводиться на вкладке **Прочие** в диалоговом окне **Свойства:**<*база данных*>. Например, с помощью вызова функции `ЗаданиеСпециальногоСвойства` новое свойство «ФамилияПользователя» добавляется в объект **Document** с именем «UserDefined» в базе данных. Аргументы, которые передаются в функцию `ЗаданиеСпециальногоСвойства`, требуются для выполнения метода **CreateProperty**.

```
Dim strName As String, strValue As String

' Определяет переменную для имени свойства.
strName = "ФамилияПользователя"
' Определяет переменную для значения свойства.
strValue = InputBox("Введите ваше полное имя")
If ЗаданиеСпециальногоСвойства (strName, dbText, strValue) <> True Then
   ' Ошибка при определении свойства.
   MsgBox "Ошибка при определении свойства."
End If

Function ЗаданиеСпециальногоСвойства(strPropName As String, intPropType _
   As Integer, strPropValue As String) As Integer

   Dim dbs As Database, cnt As Container
   Dim doc As Document, prp As Property

   Const conPropertyNotFound = 3270      ' Ошибка: свойство не найдено.
   Set dbs = CurrentDb                   ' Определяет объект Database.
   Set cnt = dbs.Containers!Databases    ' Определяет объект Container.
   Set doc = cnt.Documents!UserDefined   ' Определяет объект Document.
   On Error GoTo SetCustom_Err
   doc.Properties.Refresh
   ' Задает имя специального свойства. Если возникает ошибка,
   ' это значит, что свойство не существует, и его надо
   ' создать и добавить в семейство объектов Document.
   Set prp = doc.Properties(strPropName)
   prp = strPropValue                    ' Задает значение специального
свойства.
   ЗаданиеСпециальногоСвойства = True

SetCustom_Bye:
   Exit Function

SetCustom_Err:
   If Err = conPropertyNotFound Then
      Set prp = doc.CreateProperty(strPropName, intPropType, strPropValue)
      doc.Properties.Append prp          ' Добавляет в семейство.
      Resume Next
   Else                                  ' Неизвестная ошибка.
      ЗаданиеСпециальногоСвойства = False
      Resume SetCustom_Bye
   End If
End Function
```

В следующем примере демонстрируется вывод сведений, определенных на вкладке **Документ** в диалоговом окне **Свойства:**<*база данных*>. В полях формы выводятся название приложения,

тема и имя автора. Если значение свойства не было определено ранее, то приведенная процедура возвращает «Отсутствует». При возникновении неизвестной ошибки возвращается пустая строка ("").

```
Private Sub Form_Open(Cancel As Integer)
   Dim strTitle As String, strSubject As String, strAuthor As String

   strTitle = "Название"
   strSubject = "Тема"
   strAuthor = "Автор"

   Me!txtTitle = GetSummaryInfo(strTitle)
   Me!txtSubject = GetSummaryInfo(strSubject)
   Me!txtAuthor = GetSummaryInfo(strAuthor)
End Sub

Function GetSummaryInfo(strPropName As String) As String
   Dim dbs As Database, cnt As Container
   Dim doc As Document, prp As Property

   ' Ошибка: свойство не найдено.
   Const conPropertyNotFound = 3270
   On Error GoTo GetSummary_Err
   Set dbs = CurrentDb
   Set cnt = dbs.Containers!Databases
   Set doc = cnt.Documents!SummaryInfo
   doc.Properties.Refresh
   GetSummaryInfo = doc.Properties(strPropName)

GetSummary_Bye:
   Exit Function

GetSummary_Err:
   If Err = conPropertyNotFound Then
      Set prp = doc.CreateProperty(strPropName, dbText, "Отсутствует")
      ' Добавляет в семейство.
      doc.Properties.Append prp
      Resume
   Else
      ' Неизвестная ошибка.
      GetSummaryInfo = ""
      Resume GetSummary_Bye
   End If
End Function
```

Свойства объекта базы данных

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acproObjectPropertiesC;dacolContainer;dacolDocument;daobjDocument;daproDateCreated;daproDescripti
on;daproName;daproOwner"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproObjectPropertiesX":1}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproObjectPropertiesA"}

Свойства объекта базы данных содержат общие сведения об объектах, выводящихся в окне базы данных.

Значения

Для просмотра свойств объекта и задания свойств **Описание** или **Атрибуты** пользователь должен выполнить следующие действия.

- Выделить объект в окне базы данных и нажать кнопку **Свойства** ... на панели инструментов **База данных**.
- Выделить объект в окне базы данных и выбрать команду **Свойства** в меню **Вид**.
- Выделить объект в окне базы данных правой кнопкой мыши и выбрать команду **Свойства** в контекстном меню.

Пользователь имеет также возможность задать или определить значения свойств объекта в программе Visual Basic.

Примечание. Пользователь имеет возможность задать или изменить значения только свойств **Описание** или **Атрибуты**. Остальные свойства объекта базы данных задаются Microsoft Access и доступны только для чтения.

Дополнительные сведения

Объектами базы данных являются таблицы, запросы, формы, отчеты, макросы и модули. Каждый класс объектов базы данных представляется отдельным объектом **Document** в семействе **Containers**. Например, семейство **Containers** содержит объект **Document**, представляющий все формы в базе данных.

В окне базы данных возможен просмотр следующих свойств объектов базы данных.

Свойство	Описание
Имя	Имя объекта, являющее также значением свойства **Имя (Name)** объекта.
Тип	Тип объекта. Объект Microsoft Access может относиться к типу таблицы, запроса, формы, отчета, макроса или модуля.
Описание	Описание объекта, являющееся также значением свойства **Описание (Description)** объекта. Для таблиц и запросов пользователь имеет возможность задать значение свойства **Описание (Description)** в окне свойств объекта. Описание объекта выводится также в окне базы данных рядом с именем объекта, если в меню **Вид** выбрана команда **Таблица**.
Создан	Дата создания объекта. Для таблиц и запросов это свойство совпадает со свойством **DateCreated**.
Изменен	Дата последнего изменения объекта. Для таблиц и запросов это свойство совпадает со свойством **LastUpdated**.
Владелец	Имя владельца объекта. Для получения дополнительных сведений см. раздел справки для свойства **Owner**.
Атрибуты	Данное свойство определяет, является ли объект видимым или скрытым, а также является ли объект реплицируемым в реплике базы данных.

Если для атрибута «Скрытый» задается значение **True** (с помощью установки флажка **Скрытый**), то объект не будет выводиться в окне базы данных. Для того чтобы вывести скрытые объекты, выберите в меню **Сервис** команду **Параметры**, выберите вкладку **Вид** и установите флажок **Скрытые объекты**. Значки скрытых объектов будут выведены в окне базы данных как недоступные. После этого становится возможным снятие атрибута «Скрытый», в результате чего объект снова станет видимым в окне базы данных.

Свойства объекта базы данных, пример

В данном примере подпрограмма `ПечатьСвойствОбъекта` используется для вывода на печать значений свойств объекта в окне отладки. Подпрограмма требует передачи типа и имени объекта в качестве аргументов.

```
Dim strObjectType As String
Dim strObjectName As String
Dim strMsg As String

strMsg = "Введите тип объекта (форма, макрос, модуль" _
   & "запрос, отчет, таблица)."
' Принимает тип объекта.
strObjectType = InputBox(strMsg)
strMsg = "Введите имя формы, макроса, модуля, " _
   & "запроса, отчета или таблицы."
' Принимает имя объекта, введенное пользователем.
strObjectName = InputBox(strMsg)
' Передает тип и имя объекта
' в подпрограмму ПечатьСвойствОбъекта.
ПечатьСвойствОбъекта strObjectType, strObjectName

Sub ПечатьСвойствОбъекта(strObjectType As String, strObjectName _
   As String)
   Dim dbs As Database, ctr As Container, doc As Document
   Dim intI As Integer
   Dim strTabChar As String
   Dim prp As Property

   Set dbs = CurrentDb
   strTabChar = vbTab
   ' Определяет переменную, представляющую объект-контейнер.
   Set ctr = dbs.Containers(strObjectType)
   ' Определяет переменную, представляющую объект-документ.
   Set doc = ctr.Documents(strObjectName)
   doc.Properties.Refresh
   ' Выводит имя объекта в окно отладки.
   Debug.Print doc.Name
   ' Выводит все свойства объекта в окно отладки.
   For Each prp in doc.Properties
      Debug.Print strTabChar & prp.Name & " = " & prp.Value
   Next
End Sub
```

Свойства SelHeight, SelWidth

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproSelHeightC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproSelHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproSelHeightA;daobjQueryDef;daobjTableDef"}

Свойство **SelHeight** позволяет указать или определить число выделенных строк (записей) в текущем выделенном блоке таблицы, запроса или режиме таблицы, или выделенных записей в ленточной форме. Для того чтобы указать или определить число столбцов (полей) в текущем выделенном блоке используют свойство **SelWidth**. Например, если в таблице «Клиенты» выделен прямоугольный диапазон ячеек, то число строк и столбцов в выделенном блоке можно определить с помощью свойств **SelHeight** и **SelWidth**.

Значения

- Свойство **SelHeight**. Значение типа **Long Integer** от 0 до числа записей в режиме таблицы или в ленточной форме. Вызов этого свойства устанавливает или возвращает число строк в выделенном блоке или число выделенных записей в ленточной форме.
- Свойство **SelWidth**. Значение типа **Long Integer** от 0 до числа столбцов в таблице. Вызов этого свойства устанавливает или возвращает число столбцов в выделенном блоке.

В режиме конструктора данные свойства недоступны. Они используются только в макросах или в программах Visual Basic.

Дополнительные сведения

При отсутствии выделения все эти свойства возвращают 0. Установка для любого из них нулевого значения снимает выделение с таблицы или формы.

В случае выделения в таблице одной или более записей (с использованием области выделения), изменение значения свойства **SelWidth** (кроме присвоения ему значения 0) невозможно. В случае выделения одного или более столбцов (с использованием их заголовков), изменение значения свойства **SelHeight** (кроме присвоения ему значения 0) невозможно.

Вместе со свойствами **SelTop** и **SelLeft** данные свойства позволяют указать или определить размеры и позицию выделенного блока в объекте в режиме таблицы. Если выделенного блока нет, свойства **SelTop** и **SelLeft** возвращают номера строки и столбца ячейки, находящейся в фокусе.

Значениями свойств **SelHeight** и **SelWidth** являются координаты правого нижнего угла выделенного блока относительно его верхнего левого угла. Положение верхнего левого угла выделенного блока определяется значениями свойств **SelTop** и **SelLeft**.

Свойства SelHeight, SelWidth, SelTop и SelLeft, пример

В данном примере демонстрируется использование свойств **SelHeight**, **SelWidth**, **SelTop** и **SelLeft** для определения положения и размеров выделенного блока в форме в режиме таблицы. В общей процедуре `ВыделениеБлока` значения высоты и ширины текущего выделенного блока присваиваются переменным `lngHeight`, `lngWidth`, `lngTop` и `lngLeft` и выводятся в окне сообщения.

```
Sub SetHeightWidth(frm As Form)
   Dim lngNumRows As Long, lngNumColumns As Long
   Dim lngTopRow As Long, lngLeftColumn As Long
   Dim strMsg As String

   If frm.CurrentView = 2 Then           ' Форма в режиме таблицы.
      lngNumRows = frm.SelHeight         ' Число выделенных строк.
      lngNumColumns = frm.SelWidth       ' Число выделенных столбцов.
      lngTopRow = frm.SelTop             ' Верхняя выделенная строка.
      lngLeftColumn = frm.SelLeft        ' Левый выделенный столбец.
      strMsg = " Число строк: " & lngNumRows & vbCrLf
      strMsg = strMsg & " Число столбцов: " & lngNumColumns & vbCrLf
      strMsg = strMsg & " Верхняя строка: " & lngTopRow & vbCrLf
      strMsg = strMsg & " Левый столбец: " & lngLeftColumn
      MsgBox strMsg
   End If
End Sub
```

Свойства SelTop, SelLeft

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproSelTopC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproSelTopX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproSelTopA;daobjQueryDef;daobjTableDef"}

Свойство **SelTop** позволяет указать или определить строку (запись), являющуюся верхней строкой в выделенном блоке в таблице, запросе или в режиме таблицы, или выделенных записей в ленточной форме. Для указания или определения крайнего левого столбца (поля) в текущем выделенном блоке используют свойство **SelLeft**. Например, если в таблице «Клиенты» выделен прямоугольный диапазон ячеек, то первую строку и первый столбец в выделенном блоке в режиме таблицы можно определить с помощью свойств **SelTop** и **SelLeft**.

Значения

- Свойство **SelTop**. Значение типа **Long Integer** от 1 до числа записей в режиме таблицы или в ленточной форме. Вызов этого свойства устанавливает или возвращает число верхних строк в выделенном блоке или число выделенных записей в ленточной форме.
- Свойство **SelLeft**. Значение типа **Long Integer** от 1 до числа столбцов в таблице. Вызов этого свойства устанавливает или возвращает число столбцов слева в выделенном блоке.

Свойства **SelTop** и **SelLeft** недоступны в режиме конструктора. Данные свойства используются только в макросах и в программах Visual Basic.

Дополнительные сведения

При отсутствии выделения данные свойства принимают значения номеров строки и столбца ячейки, находящейся в фокусе.

В случае выделения в таблице одной или более записей (с использованием области выделения), изменение значения свойства **SelLeft** невозможно. В случае выделения одного или более столбцов (с использованием их заголовков), изменение значения свойства **SelTop** невозможно.

Вместе со свойствами **SelHeight** и **SelWidth** данные свойства позволяют указать или определить размеры выделенного блока. Свойства **SelTop** и **SelLeft** определяют положение верхнего левого угла выделенного блока. Координаты правого нижнего угла выделенного блока относительно его верхнего левого угла определяют свойства **SelHeight** и **SelWidth**.

Свойство CurrentRecord

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCurrentRecordC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproCurrentRecordX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproCurrentRecordA"}

Свойство **CurrentRecord** используют для определения текущей записи в текущем наборе записей, выведенной в форме.

Значения

Microsoft Access присваивает данному свойству значение типа **Long Integer,** представляющее номер текущей записи, выведенной в форме.

Свойство **CurrentRecord** доступно только для чтения в режиме формы и в режиме таблицы. В режиме конструктора это свойство недоступно. Данное свойство используется только в макросах и в программах Visual Basic.

Дополнительные сведения

Значение данного свойства соответствует значению, выводящемуся в поле номера записи в левом нижнем углу формы.

Свойство CurrentRecord, пример

В данном примере демонстрируется использование свойства **CurrentRecord** для определения номера текущей выведенной записи. В общей процедуре «ТекущаяЗапись» номер текущей записи присваивается переменной `lngrecordnum`.

```
Sub ТекущаяЗапись(frm As Form)
   Dim lngrecordnum As Long

   lngrecordnum = frm.CurrentRecord
End Sub
```

Свойства InsideHeight, InsideWidth

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproInsideHeightC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproInsideHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproInsideHeightA"}

Свойства **InsideHeight** и **InsideWidth** позволяют указать или определить (в единицах твип) внутренние размеры окна, содержащего форму.

Значения

- Свойство **InsideHeight**. Значение типа **Integer**, представляющее высоту внутренней области окна формы в единицах твип.
- Свойство **InsideWidth**. Значение типа **Integer**,представляющее ширину внутренней области окна формы в единицах твип.

Свойства **InsideHeight** и **InsideWidth** являются доступными только в макросах и в программах Visual Basic. Они допускают изменение значений в любое время.

Дополнительные сведения

Для того чтобы определить внутренние размеры самой формы следует использовать свойства **Ширина (Width)** для определения ее ширины, и сумму высот видимых областей (свойство **Высота (Height)**, применяемое только к областям, а не к формам). Внутренней областью формы называют всю область внутри границ формы за исключением полос прокрутки и области выделения записей.

Использование свойств **WindowHeight** и **WindowWidth** позволяет определить высоту и ширину окна формы.

Если окно максимизировано, установка этих не имеет результатов до тех пор, пока оно не будет возвращено к своим исходным размерам.

Свойства InsideHeight, InsideWidth, пример

В данном примере демонстрируется использование свойств **InsideHeight** и **InsideWidth** для сравнения внутренних размеров формы с шириной и высотой окна формы. Если эти размеры не совпадают, то размеры окна формы приводятся в соответствие с размерами формы.

```
Sub РазмерыОкна(frm As Form)
   Dim intWindowHeight As Integer
   Dim intWindowWidth As Integer
   Dim intTotalFormHeight As Integer
   Dim intTotalFormWidth As Integer
   Dim intHeightHeader As Integer
   Dim intHeightDetail As Integer
   Dim intHeightFooter As Integer

   ' Определяет высоту формы.
   intHeightHeader = frm.Section(acHeader).Height
   intHeightDetail = frm.Section(acDetail).Height
   intHeightFooter = frm.Section(acFooter).Height
   intTotalFormHeight = intHeightHeader _
      + intHeightDetail + intHeightFooter
   ' Определяет ширину формы.
   intTotalFormWidth = frm.Width
   ' Определяет высоту и ширину окна.
   intWindowHeight = frm.InsideHeight
   intWindowWidth = frm.InsideWidth

   If intWindowWidth <> intTotalFormWidth Then
      frm.InsideWidth = intTotalFormWidth
   End If
   If intWindowHeight <> intTotalFormHeight Then
      frm.InsideHeight = intTotalFormHeight
   End If
End Sub
```

Свойство NewRecord

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproNewRecordC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproNewRecordX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproNewRecordA"}

Свойство **NewRecord** показывает, является ли текущая текущая запись новой.

Значения

Свойство **NewRecord** может иметь следующие значения.

Значение	Описание
True (–1)	Текущая запись является новой.
False (0)	Текущая запись не является новой.

Свойство **NewRecord** доступно только для чтения в режиме формы и в режиме таблицы. В режиме конструктора это свойство недоступно. Данное свойство используется только в макросах и в программах Visual Basic.

Дополнительные сведения

Когда пользователь переходит на новую запись, свойство **NewRecord** получает значение **True** вне зависимости от того, начал ли пользователь изменять запись.

Свойство NewRecord, пример

В данном примере демонстрируется использование свойства **NewRecord** для проверки, является ли текущая запись новой записью. В общей процедуре «ПроверкаНовойЗаписи» текущая запись связывается с переменной `intnewrec`. Если запись является новой, на экран выводится соответствующее сообщение. Выполнение этой процедуры возможно после возникновения в форме текущего события.

```
Sub ПроверкаНовойЗаписи(frm As Form)
   Dim intnewrec As Integer

   intnewrec = frm.NewRecord
   If intnewrec = True Then
      MsgBox "Вы находитесь в новой записи." _
         & "@Добавить новые данные?" _
         & "@Если нет, перейдите к существующей записи."
   End If
End Sub
```

Свойство StartupForm

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproStartupFormC; dacolProperty; damthCreateProperty"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproStartupFormX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproStartupFormA;daobjDatabase"}

Свойство **StartupForm** позволяет указать имя формы, автоматически открывающейся при запуске базы данных. Например, это свойство удобно использовать для вывода на экран специальной формы, содержащей меню с именами всех доступных форм, запросов и отчетов в данном приложении Microsoft Access.

Значения

Значение свойства **StartupForm** задается с помощью строкового выражения, которое определяет имя формы, содержащейся в текущей базе данных.

Для облегчения установки данных свойств используется поле **Форма** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Значение данного свойства задается в макросе или в программе Visual Basic.

Для установки свойства **StartupForm** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

Свойство **StartupForm** имеет приоритет над значением, указанным в аргументе макрокоманды Open Form в макросе AutoExec. Microsoft Access запускает макрос AutoExec после проверки всех свойств, задающих параметры запуска, поэтому не следует использовать макрокоманду **ОткрытьФорму (OpenForm)** в макросе AutoExec, если свойство **StartupForm** имеет заданное значение.

Если значение данного свойства является пустым, то Microsoft Access использует стандартные настройки запуска базы данных (открывается окно базы данных).

После указания значения данного свойства эта настройка вступает в силу со следующей загрузки базы данных.

Свойства, определяющие параметры запуска, пример

В данном примере демонстрируется процедура «SetStartupProperties», в которой передается имя свойства, значение которого требуется задать, его тип данных и задаваемое значение. В общей процедуре «ChangeProperty» делается попытка задать значение стартового свойства и, если свойство не найдено, вызывается метод **CreateProperty** для добавления свойства в семейство свойств базы данных. Эти действия являются необходимыми, поскольку данные свойства не включаются в семейство до первого определения их значений.

```
Sub SetStartupProperties()
   ChangeProperty "StartupForm", dbText, "Клиенты"
   ChangeProperty "StartupShowDBWindow", dbBoolean, False
   ChangeProperty "StartupShowStatusBar", dbBoolean, False
   ChangeProperty "AllowBuiltinToolbars", dbBoolean, False
   ChangeProperty "AllowFullMenus", dbBoolean, True
   ChangeProperty "AllowBreakIntoCode", dbBoolean, False
   ChangeProperty "AllowSpecialKeys", dbBoolean, True
   ChangeProperty "AllowBypassKey", dbBoolean, True
End Sub

Function ChangeProperty(strPropName As String, varPropType As Variant,
varPropValue As Variant) As Integer
   Dim dbs As Database, prp As Property
   Const conPropNotFoundError = 3270

   Set dbs = CurrentDb
   On Error GoTo Change_Err
   dbs.Properties(strPropName) = varPropValue
   ChangeProperty = True

Change_Bye:
   Exit Function

Change_Err:
   If Err = conPropNotFoundError Then    ' Свойство не найдено.
      Set prp = dbs.CreateProperty(strPropName, _
         varPropType, varPropValue)
      dbs.Properties.Append prp
      Resume Next
   Else
      ' Неизвестная ошибка.
      ChangeProperty = False
      Resume Change_Bye
   End If
End Function
```

Свойство StartupMenuBar

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproStartupMenuBarC; dacolProperty; damthCreateProperty"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproStartupMenuBarX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproStartupMenuBarA;daobjDatabase"}

Свойство **StartupMenuBar** позволяет указать отдельную строку меню, занимающую место общей строки меню приложения. Например, свойство **StartupMenuBar** позволяет вывести в приложении специальную строку меню, не содержащую меню **Защита**. Это лишает пользователя доступа к командам меню **Защита** через строку меню. Также возможно указание макроса создания меню, выводящего на экран нужную в данный момент строку основного меню.

Значения

Значение свойства **StartupMenuBar** задается с помощью строкового выражения, которое определяет имя отдельной строки меню или макроса создания меню, содержащегося в текущей базе данных.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства через поле **Строка меню** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым

Для установки свойства **StartupForm** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

Если задано значение свойства **StartupMenuBar**, не следует задавать значение свойства **Строка меню (MenuBar)** объекта **Application** с помощью макрокоманды **Задать значение (SetValue)** в макросе AutoExec. Since Microsoft Access запускает макрос AutoExec после проверки всех свойств, задающих параметры запуска, поэтому главное меню, задаваемое макросом AutoExec, заменит меню, указанное в свойстве **StartupMenuBar**.

Допускается также создание специального главного меню с помощью свойства **Строка меню (MenuBar)** формы и отчета. Такие специальные меню выводятся при открытии конкретной формы или отчета и заменяют общее главное меню.

Если значение свойства **StartupMenuBar** является пустым, то Microsoft Access выводит встроенное главное меню. Если значение данного свойства не пусто (по умолчанию), то значение свойства **AllowFullMenus** игнорируется (встроенное меню замещается на главную строку меню).

Указание значения данного свойства дает такой же результат, как и указание значения свойства **MenuBar** объекта **Application** (с тем лишь отличием, что настройка, задаваемая свойством **MenuBar**, вступает в силу немедленно).

При установке в окне параметров запуска флажка **Специальные клавиши Access** или при задании для свойства **AllowSpecialKeys** значения **True** (–1) нажатие клавиш CTRL+F11 позволяет пользователю переключаться между встроенным меню и основной строкой меню.

Заданное для данного свойства новое значение вступает в силу при следующем запуске приложения.

Свойство StartupShortcutMenuBar

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproStartupShortcutMenuBarC; dacolProperty; damthCreateProperty"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproStartupShortcutMenuBarX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproStartupShortcutMenuBarA;daobjDatabase"}

Свойство **StartupShortcutMenuBar** позволяет указать отдельное контекстное меню для формирования основного контекстного меню приложения и макрос создания меню, который создает контекстное меню, являющееся общим.

Значения

Значение свойства **StartupShortcutMenuBar** задается с помощью строкового выражения, которое определяет имя контекстного меню или макроса, создающего специальное контекстное меню.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства через поле **Контекстное меню** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым.

Для установки свойства **StartupShortcutMenuBar** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

Если задано значение свойства **StartupShortcutMenuBar**, не следует задавать значение свойства **Контекстное меню (ShortcutMenuBar)** объекта **Application** с помощью макрокоманды **Задать значение (SetValue)** в макросе AutoExec. Microsoft Access запускает макрос AutoExec после проверки всех свойств, задающих параметры запуска, поэтому общее контекстное меню, задаваемое макросом AutoExec, заменит контекстное меню, указанное в свойстве **StartupShortcutMenuBar**.

Допускается также создание специальных контекстных меню с помощью свойства **Контекстное меню (ShortcutMenuBar)** форм, отчетов и элементов управления. Такие контекстные меню выводятся при нажатии правой кнопки мыши в форме, отчете или на элементе управления и замещают общее меню.

Если значение данного свойства является пустым, то Microsoft Access выводит встроенные контекстные меню.

Указание значения данного свойства дает такой же результат, как и указание значения свойства **ShortcutMenuBar** объекта **Application** (с тем лишь отличием, что настройка, задаваемая свойством **ShortcutMenuBar**, вступает в силу немедленно).

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство StartupShowDBWindow

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ.
пїЅпїЅпїЅпїЅпїЅS":"acproStartupShowDBWindowC;dacolProperty;damthCreateProperty"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"acproStartupShowDBWindowX":1} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproStartupShowDBWindowA;daobjDatabase"}

Свойство **StartupShowDBWindow** определяет, выводится ли окно базы данных на экран при открытии базы данных. Например, можно при запуске приложения открыть главную форму приложения и не открывать окно базы данных.

Значения

Свойство **StartupShowDBWindow** может иметь следующие значения.

Значение	Описание
True (–1)	Выводит окно базы данных при запуске приложения.
False (0)	Не открывает окно базы данных при запуске приложения.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства при установке или снятии флажка **Окно базы данных** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым.

Для установки свойства **StartupShowDBWindow** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

Заданное для свойства **StartupShowDBWindow** значение **False** позволяет разработчику приложения скрыть окно базы данных и сделать недоступными для пользователя списки таблиц, запросов, макросов и модулей приложения.

При установке в окне параметров запуска флажка **Специальные клавиши Access** или при установке для свойства **AllowSpecialKeys** значения **True** пользователь имеет возможность нажатием клавиши F11 открыть для просмотра окно базы данных.

Даже при одновременной установке свойств **StartupShowDBWindow** и **AllowSpecialKeys** в **False** пользователь может имеет доступ к окну базы данных. Это может произойти после нескольких попыток открыть одну и ту же базу данных из списка наиболее часто используемых, которые автоматически появляются в меню **File**. Для запрещения пользователям такого доступа необходимо исключить меню **File** из вашего меню.

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство StartupShowStatusBar

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproStartupShowStatusBarC; dacolProperty; damthCreateProperty"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproStartupShowStatusBarX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproStartupShowStatusBarA;daobjDatabase"}

Свойство **StartupShowStatusBar** определяет, выводится ли строка состояния на экран при открытии базы данных. Например, свойство **StartupShowStatusBar** позволяет отключить вывод строки состояния, если в разрабатываемом приложении не требуется выводить сообщения в строке состояния.

Значения

Свойство **StartupShowStatusBar** может иметь следующие значения.

Значение	Описание
True (–1)	Строка состояния выводится при запуске.
False (0)	Строка состояния не выводится при запуске.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства при установке или снятии флажка **Строка состояния** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым.

Для установки свойства **StartupShowStatusBar** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

Установка данного свойства влияет только на строку состояния текущего приложения. В Microsoft Access предусмотрена возможность управления выводом строки состояния для всех приложений по умолчанию. Для этого выберите в меню **Сервис** команду **Параметры**, выберите вкладку **Вид** и установите или снимите флажок **Строка состояния** в группе **Отображение на экране**. Строка состояния текущего приложения не выводится на экран в следующих случаях: снят флажок **Строка состояния** в диалоговом окне **Параметры**; снят флажок **Вывод строки состояния** в диалоговом окне **Параметры запуска**; свойство **StartupShowStatusBar** имеет значение **False**.

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство AllowBuiltInToolbars

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproAllowBuiltInToolbarsC; dacolProperty; damthCreateProperty"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproAllowBuiltInToolbarsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproAllowBuiltInToolbarsA;daobjDatabase"}

Свойство **AllowBuiltInToolbars** определяет вывод встроенных панелей инструментов. Например, свойство **AllowBuiltInToolbars** позволяет разработчику запретить пользователям приложения выводить на экран встроенные панели инструментов Microsoft Access.

Значения

Свойство **AllowBuiltInToolbars** может иметь следующие значения.

Значение	Описание
True (–1)	Разрешает вывод встроенных панелей инструментов.
False (0)	Запрещает вывод встроенных панелей инструментов.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства при установке или снятии флажка **Стандартные панели инструментов** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым.

Для установки свойства **AllowBuiltInToolbars** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

Пользователь имеет возможность вывести на экран или скрыть конкретные панели инструментов с помощью диалогового окна **Настройка**, для открытия которого следует выбрать в меню **Вид** команду **Панели инструментов** и подкоманду **Настройка**.

Если для свойства **AllowBuiltInToolbars** задано значение **False**, то в подменю панелей инструментов, которое открывается командой **Панели инструментов** в меню **Вид**, не выводятся имена встроенных панелей инструментов. При выборе в этом подменю команды **Настройка** имена встроенные панели инструментов отображаются в диалоговом окне, но эти панели являются недоступными для выбора или изменения.

Пользователь имеет возможность изменять встроенные панели инструментов только при одновременном задании значения **True** для свойств **AllowToolbarChanges** и **AllowBuiltInToolbars**.

Если для свойства **AllowBuiltInToolbars** задано значение **False**, использование макрокоманды ShowToolbar для просмотра встроенных панелей инструментов невозможно.

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство AllowFullMenus

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproAllowFullMenusC; dacolProperty; damthCreateProperty"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproAllowFullMenusX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowFullMenusA;daobjDatabase"}

Свойство **AllowFullMenus** определяет, является ли доступной полная система встроенных меню Microsoft Access при запуске приложения. Например, свойство **AllowFullMenus** позволяет отключить команды меню, с помощью которых изменяют структуру таблиц, форм, запросов или отчетов.

Значения

Свойство **AllowFullMenus** может иметь следующие значения.

Значение	Описание
True (–1)	Выводит полную систему встроенных меню при запуске.
False (0)	Не выводит полную систему встроенных меню при запуске.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства при установке или снятии флажка **Полный набор меню Access** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым.

Для установки свойства **AllowFullMenus** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

При установке данного свойства в **False** в приложении отобразится список встроенных полных меню, не включающий в себя меню и команды, позволяющие пользователю изменять объекты приложения.

В этом случае также происходит отключение кнопок панели инструментов, соответствующих запрещенным командам. Однако это не относится к контекстному меню (использование его команд все еще позволяет изменять некоторые объекты приложения). Для запрещения пользователю доступа к командам контекстного меню установите свойство **AllowShortcutMenus** в **False**.

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство AllowToolbarChanges

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ.
пїЅпїЅпїЅпїЅпїЅS":"acproAllowToolbarChangesC;dacolProperty;damthCreateProperty"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acproAllowToolbarChangesX":1} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproAllowToolbarChangesA;daobjDatabase"}

Свойство **AllowToolbarChanges** определяет, имеет ли пользователь приложения возможность изменять панель инструментов, строки меню и контекстное меню. Например, свойство **AllowToolbarChanges** позволяет запретить пользователю удалять кнопки с панели инструментов или удалять панели инструментов из приложения.

Значения

Свойство **AllowToolbarChanges** может иметь следующие значения.

Значение	Описание
True (–1)	Разрешает изменять панели инструментов.
False (0)	Запрещает изменять панели инструментов.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства при установке или снятии флажка **Изменение панелей инструментов/меню** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым.

Для установки свойства **AllowToolbarChanges** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

Значение **False** свойства **AllowToolbarChanges** запрещает пользователю изменять любые панели инструментов, строковые и контекстные меню. Такое значение этого свойства приводит к отключению команды **Настройка** в подменю **Панели инструментов** из меню **Вид** и соответствующей команды контекстного меню, открывающегося при нажатии правой кнопки мыши на панели инструментов или строке меню.

При установленном значении **False** данного свойства пользователь может перемещать панель инструментов и строку меню, изменять их размеры и закреплять их.

Пользователь имеет возможность изменять панели инструментов только при одновременном задании значения **True** для свойств **AllowToolbarChanges** и **AllowBuiltInToolbars**.

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство AllowShortcutMenus

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproAllowShortcutMenusC;dacolProperty;damthCreateProperty"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowShortcutMenusX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowShortcutMenusA;daobjDatabase"}

Свойство **AllowShortcutMenus** определяет вывод в приложении контекстных меню Microsoft Access. Например, с помощью совместного использования в приложении свойств **AllowShortcutMenus** и **AllowFullMenus** можно запретить пользователю приложения вывод любых встроенных строковых меню, панелей инструментов или команд контекстного меню, позволяющих изменение объектов приложения.

Значения

Свойство **AllowShortcutMenus** может иметь следующие значения.

Значение	Описание
True (–1)	Разрешает использование встроенных контекстных меню Microsoft Access.
False (0)	Запрещает использование встроенных контекстных меню Microsoft Access.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства при установке или снятии флажка **Стандартные контекстные меню** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым.

Для установки свойства **AllowShortcutMenus** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

Данное свойство не влияет на контекстные меню и общие контекстные меню. Для вывода на экран контекстных меню форм, элементов управления и отчетов используются свойства **ShortcutMenuBar** и **StartupShortcutMenuBar**, а для просмотра общих меню –свойство **ShortcutMenuBar** объекта **Application**.

Для отображения в приложении встроенных контекстных меню с одновременным запрещением пользователю их изменения установите для свойства **AllowShortcutMenus** значение **True,** а для свойства **AllowToolbarChanges** значение **False**.

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство AllowBreakIntoCode

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproAllowBreakIntoCodeC; dacolProperty; damthCreateProperty"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproAllowBreakIntoCodeX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowBreakIntoCodeA;daobjDatabase"}

Свойство **AllowBreakIntoCode** определяет возможность вывода текста программы Visual Basic после возникновения в модуле ошибки выполнения.

Значения

Свойство **AllowBreakIntoCode** может иметь следующие значения.

Значение	Описание
True (–1)	Активизирует кнопку **Отладка** в диалоговом окне, которое выводится при возникновении ошибки выполнения.
False (0)	Отключает кнопку **Отладка**.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства при установке или снятии флажка **Просмотр программы после ошибки** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым.

Для установки свойства **AllowBreakIntoCode** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

В режиме отладки выполняемой версии приложения или макроса свойство **AllowBreakIntoCode** должно быть включено.

При включенном свойстве **AllowSpecialKeys** для приостановки выполнения программы Visual Basic допустимо использование CTRL+BREAK, даже при выключенном свойстве **AllowBreakIntoCode**.

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство AllowSpecialKeys

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproAllowSpecialKeysC;dacolProperty;damthCreateProperty"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproAllowSpecialKeysX":1} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproAllowSpecialKeysA;daobjDatabase"}

Свойство **AllowSpecialKeys** определяет включение специальных сочетаний клавиш (ALT+F1 (F11), CTRL+F11, CTRL+BREAK и CTRL+G). Например, с помощью свойства **AllowSpecialKeys** можно запретить пользователю приложения открывать окно базы данных с помощью клавиши F11, переходить в модуле Visual Basic в режим прерывания с помощью клавиш CTRL+BREAK или открывать окно отладки с помощью клавиш CTRL+G.

Значения

Свойство **AllowSpecialKeys** может иметь следующие значения.

Setting	Описание
True (–1)	Активизирует специальные клавиши.
False (0)	Отключает специальные клавиши.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства при установке или снятии флажка **Специальные клавиши Access** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым

Для установки свойства **AllowSpecialKeys** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

В режиме отладки выполняемой версии приложения свойство **AllowSpecialKeys** должно быть включено.

Данное свойство определяет режим следующих быстрых клавиш.

Клавиши	Результат
ALT+F1 (F11)	Окно базы данных переводится в верхний слой.
CTRL+G	Открывается окно отладки.
CTRL+F11	Переключение между меню пользователя и встроенным меню.
CTRL+BREAK	Вход в режим прерывания и вывод текущего модуля в окно модуля.

При выключенном свойстве **UseSpecialKeys** и выбранном при помощи свойств **StartupMenuBar** или **MenuBar** объекта **Application** общем меню, встроенное меню недоступно.

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство AllowBypassKey

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproAllowBypassKeyC; dacolProperty; damthCreateProperty"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproAllowBypassKeyX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowBypassKeyA;daobjDatabase"}

Свойство **AllowBypassKey** включает и отключает возможность запуска приложения с нажатой клавишей SHIFT. Нажатие клавиши SHIFT при запуске базы данных Microsoft Access приводит к обходу стартовых параметров запуска и отмене выполнения макроса AutoExec. Например, при заданном для свойства **AllowBypassKey** значении **False** пользователь приложения не имеет возможности обойти стартовые настройки и макрос AutoExec.

Значения

Свойство **AllowBypassKey** может иметь следующие значения.

Значение	Описание
True (–1)	Позволяет использовать режим запуска приложения с нажатой клавишей SHIFT.
False (0)	Отключает режим запуска приложения с нажатой клавишей SHIFT и запрещает обход стартовых настроек и макроса AutoExec.

Значение данного свойства задается в макросе или в программе Visual Basic.

Для установки свойства **AllowBypassKey** при помощи макроса или программы Visual Basic необходимо создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**.

Дополнительные сведения

При отладкие выполняемой версии приложения свойство **AllowBypassKey** должно быть включено.

Настройка данного свойства вступает в силу при следующем запуске приложения.

Свойство AppIcon

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproAppIconC;dacolProperty;damthCreateProperty"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproAppIconX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproAppIconA;daobjDatabase"}

Свойство **AppIcon** определяет имя рисунка (.bmp) или значка (.ico), содержащего значок приложения. Например, с помощью свойства **AppIcon** можно выбрать файл .bmp со значком автомобиля для приложения, содержащего базу данных по запасным частям для автомобилей.

Значения

Значение свойства **AppIcon** задается с помощью строкового выражения, которое определяет имя файла, содержащего рисунок значка приложения (включая путь).

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства через поле «Значок приложения» в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым.

Для установки свойства **AppIcon** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**. Для немедленного просмотра внесенных изменений также необходимо использовать метод **RefreshTitleBar**.

Дополнительные сведения

В приложении, предназначенном для распространения, рекомендуется помещать файл .bmp или .ico, содержащий значок приложения, в один каталог с этим приложением Microsoft Access.

Если значение данного свойства не задано, или если задано недопустимое значение, то в приложении используется значок Microsoft Access.

Настройка данного свойства вступает в силу сразу после задания значения свойства в программе (пока в программе присутствует метод **RefreshTitleBar**) или после закрытия окна диалога **Параметры запуска**.

Свойства AppIcon, AppTitle, пример

В следующем примере демонстрируется изменение значений свойств **AppIcon** и **AppTitle**. Если свойства еще не были созданы или не получили значения, необходимо создать эти свойства и добавить их в семейство свойств объекта **Database** с помощью метода **CreateProperty**.

```
Sub cmdAddProp_Click()
   Dim intX As Integer

   intX = AddAppProperty("AppTitle", dbText, " Собственное приложение ")
   intX = AddAppProperty("AppIcon", dbText, "C:\Windows\Cars.bmp")
   RefreshTitleBar
End Sub

Function AddAppProperty(strName As String, varType As Variant, varValue As
Variant) As Integer
   Dim dbs As Database, prp As Property
   Const conPropNotFoundError = 3270

   Set dbs = CurrentDb
   On Error GoTo AddProp_Err
   dbs.Properties(strName) = varValue
   AddAppProperty = True

AddProp_Bye:
   Exit Function

AddProp_Err:
   If Err = conPropNotFoundError Then
      Set prp = dbs.CreateProperty(strName, varType, varValue)
      dbs.Properties.Append prp
      Resume
   Else
      AddAppProperty = False
      Resume AddProp_Bye
   End If
End Function
```

Свойство AppTitle

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproAppTitleC;dacolProperty;damthCreateProperty"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproAppTitleX":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproAppTitleA;daobjDatabase"}

Свойство **AppTitle** определяет текст, выводящийся в строке заголовка приложения. Например, свойство **AppTitle** позволяет указать строку «Контроль запасов» для заголовка базы данных по учету запасов.

Значения

Значением свойства **AppTitle** является строковое выражение, определяющее текст, выводящийся в строке заголовка приложения.

Значение данного свойства задается в макросе или в программе Visual Basic. Допускается также интерактивное задание значения данного свойства через поле **Заголовок приложения** в диалоговом окне **Параметры запуска**, которое открывается командой **Параметры запуска** в меню **Сервис**. Последний способ задания значения данного свойства является самым простым

Для установки свойства **AppTitle** при помощи макроса или программы Visual Basic необходимо или установить его в окне **Параметры запуска** или создать свойство при помощи метода **CreateProperty** и добавить его к семейству **Properties** объекта **Database**. Для немедленного просмотра внесенных изменений также необходимо использовать метод **RefreshTitleBar**.

Дополнительные сведения

Если значение данного свойства не задано, в заголовке выводится строка «Microsoft Access».

Настройка данного свойства вступает в силу сразу после включения данного свойства в программу (пока в программе присутствует метод **RefreshTitleBar**) или после закрытия окна диалога **Параметры запуска**.

Свойства DatasheetBackColor и DatasheetForeColor

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDatasheetBackForeColorC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDatasheetBackForeColorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDatasheetBackForeColorA;daobjQueryDef;daobjTableDef"}

- Свойство **DatasheetBackColor** используется в программах Visual Basic для определения цвета фона таблицы, запроса или формы в режиме таблицы.
- Также с помощью свойства **DatasheetForeColor** можно определить цвет текста в таблице, запросе или форме в режиме таблицы.

Значения

Свойства **DatasheetBackColor** и **DatasheetForeColor** имеют значения типа **Длинное целое (Long Integer),** представляющие собой настройки цвета фона и шрифта для режима таблицы.

Для задания этих свойств также можно нажать кнопку **Цвет заливки/фона** ... или **Цвет текста** ... на панели инструментов **Форматирование (режим таблицы)** и выбрать нужный цвет на раскрывшейся палитре.

Задать значения по умолчанию для свойств **DatasheetBackColor** и **DatasheetForeColor** можно на вкладке **Режим таблицы** диалогового окна **Параметры**, (команда **Параметры**, меню **Сервис)**.

Дополнительные сведения

Значения свойств **DatasheetBackColor** или **DatasheetForeColor**, заданные для таблицы или запроса, не повлияют на значения этих свойств для формы, использующей указанную таблицу или запрос в качестве источника данных.

В следующем списке приведены свойства, первоначально отсутствующие в семействе **Properties** для объектов **TableDef** или **QueryDef,** но которые можно добавить с помощью метода **CreateProperty** или задать с помощью панели инструментов **Форматирование (режим таблицы)**.

DatasheetBackColor	**DatasheetFontUnderline***
DatasheetCellsEffect	**DatasheetFontWeight***
DatasheetFontHeight*	**DatasheetForeColor***
DatasheetFontItalic*	**DatasheetGridlinesBehavior**
DatasheetFontName*	**DatasheetGridlinesColor**

Примечание. При задании свойства, обозначенного в списке звездочкой, оно автоматически добавляется в семейство **Properties**.

Свойства DatasheetBackColor и DatasheetForeColor, пример

В следующем примере процедура `SetTableProperty` используется для задания темно-голубого цвета шрифта таблицы и светло-серого фона. Если после задания свойства возникает ошибка «Свойство не найдено», то для добавления свойства в семейство **Properties** для объекта используется метод **CreateProperty**.

```
Dim dbs As Database, tdfProducts As TableDef
Const lngForeColor As Long = 8388608           ' Темно-голубой.
Const lngBackColor As Long = 12632256          ' Светло-серый.

Set dbs = CurrentDb
Set tdfProducts = dbs!Товары
SetTableProperty tdfProducts, "DatasheetBackColor", dbLong, lngBackColor
SetTableProperty tdfProducts, "DatasheetForeColor", dbLong, lngForeColor

Sub SetTableProperty(tdfTableObj As TableDef, strPropertyName As String, _
      intPropertyType As Integer, varPropertyValue As Variant)
   Const conErrPropertyNotFound = 3270
   Dim prpProperty As Property
   On Error Resume Next            ' Обработка ошибок не проводится.
   tdfTableObj.Properties(strPropertyName) = varPropertyValue
   If Err <> 0 Then                ' Ошибка при задании свойства.
      If Err <> conErrPropertyNotFound Then
         ' Неизвестная ошибка.
         MsgBox "Невозможно задать свойство '" & strPropertyName _
            & "' для таблицы '" & tdfTableObj.Name & "'", vbExclamation,
Err.Description
         Err.Clear
      Else
         ' Ошибка "Свойство не найдено", добавьте свойство в семейство.
         Set prpProperty = tdfTableObj.CreateProperty(strPropertyName, _
            intPropertyType, varPropertyValue)
         tdfTableObj.Properties.Append prpProperty
         Err.Clear
      End If
   End If
   tdfTableObj.Properties.Refresh
End Sub
```

Свойство FailOnError

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFailOnErrorC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFailOnErrorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproFailOnErrorA"}

Свойство **FailOnError** используется для указания того, какой запрос, запрос на обновление или запрос на удаление, основанный на данных из источника ODBC, будет прерван при появлении ошибки. Свойство **FailOnError** позволяет оптимизировать выполнение запросов на обновление большого количества данных из источника ODBC, так, чтобы они выполнялись на сервере, если поведение с частичным выполнением запросов ядра базы данных Microsoft Jet не желательно.

Примечание. Свойство **FailOnError** применимо только к запросам на обновление и удаление.

Значения

Свойство **FailOnError** имеет следующие значения.

Значение	Описание	Visual Basic
Да	Ошибка прерывает выполнение запроса.	**True** (–1)
Нет	(По умолчанию) Ошибка не прерывает выполнение запроса.	**False** (0)

Значения для свойства **FailOnError** могут быть заданы в окне свойств запроса или программе Visual Basic. Для задания этого свойства в программе Visual Basic необходимо сначала добавить его в семейство **Properties** объекта **QueryDef** с помощью метода **CreateProperty.**

Дополнительные сведения

Если для свойства **FailOnError** задано значение **Да**, а запрос выполняется на основе одной таблицы базы данных ODBC и не содержит выражений, которые не могли бы быть определены на сервере, то запрос будет передан для выполнения на сервер. Если для свойства **FailOnError** задано значение **Да**, то при появлении ошибки в ходе выполнения запроса на обновление или удаление запрос прерывается, и таблица возвращается в свое исходное состояние. Во всех остальных случаях, свойство **FailOnError** будет воспринято как имеющее значение **Нет**. Если для этого свойства задано значение **Нет**, то при появлении ошибки можно либо принять результаты выполнения запроса, либо отменить все внесенные им изменения.

Если для свойства **FailOnError** задано значение **Нет**, и запрос выполняется на основе локальных данных, запросы на обновление и удаление выполняются также как и в предыдущих версиях Microsoft Access; при выполнении запроса, затрагивающего несколько строк, некоторые из них могут быть обновлены, а другие, вследствие возникновения ошибок, — нет. В случае появления ошибки, пользователю выдается сообщение о ее причине и номерах записей, которых она касается, после чего можно либо принять результаты выполнения запроса, либо отменить все внесенные им изменения.

Задание для данного свойства значения **Да** аналогично использованию внутренней константы **dbFailOnError** метода объектов доступа к данным **Execute.**

Свойство «Наличие модуля» (HasModule)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproHasModuleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproHasModuleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproHasModuleA"}

Свойство **Наличие модуля (HasModule)** используется для определения того, имеет ли форма или отчет модуль класса. Задание значения **Нет** для этого свойства может оптимизировать базу данных и уменьшить ее размер.

Значения

Свойство **Наличие модуля (HasModule)** имеет следующие значения.

Значение	Описание	Visual Basic
Да	В форме или отчете есть модуль класса.	**True** (–1)
Нет	(По умолчанию) Форма или отчет не имеют модуля класса.	**False** (0)

Свойство **Наличие модуля (HasModule)** может быть задано в окне свойств формы или отчета, в макросе или в программе Visual Basic.

Задать свойство **Наличие модуля (HasModule)** можно только в режиме конструктора формы или режиме конструктора отчета, а для чтения оно доступно во всех режимах.

Дополнительные сведения

Формы или отчеты, для которых свойству **Наличие модуля (HasModule)** задано значение **Нет**, рассматриваются как простые объекты. Такие объекты меньше и, обычно, быстрее загружаются и выводятся на экран, чем объекты, имеющие модуль класса. Во многих случаях, нет необходимости использовать в форме или отчете процедуры обработки событий, и наличие модуля класса не требуется.

Если в приложении для перехода к другим формам используется кнопочная форма вместо кнопок с процедурами событий, можно использовать кнопку с макросом или гиперссылкой. Например, для открытия формы «Сотрудники» с помощью кнопки в кнопочной форме следует задать для свойства элемента управления-кнопки **Дополнительный адрес** значение «Form Сотрудники».

Простые объекты, не имеющие модуля, не отображаются в окне просмотра объектов и нельзя использовать ключевое слово **New** для создания новой копии объекта. Простая форма или отчет может использоваться как подчиненная форма или подчиненный отчет и будет добавлена в семейство **Forms** или **Reports.** В простых объектах можно использовать макросы и общие (public) процедуры, существующие в стандартных модулях при вызове из окна свойств объекта.

Для свойства **Наличие модуля (HasModule)** автоматически задается значение **True** при попытке просмотра модуля объекта, даже если в него не были добавлены программы. Например, при выборе команды **Программа** в меню **Вид** для формы в режиме конструктора автоматически будет добавлен модуль класса в объект **Form,** а для свойства **Наличие модуля (HasModule)** задано значение **True**. Также модуль класса можно добавить к объекту, задав для свойства **Наличие модуля (HasModule)** значение **Да** в окне свойств объекта.

Предупреждение. Если для свойства **Наличие модуля (HasModule)** в окне свойств объекта задать значение **Нет** или в программе Visual Basic задать значение **False**, автоматически будет удален модуль класса объекта и все содержащиеся в нем программы.

При использовании метода объекта **Module** или ссылке на свойство **Module** формы или отчета в режиме конструктора автоматически создается соответствующий модуль и для свойства объекта **Наличие модуля (HasModule)** устанавливается значение **True**. При ссылке на свойство **Module** формы или отчета во время выполнения при заданном значении **False** для

свойства объекта **Наличие модуля (HasModule)** возникает ошибка.

Объекты, создаваемые с помощью функций **CreateForm** или **CreateReport,** по умолчанию являются простыми.

Свойство Hyperlink

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproHyperlinkC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproHyperlinkX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproHyperlinkA"}

Свойство **Hyperlink** используется для возвращения ссылок на объект гиперссылки. Это свойство позволяет получить доступ к свойствам и методам объекта **Hyperlink**, связанного с элементом управления.

Значения

Свойство **Hyperlink** доступно только в программах Visual Basic.

Свойство Hyperlink, пример

В следующем примере с помощью процедуры `CreateHyperlink` свойствам гиперссылки для элементов управления - кнопки, надписи или рисунка - задаются значения адреса или дополнительного адреса, передаваемые процедуре. Адрес задавать не обязательно, поскольку гиперссылка к объекту текущей базы данных использует только дополнительный адрес. Для выполнения данного примера необходимо создать форму, содержащую два поля (`txtAddress` и `txtSubAddress`) и кнопку (`cmdFollowLink`), а затем вставить следующие строки в раздел описаний модуля формы.

```
Private Sub cmdFollowLink_Click()
   CreateHyperlink Me!cmdFollowLink, Me!txtSubAddress, Me!txtAddress
End Sub

Sub CreateHyperlink(ctlSelected As Control, strSubAddress As Textbox,
Optional strAddress As Textbox)
   Dim hlk As Hyperlink
   Select Case ctlSelected.ControlType
      Case acLabel, acImage, acCommandButton
         Set hlk = ctlSelected.Hyperlink
         With hlk
            If Not IsMissing(strAddress) Then
               .Address = strAddress
            Else
               .Address = ""
            End If
            .SubAddress = strSubAddress
            .Follow
            .Address = ""
            .SubAddress = ""
         End With
      Case Else
         MsgBox "Для элемента управления '" & ctlSelected.Name & "'
гиперссылки не используются."
   End Select
End Sub
```

Свойство «Адрес гиперссылки» (HyperlinkAddress)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproHyperlinkAddressC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproHyperlinkAddressX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproHyperlinkAddressA"}

Свойство **Адрес гиперссылки (HyperlinkAddress)** позволяет определить путь к объекту, документу, странице Web или другим файлам для таких элементов управления, как кнопка, рисунок или надпись.

Значения

Свойство **Адрес гиперссылки (HyperlinkAddress)** должно содержать строковое выражение, представляющее собой путь к файлу (в формате UNC) или путь к странице Web (адрес URL).

Свойство **Адрес гиперссылки (HyperlinkAddress)** задается в окне свойств элемента управления, в макросе или программе Visual Basic.

Также для задания значения этому свойству используется диалоговое окно **Вставить гиперссылку**, открывающееся при нажатии кнопки построителя ... справа от ячейки свойства в окне свойств.

Примечание. При создании гиперссылки с помощью диалогового окна **Вставить гиперссылку** для свойства **Адрес гиперссылки (HyperlinkAddress)** автоматически задается значение, указанное в поле **Связать с файлом/URL**. Для свойства **Дополнительный адрес (HyperlinkSubAddress)** устанавливается значение, указанное в поле **Имя объекта в документе**.

При копировании гиперссылки из другого приложения и последующей вставке в форму или отчет автоматически создается элемент управления-надпись с соответствующей подписью, задается свойство **Адрес гиперссылки (HyperlinkAddress)** и **Дополнительный адрес (HyperlinkSubAddress)**.

Дополнительные сведения

При установке указателя на кнопку, рисунок или надпись, для которых задано свойство **Адрес гиперссылки (HyperlinkAddress)**, указатель превращается в указывающую вверх руку. Если щелкнуть любой из этих элементов управления, то на экране появится объект или страница Web, связь с которыми указана для данного элемента.

Для открытия объекта текущей базы данных следует оставить ячейку свойства **Адрес гиперссылки (HyperlinkAddress)** пустой, а в свойстве **Дополнительный адрес (HyperlinkSubAddress)** указать тип и имя объекта, который нужно открыть, используя следующий синтаксис: "*типобъекта имяобъекта*". Чтобы открыть объект, находящийся в другой базе данных Microsoft Access, введите путь к этой базе данных и имя файла в ячейку свойства **Адрес гиперссылки (HyperlinkAddress)**, а в свойстве **Дополнительный адрес (HyperlinkSubAddress)** укажите нужный объект.

Для свойства **Адрес гиперссылки (HyperlinkAddress)** может быть задан абсолютный или относительный путь к файлу. Абсолютный путь представляет собой полностью заданный путь к файлу (URL или UNC). Относительный путь — это путь, связанный с базовым путем, указанным в поле **База гиперссылки** в диалоговом окне свойств базы данных (команда **Свойства базы данных**, меню **Файл**), или текущий путь базы данных. Если заданное значение для свойства **Адрес гиперссылки (HyperlinkAddress)** не распознается как путь URL или UNC, то оно принимается за путь, связанный с базовым путем, указанным в поле **База гиперссылки**, или текущий путь базы данных.

Примечание. При переходе по гиперссылке к объекту другой базы данных Microsoft Access применяются и параметры запуска, обозначенные для этой базы данных. Например, если для базы данных, в которую осуществляется переход, задано открытие формы при запуске, то эта форма будет отображаться при открытии базы данных.

В следующей таблице приведены примеры значений свойств **Адрес гиперссылки (HyperlinkAddress)** и **Дополнительный адрес (HyperlinkSubAddress)**.

Адрес гиперссылки (HyperlinkAddress)	**Дополнительный адрес (HyperlinkSubAddress)**	**Описание**
http://www.microsoft.com/		Основная страница Microsoft на Web.
C:\Program Files\Microsoft Office\Office\Samples\ Cajun.htm		Страница Cajun Delights в папке примеров Access.
C:\Program Files\Microsoft Office\Office\Samples\ Cajun.htm	НовыеТовары	Именованная ссылка «НовыеТовары» на странице Cajun Delights.
C:\Личная\Письмо.doc	Ссылки	Закладка «Ссылки» в документе Microsoft Word «Письмо.doc».
C:\Финансы\Первый квартал.xls	Лист1!ИтогиПродаж	Диапазон ячеек «ИтогиПродаж» в таблице Microsoft Excel «Первый квартал.xls».
C:\Презентации\Новые планы.ppt	10	Десятый слайд в презентации Microsoft PowerPoint «Новые планы.ppt».

Свойство «Дополнительный адрес» (HyperlinkSubAddress)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproHyperlinkSubAddressC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproHyperlinkSubAddressX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproHyperlinkSubAddressA"}

Свойство **Дополнительный адрес (HyperlinkSubAddress)** используется для указания определенного места в пределах документа, указанного в свойстве **Адрес гиперссылки (HyperlinkAddress).** Свойство **Дополнительный адрес (HyperlinkSubAddress)** может представлять собой объект базы данных Microsoft Access, закладку документа Microsoft Word, именованный диапазон ячеек таблицы Microsoft Excel, слайд презентации Microsoft PowerPoint или место в пределах документа HTML.

Значения

Свойство **Дополнительный адрес (HyperlinkSubAddress)** должно содержать строковое выражение, представляющее собой определенное именованное место в пределах документа, указанного в свойстве **Адрес гиперссылки (HyperlinkAddress)**.

Свойство **Дополнительный адрес (HyperlinkSubAddress)** задается в окне свойств элемента управления, в макросе или программе Visual Basic.

Также для задания значения этому свойству используется диалоговое окно **Вставить гиперссылку**, открывающееся при нажатии кнопки построителя ... справа от ячейки свойства в окне свойств.

Примечание. При создании гиперссылки с помощью диалогового окна **Вставить гиперссылку** для свойства **Адрес гиперссылки (HyperlinkAddress)** автоматически задается значение, указанное в поле **Связать с файлом/URL**. Для свойства **Дополнительный адрес (HyperlinkSubAddress)** устанавливается значение, указанное в поле **Имя объекта в документе**.

Дополнительные сведения

При установке указателя на кнопку, рисунок или надпись, для которых задано свойство **Дополнительный адрес (HyperlinkSubAddress)**, указатель превращается в указывающую вверх руку. Если щелкнуть любой из этих элементов управления, то на экране появится объект или страница Web, связь с которыми указана для данного элемента.

Для открытия объекта текущей базы данных следует оставить ячейку свойства **Адрес гиперссылки (HyperlinkAddress)** пустой, а в свойстве **Дополнительный адрес (HyperlinkSubAddress)** указать тип и имя объекта, который нужно открыть, используя следующий синтаксис: "*типобъекта имяобъекта*". Например, чтобы создать гиперссылку для кнопки, открывающей форму «Сотрудники», нужно задать для свойства этого элемента управления **Дополнительный адрес (HyperlinkSubAddress)** значение «Form Сотрудники». Чтобы открыть объект, находящийся в другой базе данных Microsoft Access, введите путь к этой базе данных и имя файла в ячейку свойства **Адрес гиперссылки (HyperlinkAddress)**, а в свойстве **Дополнительный адрес (HyperlinkSubAddress)** укажите нужный объект.

Примечание. При переходе по гиперссылке к объекту другой базы данных Microsoft Access применяются и параметры запуска, обозначенные для этой базы данных. Например, если для базы данных, в которую осуществляется переход, задано открытие формы при запуске, то эта форма будет отображаться при открытии базы данных.

В следующей таблице приведены примеры значений свойств **Адрес гиперссылки (HyperlinkAddress)** и **Дополнительный адрес (HyperlinkSubAddress)**.

Адрес гиперссылки (HyperlinkAddress)	Дополнительный адрес (HyperlinkSubAddress)	Описание
http://www.microsoft.com/		Основная страница

		Microsoft на Web.
C:\Program Files\Microsoft Office\Office\Samples\ Cajun.htm		Страница Cajun Delights в папке примеров Access.
C:\Program Files\Microsoft Office\Office\Samples\ Cajun.htm	НовыеТовары	Именованная ссылка «НовыеТовары» на странице Cajun Delights.
C:\Личная\Письмо.doc	Ссылки	Закладка «Ссылки» в документе Microsoft Word «Письмо.doc».
C:\Финансы\Первый квартал.xls	Лист1!ИтогиПродаж	Диапазон ячеек «ИтогиПродаж» в таблице Microsoft Excel «Первый квартал.xls».
C:\Презентации\Новые планы.ppt	10	Десятый слайд в презентации Microsoft PowerPoint «Новые планы.ppt».

Свойство IsVisible

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproIsVisibleC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproIsVisibleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproIsVisibleA"}

Свойство **IsVisible** доступно только в программах Visual Basic и служит для определения того, является ли видимым элемент управления в отчете.

Значения

Свойство **IsVisible** имеет следующие значения.

Значение	Описание
True (–1)	(По умолчанию) Элемент управления является видимым.
False (0)	Элемент управления является невидимым.

Значение для свойства **IsVisible** может быть задано только в процедуре обработки события **Печать (Print)** раздела отчета, содержащего данный элемент управления.

Дополнительные сведения

Свойство **IsVisible** может использоваться вместе со свойством **Не выводить повторы (HideDuplicates)** для определения того, когда элемент управления отчета является видимым, и в зависимости от этого, скрытия и отображения других элементов. Например, можно скрыть линию в случае, когда поле скрыто по причине содержания повторов.

Свойство IsVisible, пример

В следующем примере свойство поля **IsVisible** используется для контроля отображения линии в отчете. Отчет основан на таблице «Товары» и использует три элемента управления со следующими свойствами.

Свойства	Линия	Поле #1	Поле #2
Имя (Name)	Линия0	КодТипа	НазваниеТовара
Данные (ControlSource)		КодТипа	НазваниеТовара
Не выводить повторы (HideDuplicates)		Да	Нет
От левого края (Left)	0	0	2.0
От верхнего края (Top)	0	0,1	0,1
Ширина (Width)	4,0	1,0	1,0

Вставьте следующую программу в раздел описаний модуля отчета:

```
Private Sub Detail_Print(Cancel As Integer, PrintCount As Integer)
   If Me!КодТипа.IsVisible Then
      Me!Линия0.Visible = True
   Else
      Me!Линия0.Visible = False
   End If
End Sub
```

Свойство «Максимальное число записей» (MaxRecords)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproMaxRecordsC;daproMaxRecords"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproMaxRecordsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproMaxRecordsA"}

Свойство **Максимальное число записей (MaxRecords)** используется для определения максимального числа записей, возвращаемых запросом, который выполняется на основе данных из базы данных ODBC.

Значения

Свойство **Максимальное число записей (MaxRecords)** имеет значения типа **Длинное целое (Long Integer),** представляющие собой число возвращаемых записей.

Задать это свойство можно в окне свойств запроса или в программе Visual Basic.

При задании значения этого свойства в программе Visual Basic используется свойство DAO **MaxRecords.**

Дополнительные сведения

Записи возвращаются в порядке, указанном предложением запроса ORDER BY.

Свойство **Максимальное число записей (MaxRecords)** используется в ситуациях, когда ограниченные возможности системы могут не позволять возвращать большое число записей.

Свойство MDE

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproMDEC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproMDEX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproMDEA;daobjDatabase"}

Свойство **MDE** используется для определения того, была ли сохранена база данных как файл .mde.

Значения

Когда база данных сохранена как файл .mde, к ней автоматически добавляется свойство **MDE** и для него задается значение типа **String** «Т».

Свойство **MDE** доступно только в программах Visual Basic и допускает только чтение во всех режимах.

Важно! Попытка определения свойства **MDE** для базы данных, которая не была сохранена в формате MDE, приведет к возникновению ошибки выполнения 3270 (свойство не найдено). Свойство **MDE** не существует для базы данных до тех пор, пока она не будет сохранена как файл .mde.

Дополнительные сведения

Для преобразования базы данных в формат MDE служит подкоманда **Создать MDE-файл** (команда **Служебные программы**, меню **Сервис**). При сохранении базы данных как файла .mde Microsoft Access удаляет все изменяемые исходные программы, компилирует все модули и сжимает базу данных. Программы Visual Basic продолжают выполняться, но не могут быть изменены или просмотрены. Кроме того, нельзя добавить или изменить форму или отчет.

Предупреждение. Обязательно создайте копию исходной базы данных. Если в базе данных, сохраненной в формате MDE, понадобится изменить какую-либо программу или структуру объектов, то это следует сделать в исходной базе данных, а затем создать новый файл .mde. Исходная база данных также необходима для преобразования в более поздние версии Microsoft Access, так как файлы .mde не могут быть напрямую преобразованы.

Сохранение базы данных в формате MDE целесообразно, когда необходимо предотвратить внесение изменений в формы или отчеты, а также скрыть программы Visual Basic от пользователей базы данных. Также можно создать файл .mde при создании собственных мастеров или для интерфейсной части приложения с интерфейсом.

Реплицированная база данных не может быть сохранена как файл .mde; однако после сохранения формате MDE база данных может быть реплицирована.

Свойство MDE, пример

В следующем примере функция `IsItMDE` используется для проверки свойства **MDE** текущей базы данных.

```
Dim dbs As Database
Set dbs = CurrentDb
If IsItMDE(dbs) <> True Then
   ' Обработка базы данных.
End If

Function IsItMDE(dbs as Database) As Boolean
   Dim strMDE As String
   On Error Resume Next
   strMDE = dbs.Properties("MDE")
   If Err = 0 AND strMDE = "T" Then
      ' Это база данных в формате MDE.
      IsItMDE = True
   Else
      IsItMDE = False
   End if
End Function
```

Свойство MousePointer

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproMousePointerC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproMousePointerX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproMousePointerA"}

Свойство **MousePointer** используется вместе с объектом **Screen** для определения типа указателя мыши, отображаемого в определенный момент.

Значения

Для свойства **MousePointer** задаются значения типа **Integer,** обозначающие один из следующих указателей.

Значение	Описание
0	(По умолчанию). Вид указателя определяется Microsoft Access.
1	Обычный указатель выбора (стрелка).
3	Выбор текста (курсор).
7	Изменение размера по вертикали (стрелка вверх-вниз).
9	Изменение размера по горизонтали (стрелка вправо-влево).
11	Занято (песочные часы).

Примечание. Задание для свойства **MousePointer** целого значения, отличного от перечисленных в таблице, приведет к установке значения 0.

Свойство **MousePointer** доступно только в программах Visual Basic.

Дополнительные сведения

Свойство **MousePointer** влияет на внешний вид указателя мыши для всего экрана. Если это свойство задано для некоторых специальных элементов управления, то его значение повлияет только на вид указателя, установленного на элементе управления.

Свойство **MousePointer** может использоваться для указания того, что приложение занято в данный момент; для этого задается значение 11, отображающее указатель мыши в виде песочных часов. Также можно прочитать свойство **MousePointer**, чтобы определить, что было отображено. Это может понадобиться, чтобы предотвратить нажатие кнопки другими пользователями в то время, как указатель мыши представляет собой песочные часы.

Задание для свойства **MousePointer** значения 11 аналогично присвоению аргумента **True** (–1) методу **Hourglass** объекта **DoCmd.** И наоборот, присвоение аргумента **True** методу **Hourglass** задает для свойства **MousePointer** значение 11.

Свойство «Несколько строк» (MultiRow)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproMultiRowC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproMultiRowX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproMultiRowA"}

Свойство **Несколько строк (MultiRow)** служит для определения того, может ли такой элемент управления, как набор вкладок, отображать несколько строк вкладок.

Значения

Свойство **Несколько строк (MultiRow)** имеет следующие значения.

Значение	Описание	Visual Basic
Да	Вкладки могут выстраиваться в несколько строк.	**True** (–1)
Нет	(По умолчанию) Вкладки не могут выстраиваться в несколько строк..	**False** (0)

Значение для свойства **Несколько строк (MultiRow)** задается в окне свойств элемента управления, в макросе или программе Visual Basic.

Можно также задать значение этого свойства, используемое по умолчанию, в окне стандартных свойств элемента управления или с помощью метода **DefaultControl** в программе Visual Basic.

Дополнительные сведения

Если для свойства **Несколько строк (MultiRow)** задано значение **Да (True)**, то число строк вкладок определяется их шириной и количеством. Число строк может меняться при изменении размера элемента управления или при добавлении в него вкладок.

Если для свойства **Несколько строк (MultiRow)** задано значение **Нет (False)**, и ширина вкладок превышает ширину элемента управления, то на правой стороне набора вкладок появляются кнопки перехода. Они позволяют переходить к любой вкладке набора.

Свойство «Индекс вкладки» (PageIndex)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPageIndexC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPageIndexX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPageIndexA"}

Свойство **Индекс вкладки (PageIndex)** используется для определения позиции объекта **Page** в семействе **Pages.** Данное свойство указывает порядок, в котором вкладки располагаются в элементе управления-наборе вкладок.

Значения

Значение свойства **Индекс вкладки (PageIndex)** принадлежит к типу данных **Integer** и лежит в диапазоне от 0 до значения свойства **Count** семейства **Pages**, уменьшенного на 1.

Значение для свойства **Индекс вкладки (PageIndex)** задается в окне свойств объекта **Page**, в макросе или в программе Visual Basic.

Свойство **Индекс вкладки (PageIndex)** доступно в любом режиме.

Дополнительные сведения

Изменение значения свойства **Индекс вкладки (PageIndex)** меняет положение объекта **Page** в семействе **Pages** и порядок вкладок в наборе.

Расположение объектов **Page** в семействе **Pages** можно также изменить в диалоговом окне **Порядок страниц**, которое откроется, если щелкнуть набор вкладок правой кнопкой мыши в режиме конструктора и выбрать команду **Последовательность вкладок** в контекстном меню.

Свойство ProjectName

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproProjectNameC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproProjectNameX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproProjectNameA"}

Свойство **ProjectName** используется для определения имени объекта Microsoft Access **Application.**

Значения

Свойство **ProjectName** должно содержать строковое выражение, представляющее собой имя объекта Microsoft Access **Application**. Для этого свойства нельзя задать в качестве значения имя существующего класса объектов, например, «application», «form» или «report». При преобразовании базы данных, имеющей такое же имя, что и существующий класс объектов, к имени базы данных добавляется знак подчеркивания для предотвращения конфликтов. Например, если преобразуемая база данных называется «Application», то в Microsoft Access 97 она будет иметь имя проекта «Application_».

Значение для свойства **ProjectName** может быть задано на вкладке **Другие** диалогового окна **Параметры** (команда **Параметры**, меню **Сервис**) или с помощью метода **SetOption** в программе Visual Basic. Например:

```
Application.SetOption "Имя проекта", "MyAccessProject"
```

Дополнительные сведения

При создании новой базы данных для свойства **ProjectName** автоматически задается значение ее имени без расширения .mdb. Последующее изменение имени файла базы данных не повлияет на значение свойства **ProjectName**.

При задании свойства **ProjectName** имя объекта **Application** отображается в диалоговом окне **Ссылки**, которое открывается из окна модуля с помощью команды **Ссылки** в меню **Сервис**.

Если изменить значение свойства **ProjectName**, проект будет декомпилирован. Для его перекомпиляции необходимо закрыть и снова открыть базу данных, а затем выбрать команду **Компилировать и сохранить все модули** в меню **Отладка**.

Для задания ссылки из другого приложения, например из Microsoft Excel, можно использовать имя проекта как часть полного имени элемента проекта Microsoft Access, соблюдая следующий синтаксис: *имяпроекта***.Application.Forms!***имяформы.* Например, для ссылки из Microsoft Excel на поле «Фамилия» формы «Сотрудники» учебной базы данных «Борей» необходимо задать ссылку на базу данных «Борей», а затем использовать следующую ссылку:

```
Dim strLstName As String
strLastName = Борей.Application.Forms!Сотрудники!Фамилия
```

Ссылку на базу данных Microsoft Access можно создать, выбрав имя проекта (объект **Application**) в диалоговом окне **Ссылки**, которое открывается из окна модуля с помощью команды **Ссылки** в меню **Сервис**. Как только имя проекта будет выбрано, оно появится в списке **Проект/Библиотека** в окне просмотра объектов.

Свойство «Стиль» (Style)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproStyleC;ofproStyle"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproStyleX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproStyleA"}

Свойство **Стиль (Style)** используется для определения внешнего вида вкладок в наборе вкладок.

Значения

Свойство **Стиль (Style)** может иметь следующие значения.

Значение	Описание	Visual Basic
Ярлычки	(По умолчанию) Вкладки отображаются в виде ярлычков.	**AcTabs**
Кнопки	Вкладки отображаются в виде кнопок.	**AcButtons**
Нет	Вкладки не отображаются в наборе.	**AcNone**

Значение для свойство **Стиль (Style)** задается в окне свойств набора вкладок, в макросе или в программе Visual Basic.

Можно также задать значение этого свойства, используемое по умолчанию, в окне стандартных свойств элемента управления или с помощью метода **DefaultControl** в программе Visual Basic.

Свойство **Стиль (Style)** доступно во всех режимах.

Дополнительные сведения

Если для свойства набора вкладок **Стиль (Style)** задано значение **Ярлычки** или **Кнопки**, то внешний вид вкладок определяется свойствами **TabFixedHeight**, **TabFixedWidth** и **MultiRow.**

Значение **Нет** задается для свойства **Стиль (Style)**, если нужно сохранить полный контроль за перемещением пользователя по вкладкам. В предыдущих версиях Microsoft Access диалоговые окна мастера были созданы с помощью форм с несколькими вкладками. В текущей версии можно использовать элемент управления-набор вкладок для создания собственного мастера, помещая каждую страницу мастера на отдельную вкладку набора, задав для его свойства **Стиль (Style)** значение **Нет**.

Свойства «Высота ярлычка» (TabFixedHeight) и «Ширина ярлычка» (TabFixedWidth)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproTabFixedHeightWidthC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproTabFixedHeightWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproTabFixedHeightWidthA"}

Свойства **Высота ярлычка (TabFixedHeight)** и **Ширина ярлычка (TabFixedWidth)** используются для определения высоты и ширины вкладок в наборе вкладок.

Значения

Значения свойств **Высота ярлычка (TabFixedHeight)** и **Ширина ярлычка (TabFixedWidth)** представляют собой высоту и ширину вкладок, выраженную в единицах измерения, указанных в диалоговом окне **Языки и стандарты** на панели управления Windows. Если для этих свойств задать значение 0, то размер вкладок автоматически выравнивается по их содержимому.

Задать эти свойства можно окне свойств набора вкладок, в макросе или в программе Visual Basic.

Можно также задать значение этого свойства, используемое по умолчанию, в окне стандартных свойств элемента управления или с помощью метода **DefaultControl** в программе Visual Basic.

В Visual Basic эти свойства имеют значения типа **Long Integer,** представляющие собой высоту или ширину вкладок в твипах и доступны во всех режимах.

Примечание. Для использования единиц измерения, отличных от установленных в диалоговом окне **Язык и стандарты** на панели управления Windows, следует указать нужные единицы, сантиметры или дюймы (например, 5 см или 3").

Дополнительные сведения

Нельзя изменить цвет набора вкладок. Если вкладки не занимают всю ширину набора, то оставшаяся часть, расположенная за ними, отображается на экране. Поэтому, при помещении набора вкладок в объект, имеющий отличный от него цвет, нужно быть уверенным, что вкладки закрывают фоновую часть набора.

Свойство «Панель инструментов» (Toolbar)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproToolBarC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproToolBarX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproToolBarA"}

Свойство **Панель инструментов (Toolbar)** используется для указания панели инструментов, которая должна использоваться для формы или отчета. Эти панели можно создать в диалоговом окне **Настройка** (подкоманда **Настройка**, команда **Панели инструментов**, меню **Вид**). Для получения дополнительных сведений о создании специальных панелей инструментов см. раздел Создание специальной панели инструментов для текущей базы данных.

Значения

Значение свойства **Панель инструментов (Toolbar)** представляет собой название специальной панели инструментов, которую нужно отобразить. Если оставить ячейку свойства **Панель инструментов (Toolbar)** пустой, то для формы или отчета будут отображаться встроенные (стандартные) панели инструментов.

Задать значение для свойства **Панель инструментов (Toolbar)** можно в окне свойств формы или отчета, в макросе или в программе Visual Basic.

В Visual Basic значение для этого свойства должно содержать строковое выражение, представляющее собой название панели инструментов, которую необходимо отобразить.

Чтобы отобразить встроенную панель инструментов для формы или отчета с помощью макроса или программы Visual Basic, следует задать для свойства **Панель инструментов (Toolbar)** пустую строку ("").

Дополнительные сведения

После задания значения свойству **Панель инструментов (Toolbar)** указанная специальная панель инструментов выводится на экран при открытии формы или отчета. Эта панель отображается независимо от того, имеют ли форма или отчет фокус.

Свойство «Использовать транзакцию» (UseTransaction)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproUseTransactionC;damthBeginTrans;damthCreateProperty"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproUseTransactionX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproUseTransactionA"}

Свойство **Использовать транзакцию (UseTransaction)** используется для определения того, выполняется ли запрос на изменение как отдельная транзакция. В некоторых случаях задание значения **Нет** для этого свойства может значительно повысить быстродействие запросов.

Значения

Свойство **Использовать транзакцию (UseTransaction)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(По умолчанию) Запрос выполняется как отдельная транзакция.	**True** (–1)
Нет	Запрос не выполняется как отдельная транзакция.	**False** (0)

Свойство **Использовать транзакцию (UseTransaction)** может быть задано в окне свойств запроса или в программе Visual Basic. Для задания этого свойства в Visual Basic необходимо сначала добавить его в семейство **Properties** объекта **QueryDef** с помощью метода **CreateProperty.**

Дополнительные сведения

Если для запроса на создание таблицы, запроса на удаление, на добавление или обновление свойству **Использовать транзакцию (UseTransaction)** задано значение **Да**, то ядро базы данных Microsoft Jet хранит возвращаемые записи во временном буфере и, при необходимости, сохраняет результаты во временной базе данных на диске. Когда запрос выполнен, ядро базы данных Jet считывает записи из временной базы данных и записывает их обратно в исходную базу.

Задав для свойства **Использовать транзакцию (UseTransaction)** значение **Нет**, можно значительно повысить быстродействие при определенных обстоятельствах. В этом случае пользователь никогда не столкнется с проблемами, связанными со слишком большим количеством запросов на блокировку записей. Кроме того, ядро базы данных Jet не сохраняет результаты запроса во временной базе данных, что тоже повышает быстродействие.

Если существует большое количество записей, которые должны быть записаны во временную базу данных, то выполнение запроса на изменение как отдельной транзакции может снизить быстродействие. Кроме того, при выполнении запроса на удаление или обновление в общей базе данных, возникает большое количество запросов на блокировку записей. Это может снизить быстродействие в некоторых сетевых операционных системах, например, в NetWare выполнение запроса будет прервано, если количество запросов на блокировку записей превысит 10000.

Если завершение транзакции невозможно, появляется сообщение об ошибке и пользователь может сохранить или нет внесенные изменения.

Если для свойства **Использовать транзакцию (UseTransaction)** задано значение **Нет**, единственным способом отменить всю транзакцию является использование метода DAO **RollBack** в программе Visual Basic.

Свойство Application

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproApplicationC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproApplicationX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproApplicationA;dacolProperty"}

Свойство **Application** используется в программах Visual Basic для доступа к активному объекту **Application** Microsoft Access и его свойствам.

Значения

Значение свойства **Application** задается в Microsoft Access и во всех режимах является доступным только для чтения.

Дополнительные сведения

Каждый объект Microsoft Access имеет свойство **Application**, которое возвращает текущий объект **Application**. Данное свойство позволяет получить доступ к любому из свойств объекта **Application**. Например, для ссылки на строку меню объекта **Application** из текущей формы необходимо использовать следующий синтаксис:

```
Me.Application.MenuBar
```

Свойства Application и DBEngine, пример

Данный пример выводит в окно сообщений свойства объекта **DBEngine**.

```
Private Sub Кнопка1_Click()
   DisplayApplicationInfo Me
End Sub

Function DisplayApplicationInfo(obj As Object) As Integer
   Dim objApp As Object, intI As Integer, strProps As String
   On Error Resume Next
      ' Свойство Application формы.
      Set objApp = obj.Application
      MsgBox "Свойство Вывод на экран (Visible) приложения = " &
objApp.Visible
      If objApp.UserControl = True Then
      For intI = 0 To objApp.DBEngine.Properties.Count - 1
         strProps = strProps & objApp.DBEngine.Properties(intI).Name & ", "
      Next intI
      End If
      MsgBox Left(strProps, Len(strProps) - 2) & ".", vbOK, "Свойства
DBEngine"
End Function
```

Свойство DBEngine

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDBEngineC;daobjDBEngine"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDBEngineX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproDBEngineA"}

Свойство **DBEngine** используется в программах Visual Basic для доступа к текущему объекту **DBEngine** и его свойствам.

Значения

Значение свойства **DBEngine** задается в Microsoft Access и во всех режимах является доступным только для чтения.

Дополнительные сведения

Свойство **DBEngine** объекта **Application** представляет ядро базы данных Microsoft Jet. Объект **DBEngine** является объектом верхнего уровня в модели объектов доступа к данным (DAO), который содержит все остальные объекты, входящие в иерархию объектов доступа к данным и осуществляет управление ими.

Свойство UserControl

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproUserControlC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproUserControlX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproUserControlA"}

Свойство **UserControl** используется для проверки, было ли текущее приложение Microsoft Access запущено пользователем или вызвано из другого приложения с помощью программирования объектов, ранее называемого программированием объектов OLE.

Значения

Свойство **UserControl** может иметь следующие значения.

Значение	Описание
True (–1)	Текущее приложение запущено пользователем.
False (0)	Текущее приложение запущено из другого приложения с помощью механизма управления объектами OLE.

Значение свойства **UserControl** определяется только в программах Visual Basic.

Данное свойство во всех режимах доступно только для чтения.

Дополнительные сведения

При запуске приложения пользователем свойства **Visible** и **UserControl** объекта **Application** получают значение **True**. Если свойство **UserControl** имеет значение **True**, задать для свойства **Visible** значение **False** невозможно.

При создании объекта **Application** с помощью программирования объектов его свойства **Visible** и **UserControl** получают значение **False**.

Свойство Visible (Объект Application)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅS":"acproVisibleAppObjC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acproVisibleAppObjX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproVisibleAppObjA"}

Свойство **Visible** объекта **Application** позволяет указать, или проверить, является ли приложение Microsoft Access свернутым.

Значения

Свойство **Visible** может иметь следующие значения.

Значение	Описание
True (–1)	Приложение является видимым.
False (0)	Приложение не является видимым.

Значение свойства **Visible** объекта **Application** можно задать только в программе Visual Basic. Для задания для данного свойства значения по умолчанию можно использовать окно стандартных свойств элемента управления или метод Visual Basic **DefaultControl.**

Дополнительные сведения

При запуске приложения пользователем свойства **Visible** и **UserControl** объекта **Application** получают значение **True**. Если свойство **UserControl** имеет значение **True**, задать для свойства **Visible** значение **False** невозможно.

При создании объекта **Application** с помощью программирования объектов его свойства **Visible** и **UserControl** получают значение **False**.

Примечание. Когда свойство **Visible** объекта **Application** получает значение **False,** окно приложения свертывается, но приложение при этом не является скрытым.

Свойство CodeContextObject

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅS":"acproCodeContextObjectC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acproCodeContextObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproCodeContextObjectA"}

Свойство **CodeContextObject** позволяет определить объект, в котором выполняется макрос или программа Visual Basic.

Значения

Значение свойства **CodeContextObject** задается в Microsoft Access и во всех режимах является доступной только для чтения.

Дополнительные сведения

Свойства **ActiveForm** и **ActiveReport** объекта **Screen** всегда возвращают объект, имеющий фокус. Объект, имеющий фокус, может быть, а может и не быть тем объектом, в котором выполняется текущий макрос или программа Visual Basic. Такая ситуация возникает, например, при выполнении программы Visual Basic в процедуре обработки события **Таймер (Timer)** в скрытой форме.

Свойство CodeContextObject, пример

В данном примере свойство **CodeContextObject** используется в функции для определения имени объекта, в котором возникла ошибка. После этого имя объекта выводится в заголовке окна сообщения и в тексте сообщения об ошибке. Для создания сообщения об ошибке используется инструкция **Error** в процедуре обработки нажатия кнопки.

```
Private Sub Кнопка1_Click()
   On Error GoTo Кнопка1_Err
   Error 11                         ' Создает ошибку деления на нуль.
   Exit Sub

   Кнопка1_Err:
      If ErrorMessage("Событие Кнопка1_Click().", vbYesNo + _
            vbInformation, Err) = vbYes Then
         Exit Sub
      Else
         Resume
      End If
End Sub

Function ErrorMessage(strText As String, intType As Integer, _
      intErrVal As Integer) As Integer
   Dim objCurrent As Object
   Dim strMsgboxTitle As String
   Set objCurrent = CodeContextObject
   strMsgboxTitle = "Ошибка в " & objCurrent.Name
   strText = strText & "Ошибка #" & intErrVal _
      & " в объекте " & objCurrent.Name
   ErrorMessage = MsgBox(strText, intType, strMsgboxTitle)
   Err = 0
End Function
```

Свойство «Тип вывода» (DisplayType)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDisplayTypeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDisplayTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDisplayTypeA"}

Свойство **Тип вывода (DisplayType)** определяет отображение в Microsoft Access содержимого объекта OLE или значка объекта. Например, если объект OLE является документом Microsoft Word, и для данного свойства выбрано значение «Содержимое», то в элементе управления будет выводиться документ Word; если для свойства задано значение «Значок», то в элементе управления выводится значок Word.

Значения

Свойство **Тип вывода (DisplayType)** может иметь следующие значения.

Значение	Описание	Visual Basic
Содержимое	(Значение по умолчанию). Если элемент управления содержит объект OLE, то в элементе управления выводится содержимое объекта, например, документ или электронная таблица.	**acOLEDisplayContent** (0)
Значок	Если элемент управления содержит объект OLE, то в элементе управления выводится значок объекта.	**acOLEDisplayIcon** (1)

Значение свойства **Тип вывода (DisplayType)** задается в окне свойств, в макросе или в программе Visual Basic. Допускается также задание значения этого свойства в окне стандартных свойств элемента управления. или с помощью метода **DefaultControl** из Visual Basic.

Для присоединенной рамки объекта значение свойства **Тип вывода (DisplayType)** можно задать только в режиме конструктора; чтение этого значения допускается во всех режимах. Для свободной рамки объекта или диаграммы значение данного свойства задается в окне диалога **Вставка объекта** при создании объекта (значение по умолчанию - «Содержимое», при установке флажка **В виде значка** устанавливается значение «Значок»).

Дополнительные сведения

Значение свойства **Тип вывода (DisplayType)** определяет заданное по умолчанию состояние флажка **В виде значка** в окнах диалога **Специальная вставка**, доступного по команде **Специальная вставка** из меню **Правка,** и **Вставка объекта**, выводимом при вставке свободной рамки объекта. При выводе этих окон диалога в режиме формы, режиме таблицы или режиме конструктора флажок **В виде значка** устанавливается автоматически, если для свойства **Тип вывода (DisplayType)** выбрано значение «Значок». Например, этот флажок будет установлен, если для свойства элемента управления **Action** в программе Visual Basic заданы значения **acOLEInsertObjDlg** или **acOLEPasteSpecialDlg**.

Свойство **Тип вывода (DisplayType)** не влияет на состояние флажка **В виде значка** в окне диалога **Вставка объекта** при вставке объекта в свободную рамку объекта. При вставке объекта из буфера состояние флажка **В виде значка** определяется состоянием объекта, скопированного в буфер.

Изменение значения свойства **Тип вывода (DisplayType)** для присоединенной рамки объекта не влияет на отображение в данном элементе управления существующих объектов; однако это свойство будет влиять на отображение новых объектов, помещаемых в этот элемент

управления с помощью команды **Объект** из меню **Вставка**.

Свойство «Установка размеров» (SizeMode)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproSizeModeC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproSizeModeX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproSizeModeA"}

Свойство **Установка размеров (SizeMode)** задает способ установки размеров рисунка или другого объекта в присоединенной рамке объекта, в свободной рамке объекта или в рамке рисунка.

Значения

Свойство **Установка размеров (SizeMode)** может иметь следующие значения.

Значение	Описание	Visual Basic
Фрагмент	(Значение по умолчанию). Объект выводится в натуральную величину. Если размеры объекта превышают размеры элемента управления, то изображение объекта обрезается по краям элемента управления.	**acOLESizeClip** (0)
Вписать в рамку	Задает изменение размеров объекта по размеру рамки. Эта настройка может привести к искажению пропорций объект.	**acOLESizeStretch** (1)
По размеру рамки	Задает полное отображение объекта в существующих размерах рамки без искажения пропорций объекта. При изменении размера рамки данная настройка может привести к появлению в рамке пустых мест.	**acOLESizeZoom** (3)

Значение свойства **Установка размеров (SizeMode)** задается в окне свойств, в макросе или в программе Visual Basic. Допускается также задание значения этого свойства в окне стандартных свойств элемента управления. или с помощью метода **DefaultControl** из Visual Basic.

Совет. Объект выводится быстрее, если задать для данного свойства значение «Фрагмент». Значение «Вписать в рамку» для гистограмм и графиков не приводит к искажению пропорций. Однако это значение может привести к искажению окружностей и фотографий.

Свойство «Документ-источник» (SourceDoc)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproSourceDocC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproSourceDocX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproSourceDocA"}

Свойство **Документ-источник (SourceDoc)** указывает имя файла, используемого для создания связанного или внедренного объекта OLE при создании объекта OLE пользователем или с помощью свойства **Action** в программе Visual Basic.

Значения

При создании внедренного объекта введите имя файла, используемого в качестве источника, и задайте для свойства **Action** значение **acOLECreateEmbed**.

При создании связанного объекта введите путь и имя связываемого файла и задайте для свойства **Action** значение **acOLECreateLink**.

Значение данного свойства задается в окне свойств, в макросе или в программе Visual Basic.

Примечание. Хотя значения данного свойства можно задать в окне свойств, они действуют в программе Visual Basic только при одновременном использовании свойства **Action**.

Дополнительные сведения

Свойство элемента управления **Документ-источник (SourceDoc)** позволяет указать связываемый файл, а свойство **Источник данных** (**SourceItem**) указать связываемый фрагмент этого файла. Если требуется связать весь объект, оставьте значение свойства **Источник данных (SourceItem)** пустым.

При связывании объекта в свободную рамку объекта значение свойства **Источник данных (SourceItem)** сливается со значением свойства **Документ-источник (SourceDoc)**. В режиме формы, режиме таблицы и режиме предварительного просмотра значением свойства **Источник данных** будет являться пустая строка (""). При этом в свойстве **Документ-источник** будет указан полный путь к связанному файлу, за которым следует восклицательный знак (!) или обратная косая (\) и значение свойства **Источник данных**, как показано в следующем примере.

```
"C:\Work\Qtr1\Revenue.xls!R1C1:R30C15"
```

Свойства «Документ-источник» (SourceDoc), «Источник данных» (SourceItem), Action, «Класс» (Class), «Допустимый тип OLE» (OLETypeAllowed), «Установка размеров» (SizeMode), пример

Следующая процедура, определенная для кнопки, создает при нажатии кнопки связанный объект OLE и задает изменяемые размеры объекта, позволяющие полностью отобразить его содержимое в элементе управления.

```
Sub Кнопка1_Click
   OLE1.Class = "Excel.Sheet"                    ' Задает имя класса.
   OLE1.OLETypeAllowed = acOLELinked              ' Задает тип объекта.
   OLE1.SourceDoc = "C:\Excel\Oletext.xls"       ' Задает файл-источник.
   OLE1.SourceItem = "R1C1:R5C5"                 ' Задает связываемый
фрагмент.
   OLE1.Action = acOLECreateLink                 ' Создает связанный объект.
   OLE1.SizeMode = acOLESizeZoom                 ' Задает изменяемые размеры
элемента управления.
End Sub
```

Свойство «Параметры обновления» (UpdateOptions)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproUpdateOptionsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproUpdateOptionsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproUpdateOptionsA"}

Свойство **Параметры обновления (UpdateOptions)** определяет режим обновления связанного объекта OLE.

Значения

Свойство **Параметры обновления (UpdateOptions)** может иметь следующие значения.

Значение	Описание	Visual Basic
Автоматическое	(Значение по умолчанию). Задает обновление объекта при каждом изменении связанных данных.	**acOLEUpdateAutomatic** (0)
По запросу	Задает обновление объекта только при указании значения **acOLEUpdate** для свойства **Action** элемента управления или при обновлении связи с помощью команды **Связи OLE/DDE** из меню **Правка**.	**acOLEUpdateManual** (2)

Значение свойства **Параметры обновления (UpdateOptions)** задается в окне свойств, в макросе или в программе Visual Basic. Допускается также задание значения этого свойства в окне стандартных свойств элемента управления. или с помощью метода **DefaultControl** из Visual Basic.

Значения свойства **Параметры обновления (UpdateOptions)** могут быть также изменены с помощью команды **Связи OLE/DDE** из меню **Правка**.

Дополнительные сведения

По умолчанию объект обновляется автоматически при любом изменении связанных данных, однако, имеется возможность указать в Microsoft Access обновление объекта только при выполнении специальной команды пользователя. Например, если возможен доступ к данным электронной таблицы, связанным в форме, или изменение этих данных другими пользователями или приложениями, то разработчик формы с помощью этого свойства может указать, что связанные данные должны изменяться только при открытии базы данных для монопольного доступа одного пользователя.

Если для свойства **Параметры обновления (UpdateOptions)** задано значение «По запросу», то не производится периодическое обновление объекта в соответствие со значением параметра **Период обновления (с)**, задаваемым на вкладке **Другие** в окне диалога **Параметры**, которое открывается командой **Параметры** в меню **Сервис**.

Примечание. При изменении данных в объекте возникает событие **При обновлении (Updated).**

В данном примере с помощью свойства **Параметры обновления (UpdateOptions)** для свободной рамки объекта с именем «OLE1» задается режим обновления по запросу, а затем объект в элементе управления обновляется с помощью свойства **Action**.

```
OLE1.UpdateOptions = acOLEUpdateManual
OLE1.Action = acOLEUpdate
```

Свойство Action

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproActionC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproActionX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproActionA"}

Свойство **Action** используется в программе Visual Basic для указания операции, выполняемой над объектом OLE.

Значения

Свойство **Action** может иметь следующие значения.

Константа	Описание
acOLECreateEmbed	Создает внедренный объект. Для применения данного значения следует предварительно задать для свойства элемента управления **OLETypeAllowed** значение **acOLEEmbedded** или **acOLEEither**. В свойстве **Класс (Class)** следует указать тип создаваемого объекта OLE. Свойство **Документ-источник (SourceDoc)** позволяет использовать существующий файл в качестве образца объекта.
AcOLECreateLink	Создает связанный объект OLE, в котором выводится содержимое файла. Для применения данного значения необходимо предварительно определить значения свойств **Допустимый тип OLE (OLETypeAllowed)** и **Документ-источник (SourceDoc)**.Для свойства **Допустимый тип OLE (OLETypeAllowed)** следует указать значение **acOLELinked** или **acOLEEither**. Свойство **Документ-источник (SourceDoc)** позволяет указать файл, используемый для создания объекта OLE. Кроме того, в свойстве **Источник данных (SourceItem)** можно определить связываемый фрагмент файла (например, для указания диапазона ячеек, если создаваемый объект является электронной таблицей Microsoft Excel). При создании объекта OLE с помощью данного значения свойства в элементе управления выводится графическое изображение (в формате метафайла файла, указанного в свойстве **Документ-источник (SourceDoc)** элемента управления. При сохранении объекта OLE в действительности сохраняется только описание связи (имя приложения, в котором был создан объект, и имя связанного файла), поскольку элемент управления содержит изображение данных, но не содержит самих исходных данных.
acOLECopy	Копирует объект в буфер обмена. При копировании объекта OLE в буфер обмена помещаются все данные и сведения о связи, относящиеся к данному объекту. Допускается копирование в буфер обмена как связанных, так и внедренных объектов. Использование данного значения эквивалентно выполнению команды **Копировать** в меню **Правка**.
acOLEPaste	Копирует данные из буфера обмена в элемент управления. Если операция вставки прошла

	успешно, то свойство элемента управления **Тип OLE (OLEType)** получает значение **acOLELinked** или **acOLEEmbedded**. Если операцию вставки выполнить не удалось, то свойство **Тип OLE (OLEType)** будет иметь значение **acOLENone**. Использование значения **acOLEPaste** эквивалентно выполнению команды **Вставить** из меню **Правка**.
acOLEUpdate	Загружает текущие данные объекта OLE из приложения, в котором был создан объект, и выводит графическое изображение этих данных в элементе управления.
acOLEActivate	Открывает объект OLE для выполнения действий, например, для редактирования. Для применения данного значения необходимо предварительно задать значение свойства **Команда (Verb).** Свойство **Команда (Verb)** определяет операцию, которая будет выполнена при запуске объекта OLE.
acOLEClose	Закрывает объект OLE и заканчивает сеанс подключения к приложению, в котором был создан объект. Данное значение относится только к внедренным объектам, а его использование эквивалентно выбору команды **Закрыть** в оконном меню объекта.
acOLEDelete	Удаляет указанный объект OLE и освобождает занимаемую объектом область памяти. Данное значение позволяет явно удалить объект OLE. Объекты OLE удаляются автоматически при закрытии формы или при замене объекта на новый объект. Не допускается использование свойства **Action** для удаления присоединенного объекта OLE из базовой таблицы или запроса.
acOLEInsertObjDlg	Открывает диалоговое окно **Вставка объекта**. В режиме формы или в режиме таблицы разработчик выводит данное окно диалога, чтобы позволить пользователю создать новый объект или связать или внедрить существующий. Свойство элемента управления **Допустимый тип OLE (OLETypeAllowed)** позволяет при этом определить (с помощью констант **acOLELinked**, **acOLEEmbedded** или **acOLEEither**) тип объекта, который пользователю разрешено вставить в этом окне диалога.
acOLEPasteSpecialDlg	Открывает окно диалога **Специальная вставка**. В режимах формы или таблицы разработчик выводит данное окна диалога, чтобы позволить пользователю вставить объект из буфера обмена. Параметры окна диалога предоставляют выбор нескольких возможностей, в том числе связывание или внедрение объекта. Свойство элемента управления **Допустимый тип OLE (OLETypeAllowed)** позволяет при этом определить (с помощью констант **acOLELinked**, **acOLEEmbedded** или **acOLEEither**) тип объекта, который пользователю разрешено вставить в этом окне диалога.

acOLEFetchVerbs	Обновляет список команд, поддерживаемых объектом OLE. Вывести список команд позволяют свойства **ObjectVerbs** и **ObjectVerbsCount**.

Значение свойства **Action** задается только в программе Visual Basic. Значение свойства **Action** принадлежит к типу данных **Integer.**

Свойство **Action** недоступно в режиме конструктора, но может быть считано или задано в других режимах.

Дополнительные сведения

Если в свойстве **Доступ (Enabled)** установлено значение **Нет** или в свойстве **Блокировка (Locked)** установлено значение **Да**, то применение некоторых значений свойства **Action** становится невозможным. В следующей таблице показано, какие значения допускается или не допускается использовать в этих случаях.

Значения	**Доступ (Enabled) = Нет**	**Блокировка (Locked) = Да**
acOLECreateEmbed	Не допускается	Не допускается
acOLECreateLink	Не допускается	Не допускается
acOLECopy	Допускается	Допускается
acOLEPaste	Не допускается	Не допускается
acOLEUpdate	Не допускается	Не допускается
acOLEActivate	Допускается	Допускается
acOLEClose	Не допускается	Допускается
acOLEDelete	Не допускается	Не допускается
acOLEInsertObjDlg	Не допускается	Не допускается
acOLEPasteSpecialDlg	Не допускается	Не допускается
acOLEFetchVerbs	Не допускается	Допускается

Свойство «Автоматический запуск» (AutoActivate)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproAutoActivateC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproAutoActivateX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproAutoActivateA"}

Свойство **Автоматический запуск (AutoActivate)** определяет способ запуска объекта OLE.

Значения

Свойство **Автоматический запуск (AutoActivate)** может иметь следующие значения.

Значения	Описание	Константа
По запросу	Объект OLE, не будет запускаться при получении фокуса элементом управления и при двойном нажатии кнопки мыши. Запуск данного объекта OLE можно провести только из программы Visual Basic, указав значение **acOLEActivate** в свойстве **Action**.	**AcOLEActivateManual**
При получении фокуса	(Только для элементов управления типа свободной рамки объекта и диаграммы). Если элемент управления содержит объект OLE, то приложение, в котором создан объект, будет запущено при получении фокуса элементом управления.	**AcOLEActivateGetFocus**
Двойным нажатием кнопки	(Значение по умолчанию). Если элемент управления содержит объект OLE, то приложение, в котором создан объект, будет запущено, когда пользователь установит указатель на объект и дважды нажмет кнопку мыши, или если пользователь нажмет клавиши CTRL+ENTER в тот момент, когда элемент управления имеет фокус.	**acOLEActivateDoubleClick**

Значение данного свойства задается в окне свойств, в макросе или в программе Visual Basic.

Значение свойства **Автоматический запуск (AutoActivate)** можно задать только в режиме конструктора.

Дополнительные сведения

Некоторые объекты OLE запускаются непосредственно в элементе управления. При запуске подобного объекта его можно редактировать (или выполнить какое-либо другое действие) внутри элемента управления. Такая характеристика объектов носит название запуск по месту. Если объект поддерживает механизм запуска по месту, то сведения об этом приводятся в документации приложения, в котором создается объект.

В программе Visual Basic для проверки, содержится ли в элементе управления объект OLE, следует определить значение свойства элемента управления **OLEType.**

Примечание. Если свойство **Автоматический запуск (AutoActivate)** имеет значение **Двойным нажатием кнопки**, и для элемента управления задан макрос или процедура обработки события **Двойное нажатие кнопки (DblClick)**, то это событие возникает (и соответствующая процедура или макрос выполняются) до запуска объекта.

Свойство «Класс» (Class)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproClassC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproClassX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproClassA"}

Свойство **Класс (Class)** позволяет указать или определить имя класса внедренного объекта OLE.

Значения

Значением свойства **Класс (Class)** является строковое выражение, которое задается пользователем или автоматически подставляется Microsoft Access при создании или вставке объекта OLE.

Пользователь задает значение свойства **Класс (Class)** в окне свойств элемента управления, в макросе или в программе Visual Basic.

Дополнительные сведения

Имя класса определяет тип объекта OLE. Например, Microsoft Excel версии 5.0 поддерживает несколько типов объектов OLE, в том числе электронные таблицы и диаграммы. Именами классов для данных объектов являются, соответственно, «Excel.Sheet» и «Excel.Chart». Если пользователь создает объект OLE в режиме конструктора с помощью команды **Специальная вставка** из меню **Правка** или команды **Объект** из меню **Вставка**, то имя класса нового объекта автоматически подставляется в окно свойств.

Примечание. Для определения имени класса конкретного объекта OLE следует обратиться к документации приложения, в котором создается данный объект OLE.

Значение свойства **Класс (Class)** обновляется при копировании объекта из буфера обмена. Например, при вставке диаграммы Microsoft Excel из буфера обмена в объект OLE, который до этого содержал электронную таблицу Microsoft Excel, значение свойства **Класс (Class)** изменится с «Excel.Sheet» на «Excel.Chart». Для вставки объекта из буфера обмена в программе Visual Basic следует задать для свойства **Action** элемента управления значение **acOLEPaste** или **acOLEPasteSpecialDlg**.

Примечание. Свойства **Класс OLE (OLEClass)** и **Класс (Class)** похожи, но не тождественны. В значении свойства **Класс OLE (OLEClass)** содержится общее описание объекта OLE, тогда как значением свойства **Класс (Class)** является имя, которое следует использовать для ссылки на этот объект OLE в программе Visual Basic. Примерами значений свойства **Класс OLE (OLEClass)** могут служить «Диаграмма Microsoft Excel», «Документ Microsoft Word» и «Точечный рисунок Paintbrush».

Свойство «Элемент» (Item)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproItemC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproItemX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproItemA"}

Свойство **Элемент (Item)** возвращает описание связанных данных, отображенных в таких элементах управления, как диаграмма или свободная рамка объекта.

Значения

Значением свойства **Элемент (Item)** является строковое выражение, доступное только для чтения во всех режимах.

Примечание. Если при программировании объектов (формально называется программирование объектов OLE) необходимо указать диапазон данных, используйте свойство **Источник данных (SourceItem).**

Дополнительные сведения

Описание рамки объекта, с которой связан внешний файл, зависит от типа объекта OLE, с которым она связана. Например, если связь установлена с диапазоном ячеек электронной таблицы Microsoft Excel, то в Microsoft Access в рамке объекта отображается содержимое ячеек, а свойство **Элемент (Item)** возвращает координаты диапазона электронной таблицы. Для диаграммы значением свойства **Элемент (Item)** является слово «Chart». Если в объекте выводится фрагмент рисунка, то свойство **Элемент (Item)** возвращает координаты связанной области рисунка.

Примечание. Данные связанного объекта OLE отображаются в рамке, однако, хранятся в исходном приложении; для их изменения необходимо использовать это приложение.

При связывании объекта OLE в свободную рамку объекта с помощью команды **Специальная вставка** из меню **Правка** значения свойств **Объект-источник (SourceObject), Класс OLE (OLEClass)** и **Элемент (Item)** задаются автоматически. Ниже приводятся типичные значения этих свойств для различных объектов OLE.

Объект	Свойство объект-источник (SourceObject)	Свойство Класс OLE (OLEClass)	Свойство Элемент (Item)
Электронная таблица	C:\Excel\Xls\Mywrksht.xls	Лист Microsoft Excel	R1C1:R7C4
Диаграмма	C:\Excel\Xls\Mychart.xlc	Диаграмма Microsoft Excel	Chart
Рисунок	C:\Windows\Mybitmap.bmp	Точечный рисунок Paintbrush	0 0 72 71

Свойство lpOLEObject

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acprolpOLEObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acprolpOLEObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acprolpOLEObjectA"}

Значение свойства **lpOLEObject** используется в программе Visual Basic для определения адреса объекта OLE. Свойство **lpOLEObject** возвращает указатель IOLEObject.

Значения

Значение свойства **lpOLEObject** используется для ссылки на адрес активного объекта OLE. Если ни один объект OLE не выведен, данное свойство имеет значение 0. Значение свойства **lpOLEObject** принадлежат к типу данных **Long,** определенному в Microsoft Access.

Данное свойство недоступно в режиме конструктора и доступно только для чтения в других режимах.

Дополнительные сведения

При вызове многих функций из библиотек динамической компоновки (DLL) необходимо указать в аргументе адрес объекта OLE. следует передавать в качестве аргумента при организации вызовов из интерфейса программирования приложений (API) библиотек динамической компоновки OLE.

Примечание. Если вызываемая процедура API сама выполняет обратный вызов элемента управления, то результат подобной операции будет непредсказуемым.

Свойство Object

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproObjectC;vaconUnderstandingOleAutomation"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproObjectA"}

Свойство **Object** используется в Visual Basic для возвращения ссылки на объект программирования, который соответствует связанному или внедренному объекту OLE в элементе управления. С помощью данной ссылки можно получать доступ к ссылками или обращаться к методам объекта OLE.

Значения

Свойство **Object** возвращает ссылку на объект программирования. Чтобы назначить объект программирования переменной, используется инструкция **Set.** Тип возвращенной ссылки на объект зависит от приложения, в котором был создан данный объект OLE.

Свойство **Object** доступно только для чтения во всех режимах.

Дополнительные сведения

Если объект OLE был связан или внедрен в форму Microsoft Access, то можно установить свойства, определяющие тип объекта и действия элемента управления. Однако напрямую читать свойства объекта OLE или использовать его методы невозможно. Свойство **Object** возвращает ссылку на объект программирования, который представляет связанный или внедренный объект OLE. С помощью данной ссылки можно изменить объект OLE, установив или считав его свойства или применив методы. Например, Microsoft Excel является компонентом ActiveX, поддерживающим программирование объектов. При внедрении листа Microsoft Excel в форму Microsoft Access свойство **Object** используется для установки ссылки на объект **Worksheet,** связанный с данным листом. Далее можно использовать любые свойства и методы объекта **Worksheet**.

Для получения дополнительных сведений и свойствах и методах, поддерживаемых объектом программирования см. документацию по приложению, в котором был создан данный объект OLE.

Свойство Object, пример

В следующем примере используется свойство **Object** свободной рамки объекта с именем «OLE1». Названия и адреса клиентов вставлены во внедренный документ Microsoft Word, который имеет формат письма и содержит прототипы названия организации и адреса клиента, а также стандартный текст. Данная процедура подставляет на места прототипов данные из текущей записи и печатает очередной экземпляр письма. Копии напечатанного составного документа не сохраняются.

```
Sub PrintFormLetter_Click()
   Dim objWord As Object
   Dim strCustomer As String, strAddress As String
   Dim strCity As String, strRegion As String

   ' Записывает значение свойства в переменную.
   Set objWord = Me!OLE1.Object.Application.Wordbasic
   ' Записывает адрес клиента в переменную.
   strCustomer = Me!Название
   strAddress = Me!Адрес
   strCity = Me!Город & ", "
   If Not IsNull(Me!Область) Then
      strRegion = Me!Область
   Else
      strRegion = Me!Страна
   End If
   ' Активизирует элемент управления ActiveX.
   Me!OLE1.Action = acOLEActivate
   With objWord
      .StartOfDocument
      ' Переход к первому полю.
      .LineDown 2
      ' Выделение текста поля.
      .EndOfLine 1
      ' Ввод названия клиента.
      .Insert strCustomer
      ' Переход к следующему полю.
      .LineDown
      .StartOfLine
      ' Выделение текста поля.
      .EndOfLine 1
      ' Ввод адреса.
      .Insert strAddress
      ' Переход к последнему полю.
      .LineDown
      .StartOfLine
      ' Выделение текста поля.
      .EndOfLine 1
      ' Ввод города и области.
      .Insert strCity & strRegion
      .FilePrint
      .FileClose
   End With
   Set objWord = Nothing
End Sub
```

Свойство ObjectVerbs

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproObjectVerbsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproObjectVerbsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproObjectVerbsA"}

Свойство **ObjectVerbs** позволяет получить в программе Visual Basic список команд, поддерживаемых объектом OLE.

Значения

Значением свойства **ObjectVerbs** является массив строковых значений, нумерация элементов которого начинается с нуля.

Данное свойство недоступно в режиме конструктора и доступно только для чтения в других режимах.

Дополнительные сведения

Свойство **ObjectVerbs** применяется вместе со свойством **ObjectVerbsCount** для вывода списка команд, поддерживаемых объектом OLE. Свойство **Команда (Verb)** использует этот список команд для определения команды, которую следует выполнить при запуске объекта OLE (то есть в том случае, когда свойство **Action** получает значение **acOLEActivate**).

Свойство **Команда (Verb)** использует положение определенной команды в списке команд объекта, возвращаемых свойством **ObjectVerbs**. Например, значение определяет первую команду в списке (команду Visual Basic `ObjectVerbs(0)`, т.е. первый элемент в массиве, возвращаемом в свойстве **ObjectVerbs**), значение 2 определяет вторую команду в списке (команду Visual Basic `ObjectVerbs(1)`, т.е. второй элемент в массиве, возвращаемом в свойстве **ObjectVerbs**) и т.д.

Первая команда в массиве **ObjectVerbs,** вызываемая в программе Visual Basic как `ObjectVerbs(0)`, является командой, используемой по умолчанию. Если значение свойства **Команда (Verb)** не задано, то именно эта команда определяет действие, выполняющееся при запуске объекта OLE.

Приложения, в которых выводятся объекты OLE, обычно содержат в меню **Правка** команду **Объект**. При выборе пользователем команды **Объект** на экране появляется подменю со списком команд объекта. Свойства **ObjectVerbs** и **ObjectVerbsCount** позволяют вывести список команд в форме или отчете, вместо того чтобы выводить эти команды в меню.

Список команд, которые поддерживает объект, может изменяться в зависимости от текущего состояния объекта. Для обновления списка поддерживаемых объектом команд следует задать для свойства **Action** элемента управления значение **acOLEFetchVerbs**. Необходимо также обновить список команд объекта перед выводом этого списка для пользователя.

Свойства ObjectVerbs, ObjectVerbsCount, пример

В следующем примере возвращается список команд, поддерживаемых объектом OLE в элементе управления с именем «OLE1», и каждая команда выводится в окно сообщения.

```
Sub GetVerbList(frm As Form, OLE1 As Control)
   Dim intX As Integer, intNumVerbs As Integer
   Dim strVerbList As String

   ' Обновление списка команд.
   With frm!OLE1
      .Action = acOLEFetchVerbs
      intNumVerbs = .ObjectVerbsCount
      For intX = 0 To intNumVerbs - 1
         strVerbList = strVerbList & .ObjectVerbs(intX) & "; "
      Next intX
   End With

   ' Вывод команды в окне сообщений.
   MsgBox Left(strVerbList, Len(strVerbList) - 2)
End Sub
```

Свойство ObjectVerbsCount

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅS":"acproObjectVerbsCountC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acproObjectVerbsCountX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproObjectVerbsCountA"}

Свойство **ObjectVerbsCount** определяет в программе Visual Basic число команд, поддерживаемых объектом OLE.

Значения

Значение свойства **ObjectVerbsCount** принадлежит к типу данных **Integer,** это число определяет количество элементов в массиве команд, возвращаемых в значении свойства **ObjectVerbs.**

Данное свойство недоступно в режиме конструктора и доступно только для чтения в других режимах.

Дополнительные сведения

Список команд, которые поддерживает объект, может изменяться в зависимости от текущего состояния объекта. Для обновления списка поддерживаемых объектом команд следует установить для свойства **Action** элемента управления значение **acOLEFetchVerbs**.

Свойство «Класс OLE» (OLEClass)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproOLEClassC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproOLEClassX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproOLEClassA"}

Свойство **Класс OLE (OLEClass)** позволяет получить описание класса объекта OLE, содержащегося в элементе управления диаграммы или в свободной рамке объекта.

Значения

Значением данного свойства является строковое выражение, которое задается автоматически в окне свойств элемента управления при добавлении объекта OLE в форму с помощью команды **Объект** в меню **Вставка**. Свойство **Класс OLE (OLEClass)** доступно только для чтения во всех режимах.

Примечание. Если при программировании объектов (формально называется программирование объектов OLE) необходимо указать имя для ссылки на объект OLE, используется свойство **Класс (Class).**

Дополнительные сведения

Свойства **Класс OLE (OLEClass)** и **Класс (Class)** похожи, но не тождественны. В значении свойства **Класс OLE (OLEClass)** содержится общее описание объекта OLE, тогда как значением свойства **Класс (Class)** является имя, которое следует использовать для ссылки на этот объект OLE в программе Visual Basic. Примерами значений свойства **Класс OLE (OLEClass)** могут служить «Диаграмма Microsoft Excel», «Документ Microsoft Word» и «Точечный рисунок Paintbrush».

Свойство «Тип OLE» (OLEType)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproOleTypeC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproOLETypeX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproOleTypeA"}

Свойство **Тип OLE (OLEType)** позволяет определить, содержится ли в элементе управления объект OLE, и если да, является ли объект связанным или внедренным.

Значения

Свойство **Тип OLE (OLEType)** может иметь следующие значения.

Значения	Описание	Константа
Связанный	Элемент управления содержит связанный объект. Вся обработка данных объекта выполняется в приложении, в котором объект был создан.	**acOLELinked**
Внедренный	Элемент управления содержит внедренный объект. Обработка данных в объекте выполняется в приложении Microsoft Access.	**acOLEEmbedded**
Отсутствует	Элемент управления не содержит объект OLE.	**acOLENone**

Свойство **Тип OLE (OLEType)** доступно только для чтения во всех режимах.

Дополнительные сведения

При создании объекта OLE следует в свойстве **Допустимый тип OLE (OLETypeAllowed)** определить типы объектов, которые могут быть помещены в данный элемент управления.

Свойство «Тип OLE» (OLEType), пример

В следующей программе показано открытие диалогового окна **Вставка объекта** и вывод сообщения об ошибке в случае, если создание объекта отменяется нажатием кнопки **Отмена** в диалоговом окне **Вставка объекта**.

```
Sub ВставкаОбъекта_Click()
   Dim conUserCancelled As Integer

   ' Сообщение об ошибке при отмене пользователем.
   conUserCancelled = 2001
   On Error GoTo ButtonErr
   If OLE1.OLEType = acOLENone Then
      ' Объект OLE не создан.
      ' Открытие диалогового окна "Вставка объекта".
      OLE1.Action = acOLEInsertObjDlg
   End If
   Exit Sub

ButtonErr:
   If Err = conUserCancelled Then ' Вывод сообщения.
      MsgBox "Нажата кнопка 'Отмена', объект не создан."
   End If
   Resume Next
End Sub
```

Свойство «Допустимый тип OLE» (OLETypeAllowed)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproOleTypeAllowedC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproOleTypeAllowedX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproOleTypeAllowedA"}

Свойство **Допустимый тип OLE (OLETypeAllowed)** определяет типы объектов OLE, которые могут содержаться в элементе управления.

Значения

Свойство **Допустимый тип OLE (OLETypeAllowed)** может иметь следующие значения.

Значения	Описание	Константа
Связанный	Элемент управления может содержать только связанный объект.	**AcOLELinked**
Внедренный	Элемент управления может содержать только внедренный объект.	**AcOLEEmbedded**
Все	(Значение по умолчанию). Элемент управления может содержать связанный или внедренный объект.	**acOLEEither**

Значение свойства **Допустимый тип OLE (OLETypeAllowed)** задается в окне свойств, в макросе или в программе Visual Basic. Можно также задать значение этого свойства, используемое по умолчанию, в окне стандартных свойств элемента управления или с помощью метода **DefaultControl** в Visual Basic.

Примечание. Для свободных рамок объекта и диаграмм невозможно изменить значение свойства **Допустимый тип OLE (OLETypeAllowed)** после создания объекта. Для присоединенных рамок объекта допускается изменение значения этого свойства после создания объекта. Изменение значения свойства **Допустимый тип OLE (OLETypeAllowed)** будет влиять только на новые объекты, помещаемые в этот элемент управления.

Дополнительные сведения

Для определения типа объекта OLE, содержащегося в элементе управления, следует использовать свойство **Тип OLE (OLEType).**

Свойство «Источник данных» (SourceItem)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproSourceItemC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproSourceItemX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproSourceItemA"}

Свойство **Источник данных (SourceItem)** определяет фрагмент файла, который будет связан, при создании связанного объекта OLE.

Значения

Значение свойства **Источник данных (SourceItem)** следует задавать с использованием элементов, распознаваемых приложением, в котором был создан объект. Например, при связывании диапазона ячеек электронной таблицы Microsoft Excel в значении свойства **Источник данных (SourceItem)** следует указать ссылку на ячейку или диапазон ячеек электронной таблицы, например R1C1 или R3C4:R9C22, или имя диапазона ячеек, например, «Доходы».

Примечание. За сведениями о синтаксисе описания фрагментов данных конкретного объекта следует обращаться к документации приложения, в котором был создан объект.

Значение данного свойства задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

В программе Visual Basic значение этого свойства задается с помощью строкового выражения.

Дополнительные сведения

Для того чтобы использовать данное свойство, необходимо предварительно задать для свойства **Допустимый тип OLE (OLETypeAllowed)** значение **Связанный** или **Все**. Свойство **Документ-источник (SourceDoc)** позволяет указать связываемый файл.

Свойство «Команда» (Verb)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproVerbC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproVerbX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproVerbA"}

Свойство **Команда (Verb)** задает действие, которое следует выполнить при запуске объекта OLE (то есть когда свойство **Action** элемента управления получит значение **acOLEActivate)**.

Значения

Пользователь задает значение свойства **Команда (Verb),** указывая значение типа **Integer,** которое определяет положение команды в списке команд, возвращаемом в свойстве **ObjectVerbs.** Если свойство **Команда (Verb)** имеет значение 1, то будет выполнена первая команда из списка; значение 2 определяет вторую команду из списка и так далее.

Значение свойства **Команда (Verb)** задается в окне свойств элемента управления, в макросе или в программе Visual Basic. Можно также задать значение этого свойства, используемое по умолчанию, в окне стандартных свойств элемента управления или с помощью метода **DefaultControl** в Visual Basic.

Если свойство **ObjectVerbs** не определено, то определить действие, выполняемое при запуске объекта, позволяет одно из следующих значений свойства **Команда (Verb)**. Приведенные значения определяют стандартные команды, поддерживаемые всеми объектами.

Константа	Значение	Описание
acOLEVerbPrimary	0	Выполняет определенное для объекта действие по умолчанию.
acOLEVerbShow	–1	Запускает объект для редактирования.
acOLEVerbOpen	–2	Открывает объект в отдельном окне приложения.
acOLEVerbHide	–3	Для внедренных объектов делает невидимым приложение, в котором был создан объект.

Объекты некоторых приложений поддерживают также следующие дополнительные константы.

Константа	Значение	Описание
acOLEVerbInPlaceUIActivate	–4	Запускает объект для редактирования непосредственно в элементе управления. При этом меню и панели инструментов приложения объектов (сервера OLE) становятся доступными в приложении-контейнере OLE.
acOLEVerbInPlaceActivate	–5	Запускает объект для редактирования по месту, то есть непосредственно в элементе управления. При этом меню и панели инструментов приложения

объекта не будут доступны в приложении-контейнере.

Дополнительные сведения

Каждый объект поддерживает собственный набор команд, например, многие объекты поддерживают команды редактирования и воспроизведения. Свойства **ObjectVerbs** и **ObjectVerbsCount** позволяют определить, какие команды поддерживаются объектом.

Если свойство **Автоматический запуск (AutoActivate)** объекта имеет значение **По двойному нажатию кнопки**, то при запуске объекта с помощью двойного нажатия кнопки мыши Microsoft Access использует заданную по умолчанию команду объекта.

Свойство «Конец страницы» (ForceNewPage)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproForceNewPageC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproForceNewPageX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproForceNewPageA"}

Свойство **Конец страницы (ForceNewPage)** позволяет указать, следует ли печатать раздел формы (заголовок, область данных, примечание) или раздел отчета (заголовок, область данных, примечание) с новой страницы. Например, это свойство позволяет всегда печатать на последней странице отчета бланк заказа. Если для свойства примечания отчета **Конец страницы (ForceNewPage)** задано значение «До раздела», то бланк заказа будет напечатан на последней странице.

Примечание. Свойство **Конец страницы (ForceNewPage)** неприменимо к верхним колонтитулам и нижним колонтитулам.

Значения

Свойство **Конец страницы (ForceNewPage)** может иметь следующие значения.

Значение	Описание	Visual Basic
Значение отсутствует	Используется по умолчанию. Текущий раздел (для которого задается значение свойства) печатается на текущей странице.	0
До раздела	Текущий раздел печатается с новой страницы.	1
После раздела	Печать раздела, следующего за текущим, начинается с новой страницы.	2
До и после раздела	Текущий раздел печатается с новой страницы, и раздел, следующий за текущим, печатается с новой страницы.	3

Значение данного свойства задается в окне свойств раздела, с помощью макроса или в программе Visual Basic.

Дополнительные сведения

Ниже приводятся примеры результатов, к которым приводят различные значения свойства **Конец страницы (ForceNewPage)**.

Раздел	Пример значения	Описание
Заголовок группы, содержащий текущий год	До раздела	Заголовок группы печатается на новой странице сверху; под ним печатаются область данных, область примечаний группы и нижний колонтитул.
Область данных отчета	После раздела	Область примечаний группы печатается сверху на новой странице.
Заголовок отчета, содержащий название отчета и логотип фирмы.	После раздела	Название отчета и логотип фирмы печатаются на отдельной странице в начале отчета.

Свойство «Конец страницы» (ForceNewPage), пример

В следующем примере значение свойства **Конец страницы (ForceNewPage)** области данных отчета «Продажи по годам» считывается в переменную `intGetVal`.

```
Dim intGetVal As Integer
intGetVal = Reports![Продажи по годам].Section(acDetail).ForceNewPage
```

Свойство «Не выводить повторы» (HideDuplicates)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпї. пїЅпїЅпїЅпїЅпїЅ":"acproHideDuplicatesC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproHideDuplicatesX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproHideDuplicatesA"}

Свойство **Не выводить повторы (HideDuplicates)** позволяет не выводить элемент управления отчета, если значение этого элемента управления совпадает с его значением в предыдущей записи. Например, в отчете, содержащем имена поставщиков и перечни поставляемых ими товаров, можно выводить имя каждого поставщика только один раз в начале списка всех поставляемых им товаров и не выводить его повторно для каждой группы товаров.

Примечание. Свойство **Не выводить повторы** применяется только к элементам управления (поле, поле со списком, список, флажок, переключатель, выключатель, группа переключателей) в отчете.

Это свойство не применяется к флажкам, переключателям или выключателям, включенным в состав группы; оно применимо только к самой группе переключателей.

Значения

Свойство **Не выводить повторы (HideDuplicates)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Если значение элемента управления или его содержимое такое же, как и в предыдущей записи, то этот элемент управления не выводится.	**True** (–1)
Нет	(Значение по умолчанию). Элемент управления выводится вне зависимости от его значения в предыдущей записи.	**False** (0)

Значение данного свойства задается в окне свойств элемента управления, с помощью макроса или в программе Visual Basic.

Значение свойства **Не выводить повторы (HideDuplicates)** может быть задано только в режиме конструктора отчетов.

Дополнительные сведения

Свойство **Не выводить повторы (HideDuplicates)** позволяет создать отчет с группировкой записей, в котором выводится только область данных и не выводятся заголовок и примечание группы.

Свойство «Не выводить повторы» (HideDuplicates), пример

В следующем примере значение свойства **Не выводить повторы (HideDuplicates)** поля «КодТипа» считывается в переменную `intCurVal`.

```
Dim intCurVal As Integer
intCurVal = Me!КодТипа.HideDuplicates
```

Свойство «Не разрывать» (KeepTogether) - Разделы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproKeepTogetherSectionsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproKeepTogetherSectionsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproKeepTogetherSectionsA"}

Свойство раздела **Не разрывать (KeepTogether)** позволяет печатать весь раздел формы или отчета на одной странице. Например, это свойство удобно использовать при печати родственных данных, которые нежелательно печатать на двух страницах.

Примечание. Свойство **Не разрывать (KeepTogether)** применяется только к разделам форм и отчетов (за исключением верхних колонтитулов и нижних колонтитулов.)

Значения

Свойство **Не разрывать (KeepTogether)** раздела может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Если весь раздел не умещается на текущей странице, Microsoft Access начинает печать раздела с новой страницы.	**True** (–1)
Нет	(Значение по умолчанию). Microsoft Access печатает максимально возможную часть раздела на текущей странице, а оставшуюся часть раздела на следующей странице.	**False** (0)

Значение данного свойства задается в окне свойств раздела, с помощью макроса или в программе Visual Basic.

Значение свойства раздела **Не разрывать (KeepTogether)** можно задать только в режиме конструктора формы или в режиме конструктора отчета.

Дополнительные сведения

Если раздел не умещается полностью на текущей странице, Microsoft Access продолжает печать оставшейся части на следующей странице. Свойство раздела **Не разрывать (KeepTogether)** позволяет напечатать короткий раздел на отдельной странице. Если длина раздела превышает размер страницы, печать раздела будет продолжена на следующих страницах.

Если свойство группы **Не разрывать (KeepTogether)** имеет значение «Полную группу» или «Первую область данных», а свойство раздела **Не разрывать** имеет значение «Нет», то свойство раздела игнорируется.

Свойство «Не разрывать» (KeepTogether) - пример применения к разделам

В следующем примере значение свойства **Не разрывать (KeepTogether)** области данных отчета считывается в переменную `intGetVal`.

```
Dim intGetVal As Integer
intGetVal = Me.Section(acDetail).KeepTogether
```

Свойство «Новая строка или столбец» (NewRowOrCol)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproNewRowOrColC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproNewRowOrColX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproNewRowOrColA"}

Свойство **Новая строка или столбец (NewRowOrCol)** указывает, следует ли в отчете, печатающемся в несколько столбцов, печатать раздел и его данные, начиная с новой строки или с нового столбца. Например, данное свойство позволяет в отчете, печатающемся в несколько колонок, печатать каждый заголовок группы в начале нового столбца.

Примечание. Свойство **Новая строка или столбец (NewRowOrCol)** неприменимо к верхним колонтитулам или нижним колонтитулам.

Значения

Свойство **Новая строка или столбец (NewRowOrCol)** может иметь следующие значения.

Значение	Описание	Visual Basic
Значение отсутствует	Значение по умолчанию. Положение конца строки или столбца определяется параметрами, установленными в диалоговом окне **Параметры страницы** (которое открывается при выборе команды **Параметры страницы** в меню **Файл**), и наличием свободного места на странице.	0
До раздела	Печать текущего раздела (например, заголовка группы) начинается с новой строки или с нового столбца. Следующий раздел, например область данных, печатается в той же строке или столбце.	1
После раздела	Печать текущего раздела (например, заголовка группы) начинается в текущей строке или в текущем столбце, а печать следующего раздела (например, области данных) с новой строки или столбца.	2
До и после раздела	Печать текущего раздела начинается с новой строки или столбца, а печать следующего раздела со следующей строки или столбца.	3

Значение данного свойства задается в окне свойств раздела, с помощью макроса или в программе Visual Basic.

Значение свойства **Новая строка или столбец (NewRowOrCol) для** раздела можно задать только в режиме конструктора форм или в режиме конструктора отчетов.

Дополнительные сведения

Ниже приведены примеры использования данного свойства для заголовка группы в отчете, печатающемся в несколько колонок. При этом предполагается, что в группе **Макет столбца** на вкладке **Столбцы** диалогового окна **Параметры страницы** выбран параметр **Сверху вниз**.

Пример значения	Результат
До раздела	Заголовок группы печатается в начале нового столбца.
После раздела	Область данных печатается с начала нового столбца.

До и после раздела	Заголовок группы печатается в отдельном столбце, а область данных печатается с начала следующего столбца.

Разделы в форме или отчете обычно печатаются сверху вниз. Стандартной настройкой расположения элементов макета является **Слева направо**. Можно также разместить разделы в разных колонках, если выбрать в группе **Макет столбца** на вкладке **Столбцы** диалогового окна **Параметры страницы** параметр **Сверху вниз**.

Если для свойства **Новая строка или столбец (NewRowOrCol)** задано значение «До раздела», то размещение разделов отчета на странице будет зависеть от ориентации страницы. Если в группе **Макет столбца** на вкладке **Столбцы** диалогового окна **Параметры страницы** выбран параметр **Слева направо**, то Microsoft Access начинает печать раздела с новой строки; при выборе параметра **Сверху вниз** печать раздела начинается с нового столбца.

Свойство «Новая строка или столбец» (NewRowOrCol), пример

В следующем примере значение свойства **Новая строка или столбец (NewRowOrCol)** считывается в переменную `intGetVal`.

```
Dim intGetVal As Integer
intGetVal = Me.Section(1).NewRowOrCol
```

Ниже приводятся примеры двух вариантов макета отчета, в котором данные разбиты на четыре группы («Группа1» - «Группа4»). Каждая группа содержит от трех до шести записей, в каждой записи два поля - «a» и «b». Варианты макета отличаются только выбором параметров в группе **Макет столбца** на вкладке **Столбцы** диалогового окна **Параметры страницы** и значениями свойства **Новая строка или столбец (NewRowOrCol)**. Следует отметить, что значение в поле **Ширина** в группе **Размер столбца** на вкладке **Столбцы** должно совпадать с фактической шириной поля области данных. Кроме того, если для свойства **Новая строка или столбец** задано значение «До раздела», то для правильного применения параметра **Сверху вниз** необходимо, чтобы верхний колонтитул имел ненулевой размер.

- **Макет столбца** - **Слева направо**
- **Параметры сетки** - **число столбцов** равно 4
- **Новая строка или столбец** для заголовка группы - «До и после раздела»

Группа1
1a 1b 2a 2b 3a 3b 4a 4b
5a 5b
Группа2
1a 1b 2a 2b 3a 3b 4a 4b
Группа3
1a 1b 2a 2b 3a 3b
Группа4
1a 1b 2a 2b 3a 3b 4a 4b
5a 5b 6a 6b

- **Макет столбца** - **Сверху вниз**
- **Параметры сетки** - **число столбцов** равно 4
- **Новая строка или столбец** для заголовка группы - «До раздела»

Группа1	Группа2	Группа3	Группа4
1a 1b	1a 1b	1a 1b	1a 1b
2a 2b	2a 2b	2a 2b	2a 2b
3a 3b	3a 3b	3a 3b	3a 3b
4a 4b	4a 4b		4a 4b
5a 5b			5a 5b
			6a 6b

Свойства Page, Pages

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproPagePagesC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproPagePagesX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproPagePagesA"}

Свойства **Page** и **Pages** возвращают данные, необходимые для печати номеров страниц в форме или отчете. Например, эти свойства позволяют вывести в нижнем колонтитуле отчета номера страниц в формате «Страница *n* из *nn*».

- Свойство **Page** указывает номер текущей страницы при печати формы или отчета.
- Свойство **Pages** указывает полное число страниц в форме или отчете.

Значения

Хотя пользователь имеет возможность задать для свойства **Page** значение типа **Integer** обычно эти свойства используются для возвращения сведений о номерах страниц. При использовании в выражениях оба свойства возвращают значение типа **Integer**.

Значения свойств **Page** и **Pages** используются в выражениях, в макросах или в программах Visual Basic.

Данные свойства доступны только для предварительного просмотра или при печати.

Свойство **Pages** в любое время доступно только для чтения.

Дополнительные сведения

Чтобы обратиться к свойству **Pages** с помощью макроса или в программе Visual Basic, следует включить в форму или отчет поле, в значении свойства **Данные (ControlSource)** которого указано выражение, использующее свойство **Pages**. Например, для свойства **Данные (ControlSource)** поля, расположенного в нижнем колонтитуле, допустимыми являются следующие выражения.

Выражение	Печатается
=Page	Номер страницы (например, 1, 2, 3) в нижнем колонтитуле.
="Стр. " & Page & " из " & Pages	«Стр. *n* из *nn*» (например, Стр. 1 из 5, Стр. 2 из 5) в нижнем колонтитуле.
=Pages	Общее число страниц в форме или отчете (например, 5).

Свойства Page, Pages, пример

В следующем примере выводится сообщение о количестве страниц в отчете. Для выполнения данного примера необходимо включить в отчет поле, в свойстве **Данные (ControlSource)** которого указано выражение `=Pages`. Для проверки работы примера вставьте следующие инструкции в процедуру обработки события **Страница (Page)** формы «Список товаров».

```
Dim intTotalPages As Integer, strMsg As String
intTotalPages = Me.Pages
strMsg = "Отчет содержит " & intTotalPages & " страниц."
MsgBox strMsg
```

Свойства «Верхний колонтитул» (PageHeader), «Нижний колонтитул» (PageFooter)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnï. nïSnïSnïSnïSnïS":"acproPageHeaderFooterC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPageHeaderFooterX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPageHeaderFooterA"}

Свойства **Верхний колонтитул (PageHeader)** и **Нижний колонтитул (PageFooter)** указывают, следует ли печатать верхний колонтитул или нижний колонтитул на одной странице с заголовком отчета или с примечанием отчета. Например, если в области заголовка отчета содержится титульный лист, то можно отключить печать верхнего колонтитула, содержащего номер страницы, на первой странице отчета.

Значения

Свойства **Верхний колонтитул (PageHeader)** и **Нижний колонтитул (PageFooter)** могут иметь следующие значения.

Значение	Описание	Visual Basic
Все страницы	(Значение по умолчанию). Верхний и нижний колонтитулы печатаются на всех страницах отчета.	0
Без заголовка	Верхний или нижний колонтитул не печатается на одной странице с заголовком отчета. Microsoft Access печатает заголовок отчета с новой страницы.	1
Без примечания	Верхний или нижний колонтитул не печатается на одной странице с примечанием отчета. Microsoft Access печатает примечание отчета с новой страницы.	2
Без заголовка/примечания	Верхний или нижний колонтитул не печатается на одной странице с заголовком или примечанием отчета. Microsoft Access печатает заголовок или примечание отчета с новой страницы.	3

Значения этих свойств задаются в окне свойств отчета, с помощью макроса или в программе Visual Basic.

Значения свойств **Верхний колонтитул (PageHeader)** и **Нижний колонтитул (PageFooter)** могут быть заданы только в режиме конструктора отчетов.

Дополнительные сведения

При стандартных настройках Microsoft Access печатает верхние или нижние колонтитулы на каждой странице отчета, в том числе на первой и на последней.

Для отображения верхних и нижних колонтитулов в макете отчета выберите команду **Колонтитулы** в меню **Вид** в режиме конструктора отчета.

Примечание. При печати форм верхние и нижние колонтитулы всегда печатаются на всех страницах.

Свойства «Верхний колонтитул» (PageHeader), «Нижний колонтитул» (PageFooter), пример

В следующем примере для свойства **Верхний колонтитул (PageHeader)** отчета задается значение «Без заголовка». Для выполнения данного примера введите следующую строку в окно отладки отчета с именем «Отчет1».

```
Reports!Отчет1.PageHeader = 1
```

Свойство «Для лазерного принтера» (FastLaserPrinting)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFastLaserPrintingC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproFastLaserPrintingX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproFastLaserPrintingA"}

Свойство **Для лазерного принтера (FastLaserPrinting)** указывает, следует ли изображать линии и прямоугольники с помощью текстовых символов − символа подчеркивания (_) и символа вертикальной черты (|) − при печати формы или отчета на большинстве лазерных принтеров. Изображение линий и прямоугольников с помощью этих текстовых символов заметно ускоряет процесс печати.

Значения

Свойство **Для лазерного принтера (FastLaserPrinting)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Линии и прямоугольники заменяются текстовыми символами.	**True** (–1)
Нет	Линии и прямоугольники не заменяются текстовыми символами.	**False** (0)

Значение данного свойства задается в окне свойств формы или отчета, с помощью макроса или в программе Visual Basic.

Дополнительные сведения

Свойство **Для лазерного принтера (FastLaserPrinting)** влияет на изображение любых линий и прямоугольников при печати формы или отчета, в том числе и на внешний вид элементов управления, в состав которых входят эти фигуры (например, граница поля).

Примечание. Данное свойство не оказывает никакого влияния на принтеры PostScript, матричные принтеры и ранние версии лазерных принтеров, которые не поддерживают этот режим.

Если это свойство имеет значение «Да», и распечатываемая форма или отчет содержит перекрывающиеся прямоугольники или линии, то верхние прямоугольники или линии не закрывают прямоугольники или линии, на которые они накладываются. Если в отчете эти графические элементы перекрываются, следует задать для свойства **Для лазерного принтера (FastLaserPrinting)** значение «Нет».

Свойство «Для лазерного принтера» (FastLaserPrinting), пример

В следующем примере демонстрируется включение свойства **Для лазерного принтера (FastLaserPrinting)** для отчета «Счет», находящегося в режиме конструктора.

```
DoCmd.OpenReport "Счет", acDesign
Reports!Invoice.FastLaserPrinting = True
DoCmd.Close acReport, "Счет", acSaveYes
```

Свойство FormatCount

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFormatCountC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFormatCountX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproFormatCountA"}

Свойство **FormatCount** позволяет определить, сколько раз определялось значение свойства **Форматирование (OnFormat)** для текущего раздела отчета.

Значения

Данное свойство используется только в макросах или в процедурах Visual Basic для обработки событий, которые указываются с помощью значения свойства **Форматирование (OnFormat)** раздела.

Значение свойства **FormatCount** имеет тип данных **Long.**

Данное свойство недоступно в режиме конструктора отчетов; в остальных режимах оно доступно только для чтения.

Дополнительные сведения

Microsoft Access увеличивает значение свойства **FormatCount** при каждом обращении к свойству **Форматирование (OnFormat)** для текущего раздела. При переходе к форматированию следующего раздела свойство **FormatCount** получает значение 1.

В некоторых случаях Microsoft Access несколько раз подряд форматирует один и тот же раздел. Например, можно создать отчет, в котором свойство **Не разрывать (KeepTogether)** для области данных имеет значение «Да». При достижении конца страницы Microsoft Access выполняет первичное форматирование текущей области данных с целью проверки, умещается ли она на этой странице. Если она не умещается полностью на текущей странице, Microsoft Access переходит на следующую страницу и выполняет повторное форматирование этого раздела. В таком случае свойство **FormatCount** для области данных получает значение 2, указывающее, что перед печатью этот раздел был отформатирован дважды.

Свойство **FormatCount** также используется для того, чтобы гарантировать, что операция, влияющая на форматирование, будет выполнена только один раз для каждого раздела. В следующем примере значение функции **DLookup** вычисляется только в том случае, если свойство **FormatCount** имеет значение 1:

```
Private Sub Detail_Format(Cancel As Integer, FormatCount As Integer)
   Const conBold = 700
   Const conNormal = 400
   If FormatCount = 1 Then
      If DLookup("Название", _
            "Клиенты", "КодКлиента = Reports!" _
            & "[Наклейки для клиентов]!КодКлиента") _
            Like "B*" Then
         СтрокаНазвания.FontWeight = conBold
      Else
         СтрокаНазвания.FontWeight = conNormal
      End If
   End If
End Sub
```

Свойства MoveLayout, NextRecord, PrintSection

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproMoveLayoutC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproMoveLayoutX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproMoveLayoutA"}

Комбинация этих свойств используется в программах Visual Basic для изменения макета отчета в режиме предварительного просмотра, при выводе на печать или при записи в файл.

- Значение свойства **MoveLayout** указывает, должен ли Microsoft Access перейти к следующей позиции печати на странице.
- Значение свойства **NextRecord** указывает, должен ли выполняться переход к следующей записи раздела.
- Значение свойства **PrintSection** указывает, должен ли печататься текущий раздел.

Значения

Свойство **MoveLayout** может иметь следующие значения.

Значение	Описание
True (–1)	(Значение по умолчанию). Устанавливаются значения свойств **От левого края (Left)** и **От верхнего края (Top)** раздела, соответствующие следующей позиции печати.
False (0)	Значения свойств раздела **От левого края (Left)** и **От верхнего края (Top)** не изменяются.

Свойство **NextRecord** может иметь следующие значения.

Значение	Описание
True	(Значение по умолчанию). В разделе осуществляется переход к следующей записи.
False	Не происходит переход к следующей записи раздела.

Свойство **PrintSection** может иметь следующие значения.

Значение	Описание
True	(Значение по умолчанию). Печатается раздел.
False	Раздел не печатается.

Чтобы задать значения для этих свойств, следует определить макрос или процедуру обработки события для свойства **Форматирование (OnFormat)** раздела.

Microsoft Access устанавливает значение **True** для всех этих свойств перед событием **Форматирование (Format)** для каждого раздела.

Дополнительные сведения

Эти свойства удобно использовать, если отчет используется в качестве шаблона, заполняемого при печати с помощью макроса или программы Visual Basic.

В следующей таблице рассматриваются различные сочетания этих свойств.

MoveLayout	NextRecord	PrintSection	Описание
True	**True**	**True**	(Значения по умолчанию). Переход к следующей позиции печати, переход к следующей записи и печать данных.
True	**False**	**True**	Переход к следующей позиции печати и печать

			данных без перехода к следующей записи. Эта комбинация обычно используется, если для печати содержимого раздела требуется больше места, чем отведено в макете, поэтому оставшиеся данные печатаются на том месте, которое было бы занято следующим разделом.
False	**True**	**False**	Запись пропускается без соответствующего пустого места на странице.
True	**True**	**False**	Запись пропускается, и на странице остается соответствующее пустое место.
True	**False**	**False**	Оставляется пустое место на странице, но запись не пропускается.
False	**True**	**True**	Текущая запись печатается поверх предыдущей.
False	**False**	**True**	Не допускается.
False	**False**	**False**	Не допускается.

Свойство PrintCount

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPrintCountC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPrintCountX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPrintCountA"}

Свойство **PrintCount** позволяет определить, сколько раз определялось значение свойства **Печать (OnPrint)** для текущего раздела отчета.

Значения

Данное свойство используется только в макросах или в процедурах обработки событий, которые указываются с помощью значения свойства раздела **Печать (OnPrint)**.

Значение свойства **PrintCount** принадлежит к типу данных Long.

Данное свойство недоступно в режиме конструктора отчетов; в остальных режимах оно доступно только для чтения.

Дополнительные сведения

Microsoft Access увеличивает значение свойства **PrintCount** при каждом обращении к свойству **Печать (OnPrint)** для текущего раздела. При переходе к печати следующего раздела свойство **PrintCount** получает значение 0.

Значение свойства **PrintCount** увеличивается, например, если для текущего раздела свойство **Не разрывать (KeepTogether)** имеет значение «Нет», и раздел печатается на нескольких страницах. Предположим, что печатается отчет, содержащий сведения о заказах и сумму с накоплением по заказам. В следующем примере показано, как с помощью свойства **PrintCount** можно гарантировать, что каждое значение элемента управления «Заказ» будет включено в эту сумму только один раз:

```
Private Sub Detail_Format(Cancel As Integer, FormatCount As Integer)
   If PrintCount = 1 Then
      СуммаСНакоплением = СуммаСНакоплением + Заказ
   End If
End Sub
```

«СуммаСНакоплением» в этом примере может быть именем общей переменной или свободного элемента управления, значение которых увеличивается при каждом выводе раздела на печать.

Свойство PrtDevMode

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPrtDevModeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPrtDevModeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPrtDevModeA"}

Свойство **PrtDevMode** определяет или возвращает сведения о режиме драйвера принтера, которые указываются в диалоговом окне **Печать** для формы или отчета.

Примечание. Настоятельно рекомендуется обратиться к пакету Win32 Software Development Kit для ознакомления с полной документацией по свойствам **PrtDevMode**, **PrtDevNames** и **PrtMip**.

Значения

Значения компонентов свойства **PrtDevMode** образуют 94-байтовую структуру, которая полностью совпадает со структурой DEVMODE, описанной в руководстве разработчика Win32 Software Development Kit. Полное описание компонентов свойства **PrtDevMode** содержится в пакете Win32 Software Development Kit.

Свойство **PrtDevMode** имеет следующие компоненты.

Компонент	Описание
DeviceName	Строка, содержащая до 32 байт и указывающая имя драйвера принтера (например, «HP LaserJet IIISi», если текущим принтером является Hewlett-Packard LaserJet IIISi). Каждому драйверу принтера соответствует уникальная строка.
SpecVersion	Значение типа **Integer**, определяющее номер версии структуры DEVMODE в Win32 Software Development Kit.
DriverVersion	Значение типа **Integer**, определяющее номер версии драйвера принтера, присвоенный ему производителем.
Size	Значение типа **Integer**, определяющее размер (в байтах) структуры DEVMODE. (Это значение не учитывает необязательный компонент **dmDriverData**, предназначенный для дополнительных сведений о драйвере, которые могут быть добавлены в конец этой структуры). Если приложение обрабатывает только начальную часть структуры, не зависящую от конкретного драйвера, этот компонент можно использовать для определения длины структуры без учета различных версий.
DriverExtra	Значение типа **Integer**, определяющее размер (в байтах) необязательного компонента **dmDriverData**, предназначенного для дополнительных сведений о драйвере, которые могут быть добавлены в конец этой структуры. Если приложение не использует дополнительную часть структуры, следует установить этот компонент равным 0.
Fields	Значение типа **Long**, определяющее, какие из оставшихся компонентов структуры DEVMODE были инициализированы. Это значение может быть любой комбинацией указанных констант (в том числе, пустой). Список доступных констант содержится в разделе Константы для компонента Fields.
Orientation	Имеет тип **Integer**. Определяет ориентацию листа бумаги при печати и может принимать одно из двух значений: 1 (книжная) или 2 (альбомная).
PaperSize	Значение типа **Integer**, указывающее размеры листа бумаги при печати. Если этот компонент установлен равным 0 или

	256, то длина и ширина листа указываются с помощью компонентов PaperLength и PaperWidth соответственно. В противном случае следует задать одно из стандартных значений. Список значений для компонента PaperSize содержится в разделе Значения компонента PaperSize.
PaperLength	Значение типа **Integer**, определяющее длину листа бумаги в условных единицах, равных 1/10 миллиметра. Для специальных размеров бумаги и устройств типа матричных принтеров, которые могут печатать на бумаге разных размеров, значение этого компонента имеет приоритет над значением длины листа, указанным в компоненте PaperSize.
PaperWidth	Значение типа **Integer**, определяющее ширину листа бумаги в условных единицах, равных 1/10 миллиметра. Значение этого компонента имеет приоритет над значением ширины листа, указанным в компоненте PaperSize.
Scale	Значение типа **Integer**, определяющее масштабирование изображения при печати. Фактический размер изображения можно получить из исходного путем умножения на множитель «Scale/100». Например, при значении компонента «Scale» равном 50 на листе размером 8,5 x 11 дюймов (формат Letter) поместится столько же данных, сколько обычно помещается на странице размером 17 x 22 дюйма, поскольку при печати все размеры будут уменьшены вдвое.
Copies	Значение типа **Integer**, определяющее число копий (если принтер поддерживает печать нескольких копий).
DefaultSource	Значение типа **Integer**, определяющее используемый по умолчанию тип устройства подачи бумаги. Список имеющихся значений содержится в разделе Значения компонента DefaultSource.
PrintQuality	Имеет тип **Integer**. Определяет разрешение принтера и принимает следующие значения: –4 (высокое), –3 (среднее), –2 (низкое) и –1 (черновое).
Color	Значение типа **Integer**. Для цветного принтера указывает, следует ли печатать в цвете. Значение равно 1 (цветная печать) или 2 (монохромная печать).
Duplex	Имеет тип **Integer**. Для принтера, поддерживающего двустороннюю печать, указывает, следует ли печатать на обеих сторонах листа бумаги. Может иметь значения 1 (односторонняя печать), 2 (поперечная печать) и 3 (продольная печать).
Yresolution	Значение типа **Integer**, определяющее разрешение принтера по вертикали (в точках на дюйм). Если принтером инициализируется компонент PrintQuality, он задает разрешение принтера по горизонтали (в точках на дюйм).
TTOption	Значение типа **Integer**, задающее режим печати шрифтов TrueType. Список доступных значений содержится в разделе Значения компонента TTOption.
Collate	Значение типа **Integer**, определяющее печать с раскладкой по копиям при печати нескольких экземпляров. Печать без раскладки по копиям выполняется быстрее, так как данные посылаются на принтер только один раз.
FormName	Строка, содержащая до 16 символов, которая определяет формат используемой бумаги, например Letter или Legal.

Pad	Значение типа **Long,** заполняемое пробелами, символами или значениями для последующих версий.
Bits	Значение типа **Long**, определяющее в битах на пиксел цветовое разрешение используемого устройства отображения (экрана или принтера).
PW	Значение типа **Long**, определяющее в пикселах ширину области отображения устройства (экрана или принтера).
PH	Значение типа **Long**, определяющее высоту области отображения устройства (экране или принтере).
DFI	Значение типа **Long**, определяющее режим отображения.
DFR	Значение типа **Long**, определяющее рабочую частоту используемого режима устройства отображения в Герцах (циклах в секунду).

Значение свойства **PrtDevMode** задается в программе Visual Basic.

Значение этого свойства доступно для чтения и записи в режиме конструктора; в остальных режимах для него допускается только чтение.

Внимание! Драйверы принтеров могут добавлять дополнительные данные, зависящие от конкретного устройства, которые записываются сразу после первых 94 байт структуры DEVMODE. Поэтому вышеперечисленные значения DEVMODE в совокупности не должны превышать 94 байта.

Дополнительные сведения

Структуру DEVMODE могут использовать только драйверы принтеров, которые поддерживают экспорт функции **ExtDeviceMode**.

Приложение может получить список размеров и названий форматов бумаги, поддерживаемых принтером, вызывая функцию **DeviceCapabilities** с аргументами DC_PAPERS, DC_PAPERSIZE и DC_PAPERNAMES.

Для того, чтобы задать значение компонента TTOption, в приложении необходимо вызвать функцию **DeviceCapabilities** с аргументом DC_TRUETYPE и определить, какие режимы печати шрифтов TrueType поддерживает данный драйвер.

Константы для компонента Fields

Драйверы принтеров поддерживают только те компоненты, которые соответствуют конструктивным особенностям принтеров.

Константа	Значение
DM_ORIENTATION	&H0000001
DM_PAPERSIZE	&H0000002
DM_PAPERLENGTH	&H0000004
DM_PAPERWIDTH	&H0000008
DM_SCALE	&H0000010
DM_COPIES	&H0000100
DM_DEFAULTSOURCE	&H0000200
DM_PRINTQUALITY	&H0000400
DM_COLOR	&H0000800
DM_DUPLEX	&H0001000
DM_YRESOLUTION	&H0002000
DM_TTOPTION	&H0004000
DM_COLLATE	&H0008000
DM_FORMNAME	&H0010000

Значения компонента PaperSize

Значение	Формат бумаги
1	Letter (8,5 x 11 дюймов)
2	Letter Small (8,5 x 11 дюймов)
3	Tabloid (11 x 17 дюймов)
4	Ledger (17 x 11 дюймов)
5	Legal (8,5 x 14 дюймов)
6	Statement (5,5 x 8,5 дюймов)
7	Executive (7,25 x 10,5 дюймов)
8	A3 (297 x 420 мм)
9	A4 (210 x 297 мм)
10	A4 Small (210 x 297 мм)
11	A5 (148 x 210 мм)
12	B4 (250 x 354 мм)
13	B5 (182 x 257 мм)
14	Folio (8,5 x 13 дюймов)
15	Quarto (215 x 275 мм)
16	11 x 17 дюймов
18	Note (8,5 x 11 дюймов)
19	Конверт №9 (3,875 x 8,875 дюймов)
20	Конверт №10 (4,125 x 9,5 дюймов)
21	Конверт №11 (4,5 x 10,375 дюймов)
22	Конверт №12 (4,25 x 11 дюймов)
23	Конверт №14 (5 x 11,5 дюймов)
24	Размер C (17 x 22 дюймов)
25	Размер D (22 x 34 дюймов)
26	Размер E (34 x 44 дюймов)
27	Конверт DL (110 x 220 мм)
28	Конверт C5 (162 x 229 мм)
29	Конверт C3 (324 x 458 мм)
30	Конверт C4 (229 x 324 мм)
31	Конверт C6 (114 x 162 мм)
32	Конверт C65 (114 x 229 мм)
33	Конверт B4 (250 x 353 мм)
34	Конверт B5 (176 x 250 мм
35	Конверт B6 (176 x 125 мм)
36	Конверт (110 x 230 мм)
37	Конверт Monarch (3,875 x 7,5 дюймов)
38	Конверт 6-3/4(3,625 x 6,5 дюймов)
39	US Std Fanfold (14,875 x 11 дюймов)
40	German Std Fanfold (8,5 x 12 дюймов)
41	German Legal Fanfold (8,5 x 13 дюймов)
256	Определяется пользователем

Значения компонента DefaultSource

Значение	Описание
1	Верхний или единственный лоток принтера
2	Нижний лоток
3	Средний лоток
4	Ручная подача
5	Лоток для конвертов
6	Ручная подача конвертов
7	Автоматическая подача
8	Механическая подача
9	Лоток для бумаги малого формата
10	Лоток для бумаги большого формата
11	Лоток большой емкости
14	Кассетный лоток
256	С этого значения начинаются коды устройств подачи бумаги, соответствующие конкретным принтерам

Значения компонента TTOption

Значение	Описание
1	Печать шрифтов TrueType в виде графики. Это значение используется по умолчанию для матричных принтеров.
2	Использование шрифтов TrueType в качестве загружаемых шрифтов (т.е. они однократно загружаются в память принтера, а затем воспроизводятся). Значение по умолчанию для принтеров Hewlett-Packard, использующих язык PCL (Printer Control Language).
3	Замена шрифтов TrueType на встроенные шрифты принтера. Значение по умолчанию для принтеров PostScript.

Свойство PrtDevMode, пример

В следующем примере свойство **PrtDevMode** используется для проверки определяемых пользователем размеров листа бумаги для печати отчета.

```
Type str_DEVMODE
   RGB As String * 94
End Type

Type type_DEVMODE
   strDeviceName As String * 16
   intSpecVersion As Integer
   intDriverVersion As Integer
   intSize As Integer
   intDriverExtra As Integer
   lngFields As Long
   intOrientation As Integer
   intPaperSize As Integer
   intPaperLength As Integer
   intPaperWidth As Integer
   intScale As Integer
   intCopies As Integer
   intDefaultSource As Integer
   intPrintQuality As Integer
   intColor As Integer
   intDuplex As Integer
   intResolution As Integer
   intTTOption As Integer
   intCollate As Integer
   strFormName As String * 16
   lngPad As Long
   lngBits As Long
   lngPW As Long
   lngPH As Long
   lngDFI As Long
   lngDFr As Long
End Type

Sub CheckCustomPage(rptName As String)
   Dim DevString As str_DEVMODE
   Dim DM As type_DEVMODE
   Dim strDevModeExtra As String
   Dim rpt As Report
   Dim intResponse As Integer
   ' Открывает отчет в режиме конструктора.
   DoCmd.OpenReport rptName, acDesign
   Set rpt = Reports(rptName)
   If Not IsNull(rpt.PrtDevMode) Then
      strDevModeExtra = rpt.PrtDevMode   ' Считывает структуру DEVMODE.
      DevString.RGB = strDevModeExtra
      LSet DM = DevString
      If DM.intPaperSize = 256 Then
         ' Выводит текущие размеры.
         intResponse = MsgBox("Текущий специальный размер бумаги " _
            & DM.intPaperWidth / 254 & " в ширину и " _
            & DM.intPaperLength / 254 & " в длину." _
            & "Изменить размеры?", 4)
```

```
      Else
         ' Специальные размеры бумаги не заданы.
         intResponse = MsgBox("В отчете специальные размеры бумаги не
заданы." _
            & "Задать размеры?", 4)
      End If
      If intResponse = 6 Then
         ' Пользователь изменяет размеры.
         ' Инициализация полей.
         DM.lngFields = DM.lngFields Or DM.intPaperSize Or
DM.intPaperLength _
            Or DM.intPaperWidth
         DM.intPaperSize = 256           ' Задание специальных размеров
листа.
         ' Приглашения на ввод длины и ширины листа.
         DM.intPaperLength = InputBox("Введите длину листа " _
            & "в дюймах.") * 254
         DM.intPaperWidth = InputBox("Введите ширину листа " _
            & "в дюймах.") * 254
         LSet DevString = DM       ' Обновляет значения свойства.
         Mid(strDevModeExtra, 1, 94) = DevString.RGB
         rpt.PrtDevMode = strDevModeExtra
      End If
   End If
End Sub
```

В следующем примере демонстрируется изменение ориентации бумаги при печати отчета. Программа переключает ориентацию с книжной на альбомную или с альбомной на книжную, в зависимости от текущей ориентации отчета на странице.

```
Sub SwitchOrient(strName As String)
   Const DM_PORTRAIT = 1
   Const DM_LANDSCAPE = 2
   Dim DevString As str_DEVMODE
   Dim DM As type_DEVMODE
   Dim strDevModeExtra As String
   Dim rpt As Report
   DoCmd.OpenReport strName, acDesign
         ' Открывает отчет в режиме конструктора.
   Set rpt = Reports(strName)
   If Not IsNull(rpt.PrtDevMode) Then
      strDevModeExtra = rpt.PrtDevMode
      DevString.RGB = strDevModeExtra
      LSet DM = DevString
      DM.lngFields = DM.lngFields Or DM.intOrientation     ' Инициализирует
поля.
      If DM.intOrientation = DM_PORTRAIT Then
         DM.intOrientation = DM_LANDSCAPE
      Else
         DM.intOrientation = DM_PORTRAIT
      End If
      LSet DevString = DM
                  ' Обновляет значение свойства.
      Mid(strDevModeExtra, 1, 94) = DevString.RGB
      rpt.PrtDevMode = strDevModeExtra
   End If
End Sub
```

Свойство PrtDevNames

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPrtDevNamesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPrtDevNamesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPrtDevNamesA"}

Свойство **PrtDevNames** определяет или возвращает сведения о принтере, которые указываются в диалоговом окне **Печать** для формы или отчета.

Примечание. Настоятельно рекомендуется обратиться к пакету Win32 Software Development Kit для ознакомления с полной документацией по свойствам **PrtDevMode**, **PrtDevNames** и **PrtMip**.

Значения

Значения компонентов свойства **PrtDevNames** образуют структуру переменной длины, которая полностью совпадает со структурой DEVMODE, описанной в руководстве разработчика Win32 Software Development Kit.

Свойство **PrtDevNames** имеет следующие компоненты.

Компонент	Описание
DriverOffset	Определяет относительно начала структуры смещение строки, заканчивающейся символом **Null** и содержащей имя файла драйвера принтера (без расширения). Эта строка определяет, какой принтер выводится в диалоговом окне **Печать** при первом открытии окна.
DeviceOffset	Определяет относительно начала структуры смещение строки, заканчивающейся символом **Null** и содержащей название драйвера принтера. Длина этой строки не должна превышать 32 байта (с учетом символа **Null**) и должна совпадать с компонентом DeviceName структуры DEVMODE.
OutputOffset	Определяет относительно начала структуры смещение строки, заканчивающейся символом **Null** и содержащей имя порта MS-DOS, к которому подключен данный принтер (например, «LPT1:»).
Default	Указывает, используется ли принтер, описанный с помощью структуры DEVNAMES, по умолчанию. Если до открытия диалогового окна **Печать** значение компонента Default равняется 1, а все значения в структуре DEVNAMES совпадают с описанием принтера, который выводится в окне **Печать** как текущий, то данный принтер назначается используемым по умолчанию. Если для использования по умолчанию в окне был выбран текущий принтер, то компонент Default вновь получает значение 1.

Значение свойства **PrtDevNames** можно задать, выделив его в области принтера диалогового окна **Печать**. Кроме того, это свойство может быть определено средствами Visual Basic.

Дополнительные сведения

Microsoft Access использует структуру DEVNAMES для инициализации диалогового окна **Печать**. После того, как пользователь закрывает это окно с помощью кнопки **OK**, сведения о выбранном принтере будут занесены в структуру свойства **PrtDevNames**.

Свойство PrtMip

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPrtMipC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproPrtMipX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproPrtMipA"}

С помощью свойства **PrtMip** в программах Visual Basic задаются или возвращаются настройки режима драйвера принтера, указанного для формы или отчета в диалоговом окне **Печать**.

Значения

Значения свойства **PrtMip** образуют структуру длиной 28 байт, которая соответствует настройкам на вкладке **Поля**, заданным для формы или отчета в диалоговом окне **Параметры страницы**.

Свойство **PrtMip** имеет следующие компоненты.

Компонент	Описание
LeftMargin, RightMargin, TopMargin, BottomMargin	Значения типа **Integer**, указывающие поля — расстояния в твипах между краями страницы и печатающимся объектом.
DataOnly	Значение типа **Integer**, указывающее, какие элементы будут выведены на печать. При значении **True** (–1) печатаются только данные из таблицы или запроса в режиме таблицы, в форме или отчете, но не печатаются надписи, сетка и границы элементов управления, а также графические объекты, такие как линии и прямоугольники. При значении **False** (0) печатаются данные, надписи и все графические объекты.
ItemsAcross	Значение типа **Integer**, указывающее число столбцов в отчете или в почтовой наклейке. Этот компонент эквивалентен значению поля **Число столбцов** в группе **Параметры сетки** на вкладке **Столбцы** диалогового окна **Параметры страницы**.
RowSpacing	Значение типа **Integer**, указывающее расстояние по горизонтали между областями данных в условных единицах, равных 1/20 пункта.
ColumnSpacing	Значение типа **Integer**, указывающее расстояние по вертикали между областями данных в твипах.
DefaultSize	Имеет тип **Integer**. При значении **True** используются размеры области данных, установленные в режиме конструктора. При значении **False** используются размеры области данных, указанные с помощью компонентов ItemSizeWidth и ItemSizeHeight.
ItemSizeWidth	Значение типа **Integer**, указывающее ширину области данных в твипах. Этот компонент эквивалентен значению поля **Ширина** в группе **Размер столбца** на вкладке **Столбцы** диалогового окна **Параметры страницы**.
ItemSizeHeight	Значение типа **Integer**, указывающее высоту области данных в твипах. Этот компонент эквивалентен значению поля **Высота** в группе **Размер столбца** на вкладке **Столбцы** диалогового окна **Параметры страницы**.
ItemLayout	Значение типа **Integer**, определяющее горизонтальное (1953) или вертикальное (1954) расположение столбцов. Этот компонент эквивалентен переключателям **Слева**

	направо или **Сверху вниз** соответственно, которые выбираются в группе **Макет столбца** на вкладке **Столбцы** диалогового окна **Параметры страницы**.
FastPrint	Зарезервировано.
Datasheet	Зарезервировано.

Значение свойства **PrtMip** доступно для чтения и записи в режиме конструктора; в остальных режимах для него допускается только чтение.

Свойство PrtMip, пример

В данном примере с помощью свойства **PrtMip** определяется макет отчета с горизонтальной ориентацией столбцов.

```
Type str_PRTMIP
   strRGB As String * 28
End Type
Type type_PRTMIP
   intLeftMargin As Integer
   intTopMargin As Integer
   intRightMargin As Integer
   intBotMargin As Integer
   intDataOnly As Integer
   intWidth As Integer
   intHeight As Integer
   intDefaultSize As Integer
   intColumns As Integer
   intColumnSpacing As Integer
   intRowSpacing As Integer
   intItemLayout As Integer
   intFastPrint As Integer
   intDatasheet As Integer
End Type

Sub PrtMipCols(strName As String)
   Dim PrtMipString As str_PRTMIP
   Dim PM As type_PRTMIP
   Dim rpt As Report
   Const PM_HORIZONTALCOLS = 1953
   Const PM_VERTICALCOLS = 1954
   DoCmd.OpenReport strName, acDesign
   Set rpt = Reports(strName)
   PrtMipString.strRGB = rpt.PrtMip
   LSet PM = PrtMipString
   PM.intColumns = 2                ' Создает два столбца.
   PM.intRowSpacing = 0.25 * 1440   ' Задает интервал 0,25" между строками.
   PM.intColumnSpacing = 0.5 * 1440      ' Задает интервал 0,5" между
столбцами.
   PM.intItemLayout = PM_HORIZONTALCOLS

   LSet PrtMipString = PM           ' Обновляет значение свойства.
   rpt.PrtMip = PrtMipString.strRGB
End Sub
```

В следующем примере свойство **PrtMip** используется, чтобы задать для всех полей страницы ширину 1 дюйм.

```
Sub SetMarginsToDefault(strName As String)
   Dim PrtMipString As str_PRTMIP
   Dim PM As type_PRTMIP
   Dim rpt As Report
   DoCmd.OpenReport strName, acDesign
   Set rpt = Reports(strName)
   PrtMipString.strRGB = rpt.PrtMip
   LSet PM = PrtMipString
   PM.intLeftMargin = 1 * 1440     ' Задается ширина полей.
```

```
    PM.intTopMargin = 1 * 1440
    PM.intRightMargin = 1 * 1440
    PM.intBotMargin = 1 * 1440
    LSet PrtMipString = PM          ' Обновляет значение свойства.
    rpt.PrtMip = PrtMipString.strRGB
End Sub
```

Свойство «Псевдоним» (Alias)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproAliasC;dasqlJetSQLReservedWords"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproAliasX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproAliasA"}

Свойство **Псевдоним (Alias)** используется для указания специального имени для базовой таблицы или запроса, используемых в одном запросе несколько раз.

Примечание. Свойство **Псевдоним (Alias)** применяется только к таблицам или запросам, служащим источником данных для запроса.

Значения

Свойство **Псевдоним (Alias)** задается с помощью строкового выражения, содержащего имя, не используемое в базе данных. Чтобы определить значение этого свойства, следует в режиме конструктора запроса щелкнуть правой кнопкой список полей в бланке запроса и выбрать команду **Свойства**.

Значение свойства **Псевдоним (Alias)** может быть определено при создании нового запроса в режиме SQL окна запроса с помощью предложения AS в инструкции SQL.

Дополнительные сведения

Использование псевдонима обязательно при создании рекурсивного объединения в инструкции SQL.

При добавлении в запрос таблицы или запроса Microsoft Access автоматически задает для свойства **Псевдоним (Alias)** имя таблицы или запроса. Если в запрос добавляется второй экземпляр таблицы или запроса, то значением свойства **Псевдоним** становится имя объекта с добавленным символом подчеркивания и порядковым номером (_1).

Создание псевдонима не изменяет имя таблицы или запроса; при этом просто задается новое имя, которое также может быть использовано для ссылок на таблицу или запрос. Например, в запросе с рекурсивным объединением, в котором выводятся сведения о сотрудниках и администрации, можно использовать два экземпляра таблицы «Сотрудники». Второй таблице будет автоматически присвоено имя «Сотрудники_1». В этом случае удобно использовать более информативное имя второй таблицы, например, задать в свойстве **Псевдоним (Alias)** имя «Начальники».

Свойство «Псевдоним» (Alias), примеры

В данном примере значение свойства формы **Источник записей (RecordSource)** задается с помощью инструкции SQL, в которой предложение AS вводит псевдоним «Начальники» для таблицы «Сотрудники»

```
Dim strGetSQL AS String
strGetSQL = "SELECT Сотрудники.Имя, Сотрудники.Фамилия, " _
   & "Сотрудники.Должность, Начальники.Имя, Начальники.Фамилия FROM
Сотрудники " _
   & "INNER JOIN Сотрудники AS Начальники " _
   & "ON Сотрудники.Подчиняется = Начальники.КодСотрудника; "
Forms!Форма1.RecordSource = strGetSQL
```

В следующем примере приведен запрос с рекурсивным объединением, который показывает сотрудников и их начальников. С помощью предложения AS свойству **Псевдоним (Alias)** копии таблицы «Сотрудники» присваивается значение «Начальники». Эта инструкция SQL может быть введена в окно запроса в режиме SQL.

```
SELECT Сотрудники.Имя, Сотрудники.Фамилия, Сотрудники.Должность,
Начальники.Имя, Начальники.Фамилия
FROM Сотрудники INNER JOIN Сотрудники AS Начальники
ON Сотрудники.Подчиняется = Начальники.КодСотрудника;
```

Свойство «Заголовки столбцов» (ColumnHeadings)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproColumnHeadingsC;daSQLTRANSFORM"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproColumnHeadingsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproColumnHeadingsA"}

Свойство **Заголовки столбцов (ColumnHeadings)** позволяет указать порядок или ограничить число заголовков столбцов, выводящихся в перекрестном запросе. Например, в таком запросе можно указать вывод названий стран в определенном порядке, сначала «Россия», затем «Украина» и далее «Литва».

Значения

Заголовки столбцов вводятся в строковое выражение в том порядке, в котором они должны выводиться в режиме таблицы перекрестного запроса. В качестве разделителей имен заголовков следует использовать символ разделителя списка, установленный в окне **Язык и стандарты** панели управления Windows (в режиме SQL разделителем списков служит запятая). В следующей таблице представлены примеры допустимых значений свойства **Заголовки столбцов (ColumnHeadings)** и результаты, к которым они приводят при выводе перекрестного запроса в режиме таблицы.

Пример	Результат
«Кв 1», «Кв 2», «Кв 3», «Кв 4»	Заголовки столбцов для вывода данных по кварталам.
«Россия», «Грузия», «Литва»	Заголовки столбцов для вывода данных по странам.

Значение данного свойства задается в окне свойств перекрестного запроса или в программе Visual Basic. Можно также задать значение этого свойства в режиме SQL окна запроса с помощью предложения PIVOT в инструкции SQL.

Примечание. Имена заголовков, указанные при определении свойства **Заголовки столбцов (ColumnHeadings)**, должны точно совпадать с содержимым поля, указанного в бланке запроса. Иначе данные не будут выводиться в столбцах.

Дополнительные сведения

Свойство **Заголовки столбцов (ColumnHeadings)** дает дополнительные возможности контроля над представлением данных в перекрестном запросе. По умолчанию Microsoft Access создает отдельный столбец для каждого значения и располагает столбцы в порядке возрастания значений. Например, если имена заголовков берутся из поля «Месяц», то в режиме таблицы выводятся столбцы «Август», «Апрель», «Декабрь» и т.д. Свойство **Заголовки столбцов** позволяет вывести данные в естественном порядке: «Январь», «Февраль», «Март» и т.д.

Столбец, заголовок которого указан в значении свойства **Заголовки столбцов (ColumnHeadings)**, всегда выводится в режиме таблицы, даже если он не содержит данные. Это особенно удобно для отчетов, базирующихся на перекрестных запросах, поскольку позволяет сохранить последовательность заголовков столбцов в напечатанном отчете.

Совет. Свойство **Заголовки столбцов (ColumnHeadings)** позволяет увеличить скорость выполнения некоторых перекрестных запросов за счет ограничения числа выводящихся столбцов.

Свойство «Заголовки столбцов» (ColumnHeadings), пример

В данном примере определяются четыре заголовка столбцов («Кв 1», «Кв 2», «Кв 3» и «Кв 4»), которые должны выводиться в режиме таблицы перекрестного запроса «Распределение товаров по заказам» (база данных Борей.MDB). Эту инструкцию можно ввести в окне запроса в режиме SQL.

```
TRANSFORM Sum(CCur([Заказано].Цена * [Заказано].Количество *
(1-[Скидка])/100) * 100)
AS ОбъемЗаказа SELECT Товары.Марка FROM Товары
INNER JOIN (Заказы INNER JOIN [Заказано] ON Заказы.КодЗаказа =
[Заказано].КодЗаказа)
ON Товары.КодТовара = [Заказано].КодТовара
WHERE (((Заказы.ДатаРазмещения) Between #1/1/94# And #12/31/94#))
GROUP BY Товары.Марка PIVOT "Кв " & DatePart("q",[ДатаРазмещения],1,0);
```

Свойство «Вывод всех полей» (OutputAllFields)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproOutputAllFieldsC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproOutputAllFieldsX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproOutputAllFieldsA"}

Свойство **Вывод всех полей (OutputAllFields)** используется для вывода всех полей в базовом источнике данных запроса и в списке полей формы или отчета. Данное свойство позволяет включить в список все поля без необходимости устанавливать флажок **Вывод на экран** в бланке запроса для каждого поля.

Примечание. Свойство **Вывод всех полей (OutputAllFields)** применяется только к запросам на добавление записей, на создание таблицы и на выборку.

Значения

Свойство **Вывод всех полей (OutputAllFields)** может принимать следующие значения.

Значение	Описание
Да	Задает вывод всех полей из базовых таблиц и их включение в список полей формы или отчета.
Нет	(Значение по умолчанию). Задает вывод только тех полей, для которых установлен флажок **Вывод на экран** в бланке запроса.

Значение данного свойства задается в окне свойств запроса.

Примечание. В инструкции SQL значению «Да» свойства **Вывод всех полей (OutputAllFields)** соответствует звездочка (*) вместо имени поля.

Дополнительные сведения

Если для свойства **Вывод всех полей (OutputAllFields)** установлено значение «Да», то требуется включить в бланк запроса только поля, для которых проводится сортировка или для которых определено условие.

При сохранении фильтра в качестве запроса для свойства **Вывод всех полей (OutputAllFields)** автоматически устанавливается значение «Да».

Свойство «Вывод всех полей» (OutputAllFields), примеры

В данном примере звездочка в инструкции SQL определяет отбор всех полей из таблицы «Доставка». После этого с помощью свойства **Данные (ControlSource)** задается вывод в полях формы содержимого отдельных полей запроса.

```
Dim strGetSQL AS String
strGetSQL = "SELECT * FROM Доставка"
Forms!Форма1.RecordSource = strGetSQL
Поле1.ControlSource = "Доставка"
Поле2.ControlSource = "Название"
```

В следующем примере инструкция SQL задает сортировку по возрастанию значений полей «Название» и «Марка» и вывод всех полей из таблиц «Поставщики» и «Товары» в списке полей и в режиме таблицы. Эта инструкция SQL может быть введена в окно запроса в режиме SQL.

```
SELECT * FROM Поставщики INNER JOIN Товары
ON Поставщики.[Поставщик] = Товары.[Поставщик]
ORDER BY Поставщики.[Название], Товары.[Марка];
```

Свойство «Возврат записей» (ReturnsRecords)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS.
nїSnїSnїSnїSnїS":"acproReturnsRecordsC;daproODBCTimeout;daproReturnsRecords"} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproReturnsRecordsX":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproReturnsRecordsA"}

Свойство **Возврат записей (ReturnsRecords)** определяет, должен ли запрос к серверу SQL возвращать записи. Например, значение «Да» свойства **Возврат записей** в запросе к серверу `SELECT * FROM СОТРУДНИКИ` определяет вывод в режиме таблицы всех записей из таблицы «Сотрудники».

Примечание. Свойство **Возврат записей (ReturnsRecords)** применяется только к запросам к серверу.

Значения

Свойство **Возврат записей (ReturnsRecords)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	(Значение по умолчанию). Запрос возвращает записи (аналогично запросу на выборку).	**True** (–1)
Нет	Запрос не возвращает записи (аналогично запросу на изменение).	**False** (0)

Значение данного свойства задается в окне свойств запроса или в программе Visual Basic.

Совет. Хотя возможен доступ к свойству запроса **Возврат записей (ReturnsRecords)** в программе Visual Basic, рекомендуется использовать свойство **ReturnsRecords** объектов доступа к данным.

Дополнительные сведения

Запрос к серверу может использоваться как для возвращения записей, так и для изменения данных, создания объекта базы данных или выполнения действий, аналогичных запросу на изменение.

Если свойство **Возврат записей (ReturnsRecords)** принимает значение «Да», запрос к серверу можно поместить внутрь другого запроса или использовать в качестве источника данных для поля со списком, списка, формы или отчета. Если свойство **Возврат записей** принимает значение «Нет», запрос к серверу нельзя использовать в качестве источника данных для объекта или элемента управления, поскольку записи при этом не возвращаются.

Свойство «При запуске предоставляются права» (RunPermissions)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproRunPermissionsC;daSQLWITHOWNERACCESSOption"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproRunPermissionsX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproRunPermissionsA"}

Свойство **При запуске предоставляются права (RunPermissions)** позволяет при работе в сети в составе защищенной рабочей группы преодолевать ограничения, накладываемые существующими разрешениями. Это дает возможность предоставить пользователям, в обычных условиях не обладающим достаточными правами, возможность просматривать результат выполнения запроса или выполнять запрос на добавление, удаление, создание таблицы или обновление. Если владелец указывает в свойстве **При запуске предоставляются права** права владельца для пользователей, то пользователи получают возможность выполнять запрос на добавление записей в таблицу.

Значения

Свойство **При запуске предоставляются права (RunPermissions)** может иметь следующие значения.

Значение	Описание
Владельца	Всем пользователям предоставляются права владельца для просмотра или запуска этого запроса.
Пользователя	(Значение по умолчанию). Пользователи не получают дополнительные права.

Значение данного свойства задается в окне свойств запроса.

Допускается также указание значения свойства **При запуске предоставляются права (RunPermissions)** в режиме SQL окна запроса с помощью описания WITH OWNERACCESS OPTION в инструкции SQL.

Свойство «При запуске предоставляются права» (RunPermissions), пример

В данном примере приведен запрос-выборка, источником данных которого является таблица «Типы», в котором всем пользователям предоставляются права владельца. Инструкция SQL вводится в режиме SQL в окне запроса.

```
SELECT Типы.КодТипа, Типы.Категория, Типы.Описание,
FROM Типы WITH OWNERACCESS OPTION;
```

Свойство «Источник» (Source) - (Microsoft Access)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproSourceC;daproConnect"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproSourceX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproSourceA"}

Свойство **Источник (Source)** позволяет указать в запросе строку подключения и базу данных-источник для базовой таблицы или запроса. Значение данного свойство необходимо указать, если требуется получить доступ к данным во внешней таблице без ее присоединения.

Совет. Доступ к данным в таблице по протоколу ODBC с помощью свойств **Источник (Source)** и **SourceConnectStr** осуществляется значительно медленнее, чем при использовании связанных таблиц.

Значения

Значение свойства **Источник (Source)** задается с помощью строкового выражения. Оно может быть определено непосредственно в программе. Значение свойства устанавливается при создании нового запроса с помощью инструкции SQL. Предложение FROM в инструкции соответствует значению свойства **Источник.**

Ниже приводятся примеры использования свойства **Источник (Source)** для подключения к внешним базам данных различных типов.

Тип базы данных	Следует указать	Пример
Microsoft Access	Путь и имя базы данных. Microsoft Access добавит расширение имени файла автоматически.	C:\Accts\Customers
dBASE	Тип базы данных и путь к ней. Список требуемых спецификаций см. в описании свойства **Connect** объектов доступа к данным.	dBASE IV;DatabaseTable=C:\DBDATA
SQL Server (ODBC)	Имя базы данных-источника и другие сведения, требуемые программным продуктом, например, код пользователя и пароль. Для создания строки подключения можно использовать построитель строки подключения ODBC.	ODBC;DSN=salessrv;UID=igor; PWD=password;DATABASE=sales;

Значение данного свойства задается окне свойств списка полей запроса.

Допускается также указание значения данного свойства в режиме SQL окна запроса с помощью предложений FROM и IN в инструкции SQL.

Дополнительные сведения

Если все базовые таблицы запроса содержатся в одной и той же внешней базе данных, определите свойства запроса **Строка подключения-источник (SourceConnectStr)** и **База данных-источник (SourceDatabase)** вместо определения свойства **Источник (Source)** для каждой базовой таблицы или запроса.

Свойство «Источник» (Source), примеры

В данном примере в качестве значения свойства поля **Источник строк (RowSource)** для списка «Контакты» задается поле из таблицы dBASE IV.

```
Dim strGetSQL AS String
strGetSQL = "SELECT Клиенты.Название, Клиенты.Телефон FROM Клиенты IN 'c:\
dbdata'[dBASE IV;];"
Me.Контакты.RowSource = strGetSQL
```

В следующем примере источником данных запроса служит таблица dBASE IV с именем «Клиенты», находящаяся в каталоге C:\Dbdata. Эта инструкция SQL может быть введена в окно запроса в режиме SQL.

```
SELECT Клиенты.Название, Клиенты.Телефон
FROM Клиенты IN 'c:\dbdata'[dBASE IV;];
```

Свойство «Набор значений» (TopValues)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproTopValuesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproTopValuesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproTopValuesA"}

Свойство **Набор значений (TopValues)** указывает, сколько самых больших или самых маленьких значений должно быть возвращено в запросе. Например, можно указать вывод в запросе 10 нижних значений поля или 25% верхних.

Примечание. Свойство **Набор значений (TopValues)** применяется только к запросам на добавление записей, на создание таблицы и на выборку.

Значения

Значением свойства **Набор значений (TopValues)** является значение типа **Integer** (целое), представляющее число возвращаемых значений или долю общего числа значений в процентах (%). Например, для вывода 10 верхних значений поля, в качестве значения свойства **Набор значений** следует ввести 10, а для вывода значений, попадающих в верхние 10 процентов, следует указать 10%.

Допускается также задание значения данного свойства в режиме SQL окна запроса с помощью предложений TOP *n* или TOP *n* PERCENT в инструкции SQL.

Значение свойства **Набор значений (TopValues)** задается в окне свойств запроса или в поле **Набор значений** 

на панели инструментов **Конструктор запросов**.

Примечание. Стандартные значения свойства **Набор значений (TopValues)** выводятся в раскрывающемся списке на панели инструментов **Конструктор запросов**. Пользователь имеет возможность выбрать значение из списка или ввести в поле элемента управления другое значение.

Дополнительные сведения

Обычно свойство **Набор значений (TopValues)** используют для сортируемых полей. Поле, в котором требуется ограничить число отображаемых значений, должно быть крайним левым полем с заданным условием сортировки в бланке запроса. Настройка «По возрастанию» возвращает младшие значения, а настройка «По убыванию» старшие. Если в свойстве **Набор значений** задано конкретное число, Microsoft Access возвращает указанное число записей, а также все записи, у которых значение в первом поле совпадает со значением этого поля в последней отобранной записи.

Рассмотрим приведенный ниже анализ выработки сотрудников.

Продажи	Сотрудник
90,000	Шварц
80,000	Петров
70,000	Сидоров
70,000	Ли
60,000	Смит
50,000	Иванов

Если для свойства **Набор значений (TopValues)** заданы значение 3 и сортировка поля «Продажи» по убыванию, будут возвращены следующие четыре записи.

Продажи	Сотрудник
90,000	Шварц
80,000	Петров

70,000 Сидоров
70,000 Ли

Примечание. Для того чтобы в возвращаемом наборе значений не было повторяющихся значений, установите для свойства **Уникальные значения (UniqueValues)** значение «Да».

Свойство «Набор значений» (TopValues), пример

В данном примере значение свойства формы **Источник записей (RecordSource)** формы задается с помощью строки SQL, определяющей 10 самых дорогих товаров.

```
Dim strGetSQL As String
strGetSQL = "SELECT TOP 10 Товары.[Марка] " _
   & "AS ДорогаяДесятка, Товары.Цена FROM Товары " _
   & "ORDER BY Товары.[Цена] DESC;"
Me.RecordSource = strGetSQL
```

Свойства «Строка подключения-получатель» (DestConnectStr), «База данных-получатель» (DestinationDB), «Таблица-получатель» (DestinationTable)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproDestConnectStrC;daproConnect;dasqlIn;dasqlInsertInto"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproDestConnectStrX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproDestConnectStrA"}

- Свойство **База данных-получатель (DestinationDB)** определяет тип приложения, в котором была создана внешняя база данных.
- Свойство **Строка подключения-получатель (DestConnectStr)** определяет строку подключения к базе данных, в которую будет добавлена новая таблица (в запросе на создание таблицы), или к базе данных, содержащей таблицу, в которую будут добавлены записи (в запросе на добавление записей).
- Свойство **Таблица-получатель (DestinationTable)** определяет имя новой таблицы или имя таблицы, в которую будут добавляться записи.

Примечание. Свойства **Строка подключения-получатель (DestConnectStr)**, **База данных-получатель (DestinationDB)** и **Таблица-получатель (DestinationTable)** применяются только к запросам на создание таблицы и к запросам на добавление.

Значения

Значения свойств **Строка подключения-получатель (DestConnectStr)**, **База данных-получатель (DestinationDB)** и **Таблица-получатель (DestinationTable)** задаются с помощью строкового выражения.

Свойство **База данных-получатель (DestinationDB)** по умолчанию получает значение «(текущая)», определяющее базу данных, являющуюся активной в данный момент.

Значения этих свойств задаются в окне свойств запроса или в режиме SQL в окне запроса.

В инструкции SQL запроса на добавление имя таблицы в инструкции INSERT INTO соответствует значению свойства **Таблица-получатель (DestinationTable)**. Предложение IN соответствует значениям свойств **База данных-получатель (DestinationDB)** и **Строка подключения-получатель (DestConnectStr)**.

В инструкции SQL запроса на создание таблицы имя таблицы в инструкции INTO соответствует значению свойства **Таблица-получатель (DestinationTable)**. Предложение IN соответствует значениям свойств **База данных-получатель (DestinationDB)** и **Строка подключения-получатель (DestConnectStr)**.

Примечание. Microsoft Access задает значения этих свойств автоматически с помощью сведений, введенных в окно свойств запроса или в окно запроса в режиме SQL.

Дополнительные сведения

После выбора в меню **Запрос** команды **Создание таблицы** или **Добавление** на экране открывается диалоговое окно, предназначенное для ввода имени таблицы, которую следует создать или дополнить, и для указания базы данных-получателя. Значение, введенное в поле **Имя таблицы**, становится значением свойства **Таблица-получатель (DestinationTable)**, а значение поля **Имя файла** используется для определения свойств **Строка подключения-получатель (DestConnectStr)** и **База данных-получатель (DestinationDB)**.

Кроме того, эти свойства можно определить с помощью окна свойств. Для того чтобы указать таблицу в базе данных Microsoft Access, введите имя таблицы (например, «Клиенты») в ячейку **Таблица-получатель**, а в ячейку **База данных-получатель** путь и имя файла базы данных без расширения (например, C:\Счета\Клиенты). Нужное расширение имени файла будет добавлено автоматически. Вводить значение в ячейку свойства **Строка подключения-**

получатель при этом не требуется.

Для того чтобы указать таблицу в базе данных, созданной в другом приложении (например, Paradox), введите имя таблицы (например, «Клиенты») в ячейку **Таблица-получатель**, а в ячейку **База данных-получатель** путь к нужной базе данных (например, C:\Paradoxdata). В ячейку **Строка подключения-получатель** введите указатель типа базы данных (например, Paradox 3.5). Список указателей типа базы данных см. в описании свойства объектов доступа к данным **Connect.**

Для того чтобы указать базу данных ODBC, введите в ячейку **Строка подключения-получатель** имя базы данных и другие необходимые сведения (такие как код пользователя и пароль). Например, для базы данных Microsoft SQL Server строка подключения может выглядеть следующим образом:

```
ODBC;DSN=salessrv;UID=igor;PWD=password;DATABASE=sales;
```

Для получения более подробных сведений о драйверах ODBC (например, Microsoft SQL Server) следует обратиться к встроенной справочной системе конкретного драйвера.

Для баз данных ODBC свойство **База данных-получатель (DestinationDB)** определять не требуется.

Дополнительные сведения по доступу к базам данных ODBC см. в главе 18 «Работа с внешними данными» руководства *Разработка приложений для Microsoft Access 97.*

Свойства «Строка подключения-получатель» (DestConnectStr), «База данных-получатель» (DestinationDB), «Таблица-получатель» (DestinationTable), пример

В данном примере записи из таблицы «Клиенты» добавляются в таблицу «Телефоны» в базе данных Microsoft Access с именем файла Клиенты в каталоге C:\Данные. Эта инструкция SQL вводится в окно запроса в режиме SQL.

```
INSERT INTO Телефоны IN 'C:\Данные\Клиенты.mdb'
SELECT Клиенты.[Название], Клиенты.Телефон
FROM Клиенты;
```

В следующем примере создается новая таблица с именем «Contacts» в базе данных Paradox версии 3.5 в каталоге C:\Paradoxdata.

```
SELECT Клиенты.Название, Клиенты.Телефон
INTO Contacts IN 'C:\Pdoxdata' [Paradox 3.x;] FROM Клиенты;
```

Свойство «Таблица сообщений» (LogMessages)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproLogMessagesC;daproLogMessages"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproLogMessagesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproLogMessagesA"}

Свойство **Таблица сообщений (LogMessages)** в запросе к серверу SQL определяет, должны ли сообщения, возвращаемые из базы данных SQL, сохраняться в таблице сообщений в текущей базе данных Microsoft Access.

Примечание. Свойство **Таблица сообщений (LogMessages)** применяется только к запросам к серверу.

Значения

Свойство **Таблица сообщений (LogMessages)** может принимать следующие значения.

Значение	Описание	Visual Basic
Да	Microsoft Access сохраняет сообщения, возвращаемые из базы данных SQL, в таблице сообщений.	**True** (–1)
Нет	(Значение по умолчанию). Microsoft Access не сохраняет сообщения, возвращаемые из базы данных SQL, в таблице сообщений.	**False** (0)

Значение данного свойства задается в окне свойств запроса или в программе Visual Basic.

Дополнительные сведения

Таблице сообщений присваивается имя *имяПользователя - nn*, где *имяПользователя* представляет имя, под которым подключился пользователь, запустивший запрос к серверу, а *nn* порядковый номер (целое число), начинающийся с 00 и увеличивающийся на 1. Например, если пользователь «OlgaW» установит значение «Да» свойства **Таблица сообщений (LogMessages)** и получит первое сообщение из базы данных SQL, то таблице сообщений будет присвоено имя «OlgaW-00». Если этот пользователь получит сообщения в следующем сеансе Microsoft Access (а первая таблица не будет удалена), то будет создана новая таблица с именем «OlgaW-01».

Примечание. Сообщения об ошибках, полученные от сервера SQL, не заносятся в таблицу сообщений.

Свойство «Строка подключения ODBC» (ODBCConnectStr)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅS. пїЅпїЅпїЅпїЅпїЅS":"acproODBCConnectStrC;daproConnect;daproODBCTimeout"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"acproODBCConnectStrX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproODBCConnectStrA"}

Свойство **Строка подключения ODBC (ODBCConnectStr)** в запросе к серверу SQL определяет строку подключения ODBC.

Примечание. Свойство **Строка подключения ODBC (ODBCConnectStr)** применяется только к запросам к серверу.

Значения

Введите строку подключения ODBC, предназначенную для подключения к нужной базе данных SQL.

Значение данного свойства задается в окне свойств запроса или в программе Visual Basic.

Кроме того, определить значение этого свойства позволяет построитель строк подключения ODBC. Этот построитель устанавливает связь с нужным сервером базы данных SQL и прерывает связь после создания строки подключения ODBC.

Дополнительные сведения

По умолчанию, значением этого свойства является строка "ODBC;", которая выводится в окне свойств и восстанавливается при удалении текущего значения. Если это свойство имеет значение «ODBC;», Microsoft Access выводит приглашение ввести строку подключения при каждом выполнении запроса. Введенная пользователем строка подключения при этом не сохраняется. Для сохранения строки подключения ее необходимо ввести в ячейку **Строка подключения ODBC** в окне запроса.

Совет. Если полная строка подключения к базе данных SQL известна, введите ее в ячейку **Строка подключения ODBC** в окне свойств запроса. Это позволит не вводить сведения, необходимые для подключения, при каждом выполнении запроса.

Для разных типов источников данных ODBC используются разные строки подключения. Например, строка подключения, в которую включаются код пользователя «Ivanov» и пароль «Sesame», для источника данных с именем «Human Resources», имеющего на сервере Microsoft SQL Server имя HRSRVR, выглядит следующим образом:

```
ODBC;DSN=Human Resources;SERVER=HRSRVR;UID=Ivanov;PWD=Sesame
```

Дополнительные сведения по доступу к базам данных ODBC см. в главе 18 «Работа с внешними данными» руководства *Разработка приложений для Microsoft Access 97.*

Свойство «Время ожидания ODBC» (ODBCTimeout)

{ewc HLP95EN.DLL,DYNALINK,"пїSпїS.
пїSпїSпїSпїSпїS":"acproODBCTimeoutC;daobjQueryDefU;daproODBCTimeout;daproQueryTimeout"} {ewc HLP95EN.DLL,DYNALINK,"пїSпїSпїSпїSпїSпїS":"acproODBCTimeoutX":1} {ewc HLP95EN.DLL,DYNALINK,"пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acproODBCTimeoutA"}

Свойство **Время ожидания ODBC (ODBCTimeout)** указывает число секунд, в течение которых Microsoft Access будет повторять попытки выполнения запроса к базе данных ODBC до возвращения ошибки.

Значения

Свойство **Время ожидания ODBC (ODBCTimeout)** имеет значение типа **Integer,** определяющее время ожидания в секундах. По умолчанию задается время ожидания 60 секунд. Если данное свойство имеет значение 0, ошибка превышения времени ожидания не возникает.

Значение данного свойства задается в окне свойств или в программе Visual Basic.

Дополнительные сведения

При работе с базой данных ODBC, например, Microsoft SQL Server, могут возникнуть задержки из-за высокой нагрузки на сеть или на сервер ODBC. Свойство **Время ожидания ODBC (ODBCTimeout)** позволяет указать время, в течение которого Microsoft Access будет повторять попытки выполнения запроса, прежде чем возникнет ошибка превышения времени ожидания.

Дополнительные сведения по доступу к базам данных ODBC см. в главе 18 «Работа с внешними данными» руководства *Разработка приложений для Microsoft Access 97.*

Свойства «Строка подключения-источник» (SourceConnectStr), «База данных-источник» (SourceDatabase)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproSourceConnectStrConnectDBC;daproConnect;dasqlIn"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproSourceConnectStrConnectDBX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproSourceConnectStrConnectDBA"}

Ниже перечислены свойства, используемые для доступа к данным при связывании внешних таблиц с текущей базой данных.

- Свойство **Строка подключения-источник (SourceConnectStr)** указывает имя приложения, в котором была создана внешняя база данных.
- Свойство **База данных-источник (SourceDatabase)** указывает внешнюю базу данных, которая содержит базовые таблицы и запросы запроса.

Примечание. Свойства **Строка подключения-источник (SourceConnectStr)** и **База данных-источник (SourceDatabase)** применяются ко всем запросам, за исключением управляющих запросов, запросов к серверу и запросов на объединение.

Значения

Значения свойств **Строка подключения-источник (SourceConnectStr)** и **База данных-источник (SourceDatabase)** задаются с помощью строкового выражения.

Значения этих свойств задаются в окне свойств запроса или в режиме SQL окна запроса. В инструкции SQL данным свойствам соответствует предложение IN.

Примечание. При доступе к источникам, находящимся в нескольких базах данных, используется свойство **Источник (Source).** вместо свойств **Строка подключения-источник (SourceConnectStr)** и **База данных-источник (SourceDatabase)**

Дополнительные сведения

Свойства **Строка подключения-источник (SourceConnectStr)** и **База данных-источник (SourceDatabase)** необходимо использовать для доступа к таблицам во внешних базах данных, которые создаются в приложениях, не поддерживающих связывание таблиц.

Ниже приводятся примеры возможных значений этих свойств.

- Для базы данных Microsoft Access в качестве значения свойства **База данных-источник (SourceDatabase)** следует указать путь и имя базы данных (например, C:\Счета\Клиенты). Нужное расширение имени файла добавляется автоматически. Значение свойства **Строка подключения-источник (SourceConnectStr)** при этом определять не требуется.
- Для базы данных, созданной в другом приложении (например, Paradox), в качестве значения свойства **База данных-источник (SourceDatabase)** следует указать путь к нужной базе данных (например, C:\Paradoxdata), а в качестве значения свойства **Строка подключения-источник (SourceConnectStr)** указатель типа базы данных (например, Paradox 3.x). Список указателей типов баз данных см. в описании свойства **Connect**
- В следующем примере источником данных для запроса служат таблицы dBASE IV из каталога C:\Dbdata.

```
SELECT Клиенты.Название, Заказы.Заказ, Заказы.ДатаРазмещения
FROM Клиенты INNER JOIN Заказы
ON Клиенты.КодКлиента = Заказы.КодКлиента
IN 'C:\Dbdata'[dBASE IV;];
```

- Для базы данных ODBC (ODBC) в качестве значения свойства **Строка подключения-источник (SourceConnectStr)** следует указать имя базы данных-источника и другие необходимые сведения (такие как код пользователя и пароль). Например, для базы данных Microsoft SQL Server строка подключения может выглядеть следующим образом:

```
ODBC;DSN=salessrv;UID=igor;PWD=password;DATABASE=sales;
```

Для баз данных ODBC свойство **База данных-источник (SourceDatabase)** определять не требуется.

Дополнительные сведения по доступу к базам данных ODBC см. в главе 18 руководства *Разработка приложений для Microsoft Access 97.*

Свойство «Уникальные записи» (UniqueRecords)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproUniqueRecordsC;dasqlAllDistinct"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproUniqueRecordsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproUniqueRecordsA"}

Свойство **Уникальные записи (UniqueRecords)** определяет, следует ли возвращать из источника данных только записи, содержащие уникальные наборы значений полей, или все записи.

Примечание. Свойство **Уникальные записи (UniqueRecords)** применяется ко всем запросам, за исключением запросов на добавление, на создание таблицы и на выборку.

Значения

Свойство **Уникальные записи (UniqueRecords)** может иметь следующие значения.

Значение	Описание
Да	Не возвращаются повторяющиеся записи.
Нет	(Значение по умолчанию). Возвращаются повторяющиеся записи.

Значение свойства **Уникальные записи (UniqueRecords)** задается в окне свойств запроса или в режиме SQL в окне запроса.

Примечание. В инструкции SQL значению свойства **Уникальные записи (UniqueRecords)** соответствует предикат DISTINCTROW.. Свойству **Уникальные значения (UniqueValues)** соответствует предикат DISTINCT.

Дополнительные сведения

Свойство **Уникальные записи (UniqueRecords)** используют в тех случаях, когда из результата запроса требуется исключить повторяющиеся записи, а не только повторяющиеся поля. В Microsoft Access повторяющимися считаются записи, у которых совпадают значения во всех полях.

Свойство **Уникальные записи (UniqueRecords)** влияет на результат выполнения запроса только в том случае, если в запросе используется несколько таблиц, причем в запрос включены поля из нескольких таблиц. Если в запросе используется одна таблица, свойство **Уникальные записи (UniqueRecords)** игнорируется.

Свойства **Уникальные записи (UniqueRecords)** и **Уникальные значения (UniqueValues)** не могут одновременно иметь значения «Да». Например, при значении «Да» свойства **Уникальные записи** автоматически устанавливается значение «Нет» свойства **Уникальные значения**. Однако оба свойства могут одновременно иметь значение «Нет». В этом случае возвращаются все записи.

Свойство «Уникальные записи» (UniqueRecords), пример

Запрос, приведенный в данном примере, возвращает из таблицы «Клиенты» список клиентов, разместивших хотя бы один заказ, сведения о котором хранятся в таблице «Заказы».

Таблица «Клиенты»

Название	Код клиента
Ирбис	IRBI
Клуб гурманов	KLUG
Оазис	OAZS
Варшава	WARS

Таблица «Заказы»

Код клиента	Заказ
IRBI	10698
KLUG	10512
KLUG	10725
WARS	10763
WARS	10408

Результатом выполнения следующей инструкции SQL будет приведенная ниже таблица.

```
SELECT DISTINCTROW Клиенты.Название, Клиенты.КодКлиента
FROM Клиенты INNER JOIN Заказы
ON Клиенты.КодКлиента = Заказы.КодКлиента;
```

Название	Код клиента
Ирбис	IRBI
Клуб гурманов	KLUG
Варшава	WARS

Свойство «Уникальные значения» (UniqueValues)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproUniqueValuesC;dasqlAllDistinct"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproUniqueValuesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproUniqueValuesA"}

Свойство **Уникальные значения (UniqueValues)** определяет, следует ли опускать записи, содержащие повторяющиеся значения в полях, выводящихся в режиме таблицы. Например, если в запросе выводятся несколько полей, то каждая выводящаяся запись должна содержать уникальное сочетание значений этих полей.

Примечание. Свойство **Уникальные значения (UniqueValues)** применяется ко всем запросам, за исключением запросов на добавление, на создание таблицы и на выборку.

Значения

Свойство **Уникальные значения (UniqueValues)** может принимать следующие значения.

Значение	Описание
Да	Возвращает только те записи, у которых значения всех полей, выводящихся в режиме таблицы, являются уникальными.
Нет	(Значение по умолчанию). Возвращает все записи.

Значение свойства **Уникальные значения (UniqueValues)** задается в окне свойств запроса или в режиме SQL в окне запроса.

Примечание. В инструкции SQL значению свойства **Уникальные значения (UniqueValues)** соответствует предикат DISTINCT. Свойству **Уникальные записи (UniqueRecords)** соответствует предикат DISTINCTROW.

Дополнительные сведения

При значении «Да» свойства **Уникальные значения (UniqueValues)** набор записей, полученных в результате выполнения этого запроса, является необновляемым и не отражает изменения, внесенные в базовые таблицы другими пользователями.

Свойства **Уникальные значения (UniqueValues)** и **Уникальные записи (UniqueRecords)** не могут одновременно иметь значения «Да». Например, при значении «Да» свойства **Уникальные значения** автоматически устанавливается значение «Нет» для свойства **Уникальные записи**. Однако оба свойства могут одновременно иметь значение «Нет». В этом случае возвращаются все записи.

Совет. Для подсчета числа повторений конкретного значения в поле следует создать итоговый запрос.

Свойство «Уникальные значения» (UniqueValues), пример

Инструкция SELECT, приведенная в данном примере, возвращает список названий стран, в которых расположены фирмы клиентов. Поскольку в любой стране могут находиться несколько клиентов, название каждой страны может встретиться в таблице «Клиенты» несколько раз. Однако в результате запроса каждая страна будет представлена только один раз.

В данном примере используется таблица «Клиенты», которая содержит следующие данные.

Страна	Название
Россия	Ирбис
Россия	Клуб гурманов
Россия	Приятного аппетита!
Украина	Оазис
Украина	Карбованец
Армения	Ван
Грузия	Сулико

Следующая инструкция SQL возвращает приведенную ниже таблицу.

```
SELECT DISTINCT Клиенты.Страна
FROM Клиенты;
```

Результат
Россия
Украина
Армения
Грузия

Свойства Form, Report

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFormC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproFormX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproFormA"}

Свойства **Form** и **Report** используются для ссылок на форму или отчет, а также для ссылок на форму или отчет, связанные с элементом управления подчиненной формой или подчиненным отчетом.

Значения

Данные свойства содержат ссылку на объект-форму или объект-отчет. Они доступны только для чтения во всех режимах.

Эти свойства используются в макросе или в программе Visual Basic.

Дополнительные сведения

Данные свойства позволяют ссылаться на форму или отчет, содержащиеся в элементе управления-подчиненная форма/отчет. Например, в следующей конструкции свойство **Form** обеспечивает доступ к полю «Заказ» в подчиненной форме «Заказано».

```
Dim intOrderID As Integer
intOrderID = Forms!Заказы!Заказано.Form!КодЗаказа
```

В следующем примере свойство **Form** используется для ссылки на активную форму, содержащую элемент управления «КодКлиента».

```
=MyFunction(Form!КодКлиента)
```

В подобных конструкциях свойство **Form** определяет активную форму, поэтому можно ссылаться на форму, не зная ее имени.

Ниже приводится пример эквивалентной конструкции Visual Basic.

```
X = MyFunction(Forms!Клиенты!КодКлиента)
```

Примечание. При ссылках на форму или отчет через семейство **Forms** или **Reports** необходимо указывать имя формы или отчета.

Свойства Form, Report, пример

В следующем примере свойства **Form** и **Report** используются для ссылок на элементы управления в подчиненной форме и в подчиненном отчете.

```
Dim curTotalAmount As Currency
Dim curTotalSales As Currency
curTotalAmount = Forms!Заказы!Заказано.Form!Итого
curTotalSales = Reports!Sales!Сотрудники.Report!ОбъемПродаж
```

Свойства «Подчиненные поля» (LinkChildFields), «Основные поля» (LinkMasterFields)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproLinkChildMasterFieldsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproLinkChildMasterFieldsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproLinkChildMasterFieldsA"}

Свойства **Подчиненные поля (LinkChildFields)** и **Основные поля (LinkMasterFields)** определяют связи, устанавливаемые между записями главной формы или отчета и записями подчиненного объекта (подчиненной формы, подчиненного отчета или внедренного объекта, например, диаграммы). Если эти свойства определены, Microsoft Access автоматически обновляет связанные записи в подчиненном объекте при изменении записей в главной форме или отчете.

Значения

Значения свойств **Подчиненные поля (LinkChildFields)** и **Основные поля (LinkMasterFields)** подчиненной формы, подчиненного отчета или внедренного объекта определятся в окне свойств следующим образом.

- **Подчиненные поля (LinkChildFields)** – Введите имена одного или нескольких связанных полей подчиненной формы, подчиненного отчета или внедренного объекта.
- **Основные поля (LinkMasterFields)** – Введите имена одного или нескольких связанных полей или элементов управления главной формы или отчета.

Определить значения этих свойств позволяет мастер по связыванию полей, для запуска которого следует нажать кнопку построителя ... справа от ячейки в окне свойств.

Кроме того, значения этих свойств могут быть определены с помощью строкового выражения в макросе или в программе Visual Basic.

Значения данных свойств задаются только в режиме конструктора или во время события **Открытие (Open)** формы или отчета.

Дополнительные сведения

Поля или элементы управления, используемые для связи, не обязаны иметь одинаковые имена, однако, должны содержать однотипные данные и иметь совместимые типы и размеры. Например, поле счетчика совместимо с числовым полем, у которого свойство **Размер поля** (**FieldSize**) имеет значение **Длинное целое.**

В качестве значения свойства **Основные поля (LinkMasterFields)** можно указать имя элемента управления (в том числе, вычисляемого). Не допускается использование имени элемента управления в качестве значения свойства **Подчиненные поля (LinkChildFields)**. Если для связи между объектами следует использовать вычисляемое значение, определите вычисляемое поле в базовом запросе подчиненного объекта и укажите имя этого поля в свойстве **Подчиненные поля**.

При указании нескольких связанных полей или элементов управления в значениях данных свойств их имена следует разделять символом точки с запятой (;).

При создании подчиненного объекта путем переноса одной формы или отчета из окна базы данных в другую форму или отчет с помощью мыши или при создании составной формы с помощью мастера Microsoft Access пытается установить связь между двумя формами. В большинстве случаев это удается. Microsoft Access автоматически задает значения свойств **Подчиненные поля (LinkChildFields)** и **Основные поля (LinkMasterFields)** при выполнении следующих условий.

- Источниками данных для обоих объектов являются таблицы (а не запросы), между которыми предварительно установлена связь с помощью команды **Схема данных**. В качестве связанных полей используются поля, по которым связываются эти таблицы.

- Источником данных для главной формы или отчета является таблица, в которой определен ключ, а источником данных для подчиненного объекта служит таблица, которая содержит поле с тем же именем, что и ключ первой таблицы, и с тем же или совместимым типом данных. В этом случае в качестве связанных полей используются два поля с одинаковыми именами.

Примечание. Связанные поля не обязательно должны выводиться в главном или подчиненном объектах. Любые поля, включенные в структуру базовых таблиц или запросов, могут быть использованы для связи объектов. При использовании мастера связанные поля определятся автоматически.

Свойство «Имя» (Name)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproNameC;daproName"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproNameX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproNameA;vaobjReference"}

Значением свойства **Имя (Name)** является строковое выражение, определяющее имя формы, отчета, раздела или элемента управления. или объекта **Reference.**

Значения

Имя должно удовлетворять правилам наименования объектов Microsoft Access. Имена форм и отчетов должны содержать до 64. Максимальная длина имени элемента управления составляет 255 символов.

Для того чтобы определить значение свойства **Имя (Name)** для формы или отчета выберите в режиме конструктора в меню **Файл** команду **Сохранить** и введите допустимое имя. Для изменения имени формы или отчета выделите нужный объект в окне базы данных и выберите команду **Переименовать** в меню **Правка** объекта или снова выберите объект и введите новое имя. Переименовать объект позволяет также соответствующая команда контекстного меню объекта. Для изменения имени открытой формы или отчета выберите в меню **Файл** команду **Сохранить как**.

Для элемента управления или раздела значение данного свойства можно задать в окне свойств, в макросе или в программе Visual Basic. При определении свойства **Имя (Name)** для форм и отчетов допускается использование выражений.

Значение этого свойства задается только в режиме конструктора.

Примечание. Для объекта **Reference** данное свойство доступно во всех режимах только для чтения.

Дополнительные сведения

По умолчанию, новым формам и отчетам присваивается имя, состоящее из имени объекта, к которому добавляется уникальное число типа **Integer**. Например, первая новая форма получает имя «Форма1», если форма с именем «Форма1» уже существует, то следующая форма по умолчанию получит имя «Форма2» и т.д. Имя формы не должно совпадать с именем любого другого системного объекта, например, объекта **Screen.**

Для свободного элемента управления стандартное имя состоит из названия типа объекта и уникального числа типа **Integer**. Например, если первым элементом управления, добавленным в форму, является поле, то его свойство **Имя (Name)** получает значение «Поле0».

Для присоединенного элемента управления именем, задающимся по умолчанию, является имя базового источника данных. Если элемент управления создается путем переноса поля с помощью мыши из списка полей, то в свойство **Имя (Name)** копируется значение свойства **Имя поля** (**FieldName**).

Нельзя присвоить элементу управления имя «Form» или «Report».

Не допускается существование в одной форме или отчете двух элементов управления с одинаковыми именами, однако, в разных формах или отчетах элементы управления могут иметь одинаковые имена. Нельзя присвоить одинаковые имена элементу управления и разделу в одной форме.

Свойство Parent

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproParentC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproParentX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproParentA"}

Свойство **Parent** возвращает ссылку на основной объект, содержащий элемент управления или раздел, или ссылку на элемент управления, содержащий другие элементы управления. Свойство **Parent** возвращает элемент управления, если элемент управления содержится в другом элементе управления; оно возвращает объект форму или объект отчет, если элемент управления содержится в форме или в отчете.

Значения

Свойство **Parent** содержит ссылку на объект форму, отчет или элемент управления. Это свойство доступно только для чтения во всех режимах.

Дополнительные сведения

Свойство **Parent** позволяет определить, какая форма или какой отчет является основным объектом, если подчиненная форма или подчиненный отчет были вставлены в несколько форм или отчетов.

Предположим, например, что подчиненная форма «Заказано» вставлена в форму и отчет. В следующей конструкции свойство **Parent** позволяет определить ссылку на поле «Заказ», включенное и в главную форму, и в главный отчет. Это выражение может быть введено в окно свойств присоединенного элемента управления, размещенного в подчиненной форме:

```
=Parent!КодЗаказа
```

Значением свойства **Parent** элемента управления-надписи является имя элемента управления, к которому присоединена надпись. Значением свойства **Parent** флажка, выключателя или переключателя, помещенного в рамку группы, является имя элемента управления-группы. Для группы значением свойства **Parent** является имя формы.

Свойство Parent, пример

В данном примере свойство **Parent** используется для определения основного элемента управления надписи **Надпись1**, флажка **Флажок2** и группы **СпособДоставки**.

```
Dim frm As Form
Dim ctl As Control
Set frm = Forms!Заказы
Set ctl = frm.[Надпись1]
' Возвращает имя элемента управления, связанного с надписью.
MsgBox "Имя основного элемента управления: " & ctl.Parent.Name
Set ctl = frm.Флажок2
' Возвращает имя элемента управления, содержащего флажок.
MsgBox "Имя основного элемента управления: " & ctl.Parent.Name
Set ctl = frm.СпособДоставки
' Возвращает имя формы, содержащей группу.
MsgBox "Имя основного элемента управления: " & ctl.Parent.Name
```

Следующая конструкция также возвращает имя формы, содержащей группу.

```
MsgBox Forms!Заказы!_
   Надпись1.Parent.Parent.Parent.Name
```

Свойство Section

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproSectionC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproSectionX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproSectionA"}

Свойство **Section** определяет раздел формы или отчета и обеспечивает доступ к свойствам данного раздела. Кроме того, это свойство позволяет определить раздел формы или отчета, в котором содержится элемент управления.

Значения

Значением свойства **Section** является число типа **Integer,** соответствующее конкретному разделу. При этом свойству присваивают значение типа **Integer** или одну из перечисленных ниже констант. Рекомендуется использовать константы, чтобы программы были более понятными.

Значения	Константа	Описание
0	acDetail	Область данных формы или область данных отчета.
1	acHeader	Заголовок формы или отчета.
2	acFooter	Примечание формы или отчета.
3	acPageHeader	Верхний колонтитул формы или отчета.
4	acPageFooter	Нижний колонтитул формы или отчета.
5	acGroupLevel1Header	Заголовок группы первого уровня (только отчеты).
6	acGroupLevel1Footer	Примечание группы первого уровня (только отчеты).
7	acGroupLevel2Header	Заголовок группы второго уровня (только отчеты).
8	acGroupLevel2Footer	Примечание группы второго уровня (только отчеты).

Если отчет содержит дополнительные уровни группировки, то пары заголовков/примечаний групп нумеруются последовательно, начиная с 9.

Свойство **Section** доступно только для чтения во всех режимах.

Для форм и отчетов значение свойства **Section** является массивом, в который записываются все существующие разделы формы или отчета, определяемые по номерам. Например, `Section(0)` является ссылкой на область данных формы, а `Section(3)` на область заголовка формы.

Кроме того, допускаются ссылки на разделы по их именам. Следующие конструкции, содержащие ссылку на область данных формы «Клиенты» являются эквивалентными.

```
Forms!Клиенты.Section(acDetail).Visible

Forms!Клиенты.Section(0).Visible

Forms!Клиенты.ОбластьДанных.Visible
```

Дополнительные сведения

В случае форм и отчетов, свойство **Section** можно использовать только вместе со свойствами разделов формы или отчета. В следующем примере демонстрируется ссылка на свойство **Вывод на экран** (**Visible**) верхнего колонтитула формы «Клиенты».

```
Forms!Клиенты.Section(acPageHeader).Visible

Forms!Клиенты.Section(3).Visible
```

В случае элементов управления свойство **Section** позволяет определить раздел формы или отчета, в котором содержится данный элемент управления. В следующем примере определяется раздел, в котором находится поле «КодКлиента»:

```
Dim intSectionNumber As Integer
intSectionNumber = Forms!Клиенты!КодКлиента.Section
```

Свойство «Объект-источник» (SourceObject)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproSourceObjectC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproSourceObjectX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproSourceObjectA"}

Свойство **Объект-источник (SourceObject)** определяет форму или отчет, являющиеся источником данных, выводящихся в элементе управления-подчиненной форме или подчиненном отчете. Кроме того, для связанного объекта в свободной рамке объекта это свойство позволяет определить полный путь и имя файла, содержащего данные, выводящиеся в рамке объекта.

Значения

Имя формы или отчета, являющихся источником данных, выводящихся в элементе управления-подчиненной форме или подчиненном отчете, вводится в окно свойств элемента управления. Если элемент управления-подчиненная форма/отчет добавляется в форму или отчет путем переноса из окна базы данных с помощью мыши, значение свойства **Объект-источник (SourceObject)** автоматически вводится в окно свойств.

Для свободной рамки объекта, связанной с внешним файлом, значение свойства **Объект-источник (SourceObject)** задается автоматически при вставке связанного объекта OLE с помощью команды **Объект** из меню **Вставка**.

Для подчиненных форм и подчиненных отчетов значение этого свойства задается в окне свойств, в макросе или в программе Visual Basic.

В программе Visual Basic значение этого свойства определяется с помощью строкового выражения, содержащего имя формы или отчета.

Для связанного объекта в свободной рамке объекта задать значение свойства **Объект-источник (SourceObject)** нельзя в любом режиме.

Примечание. Не допускается задание или изменение значения свойства **Объект-источник (SourceObject)** в событиях **Открытие (Open)** или **Форматирование (Format)** отчета.

Дополнительные сведения

Если значение свойства **Объект-источник (SourceObject)** удаляется из окна свойств подчиненной формы или подчиненного отчета, то элемент управления подчиненная форма/отчет остается в форме, но становится не связанным с источником данных.

Свойство «Объект-источник» (SourceObject), пример

В следующем примере в окно отладки выводится имя формы, являющейся источником данных для элемента управления-подчиненной формы.

```
Debug.Print Forms!Клиенты!
   Заказано.SourceObject
```

Свойство ActiveControl

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproActiveControlC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproActiveControlX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproActiveControlA"}

Свойство **ActiveControl**, используемое вместе с объектом **Screen,** задает или возвращает ссылку на элемент управления, обладающий фокусом.

Значения

Данное свойство содержит ссылку на объект **Control,** имеющий фокус во время выполнения.

Это свойство доступно в макросе или программе Visual Basic и допускает только чтение во всех режимах.

Дополнительные сведения

Свойство **ActiveControl** позволяет во время выполнения создать ссылку на элемент управления, имеющий фокус, для использования его методов или свойств. В следующем примере имя обладающего фокусом элемента управления присваивается переменной `strControlName`.

```
Dim ctlCurrentControl As Control
Dim strControlName As String
Set ctlCurrentControl = Screen.ActiveControl
strControlName = ctlCurrentControl.Name
```

Если при вызове свойства **ActiveControl** ни один из элементов управления не имеет фокуса, а также если все элементы управления активной формы являются скрытыми или отключенными, возникает ошибка.

Свойство ActiveControl, пример

В следующем примере активный элемент управления связывается с переменной `ctlCurrentControl`, после чего выполняются действия, зависящие от значения свойства **Имя (Name)** активного элемента.

```
Dim ctlCurrentControl As Control

Set ctlCurrentControl = Screen.ActiveControl
If ctlCurrentControl.Name = "fldКлиент" Then
   .         ' Действия для поля "Клиент".
   .
   .
ElseIf ctlCurrentControl.Name = "btnДополнительно" Then
   .         ' Действия для кнопки "Дополнительно".
   .
   .
End If
```

Свойство ActiveForm

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproActiveFormC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproActiveFormX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproActiveFormA"}

Свойство **ActiveForm**, используемое вместе с объектом **Screen,** задает или возвращает ссылку на форму, обладающую фокусом.

Значения

Данное свойство содержит ссылку на объект **Form,** имеющий фокус во время выполнения.

Это свойство доступно в макросе или программе Visual Basic и допускает только чтение во всех режимах.

Дополнительные сведения

Свойство **ActiveForm** позволяет во время выполнения создать ссылку на активную форму, имеющую фокус, для использования ее методов или свойств. Следующий пример выводит на экран значение свойства **Имя (Name)** активной формы.

```
Dim frmCurrentForm As Form
Set frmCurrentForm = Screen.ActiveForm
MsgBox "Текущая форма: " & frmCurrentForm.Name
```

Если фокус установлен на подчиненную форму, свойство **ActiveForm** возвращает ссылку на главную форму. Если при вызове свойства **ActiveForm** ни одна из главных или подчиненных форм не имеет фокуса, возникает ошибка.

Свойство ActiveReport

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅS":"acproActiveReportC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acproActiveReportX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproActiveReportA"}

Свойство **ActiveReport**, используемое вместе с объектом **Screen,** задает или возвращает ссылку на отчет, обладающий фокусом.

Значения

Данное свойство содержит ссылку на объект **Report,** имеющий фокус во время выполнения.

Это свойство доступно в макросе или программе Visual Basic и допускает только чтение во всех режимах.

Дополнительные сведения

Свойство **ActiveReport** позволяет во время выполнения создать ссылку на активный отчет, имеющий фокус, для использования его методов или свойств. Следующий пример выводит на экран значение свойства **Имя (Name)** активного отчета.

```
Dim rptCurrentReport As Report
Set rptCurrentReport = Screen.ActiveReport
MsgBox "Текущий отчет: " & rptCurrentReport.Name
```

Если при вызове свойства **ActiveReport** ни один отчет не имеет фокус, возникает ошибка.

Свойство Bookmark

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅS. пїЅпїЅпїЅпїЅпїЅS":"acproBookmarkC;daproBookmark;daproBookmarkable"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅS":"acproBookmarkX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproBookmarkA"}

Свойство **Bookmark** используется в формах для установки закладки, однозначно определяющей запись в базовой таблице, запросе или инструкции SQL формы.

Значения

Свойство **Bookmark** содержит строковое выражение, создаваемое Microsoft Access.

Доступ к данному свойству осуществляется в макросе или программе Visual Basic.

Примечание. Значение свойства **Bookmark** формы считывается или задается независимо от свойства объектов доступа к данным **Bookmark** базовой таблицы или запроса.

Дополнительные сведения

При открытии в режиме формы формы, связанной с базовой таблицей, для каждой записи автоматически создается уникальная закладка. В программе Visual Basic имеется возможность сохранить закладку текущей записи, для чего следует присвоить строковой переменной значение свойства формы **Bookmark**. Чтобы вернуться на эту запись после переходов по записям формы, следует присвоить свойству **Bookmark** значение закладки, сохраненной в переменной. Для сравнения строковой переменной или переменной **Variant** с закладкой, а также при сравнении двух закладок, используется функция **StrComp.** Третий аргумент этой функции должен иметь нулевое значение.

Примечание. Закладки не сохраняются вместе с записями, которые они представляют, и существуют, только пока форма открыта. При каждом открытии формы, связанной с источником данных, закладки создаются заново.

Количество создаваемых закладок не ограничено, если каждой закладке соответствует уникальная строковая переменная.

Свойство **Bookmark** является доступным только для текущей записи формы. Для сохранения закладки, которая соответствует записи, не являющейся текущей, следует перейти к нужной записи, а затем присвоить текущее значение свойства **Bookmark** строковой переменной, предназначенной для хранения этой закладки.

Закладки можно использовать в любой форме, в основе которой лежат только таблицы Microsoft Access, поскольку не все СУБД поддерживают использование закладок. Например, нельзя использовать закладки в форме, источником данных которой является связанная таблица, в которой не определен ключ.

После повторного выполнения базового запроса формы все сохраненные закладки перестают соответствовать конкретным записям. Однако обновление формы с помощью команды **Обновить** в меню **Записи** не влияет на закладки.

Закладки, создаваемые Microsoft Access, однозначно определяют каждую запись в наборе записей, который отбирается при открытии формы, поэтому эти закладки являются неприменимыми к другому набору записей, даже если оба набора записей создаются на основе одной таблицы, запроса или инструкции SQL. Предположим, например, что открывается форма, связанная с таблицей «Клиенты». Если затем таблица «Клиенты» открывается в программе Visual Basic и проводится поиск конкретной записи в этой таблице с помощью метода **Seek,** то нельзя указать запись таблицы в качестве значения свойства формы **Bookmark**. Подобные операции выполняются с помощью метода **FindFirst** и свойства формы **RecordsetClone.**

Если запись, закладка которой сохранена в строковой переменной, удаляется, то при попытке

обращения к этой записи с помощью сохраненного значения свойства **Bookmark** возникает перехватываемая ошибка.

Значение свойства **Bookmark** не совпадает с номером записи.

Свойство Bookmark, пример

Для проверки следующего примера с помощью учебной базы данных «Борей» необходимо добавить кнопку «Поиск контактов» в форму «Поставщики», а затем к ее процедуре обработки события «Нажатие клавиши» (OnClick) добавить приведенную ниже программу. При нажатии этой кнопки пользователю будет выведено приглашение ввести часть искомого имени. Если имя будет найдено, свойству формы **Bookmark** присваивается значение **Recordset** свойства объектов доступа к данным **Bookmark**, которое перемещает текущую запись формы к найденному имени.

```
Private Sub cmdПоискКонтактов_Click()
   Dim rst As Recordset, strCriteria As String
   strCriteria = "[ОбращатьсяК] Like '*" & InputBox("Введите " _
      & "первые буквы искомого имени") & "*'"

   Set rst = Me.RecordsetClone
   rst.FindFirst strCriteria
   If rst.NoMatch Then
      MsgBox "Записи не найдены"
   Else
      Me.Bookmark = rst.Bookmark
   End If
End Sub
```

Свойства CurrentObjectType и CurrentObjectName

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCurrentObjectTypeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproCurrentObjectTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproCurrentObjectTypeA"}

Свойства **CurrentObjectType** и **CurrentObjectName**, используемые с объектом **Application,** позволяют определить тип активного объекта базы данных (таблица, запрос, форма, отчет, макрос или модуль) и его имя. Активным называется объект, имеющий фокус, или тот, в котором выполняется программа.

Значения

Microsoft Access автоматически присваивает свойству **CurrentObjectType** значение одной из следующих встроенных констант.

Значение	Описание
acTable	Активный объект является таблицей.
acQuery	Активный объект является запросом.
acForm	Активный объект является формой.
acReport	Активный объект является отчетом.
acMacro	Активный объект является макросом.
acModule	Активный объект является модулем.

Свойству **CurrentObjectName** автоматически присваивается строковое выражение, содержащее имя активного объекта.

Значения этих свойств доступны только в программах Visual Basic и во всех режимах допускают только чтение.

Дополнительные сведения

Ниже перечислены условия, определяющие, какой объект считается активным.

- Если активным объектом является окно свойств, панель команд, меню, палитра или список полей объекта, то свойства **CurrentObjectType** и **CurrentObjectName** возвращают тип и имя базового объекта.
- Если активным объектом является всплывающая форма, то свойства **CurrentObjectType** и **CurrentObjectName** определяют имя и тип всплывающей формы, а не формы, из которой она была открыта.
- Если активным объектом является окно базы данных, то свойства **CurrentObjectType** и **CurrentObjectName** возвращают тип и имя объекта, выделенного в окне базы данных.
- Если не выделен ни один объект, свойство **CurrentObjectType** возвращает значение **True** (–1), а в свойстве **CurrentObjectName** возвращается пустая строка ("").
- Если текущее состояние не определено (активный объект не является таблицей, запросом, формой, отчетом, макросом или модулем) – например, если фокус имеет диалоговое окно – свойство **CurrentObjectType** возвращает значение **True**, а свойство **CurrentObjectName** возвращает имя диалогового окна.

Вызываемые вместе с функцией **SysCmd** эти свойства позволяют определить активный объект и его состояние (например, является ли объект открытым, новым или содержит несохраненные изменения).

Свойства CurrentObjectType и CurrentObjectName, пример

В следующем примере значения свойств **CurrentObjectType** и **CurrentObjectName** подставляются в функцию **SysCmd** для проверки того, является ли активным объектом форма «Товары» и содержит ли она несохраненные изменения. Если да, то форма сохраняется и закрывается.

```
Sub CheckProducts()
   Dim intState As Integer
   Dim intCurrentType As Integer
   Dim strCurrentName As String

   intCurrentType = Application.CurrentObjectType
   strCurrentName = Application.CurrentObjectName
   If intCurrentType = acForm And strCurrentName = "Товары" Then
      intState = SysCmd(acSysCmdGetObjectState, intCurrentType,
strCurrentName)
      ' Форма "Товары" содержит несохраненные изменения.
      If intState = acObjStateDirty + _
         acObjStateOpen Then
         ' Закрытие формы "Товары" с сохранением изменений.
         DoCmd.Close intCurrentType, _
            strCurrentName, acSaveYes
      End If
   End If
End Sub
```

Свойство Me

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproMeC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproMeX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproMeA"}

Свойство **Me** используется в программах Visual Basic для ссылок на форму, отчет, форму или отчет, связанные с подчиненной формой или подчиненным отчетом, а также на модуль класса, в котором выполняется программа Visual Basic.

Значения

Свойство **Me** доступно только в программах Visual Basic и допускает только чтение во всех режимах.

Дополнительные сведения

Свойство **Me** содержит ссылку объекта на активную форму или отчет и работает быстрее, чем полная ссылка на объект. Например, следующие два фрагмента программы ссылаются на значение элемента управления «Фамилия» для текущей записи формы «Сотрудники».

```
strLastName = Forms!Сотрудники.Фамилия
```

```
strLastName = Me!Фамилия
```

В большинстве случаев свойство **Me** возвращает ссылку на ту же форму или отчет, что и свойства **ActiveForm** или **ActiveReport** объекта **Screen.** Однако эти свойства возвращают ссылку на форму или отчет, которые обладают фокусом, в то время, как свойство **Me** всегда содержит ссылку на форму или отчет, в которых выполняется программа. Например, при возникновении события **Таймер (Timer)** в форме, не обладающей фокусом, конструкция `Screen.ActiveForm` возвратит ссылку на форму, обладающую фокусом, а свойство **Me** возвратит ссылку на форму, в которой возникло событие **Таймер (Timer)**. Поэтому при создании внутренних процедур, предназначенных для обработки активной формы, вместо использования свойства **ActiveForm** удобнее передать форму в процедуру с помощью свойства **Me**.

Ключевое слово **Me** можно использовать в модуле класса для ссылки на текущую копию этого класса. Аналогично использованию **Me** в модулях класса формы или отчета для возвращения ссылки на текущую форму или отчет, можно использовать **Me** в модуле класса для возвращения ссылки на текущий объект модуля класса.

Свойство Me, пример

В следующем примере показано изменение заголовка формы «Клиенты» на значение, предложенное пользователем. Вызывается процедура `ChangeCaption`, в которую с помощью свойства **Me** (представляющего объект **Form**) передается имя активной формы. Поскольку процедура ChangeCaption изменяет заголовок активной формы (а не какой-либо определенной), эту процедуру можно вызывать из другой процедуры в том же приложении.

```
ChangeCaption Me, InputBox("Введите новый заголовок "
   & "формы")

Sub ChangeCaption(frmCurrentForm As _
   Form, strCaption As String)
' Изменение заголовка.
   If TypeName(frmCurrentForm.Caption) = "Null" Then
      frmCurrentForm.Caption = strCaption
      Exit Sub
   End If
   If UCase(strCaption) _
         <> UCase(frmCurrentForm.Caption) Then
      frmCurrentForm.Caption = strCaption
   End If
End Sub
```

Свойство Module

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproModuleC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproModuleX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproModuleA"}

Свойство **Module** позволяет указать модуль формы или модуль отчета.

Значения

Свойство **Module** доступно только в программах Visual Basic и допускает только чтение во всех режимах.

Примечание. Свойство **Module** также возвращает ссылку на указанный объект **Module.**

Дополнительные сведения

Свойство **Module** позволяет осуществлять доступ к свойствам и методам объекта **Module**, связанного с объектом **Form** или **Report**.

Значение свойства **Наличие модуля (HasModule)** для формы или отчета определяет, имеет ли форма или отчет соответствующий модуль. Если для этого свойства задано значение **Нет (False)**, то форма или отчет не имеют модуля. При ссылке в режиме конструктора на свойство **Module** этой формы или отчета Microsoft Access создает соответствующий модуль и задает для свойства **Наличие модуля (HasModule)** значение **Да (True)**. Ссылка на свойство **Module** формы или отчета во время выполнения при заданном значении **Нет (False)** для свойства объекта **Наличие модуля (HasModule)** приведет к возникновению ошибки.

Это свойство может использоваться со всеми свойствами и методами объекта-модуля.

Свойство Module, пример

В следующем примере свойство **Module** используется для вставки метода **Beep** в процедуру обработки события формы **Открытие (Open)**.

```
Dim strFormOpenCode As String
Dim mdl As Module

Set mdl = Forms!Форма1.Module
strFormOpenCode = "Sub Form_Open(Cancel As Integer)" _
   & vbCrLf & "Beep" & vbCrLf & "End Sub"
   With mdl
      .InsertText strFormOpenCode
   End With
```

Свойство RecordsetClone

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ.
пїЅпїЅпїЅпїЅпїЅпїЅS":"acproRecordsetCloneC;damthFindFirst;damthMoveFirst;daobjRecordset;daproBookmark"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acproRecordsetCloneX":1} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproRecordsetCloneA"}

Свойство **RecordsetClone** используется для ссылок на объект формы **Recordset,** указанный в свойстве формы **Источник записей (RecordSource).**

Значения

Значение свойства формы **RecordsetClone** определяет копию набора записей базовой таблицы или запроса, указанных в свойстве формы **Источник записей (RecordSource)**. Например, если форма основана на запросе, то обращение к свойству **RecordsetClone** эквивалентно созданию копии объекта **Recordset** с помощью того же запроса. Если к форме будет дополнительно применен фильтр, то объект **Recordset** создается с учетом фильтра.

Свойство **RecordsetClone** доступно только в программах Visual Basic и допускает только чтение во всех режимах.

Дополнительные сведения

Свойство **RecordsetClone** позволяет выполнять перемещение по записям или изменение записей формы без изменения самой формы. Например, свойство **RecordsetClone** используется для вызова методов, таких как **FindFirst,** которые нельзя использовать с формами.

При открытии нового объекта **Recordset** текущей записью является его первая запись. Если текущая запись определена с помощью одного из методов **Find** или **Move,** то для того, чтобы сделать текущей другую запись объекта **Recordset**, необходимо синхронизировать текущую запись объекта **Recordset** с текущей записью формы, присвоив значение свойства объектов доступа к данным **Bookmark** свойству формы **Bookmark.**

Для подсчета числа записей в объекте **Recordset** используется свойство **RecordCount.** В следующем примере приведен подсчет числа записей в форме с помощью комбинации свойств **RecordCount** и **RecordsetClone**.

```
Forms!Заказы.RecordsetClone.MoveLast
MsgBox "Форма1 содержит следующее число записей: " _
   & Forms!Заказы.RecordsetClone.RecordCount, _
   vbInformation, "Число записей"
```

Примечание. Если пользователь закроет форму или изменит значение свойства формы **Источник записей (RecordSource)**, прежний объект **Recordset** станет неопределенным. Последующие ссылки на этот объект или на закладки, сохраненные в форме или в этом объекте, приведут к возникновению ошибки.

Свойство RecordsetClone, пример

В данном примере свойство **RecordsetClone** используется для создания новой копии объекта **Recordset** на основе формы «Заказы» и последующего вывода имен полей в окне отладки.

```
Sub Print_Field_Names()
   Dim rst As Recordset, intI As Integer
   Dim fld As Field

   Set rst = Me.RecordsetClone
   For Each fld in rst.Fields
      ' Вывод имен полей.
      Debug.Print fld.Name
   Next
End Sub
```

В следующем примере свойство **RecordsetClone** и объект **Recordset** используются для синхронизации записи в наборе записей с текущей записью формы. После выбора названия организации в раскрывающемся списке метод **FindFirst** вызывается для поиска записи, относящейся к этой организации, после чего значение свойства **Bookmark** объекта **Recordset** присваивается свойству **Bookmark** формы. В результате, в форме выводится найденная запись.

```
Sub КодПоставщика_AfterUpdate()
   Dim rst As Recordset
   Dim strSearchName As String

   Set rst = Me.RecordsetClone
   strSearchName = Str(Me!КодПоставщика)
   rst.FindFirst "КодПоставщика = " & strSearchName
      If rst.NoMatch Then
         MsgBox "Записи не найдены"
      Else
         Me.Bookmark = rst.Bookmark
      End If
   rst.Close
End Sub
```

Свойство «Неразрывная группа» (GrpKeepTogether)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproGrpKeepTogetherC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproGrpKeepTogetherX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproGrpKeepTogetherA"}

Свойство **Неразрывная группа (GrpKeepTogether)** определяет удерживание на странице или в столбце групп в отчете, у которых для свойства **Не разрывать** (**KeepTogether**) заданы значения «Полную группу» или «Первую область данных».

Значения

Свойство **Неразрывная группа (GrpKeepTogether)** может иметь следующие значения.

Значение	Описание	Visual Basic
На странице	Группы удерживаются целиком на странице.	0
В столбце	(Значение по умолчанию). Группы удерживаются целиком в столбце.	1

Значение свойства **Неразрывная группа (GrpKeepTogether)** задается в окне свойств отчета, в макросе или в программе Visual Basic.

Данное свойство можно задать только в режиме конструктора отчета.

Дополнительные сведения

Данное свойство позволяет указать, должно ли все содержимое группы выводиться в одном столбце. Например, если имеется список сотрудников с группировкой по отделам, выводящийся в несколько столбцов, данное свойство позволяет помещать все фамилии сотрудников из каждого отдела в отдельный столбец.

Если свойство **Не разрывать (KeepTogether)** принимает значение «Нет», то значение свойства **Неразрывная группа (GrpKeepTogether)** не учитывается.

Свойство HasContinued

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproHasContinuedC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproHasContinuedX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproHasContinuedA"}

Свойство **HasContinued** определяет, печатается ли часть текущего раздела на предыдущей странице.

Значения

Свойство **HasContinued** задается Microsoft Access и доступно только для чтения во всех режимах.

Значение	Описание
True (–1)	Часть текущего раздела печатается на предыдущей странице.
False (0)	На предыдущей странице текущий раздел не печатается.

Значение свойства **HasContinued** проверяется в макросе или в программе Visual Basic.

Дополнительные сведения

Данное свойство используется для управления выводом некоторых элементов управления. Например, можно поместить в верхнем колонтитуле скрытую надпись «(Продолжение с предыдущей страницы)» и при значении **True** свойства **HasContinued** сделать эту надпись видимой.

Свойство HasData

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproHasDataC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproHasDataX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproHasDataA"}

Свойство **HasData** позволяет определить, связаны ли форма или отчет с пустым набором записей.

Значения

Свойство **HasData** задается Microsoft Access и доступно только для чтения во всех режимах. Значение данного свойства может быть считано только при печати или в режиме предварительного просмотра.

Значение	Описание
–1	Объект содержит данные.
0	Объект не содержит данных.
1	Объект является свободным.

Значение свойства **HasData** проверяется в макросе или в программе Visual Basic.

Дополнительные сведения

Свойство **HasData** позволяет сделать скрытым подчиненный отчет, не содержащий данных. Например, следующее выражение делает скрытым элемент управления-подчиненный отчет, если присоединенный к нему отчет не содержит данных.

```
Me!имяЭлемента.Visible = Me!имяЭлемента.Report.HasData
```

Свойство «Повторение раздела» (RepeatSection)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproRepeatSectionC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproRepeatSectionX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproRepeatSectionA"}

Свойство **Повторение раздела (RepeatSection)** определяет, следует ли повторять печать заголовков групп на следующей странице или столбце при печати группы на нескольких страницах или столбцах.

Примечание. Свойство **Повторение раздела (RepeatSection)** применимо только для заголовков группы в отчете.

Значения

Свойство **Повторение раздела (RepeatSection)** может иметь следующие значения.

Значения	Описание	Visual Basic
Да	Заголовок группы повторяется.	**True** (–1)
Нет	(Значение по умолчанию). Заголовок группы не повторяется.	**False** (0)

Значение свойства **Повторение раздела (RepeatSection)** задается в окне свойств раздела, в макросе или в программе Visual Basic.

Дополнительные сведения

При печати отчета, содержащего подчиненный отчет, свойство подчиненного отчета **Повторение раздела (RepeatSection)** определяет, повторятся ли заголовки групп подчиненного отчета на следующей странице или столбце.

Свойство WillContinue

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproWillContinueC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproWillContinueX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproWillContinueA"}

Свойство **WillContinue** определяет, печатается ли часть текущего раздела на следующей странице.

Значения

Свойство **WillContinue** задается Microsoft Access и доступно только для чтения во всех режимах.

Значение	Описание
True (–1)	Текущий раздел продолжается на следующей странице.
False (0)	Текущий раздел не продолжается на следующей странице.

Значение свойства **WillContinue** проверяется в макросе или в программе Visual Basic.

Дополнительные сведения

Данное свойство используется для управления выводом некоторых элементов управления. Например, можно поместить в нижнем колонтитуле скрытую надпись «(Продолжение на следующей странице)» и при значении **True** свойства **WillContinue** сделать эту надпись видимой.

Свойство ColumnHidden

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproColumnHiddenC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproColumnHiddenX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproColumnHiddenA;daobjQueryDef;daobjTableDef"}

Свойство **ColumnHidden** позволяет вывести или скрыть указанный столбец в режиме таблицы. Например, может оказаться удобным скрыть поле с адресом клиента, чтобы на экран умещались поля названия и телефона.

Примечание. Свойство **ColumnHidden** применяется ко всем полям в режиме таблицы и к элементам управления в формы в режиме таблицы.

Значения

Значение свойства **ColumnHidden** задается командами **Скрыть столбцы** и **Отобразить столбцы** в меню **Формат** режима таблицы.

Кроме того, значение данного свойства можно задать в программе Visual Basic с помощью значения типа **Long Integer.**

Значение	Описание
True (–1)	Столбец не выводится на экран.
False (0)	(Значение по умолчанию) Столбец выводится на экран.

Для того, чтобы задать или изменить значение данного свойства для таблицы или запроса в программе Visual Basic, следует использовать семейство **Properties**. Более подробное описание см. в разделе Объект **Property**, семейство **Properties**. Общие сведения.

Примечание. Свойство **ColumnHidden** недоступно в режиме конструктора.

Дополнительные сведения

Скрытие столбца при помощи свойства **ColumnHidden** в режиме таблицы не приводит к удалению этого поля с экрана в режиме формы. Аналогично, указание для свойства **Вывод на экран (Visible)** значения **False** в режиме формы не приводит к скрытию соответствующего столбца в режиме таблицы.

Поле может быть выведено в запросе, даже если соответствующий этому полю столбец не выводится на экран в режиме таблицы.

Значения из скрытых столбцов могут использоваться в условиях отбора в фильтре, даже если столбец остается скрытым после применения фильтра.

Для операций со скрытыми столбцами нельзя использовать команды **Копировать**, **Вставить**, **Найти** и **Заменить** из меню **Правка**.

Заданное для свойства поля **ColumnWidth** значение 0 или нулевая ширина столбца, установленная с помощью мыши в режиме таблицы, приводит к автоматическому заданию для свойства **ColumnHidden** значения **True.** Вывод столбца на экран возвращает значение, которое свойство **ColumnWidth** имело до скрытия поля.

Свойство ColumnHidden, пример

В данном примере поле «КодТовара» в форме «Товары» делается скрытым в режиме таблицы.

```
Forms!Товары!КодТовара.ColumnHidden = -1
```

В следующем примере также делается скрытым в режиме таблицы поле «КодТовара» в форме «Товары». Для определения свойства **ColumnHidden** используются процедуры `ShowColumn` и `SetFieldProperty`, определенные в стандартном модуле базы данных.

```
Dim dbs As Database
Set dbs = CurrentDb
ShowColumn dbs.TableDefs!Товары.Fields!КодТовара, False

Sub ShowColumn(fldObject As Field, intShow As Integer)
   ' Задает значение свойства ColumnHidden.
   SetFieldProperty fldObject, "ColumnHidden", dbLong, Not intShow
End Sub

Sub SetFieldProperty(fldField As Field, strPropertyName As String, _
      intPropertyType As Integer, varPropertyValue As Variant)
   ' Задает значение свойства поля без генерирования кода
невосстанавливаемой ошибки выполнения.
   Const conErrPropertyNotFound = 3270
   Dim prpProperty As Property
   On Error Resume Next            ' Не перехватывать ошибки.
   fldField.Properties(strPropertyName) = varPropertyValue
   If Err <> 0 Then                ' Ошибка при задании значения.
      If Err <> conErrPropertyNotFound Then
         On Error GoTo 0
         MsgBox "Не удается задать свойство '" & strPropertyName _
            & "' для поля '" & fldField.name & "'", 48, "SetFieldProperty"
      Else
         On Error GoTo 0
         Set prpProperty = fldField.CreateProperty(strPropertyName, _
            intPropertyType, varPropertyValue)
         fldField.Properties.Append prpProperty
      End If
   End If
End Sub
```

Свойство ColumnOrder

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproColumnOrderC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproColumnOrderX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproColumnOrderA;daobjQueryDef;daobjTableDef"}

Свойство **ColumnOrder** определяет порядок столбцов в режиме таблицы.

Примечание. Свойство **ColumnOrder** применяется ко всем полям таблицы в режиме таблицы и к элементам управления в форме в режиме таблицы.

Значения

Свойство **ColumnOrder** получает значение при выделении столбца в режиме таблицы и переносе его с помощью мыши в новое положение.

Также значение данного свойства можно задать в программе Visual Basic с помощью значения типа **Long Integer.**

Для того чтобы задать или изменить значение данного свойства для таблицы или запроса в программе Visual Basic, следует использовать семейство **Properties.** Более подробное описание см. в разделе Объект **Property**, семейство **Properties**. Общие сведения.

Примечание. Свойство **ColumnOrder** недоступно в форме в режиме конструктора.

Дополнительные сведения

В режиме таблицы значение свойства **ColumnOrder** определяется текущим положением поля. Крайнему левому столбцу в режиме таблицы соответствует значение 1, следующему – 2 и т.д. Изменение значения свойства **ColumnOrder** для одного поля влечет за собой изменение свойства **ColumnOrder** для всех полей, расположенных слева от исходного положения этого поля в режиме таблицы.

Во всех остальных режимах значение свойства **ColumnOrder** равняется 0, за исключением случая, когда пользователь явно изменит положение одного или нескольких полей в режиме таблицы (переместив их с помощью мыши или установив значения их свойств **ColumnOrder).** Во всех режимах, кроме режима таблицы, значение свойства **ColumnOrder** для полей, расположенных справа от нового положения перемещенного поля, равняется 0.

Порядок полей в режиме таблицы не влияет на порядок полей в режиме конструктора таблицы или в режиме формы.

Свойство ColumnOrder, пример

В данном примере поля «Марка» и «ЕдиницаИзмерения» формы «Товары» выводятся в режиме таблицы в первых двух столбцах.

```
Forms!Товары!Марка.ColumnOrder = 1
Forms!Товары!ЕдиницаИзмерения.ColumnOrder = 2
```

В следующем примере поля «Марка» и «ЕдиницаИзмерения» формы «Товары» также выводятся в режиме таблицы в первых двух столбцах. Для определения свойства **ColumnOrder** в этой программе используется процедура `SetFieldProperty`. Если она выполняется при открытой таблице, изменения не выводятся на экран до тех пор, пока эта таблица не будет закрыта и открыта заново.

```
Dim dbs As Database, tdfProducts As TableDef
Set dbs = CurrentDb
Set tdfProducts = dbs!Товары
SetFieldProperty tdfProducts!Марка, "ColumnOrder", dbLong, 2
SetFieldProperty tdfProducts!ЕдиницаИзмерения, "ColumnOrder", dbLong, 3

Sub SetFieldProperty(fldField As Field, strPropertyName As String, _
      intPropertyType As Integer, varPropertyValue As Variant)
   ' Задает значение свойства поля без генерирования кода
невосстанавливаемой ошибки выполнения.
   Const conErrPropertyNotFound = 3270
   Dim prpProperty As Property
   On Error Resume Next            ' Не перехватывать ошибки.
   fldField.Properties(strPropertyName) = varPropertyValue
   If Err <> 0 Then                ' Ошибка при задании значения.
      If Err <> conErrPropertyNotFound Then
         On Error GoTo 0
         MsgBox "Не удается задать свойство '" & strPropertyName _
            & "' для поля '" & fldField.name & "'", 48, "SetFieldProperty"
      Else
         On Error GoTo 0
         Set prpProperty = fldField.CreateProperty(strPropertyName, _
            intPropertyType, varPropertyValue)
         fldField.Properties.Append prpProperty
      End If
   End If
End Sub
```

Свойство ColumnWidth

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproColumnWidthC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproColumnWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproColumnWidthA;daobjQueryDef;daobjTableDef"}

Свойство **ColumnWidth** определяет ширину столбца в объекте в режиме таблицы.

Примечание. Свойство **ColumnWidth** применяется ко всем полям в объекте в режиме таблицы и к элементам управления в форме в режиме таблицы.

Значения

Значение данного свойства можно задать путем перетаскивания правой границы области выделения столбца или посредством выбора в режиме таблицы в меню **Формат** команды **Ширина столбца** с последующим вводом нужного значения. Если значение свойства **ColumnWidth** задается с помощью команды **Ширина столбца**, то оно выражается в пунктах.

В программах Visual Basic значение свойства **ColumnWidth** имеет тип **Integer** (целое) и представляет ширину столбца в твипах. Пользователь имеет возможность указать конкретное значение ширины столбца или выбрать одно из следующих встроенных значений.

Значение	Описание
0	Делает столбец скрытым.
–1	(Значение по умолчанию). Восстанавливает стандартную ширину столбца.

Дополнительные сведения

Значение 0 свойства **ColumnWidth**, т.е. нулевая ширина столбца, указанная в режиме таблицы, автоматически задает значение **True** (–1) для свойства поля **ColumnHidden** и делает поле скрытым в режиме таблицы.

Задание для свойства поля **ColumnHidden** значения **False** (0) приводит к восстановлению того значения свойства **ColumnWidth**, которое поле имело до скрытия. Например, если до скрытия поля свойство **ColumnWidth** имело значение –1, то после указания значения **False** для свойства **ColumnHidden** свойство **ColumnWidth** получит прежнее значение –1.

Если для свойства поля **ColumnHidden** задано значение **True**, то свойство этого поля **ColumnWidth** становится недоступным.

Свойства DatasheetFontItalic, DatasheetFontUnderline

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDatasheetFontItalicC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDatasheetFontItalicX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDatasheetFontItalicA"}

Свойства **DatasheetFontItalic** и **DatasheetFontUnderline** определяют использование курсива или подчеркнутого шрифта для данных и имен полей в режиме таблицы.

Примечание. Свойства **DatasheetFontItalic** и **DatasheetFontUnderline** применяются ко всем полям в объекте в режиме таблицы и к элементам управления в форме в режиме таблицы.

Значения

Значения данных свойств задаются при нажатии кнопок **Курсив**  или **Подчеркнутый**  на панели инструментов **Форматирование (режим таблицы)**.
Кроме того, значения этих свойств можно задать в диалоговом окне **Шрифт**, которое открывается с помощью команды **Шрифт** из меню **Формат** в режиме таблицы.
В программах Visual Basic свойство **DatasheetFontItalic** может иметь следующие значения.

Значение	Описание
True (–1)	Применяется курсив.
False (0)	(Значение по умолчанию). Курсив не применяется.

В программах Visual Basic свойство **DatasheetFontUnderline** может иметь следующие значения.

Значение	Описание
True	Применяется подчеркивание.
False	(Значение по умолчанию). Подчеркивание не применяется.

В программах Visual Basic значения данных свойств имеют тип **Boolean**.

Дополнительные сведения

В следующей таблице представлены свойства, которые отсутствуют в семействе **Properties** для объектов **TableDef** или **QueryDef**, пока пользователь не добавит их с помощью метода **CreateProperty** или не определит их на панели инструментов **Форматирование (режим таблицы)**.

DatasheetFontItalic*	**DatasheetForeColor***
DatasheetFontHeight*	**DatasheetBackColor**
DatasheetFontName*	**DatasheetGridlinesColor**
DatasheetFontUnderline*	**DatasheetGridlinesBehavior**
DatasheetFontWeight*	**DatasheetCellsEffect**

Примечание. При добавлении или определении одного из вышеперечисленных свойств, отмеченного звездочкой, Microsoft Access автоматически добавляет в семейство **Properties** все остальные отмеченные звездочкой свойства.

Свойства DatasheetFontItalic, DatasheetFontUnderline, примеры

В этом примере для формы «Товары» задается использование курсива и подчеркивания для представления данных и имен полей в режиме таблицы.

```
Forms![Товары].DatasheetFontItalic = True
Forms![Товары].DatasheetFontUnderline = True
```

В следующем примере для таблицы «Товары» задается использование курсива и подчеркивания для представления данных и имен полей в режиме таблицы.

Для определения свойств **DatasheetFontItalic** и **DatasheetFontUnderline** в следующей программе используется процедура SetTableProperty, определенная в стандартном модуле базы данных.

```
Dim dbs As Database, tdfТовары As TableDef
Set dbs = CurrentDb
Set tdfТовары = dbs![Товары]
SetTableProperty tdfТовары, "DatasheetFontItalic", dbBoolean, True
SetTableProperty tdfТовары, "DatasheetFontUnderline", dbBoolean, True

Sub SetTableProperty(tdfTableObj As TableDef, strPropertyName As String, _
      intPropertyType As Integer, varPropertyValue As Variant)
   ' Определяет свойство таблицы Microsoft Access без возникновения
   ' критической ошибки выполнения.
   Const conErrPropertyNotFound = 3270
   Dim prpProperty As Property
   On Error Resume Next            ' Не перехватывать ошибки.
   tdfTableObj.Properties(strPropertyName) = varPropertyValue
   If Err <> 0 Then                ' Ошибка при задании значения.
      If Err <> conErrPropertyNotFound Then
         On Error GoTo 0
         MsgBox "Не удается определить свойство '" & strPropertyName _
            & "' для таблицы '" & tdfTableObj.Name & "'", 48,
"SetTableProperty"
      Else
         On Error GoTo 0
         Set prpProperty = tdfTableObj.CreateProperty(strPropertyName, _
            intPropertyType, varPropertyValue)
         tdfTableObj.Properties.Append prpProperty
      End If
   End If
   tdfTableObj.Properties.Refresh
End Sub
```

Свойства DatasheetFontName, DatasheetFontHeight

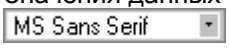
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDatasheetFontNameC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDatasheetFontNameX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDatasheetFontNameA"}

Свойства **DatasheetFontName** и **DatasheetFontHeight** позволяют задать название шрифта и его размер в пунктах, которые будут использованы при отображении или печати данных и имен полей в режиме таблицы.

Примечание. Свойства **DatasheetFontName** и **DatasheetFontHeight** применяются ко всем полям в объекте в режиме таблицы и к элементам управления в форме в режиме таблицы.

Значения

- Свойство **DatasheetFontName**. В программах Visual Basic значение данного свойства задается с помощью строкового выражения, значением которого является имя шрифта.
- Свойство **DatasheetFontHeight**. В программах Visual Basic данное свойство задается с помощью значения типа **Integer** в диапазоне от 1 до 127, представляющего размер шрифта в пунктах.

Значения данных свойств задаются при выборе имени шрифта в поле **Шрифт** 

или размера шрифта в поле **Размер**

на панели инструментов **Форматирование (режим таблицы)**.
Кроме того, значения этих свойств можно задать в полях **Шрифт** и **Размер** диалогового окна **Шрифт**, которое открывается с помощью команды **Шрифт** в меню **Формат** в режиме таблицы.

Дополнительные сведения

Набор названий шрифтов, которые могут быть указаны в свойстве **DatasheetFontName**, определяется шрифтами, установленными на компьютере и принтере. Недоступные и неустановленные шрифты автоматически заменяются похожими.

Значением свойства **DatasheetFontHeight** должен быть допустимый размер шрифта, указанного в свойстве **DatasheetFontName**. Например, для шрифта MS Sans Serif допустимыми являются только следующие размеры: 8, 10, 12, 14, 18 и 24.

Имеется возможность выбрать для свойств **DatasheetFontName** и **DatasheetFontHeight** значения по умолчанию в группе **Шрифт по умолчанию** на вкладке **Режим таблицы** диалогового окна **Параметры**, которое открывается при выборе в меню **Сервис** команды **Параметры**.

В следующей таблице представлены свойства, которые отсутствуют в семействе **Properties** для объектов **TableDef** или **QueryDef**, пока пользователь не добавит их с помощью метода **CreateProperty** или не определит их на панели инструментов **Форматирование (режим таблицы)**.

DatasheetFontItalic*	**DatasheetForeColor***
DatasheetFontHeight*	**DatasheetBackColor**
DatasheetFontName*	**DatasheetGridlinesColor**
DatasheetFontUnderline*	**DatasheetGridlinesBehavior**
DatasheetFontWeight*	**DatasheetCellsEffect**

Примечание. При добавлении или определении одного из вышеперечисленных свойств, отмеченного звездочкой, Microsoft Access автоматически добавляет в семейство **Properties** все остальные отмеченные звездочкой свойства.

Свойства DatasheetFontName, DatasheetFontHeight и DatasheetFontWeight, примеры

В следующем примере для таблицы «Товары» в режиме таблицы задается шрифт MS Serif средней насыщенности (500) размером 10 пунктов.

Для определения этих свойств в следующей программе используется процедура SetTableProperty, представленная в примере для свойств **DatasheetFontItalic** и **DatasheetFontUnderline**.

```
Dim dbs As Database, tdfТовары As TableDef
Set dbs = CurrentDb
Set tdfТовары = dbs!Товары
SetTableProperty tdfТовары, "DatasheetFontName", dbText, "MS Serif"
SetTableProperty tdfТовары, "DatasheetFontHeight", dbInteger, 10
SetTableProperty tdfТовары, "DatasheetFontWeight", dbInteger, 500
```

В следующем примере такие же изменения вносятся в форму «Товары», открытую в режиме таблицы.

```
Forms!Товары.DatasheetFontName = "MS Serif"
Forms!Товары.DatasheetFontHeight = 10
Forms!Товары.DatasheetFontWeight = 500
```

Свойство DatasheetFontWeight

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDatasheetFontWeightC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDatasheetFontWeightX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproDatasheetFontWeightA"}

Свойство **DatasheetFontWeight** определяет ширину линии шрифта, который используется для изображения и печати символов в данных и именах полей в режиме таблицы.

Примечание. Свойство **DatasheetFontWeight** применяется ко всем полям объекта в режиме таблицы и к элементам управления в форме в режиме таблицы.

Значения

В программах Visual Basic свойство **DatasheetFontWeight** может иметь одно из следующих значений типа **Integer**.

Значение	Описание
100	Тонкий
200	Сверхсветлый
300	Светлый
400	(Значение по умолчанию). Обычный
500	Средний
600	Плотный
700	Полужирный
800	Жирный
900	Сверхжирный

Кроме того, значения данного свойства, соответствующие атрибутам «Обычный» и «Полужирный», задаются в диалоговом окне **Шрифт**, которое открывается с помощью команды **Шрифт** в меню **Формат** в режиме таблицы. В поле **Начертание** диалогового окна **Шрифт** доступны только два значения насыщенности шрифта: «Обычный» и «Полужирный».

Задавать только атрибуты «Обычный» и «Полужирный» позволяет также кнопка **Полужирный**  на панели инструментов **Форматирование (режим таблицы)**.

Имеется возможность выбрать для свойства **DatasheetFontWeight** значение по умолчанию в группе **Шрифт по умолчанию** на вкладке **Режим таблицы** диалогового окна **Параметры**, которое открывается при выборе в меню **Сервис** команды **Параметры**.

В следующей таблице представлены свойства, которые отсутствуют в семействе **Properties** для объектов **TableDef** или **QueryDef**, пока пользователь не добавит их с помощью метода **CreateProperty** или не определит их на панели инструментов **Форматирование (режим таблицы)**.

DatasheetFontItalic*	**DatasheetForeColor***
DatasheetFontHeight*	**DatasheetBackColor**
DatasheetFontName*	**DatasheetGridlinesColor**
DatasheetFontUnderline*	**DatasheetGridlinesBehavior**
DatasheetFontWeight*	**DatasheetCellsEffect**

Примечание. При добавлении или определении одного из вышеперечисленных свойств, отмеченного звездочкой, Microsoft Access автоматически добавляет в семейство **Properties** все остальные отмеченные звездочкой свойства.

Свойство FrozenColumns

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFrozenColumnsC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproFrozenColumnsX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproFrozenColumnsA"}

Свойство **FrozenColumns** возвращает число столбцов, закрепленных на экране в режиме таблицы. Закрепленные столбцы располагаются в левой части таблицы; их положение не меняется при горизонтальной прокрутке таблицы.

Примечание. Свойство **FrozenColumns** применяется только к таблицам, формам и запросам в режиме таблицы.

Значения

Значение свойства **FrozenColumns** устанавливается автоматически при выполнении команды **Закрепить столбцы** в меню **Формат**.

В программах Visual Basic значение этого свойства имеет тип **Integer** и указывает число столбцов, закрепленных в режиме таблицы с помощью команды **Закрепить столбцы**. Область выделения записей всегда является закрепленной, поэтому по умолчанию данное свойство имеет значение 1. При последующем закреплении одного столбца свойство **FrozenColumns** принимает значение 2, при фиксации двух столбцов 3 и т.д.

Данное свойство во всех режимах доступно только для чтения.

Дополнительные сведения

При фиксации столбцов с помощью команды **Закрепить столбцы** Microsoft Access автоматически перемещает указанные столбцы на левый край таблицы и располагает их в том порядке, в котором они были зафиксированы. Например, если будут закреплены три столбца, они станут первым, вторым и третьим столбцами таблицы. Поскольку область выделения записей всегда является закрепленной, свойство **FrozenColumns** в этом случае примет значение 4. Для трех закрепленных столбцов свойство **ColumnOrder** получит значения 1, 2 и 3 (в соответствии с последовательностью фиксации столбцов).

При выборе в меню **Формат** команды **Освободить все столбцы** снимается закрепление всех столбцов, а свойство **FrozenColumns** принимает значение 1.

Примечание. Если применение команды **Закрепить столбцы** вызвало изменение расположения столбцов в таблице, то использование команды **Освободить все столбцы** не приведет к восстановлению исходного порядка столбцов.

Свойство FrozenColumns, пример

В данном примере свойство **FrozenColumns** используется для определения числа столбцов, закрепленных в режиме таблицы. Если зафиксировано более трех столбцов, таблица развертывается на полный экран.

```
Sub CheckFrozen(strTableName As String)
   Dim dbs As Database
   Dim tdf As TableDef
   Dim prp As Property
   Const conPropertyNotFound = 3270  ' Ошибка: свойство не найдено.
   Set dbs = CurrentDb    ' Текущая база данных.
   Set tdf = dbs.TableDefs(strTableName)   ' Объектная переменная для
таблицы.
   DoCmd.OpenTable strTableName, acNormal  ' Открывает таблицу.
   tdf.Properties.Refresh
   On Error GoTo Frozen_Err
   If tdf.Properties("FrozenColumns") > 3 Then   ' Проверяет значение
свойства.
      DoCmd.Maximize
   End If
Frozen_Bye:
   Exit Sub
Frozen_Err:
   If Err = conPropertyNotFound Then       ' Свойство не включено в
семейство.
      Set prp = tdf.CreateProperty("FrozenColumns", dbInteger, 1)
      tdf.Properties.Append prp
      Resume Frozen_Bye
   End If
End Sub
```

Свойство RowHeight

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproRowHeightC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproRowHeightX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproRowHeightA"}

Свойство **RowHeight** задает высоту всех строк в режиме таблицы.

Примечание. Свойство **RowHeight** применяется ко всем полям объекта в режиме таблицы и к элементам управления в форме в режиме таблицы.

Значения

Значение свойства **RowHeight** задается в режиме таблицы путем перетаскивания с помощью мыши нижней границы области выделения записи, или с помощью команды **Высота строки** в меню **Формат**. При определении свойства **RowHeight** с помощью команды **Высота строки** значение выражается в пунктах.

В программах Visual Basic свойство **RowHeight** имеет значение типа **Long Integer**, представляющее высоту строк таблицы в твипах. Для того чтобы определить для текущего шрифта высоту, используемую по умолчанию, следует задать для свойства **RowHeight** значение **True** (–1).

Свойства RowHeight, ColumnWidth, примеры

В следующем примере для таблицы «Клиенты» задается высота строки в режиме таблицы (450 твип), достаточная для вывода двух строк данных. Для поля «Адрес» задается ширина столбца 1,5 дюйма (2160 твип). Эти настройки вступят в силу при следующем открытии таблицы «Клиенты» в режиме таблицы.

Для определения свойств **RowHeight** и **ColumnWidth** в следующей программе используется процедура SetTableProperty, представленная в примере для свойств **DatasheetFontItalic** и **DatasheetFontUnderline**, а также процедура SetFieldProperty, представленная в примере для свойства **ColumnHidden**.

```
Dim dbs As Database, tdfКлиенты As TableDef
Set dbs = CurrentDb
Set tdfКлиенты = dbs![Клиенты]
SetTableProperty tdfКлиенты, "RowHeight", dbLong, 450
SetFieldProperty tdfКлиенты![Адрес], "ColumnWidth", dbInteger, 2160
```

В следующем примере такие же изменения вносятся в форму «Клиенты», открытую в режиме таблицы. Задается высота строк 450 твип и ширина столбцов по размеру видимого текста.

```
Forms![Клиенты].RowHeight = 450
Forms![Клиенты]![Адрес].ColumnWidth = -2
```

Свойства «Расширение» (CanGrow), «Сжатие» (CanShrink)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCanGrowShrinkC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproCanGrowShrinkX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproCanGrowShrinkA"}

Свойства **Расширение (CanGrow)** и **Сжатие (CanShrink)** определяют вид разделов и элементов управления в форме или в отчете при просмотре или при печати. Например, при заданном для обоих свойств значении «Да» вертикальные размеры раздела или элемента управления автоматически изменяются таким образом, чтобы отобразить или напечатать все данные, содержащиеся в них.

Примечания

- Свойства **Расширение (CanGrow)** и **Сжатие (CanShrink)** неприменимы к заголовкам и примечаниям страниц форм и отчетов, но применимы к элементам управления этих же разделов.
- Эти свойства влияют на вид разделов и элементов управления формы только во время ее печати или просмотре. В режиме формы, режиме таблицы или режиме конструктора влияния на вид формы эти свойства не оказывают.

Значения

Свойство **Расширение (CanGrow)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Задает автоматическое увеличение высоты раздела или элемента управления для печати или просмотра всех содержащихся в нем данных.	**True** (–1)
Нет	(Значение по умолчанию). Высота раздела или элемента управления автоматически не увеличивается. Данные не помещаются в отведенную область и не печатаются.	**False** (0)

Свойство **Сжатие (CanShrink)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Задает автоматическое уменьшение высоты раздела или элемента управления для печати всех содержащиеся в нем данных без лишних пустых строк.	**True**
Нет	(Значение по умолчанию). Высота раздела или элемента управления автоматически не уменьшается.	**False**

Значения данных свойств можно задать только в окне свойств раздела или элемента управления.

Для элементов управления указать используемые по умолчанию значения этих свойств можно в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Значения данных свойств являются доступными только для чтения и используются в макросах и в программах Visual Basic в любых режимах, за исключением режима конструктора.

Дополнительные сведения

Данные свойства позволяют управлять внешним видом печатаемой формы или отчета. При задании для обоих свойств значения «Да» объект автоматически устанавливается так, чтобы все данные были выведены на печать. При расширении или сжатии элемента управления

элемент управления, находящийся под изменяемым, перемещается соответственно вниз и вверх.

Если задать значение «Да» для свойства **Расширение (CanGrow)** элемента управления, то автоматически задается значение «Да» для свойства **Сжатие (CanShrink)** раздела, содержащего этот элемент управления. (Однако при задании значения «Да» для свойства **Сжатие (CanShrink)** элемента управления, автоматического задания значения «Да» для этого же свойства раздела не происходит.)

Изменение высоты раздела происходит по всей ширине раздела. Например, пусть в разделе формы имеются два соседних поля и свойство **Сжатие (CanShrink)** каждого из них имеет значение «Да». Если в одном поле содержится одна строка данных, а в другом две строки, оба поля будут длиной в две строки, так как раздел ограничен по внутренней ширине. Для независимого расширения и сжатия данных необходимо добавить два соседних элемента управления подчиненной формы или подчиненного отчета и установить для их свойств **Расширение (CanGrow)** и **Сжатие (CanShrink)** значения «Да».

При использовании свойств **Расширение (CanGrow)** и **Сжатие (CanShrink)** необходимо помнить следующее:

- значения этих свойств не влияют на интервалы между элементами управления по горизонтали; изменяются только вертикальные размеры самих элементов управления;
- размеры перекрывающихся элементов управления не могут ни уменьшаться, ни увеличиваться автоматически;
- высота большого элемента управления может помешать уменьшению расположенных рядом элементов управления. Например, если в левой части области данных располагаются несколько маленьких элементов управления, а в правой части - один большой элемент управления (например, свободная рамка объекта), то размеры расположенных слева элементов не будут уменьшаться, даже если они не содержат данных.

Свойства «Высота» (Height), «Ширина» (Width)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproHeightWidthC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproHeightWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproHeightWidthA":1}

Свойства **Высота (Height)** и **Ширина (Width)** позволяют точно указать размеры объекта. Это полезно, например, когда необходимо создать различные объект одного размера или имеющие одинаковую ширину или высоту.

Примечания

- Свойство **Высота (Height)** применимо только к разделам формы и разделам отчета, а не к самим формам и отчетам.
- Свойство **Ширина (Width)** применимо только к формам и отчетам, а не к их разделам.
- Оба свойства применяются к элементам управления в формах и отчетах.

Значения

Введите требуемые значения ширины или высоты объекта в текущих единицах измерения. Для использования единиц измерения, отличных от текущих, заданных в диалоговом окне **Язык и стандарты** панели управления Windows, необходимо явно указать символ единицы измерения (например, 5 см или 3"). Значение свойства **Ширина (Width)** должно лежать в диапазоне от 0 до 22 дюймов (55,87 см).

Значения данных свойств задаются в окне свойств, в макросе или в программе Visual Basic.

Для элементов управления используемые по умолчанию значения этих свойств можно указать в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

В программе Visual Basic для задания значения этих свойств используются числовые выражения. Значения представлены в единицах твип.

Для разделов отчета использование макроса или программы Visual Basic для установки свойства **Высота (Height)** при просмотре или печати отчета невозможно. Для элементов управления отчета во время просмотра или печати отчета для установки свойств **Высота (Height)** и **Ширина (Width)** допустимо использование только макросов и процедуры обработки события, указанного в значении свойства события **Форматирование (OnFormat).**

После начала процесса печати задать значения этих свойств невозможно. Например, попытка определить свойство **Высота (Height)** в процедуре обработки события отчета **Печать (Print)** приведет к ошибке.

Дополнительные сведения

Microsoft Access автоматически устанавливает значения свойств **Высота (Height)** и **Ширина (Width)** при создании элементов управления или при изменении их размеров или при изменении размеров окна в режиме конструктора формы или в режиме конструктора отчета.

Высотой и шириной форм и отчетов называют расстояние между внутренними сторонами соответствующих границ. Высотой и шириной элементов управления считается расстояние между серединами соответствующих границ; это обеспечивает правильное выравнивание элементов управления с разными типами границ. Ширина полей (отступов) для форм и отчетов задается командой **Параметры страницы** из меню **Файл.**

Примечание. Для указания положения верхнего левого угла объекта используйте свойства **От левого края (Left)** и **От верхнего края (Top).**

Свойства «Высота» (Height), «Ширина» (Width), пример

В данном примере создается квадратная кнопка с размером 1 дюйм (стандартными единицами измерения в Visual Basic являются единицы твип; 1440 единиц твип равны одному дюйму):

```
Me!cmdSizeButton.Height = 1440      ' 1440 твип = 1 дюйм.
Me!cmdSizeButton.Width = 1440
```

Свойства «Высота» (Height), «Ширина» (Width), применение

Свойство **Высота (Height)**

Свойство **Ширина (Width)**

Свойства «Позиция подписи X» (LabelX), «Позиция подписи Y» (LabelY)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproLabelXYC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproLabelXYX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproLabelXYA"}

Свойства **Позиция подписи X (LabelX)** и **Позиция подписи Y (LabelY)** определяют положение подписи, присоединенной к создаваемому новому элементу управления. Например, подпись поля можно вывести не слева, а справа от поля.

Значения

Введите положительное или отрицательное число, задающее положение подписи относительно верхнего левого угла присоединенного элемента управления. Для использования единиц измерения, отличных от текущих, заданных в диалоговом окне **Язык и стандарты** панели управления Windows, необходимо явно указать символ единицы измерения (например, 2 см или 1,5 ").

Используемые по умолчанию значения данных свойств можно указать только в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

Свойство **Позиция подписи X (LabelX)** задает смещение подписи по горизонтальной оси, а свойство **Позиция подписи Y (LabelY)** смещение по вертикальной оси. Отрицательное значение свойства **Позиция подписи X (LabelX)** помещает верхний левый угол подписи слева от элемента управления. Отрицательное значение свойства **Позиция подписи Y (LabelY)** помещает верхний левый угол подписи над элементом управления. Положительные значения помещают подпись правее и ниже верхнего левого угла элемента управления.

Если свойство **Выравнивание подписи (LabelAlign)** имеет значение «По правому краю» или «По левому краю», то свойства **Позиция подписи X (LabelX)** и **Позиция подписи Y (LabelY)** задают расстояние между правым или левым краем подписи и верхним левым углом элемента управления. Если свойство **Выравнивание подписи (LabelAlign)** имеет значение «По центру», то свойства **Позиция подписи X (LabelX)** и **Позиция подписи Y (LabelY)** задают расстояние между центром подписи и верхним левым углом элемента управления.

Комбинация различных значений свойств **Позиция подписи X (LabelX), Позиция подписи Y (LabelY)** и **Выравнивание подписи (LabelAlign)** дают следующие результаты:

Позиция подписи X (LabelX)	Позиция подписи Y (LabelY)	Выравнивание подписи (LabelAlign)	Результат
–3 см	0	Обычное	Верхний левый угол подписи находится на 3 см слева от верхнего левого угла элемента управления.
–0,5 см	0	По правому краю	Верхний правый угол подписи находится на 3 см слева от верхнего левого угла элемента управления.
1,5 см	–1 см	По центру	Центр верхней границы подписи находится на 1,5 см справа и на 1 см выше верхнего левого угла элемента управления.

Свойства «От левого края» (Left), «От верхнего края» (Top)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproLeftTopC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproLeftTopX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproLeftTopA"}

Свойства **От левого края (Left)** и **От верхнего края (Top)** задают положение объекта в форме или отчете. Например, они позволяют выравнивать элементы управления относительно правого верхнего угла раздела.

Значения

Положение элемента управления – это расстояние от его левого или верхнего края до левого или верхнего края раздела отчета, содержащего этот элемент управления. Нулевые значения свойств **От левого края (Left)** и **От верхнего края (Top)** соответствуют размещению элемента управления в верхнем левом углу раздела отчета. Для того, чтобы использовать единицы измерения, отличные от текущих, заданных в диалоговом окне **Язык и стандарты** панели управления Windows, следует явно указать символ единицы измерения (например, 3 см или 2 ").

В программе Visual Basic значения данных свойств задаются с помощью числовых выражений. Значения задаются в единицах твип.

Для элементов управления задать значения этих свойств можно в окне свойств, в макросе или в программе Visual Basic.

Для отчетов значения данных свойств можно задать только в макросе или в процедуре обработки события Visual Basic отчета в режиме предварительного просмотра или выведенного на печать.

Дополнительные сведения

При перемещении элемента управления новые значения свойств **От левого края (Left)** и **От верхнего края (Top)** автоматически заносятся в окно свойств. При просмотре формы или отчета в режиме предварительного просмотра или при печати формы положение элемента управления определяется значениями его свойств **От левого края (Left)** и **От верхнего края (Top),** а также шириной полей, заданных в диалоговом окне **Параметры страницы,** вызываемого командой **Параметры страницы** из меню **Файл**.

Для отчетов значением свойства **От верхнего края (Top)** является смещение текущего раздела относительно верхнего края страницы, а значение свойства **От левого края (Left)** смещение текущего раздела относительно левой границы. Значения обоих свойств выражаются в единицах твип. Эти свойства можно использовать в процедуре обработки события **Форматирование (Format)** для размещения раздела на странице.

Свойства «От левого края» (Left), «От верхнего края» (Top), пример

В данном примере проверяется значение свойства **От левого края (Left)** в текущем отчете. Если это значение меньше минимальной ширины полей, то для свойств **NextRecord** и **PrintSection** устанавливаются значения **False** (0). Раздел не расширяется до следующей записи и следующий раздел не печатается.

```
Sub Detail1_Format(Cancel As Integer, FormatCount As Integer)
Const conLeftMargin = 1880
' Если значение свойства "От левого края" (Left) меньше 1880 твип, то
' ни переход к следующей записи, ни печать раздела не выполняются.
   If Me.Left < conLeftMargin Then
      Me.NextRecord = False
      Me.PrintSection = False
   End If
End Sub
```

Свойство «Выравнивание по центру» (AutoCenter)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproAutoCenterC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproAutoCenterX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproAutoCenterA"}

Свойство **Выравнивание по центру (AutoCenter)** позволяет указать автоматическое выравнивание формы по центру окна приложения при ее открытии.

Значения

Свойство **Выравнивание по центру (AutoCenter)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Форма при открытии автоматически выравнивается по центру окна.	**True** (–1)
Нет	(Значение по умолчанию). При открытии верхний левый угол формы помещается в то же место, которое он занимал во время ее последнего сохранения.	**False** (0)

Значение данного свойства задается в окне свойств формы, в макросе или в программе Visual Basic.

Данное свойство можно определить только в режиме конструктора формы.

Дополнительные сведения

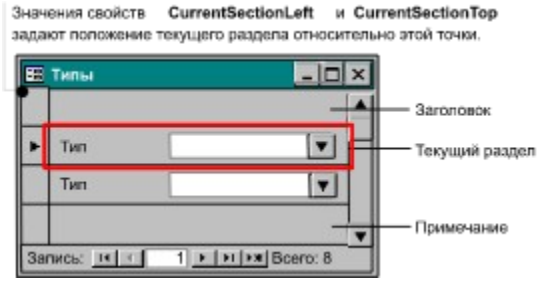
В зависимости от размера и места окна приложения форма может оказаться сдвинутой так, что ее часть не будет видна в окне приложения. Автоматическое выравнивание формы по центру при открытии делает работу с формой более удобной.

При изменении в режиме конструктора формы с отключенным свойством **Автоматический размер (AutoResize)** и установленным свойством **Выравнивание по центру** (**AutoCenter)** перед ее сохранением необходимо переключиться в режим формы. В противном случае Microsoft Access не выравнивает форму по ее правой и нижней границам при ее следующем открытии.

Свойства CurrentSectionLeft, CurrentSectionTop

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproCurrentSectionLeftTopC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproCurrentSectionLeftTopX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproCurrentSectionLeftTopA"}

Данные свойства определяют координаты текущего раздела в единицах твип относительно левой и верхней границы формы.



Значения

- Свойство **CurrentSectionLeft**. Значение типа **Integer** (целое), представляющее расстояние от левого края текущего раздела до левого края формы.
- Свойство **CurrentSectionTop**. Значение типа **Integer**, представляющее расстояние от верхнего края текущего раздела до верхнего края формы.

Значения этих свойств доступны только для чтения в макросе или в программе Visual Basic.

Дополнительные сведения

Значения свойств **CurrentSectionLeft** и **CurrentSectionTop** изменяются по мере прокрутки формы.

Если при прокрутке формы со значением свойства **Режим по умолчанию (DefaultView)** «Простая форма» осуществляется переход в положение над верхним краем раздела или вправо от левого края формы, значения данных свойств становятся отрицательными.

Для невидимого раздела формы со значением свойства **Режим по умолчанию (DefaultView)** «Ленточная форма» значение свойства **CurrentSectionTop** совпадает со значением свойства **InsideHeight** формы.

Данные свойства позволяют определить положение областей данных, выводящихся в режиме формы в ленточных формах или в режиме таблицы. Каждая из областей данных имеет собственное значение свойства **CurrentSectionTop**, зависящее от положения области в форме.

Свойства CurrentSectionLeft, CurrentSectionTop, пример

В данном демонстрируются значения свойств **CurrentSectionLeft** и **CurrentSectionTop** элемента ленточной форме. Когда пользователь переходит на новую запись, значения этих свойств для текущего раздела выводятся в подписи `lblStatus`, размещенной в заголовке формы.

```
Private Sub Form_Current()
   Dim intCurTop As Integer, intCurLeft As Integer

   intCurTop = Me.CurrentSectionTop
   intCurLeft = Me.CurrentSectionLeft
   Me!lblStatus.Caption = intCurLeft & " , " & intCurTop
End Sub
```

Свойства WindowHeight, WindowWidth

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acproWindowHeightWidthC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproWindowHeightWidthX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproWindowHeightWidthA"}

Данные свойства возвращают высоту и ширину формы в единицах твип.

Значения

Значения свойств **WindowHeight** и **WindowWidth** имеют тип **Integer**.

Значения данных свойств доступны только для чтения в макросе или в программе Visual Basic.

Дополнительные сведения

Свойство **WindowHeight** возвращает расстояние от верхнего левого угла формы до ее нижнего левого угла. Свойство **WindowWidth** возвращает расстояние от верхнего левого угла формы до ее правого верхнего угла.

Свойство «Данные» (ControlSource)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproControlSourceC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproControlSourceX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproControlSourceA"}

Свойство **Данные (ControlSource)** позволяет указать, какие данные выводятся в элементе управления. Пользователь имеет возможность отображать и редактировать данные, в элементе управления, присоединенном к полю таблицы, запроса или инструкции SQL. Допускается также вывод в элементе управления результата выражения. Для уровня группировки в отчете свойство **Данные (ControlSource)** определяет поле или выражение, по которому проводится группировка.

Примечание. Это свойство неприменимо к флажкам, переключателям, или выключателям, включенным в состав группы; оно применимо только к самой группе.

В отчетах свойство **Данные (ControlSource)** является применимым только к группам отчета.

Значения

Свойство **Данные (ControlSource)** может иметь следующие значения.

Значение	Описание
Имя поля	Элемент управления является присоединенным к полю в таблице, запросе или инструкции SQL. Содержимое поля отображается в элементе управления. Изменение данных в элементе управления приводит к изменению соответствующих данных в поле. (Для того чтобы сделать элемент управления доступным только для чтения, следует задать для его свойства **Блокировка** (**Locked**) значение «Да»). При выборе присоединенного к полю элемента управления, содержащего тип данных **Гиперссылка,** произойдет переход по адресу, заданному в гиперссылке.
Выражение	Выражение является источником данных для элемента управления. Эти значения допускают только чтение и не сохранятся в базе данных.

Значение свойства **Данные (ControlSource)** задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Значение свойства **Данные (ControlSource)** для элемента управления поле можно также задать, если непосредственно ввести в него выражение или имя поля в режиме конструктора формы или отчета.

Для того чтобы определить это свойство для отчета, выберите поле или введите выражение в столбец **Поле/выражение** в окне **Сортировка и группировка.** Более подробное описание см. в описании свойства **Уровень группировки** (**GroupLevel**).

В программе Visual Basic значение данного свойства задается с помощью строкового выражения.

Дополнительные сведения

Формы и отчеты можно рассматривать как «окна» в базу данных. Для того чтобы определить основной источник данных для формы или отчета, следует указать таблицу, запрос или инструкцию SQL в свойстве **Источник записей** (**RecordSource**). После этого становится возможным указание в свойстве **Данные (ControlSource)** поля или выражения. Если в значении свойства **Данные (ControlSource)** указано выражение, его значение допускает только чтение и не сохраняется в базе данных. Например:

Примеры значений	Описание

Фамилия	Для элемента управления содержимое поля «Фамилия» выводится в элементе управления. Для группы в отчете выполняется группировка по полю «Фамилия».
`=Date( ) + 7`	В элементе управления выводится дата, отстоящая на семь дней от текущей.
`=DatePart("q",`ДатаИсполнения`)`	В элементе управления выводится квартал для даты выполнения заказа. Для группы в отчете выполняется группировка данных по кварталам дат выполнения заказов.

Свойство «Данные» (ControlSource), пример

В данном примере в свойстве **Данные (ControlSource)** поля «Адрес1» указано поле «Город».

```
Forms!Клиенты! Адрес1.ControlSource = "Город"
```

В следующем примере свойство **Данные (ControlSource)** для поля «Отправить» определяется выражением `= Date() + 7`.

```
Forms!Счет!Отправить.ControlSource = "= Date() + 7"
```

Свойство «Значение параметра» (OptionValue)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproOptionValueC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproOptionValueX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproOptionValueA"}

Каждому элементу в группе параметров присваивается числовое значение, которое может быть задано с помощью свойства **Значение параметра (OptionValue)**. При выборе элемента управления его значение присваивается группе. Если группа является присоединенной к полю, значение свойства **Значение параметра** записывается в поле.

В данном примере группа **Регион** присоединена к полю «Регион» в таблице. Для переключателя **Европа** свойство **Значение параметра (OptionValue)** имеет значение 1, для переключателя **Азия** значение 2, а для переключателя **Африка** значение 3. При выборе переключателя в форме значение параметра заносится в поле «Регион». В данном случае в поле «Регион» в таблице будет занесено значение 2.

Примечание. Это свойство применимо только к флажкам, переключателям или выключателям, включенным в состав группы.

Значения

Свойство **Значение параметра (OptionValue)** задается с помощью значения типа **Long Integer** или числового выражения, присвоенного элементу управления в группе.

Значение данного свойства задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Дополнительные сведения

Если пользователь не изменяет значения свойства **Значение параметра (OptionValue)**, то по умолчанию первому элементу в группе присваивается значение 1, второму - 2 и т.д.

Свойство **Значение параметра (OptionValue)** является доступным только для элемента управления, входящего в группу. Переключатель, выключатель или флажок, не включенный в состав группы, не имеет свойства **Значение параметра**. Вместо этого каждый такой элемент управления имеет свойство **Данные (ControlSource),** которое принимает значения **True** (-1), когда элемент управления выделен, и **False** (0) в остальных случаях.

Свойство «Источник записей» (RecordSource)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproRecordSourceC":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproRecordSourceX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproRecordSourceA"}

Свойство **Источник записей (RecordSource)** определяет в качестве источника данных для формы или отчета таблицу, запрос или инструкцию SQL. Например, для того чтобы выводить и редактировать в форме данные из таблицы «Сотрудники», необходимо в свойстве **Источник записей (RecordSource)** указать имя этой таблицы. После этого становится возможным создание в форме или отчете элементов управления, присоединенных к конкретным полям таблицы «Сотрудники». Для этого в свойствах **Данные** (**ControlSource**) элементов управления требуется указать имена полей таблицы. Например, для присоединения элемента управления к полю «Фамилия» из таблицы «Сотрудники» необходимо задать для свойства **Источник записей** значение «Фамилия».

Значения

Значением свойства **Источник записей (RecordSource)** может быть имя таблицы, имя запроса или инструкция SQL. Например, правильными являются следующие настройки.

Пример значения	Описание
Сотрудники	Имя таблицы, которое определяет, что таблица «Сотрудники» будет источником данных.
`SELECT Заказы!ДатаИсполнения FROM Заказы;`	Инструкция SQL, определяющая поле «ДатаИсполнения» из таблицы «Заказы» в качестве источника данных. Можно присоединить элемент управления в форме или отчете к полю «ДатаИсполнения» в таблице «Заказы», указав поле «ДатаИсполнения» в свойстве **Данные (ControlSource)**.

Значение данного свойства задается в окне свойств формы или отчета, в макросе или в программе Visual Basic.

В программе Visual Basic значение данного свойства задается с помощью строкового выражения.

Примечание. При изменении источника записей для открытой формы или отчета автоматически обновляются выводящиеся в документе данные.

Дополнительные сведения

После создания формы или отчета допускается изменение источника данных. Для этого следует изменить значение свойства **Источник записей (RecordSource)**. Кроме того, свойство **Источник записей** обеспечивает многократное использование макета документа. Предположим, например, что создана форма со стандартным макетом. Для того чтобы вывести в этой форме данные из другого источника, скопируйте форму и укажите в свойстве **Источник записей (RecordSource)** другую таблицу, запрос или инструкцию SQL.

Ограничение количества записей в источнике записей формы позволяет повысить производительность, особенно при работе в сети. Например, полезным приемом является указание в свойстве **Источник записей (RecordSource)** единственной записи и изменение значения этого свойства в зависимости от условий отбора, задаваемых пользователем.

Свойство «Источник записей» (RecordSource), пример

В данном примере в свойстве **Источник записей (RecordSource)** формы «Клиенты» задается таблица «Клиенты».

```
Forms!Клиенты.RecordSource = "Клиенты"
```

В следующем примере в качестве источника записей формы задается единственная запись из таблицы «Клиенты», отбираемая по текущему значению в поле со списком `выборКлиента`. Поле со списком заполняется с помощью инструкции SQL, в которой возвращается код клиента (значение поля «КодКлиента») в присоединенном столбце и название организации.

```
Sub выборКлиента_AfterUpdate()
   Dim strNewRecord As String
   strNewRecord = "SELECT * FROM Клиенты " _
      & " WHERE КодКлиента = '" _
      & Me!выборКлиента.Value & "'"
   Me.RecordSource = strNewRecord
End Sub
```

Свойства «Тип источника строк» (RowSourceType), «Источник строк» (RowSource)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproRowSourceRowSourceTypeC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproRowSourceRowSourceTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproRowSourceRowSourceTypeA"}

Свойства **Тип источника строк (RowSourceType)** и **Источник строк (RowSource)** определят источник данных для списка, поля со списком или для свободного объекта OLE, такого как диаграмма. Например, для того чтобы вывести в строках списка данные из запроса «Список клиентов», следует выбрать для свойства **Тип источника строк** значение «Таблица/запрос» и указать в свойстве **Источник строк** имя запроса «Список клиентов».

Значения

Свойство **Тип источника строк (RowSourceType)** может иметь следующие значения.

Значения	Описание
Таблица/запрос	(Значение по умолчанию). Источником данных является таблица, запрос или инструкция SQL, указанная в значении свойства **Источник строк (RowSource)**.
Список значений	Источником данных является список значений, указанный в свойстве **Источник строк (RowSource)**.
Список полей	Источником данных является список имен полей из таблицы, запроса или инструкции SQL, указанной в свойстве **Источник строк (RowSource)**.

Примечание. Допускается также определение свойства **Тип источника строк (RowSourceType)** с помощью функции, определяемой пользователем. Имя функции вводится без предшествующего знака равенства (=) и без круглых скобок после имени функции. Пользователь должен ввести специальные кодовые аргументы функции, указывающие способ заполнения элемента управления.

Значение свойства **Источник строк (RowSource)** зависит от значения свойства **Тип источника строк (RowSourceType)**.

Тип источника строк	Источник строк
Таблица/запрос	Имя таблицы или запроса или инструкция SQL.
Список значений	Список элементов, разделяемых точкой с запятой (;).
Список полей	Имя таблицы, запроса или инструкции SQL.

Примечание. Если свойство **Тип источника строк (RowSourceType)** задается с помощью определяемой пользователем функцию, то значение свойства **Источник строк (RowSource)** может быть оставлено пустым.

Значения свойств **Тип источника строк (RowSourceType)** и **Источник строк (RowSource)** задаются в окне свойств элемента управления, в макросе или в программе Visual Basic.

Для полей таблиц, у которых в свойстве **Тип элемента управления** (**DisplayControl**) указан список или поле со списком, значения данных свойств задаются в режиме конструктора таблицы. на вкладке **Подстановка** раздела «Свойства поля».

Примечание. Для полей таблиц, у которых в режиме конструктора таблицы в типе данных указан «Мастер подстановок», значения данных свойств задаются автоматически.

В программе Visual Basic значение свойства **Тип источника строк (RowSourceType)** задается

с помощью строкового выражения с одним из следующих значений: «Таблица/запрос», «Список значений» или «Список полей». Допускается определение значения свойства **Источник строк (RowSource)** с помощью строкового выражения. Для того чтобы задать значение свойства **Тип источника строк** с помощью функции, определяемой пользователем, следует указать имя этой функции.

Дополнительные сведения

Если список должен содержать небольшое число значений, которые не должны изменяться, можно выбрать в свойстве **Тип источника строк (RowSourceType)** «Список значений» и ввести образующие список значения в ячейку свойства **Источник строк (RowSource)**.

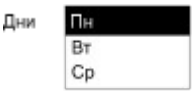
Если создана определяемая пользователем функция, заполняющая список или поле со списком, Microsoft Access повторно вызывает эту функцию для считывания необходимой информации. Функции, определяющие тип источника строк, должны строго удовлетворять специальному формату функций.

Свойства «Тип источника строк» (RowSourceType), «Источник строк» (RowSource), примеры значений

Ниже приведены примеры значений свойства **Источник строк (RowSource)** для каждого значения свойства **Тип источника строк (RowSourceType)**.

Тип источника строк = «Список значений»

Если в свойстве **Тип источника строк (RowSourceType)** задается «Список значений», то допустимыми являются приведенные ниже значения свойств **Источник строк (RowSource)** и **Число столбцов (ColumnCount)**, приводящие к следующим результатам.

Источник строк (RowSource)	Список
Пн;Вт;Ср (**Число столбцов** = 1)	 Список из одного столбца и трех строк
1;Понедельник;2;Вторник;3;Среда;4;Четверг (**Число столбцов** = 2)	 Список из двух столбцов и четырех строк
Страна;Столица;Россия;Москва;Китай;Пекин (**Число столбцов** = 2, **Заглавия столбцов** = Да)	 Список из двух столбцов со строкой заголовков и двумя строками данных

Тип источника строк (RowSourceType) = «Таблица/запрос»

Если в свойстве **Тип источника строк (RowSourceType)** задается «Таблица/запрос», в свойстве **Источник строк (RowSource)** следует указать имя таблицы или запроса, например, «Список типов».

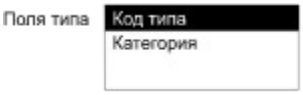
Ниже приводится правильная инструкция SQL, с помощью которой список из двух столбцов заполняется значениями полей таблицы «Типы» из базы данных «Борей».

```
SELECT КодТипа, Категория FROM Типы ORDER BY Категория;
```

Если запрос «Список типов» и инструкция SQL определят один набор записей, то при любом из значений данного свойства создается следующий список.

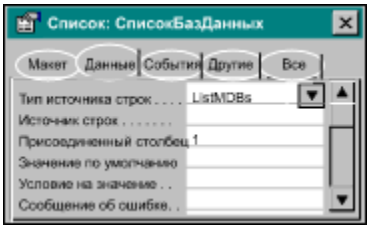
Тип источника строк (RowSourceType) = «Список полей»

Если в свойстве **Тип источника строк (RowSourceType)** задается «Список полей», а в свойстве **Источник строк (RowSource)** указан запрос «Список клиентов» (описанный в предыдущем примере), результат выглядит следующим образом.



Тип источника строк (RowSourceType) = имя функции

Если в свойстве **Тип источника строк (RowSourceType)** задано имя функции, определяемой пользователем, то значение свойства **Источник строк (RowSource)** следует оставить пустым. Например, определенную пользователем функцию ListMDBs, заполняющую поле со списком именами баз данных из текущего каталога, следует ввести в окно свойств следующим образом.



Свойства «Тип источника строк» (RowSourceType), «Источник строк» (RowSource), пример

В следующем примере для свойства **Тип источника строк (RowSourceType)** поля со списком задается «Таблица/запрос», а в свойстве **Источник строк (RowSource)** указывается запрос «Список сотрудников».

```
Forms!Форма1!ПолеСоСписком2.RowSourceType = "Таблица/запрос"
Forms!Форма1!ПолеСоСписком2.RowSource = "Сотрудники"
```

Свойство RowSourceType (функция, определяемая пользователем) - значения кодовых аргументов

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproRowSourceTypeFunctionParametersC;vastmStatic"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproRowSourceTypeFunctionParametersX":1}

Создаваемая пользователем функция Visual Basic должна принимать пять аргументов. Первый аргумент должен быть описан с типом элемент управления, а остальные иметь тип **Variant.** Функция должна возвращать значение типа **Variant**.

Синтаксис

Function *имяФункции*(*элемент* **As Control**,*метка* **As Variant**, *строка* **As Variant**, *столбец* **As Variant**,*код* **As Variant) As Variant**

Следующие пять аргументов являются обязательными.

Аргумент	Описание
элемент	Объектная переменная объекта, задающая ссылку на заполняемый список или поле со списком.
метка	Уникальное значение, определяющее заполняемый элемент управления; данный аргумент позволяет различать списки и использовать одну определяемую пользователем функцию для заполнения нескольких списков. (В примере для данного раздела уникальной меткой служит значение функции **Timer**).
строка	Номер заполняемой строки (отсчитывается от 0).
столбец	Номер заполняемого столбца (отсчитывается от 0).
код	Встроенная константа, указывающая тип запрашиваемых данных.

Примечание. Поскольку Microsoft Access при заполнении списка несколько раз вызывает функцию, создающую список, необходимо обеспечить сохранение данных между вызовами. Лучшим способом является их сохранение в переменных типа **Static.**

Microsoft Access повторно вызывает функцию с различными значениями аргумента *код*, определяющими требуемые данные. Аргумент *код* может иметь следующие значения.

Код	Объяснение	Результат функции
acLBInitialize	Инициализация	Ненулевое значение, если заполнение списка возможно; в противном случае **False** или **Null.**
AcLBOpen	Открытие	Ненулевое значение метки, если заполнение списка возможно; в противном случае **False** или **Null**.
AcLBGetRowCount	Число строк	Число строк в списке (допускается нулевое значение); -1, если число строк неизвестно.
AcLBGetColumnCount	Число столбцов	Число столбцов в списке (допускается нулевое значение); должно соответствовать значению, указанному в окне свойств.
acLBGetColumnWidth	Ширина столбца	Ширина (в единицах твип) столбца, определяемого аргументом столбец; -1 задает стандартную ширину.
acLBGetValue	Элемент списка	Элемент списка, выводящийся в позиции, определяемой аргументами

		строка и *столбец*.
acLBGetFormat	Строка форматирования	Строка форматирования, используемая для форматирования элемента списка, определяемого аргументами *строка* и *столбец*; значение -1 задает использование стандартного формата.
acLBEnd	Завершение (в последнем вызове функции заполнения списка всегда используется это значение)	Отсутствует
acLBClose	(Не используется)	Не используется.

Microsoft Access вызывает функцию по одному разу для значений кода **acLBInitialize**, **acLBOpen**, **acLBGetRowCount** и **acLBGetColumnCount**. При этом инициализируется список, открывается запрос и определяется число строк и столбцов.

С кодом **acLBGetColumnWidth** функция вызывается два раза – первый раз для определения полной ширины списка или поля со списком и второй раз для задания ширины столбцов.

Число вызовов функции с кодами **acLBGetValue** и **acLBGetFormat** для считывания и форматирования элементов списка заранее неизвестно; оно определяется числом элементов списка, прокруткой и другими факторами.

С кодом **acLBEnd** функция вызывается при закрытии формы и при каждом обновлении списка или поля со списком.

Если требуется возврат определенного значения (например, числа столбцов), а возвращается пустое (**Null**) или недопустимое значение, вызов функции с данным кодом прекращается.

Совет. Приведенная в примере конструкция **Select Case** может быть использована как шаблон для создания собственных функции, определяющих свойство **RowSourceType**.

Свойство RowSourceType (функция, определяемая пользователем) - значения кодовых аргументов, пример

Следующая функция возвращает список дат четырех ближайших понедельников. Для того чтобы вызвать эту функцию из элемента управления-списка, следует ввести имя **ListMondays** в значении свойства **Тип источника строк (RowSourceType)** и оставить значение свойства **Источник строк (RowSource)** пустым.

```
Function ListMondays(fld As Control,id As Variant,row As Variant,col As
Variant,code As Variant) As Variant
   Dim intOffset As Integer
   Select Case code
      Case acLBInitialize                ' Инициализация.
         ListMondays = True
      Case acLBOpen                      ' Открытие.
         ListMondays = Timer             ' Уникальная метка элемента
управления.
      Case acLBGetRowCount               ' Число строк.
         ListMondays = 4
      Case acLBGetColumnCount            ' Число столбцов.
         ListMondays = 1
      Case acLBGetColumnWidth            ' Ширина столбца.
         ListMondays = -1                ' Задает использование значения по
умолчанию.
      Case acLBGetValue                  ' Определятся данные.
         intOffset = Abs((9 - Weekday(Now))Mod 7)
         ListMondays = Format(Now() + intOffset + 7 * row,"mmmm d")
   End Select
End Function
```

В следующем примере для сохранения имен баз данных, содержащихся в текущем каталоге, используется статический массив. Для вызова этой функцию введите имя **ListMDBs** в качестве значения свойства **Тип источника строк (RowSourceType)** и оставьте значение свойства **Источник строк (RowSource)** пустым.

```
Function ListMDBs(fld As Control, id As Variant, row As Variant, col As
Variant, code As Variant) As Variant
   Static dbs(127) As String, Entries As Integer
   Dim ReturnVal As Variant
   ReturnVal = Null
   Select Case code
      Case acLBInitialize                ' Инициализация.
         Entries = 0
         dbs(Entries ) = Dir("*.MDB")
         Do Until dbs(Entries) = "" Or Entries >= 127
            Entries = Entries+1
            dbs(Entries) = Dir
         Loop
         ReturnVal = Entries
      Case acLBOpen                      ' Открытие.
         ReturnVal = Timer               ' Уникальная метка элемента
управления.
      Case acLBGetRowCount               ' Число строк.
         ReturnVal = Entries
      Case acLBGetColumnCount            ' Число столбцов.
         ReturnVal = 1
      Case acLBGetColumnWidth            ' Ширина столбца.
```

```
         ReturnVal = -1                ' -1 Задается значение по
умолчанию.
      Case acLBGetValue                ' Определяются данные.
         ReturnVal = dbs(row)
      Case acLBEnd                     ' Конец.
         Erase dbs
   End Select
   ListMDBs = ReturnVal
End Function
```

Свойство «Сумма с накоплением» (RunningSum)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproRunningSumC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproRunningSumX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproRunningSumA"}

Свойство **Сумма с накоплением (RunningSum)** используют для расчета в отчетах сумм с накоплением по записям или по группам. Свойство **Сумма с накоплением** позволяет указать для поля в отчете расчет суммы с накоплением и задать диапазон, по которому выполняется суммирование. Например, если данные в отчете группируются по месяцам и в области примечаний группы выводится сумма продаж за каждый месяц, то рядом с ней можно вывести общую сумму продаж с начала года (продажи за январь в январской группе, продажи за январь и февраль в февральской группе и т.д. Для этого следует добавить в область примечаний группы поле и выбрать в ячейке свойства **Сумма с накоплением (RunningSum)** значение «Для всего».

Примечание. Свойство **Сумма с накоплением (RunningSum)** применимо только к элементу управления-полю в отчете.

Значения

Свойство **Сумма с накоплением (RunningSum)** может иметь следующие значения.

Значения	Описание	Visual Basic
Отсутствует	(Значение по умолчанию). В поле выводятся данные из базового поля текущей записи.	0
Для группы	В поле выводится сумма с накоплением для данного уровня группировки. Суммирование выполняется до тех пор, пока не встретится раздел с другим уровнем группировки.	1
Для всего	В поле выводится сумма с накоплением по текущему уровню группировки. Суммирование выполняется до конца отчета.	2

Значение данного свойства задается в окне свойств поля, в макросе или в программе Visual Basic. Значение свойства **Сумма с накоплением (RunningSum)** можно задать только в режиме конструктора отчета.

Дополнительные сведения

Для расчета сумм с накоплением по записям поле следует помещать в область данных. Например, для того чтобы подсчитать число записей, выводящихся в области данных группы, введите в ячейку свойства **Данные (ControlSource)** выражение **=Sum(1)** и выберите для свойства **Сумма с накоплением (RunningSum)** значение «Для группы».

Для того чтобы рассчитать сумму с накоплением по группам следует помещать поле в область заголовка или примечаний группы.

Один отчет может содержать до 10 вложенных уровней группировки.

Свойство «Сумма с накоплением» (RunningSum), пример отчета о продажах

В данном примере показан отчет с двумя уровнями группировки: «Месяц» и «Тип». Каждое поле в заголовке группы «Тип» («Январь; Чай», «Январь; Кофе» и т.п.) содержит следующее выражение:

```
=Sum([Цена])
```

Свойство **Сумма с накоплением (RunningSum)** имеет разные значения для каждого поля: «Отсутствует», «Для группы» и «Для всего».

Группы	**Нет**	**Группа**	**Весь**
Месяц: Январь			
Тип: Чай			
02-янв	25		
22-янв	75		
Январь; Чай	100	Январь: 100	Итого: 100
Тип: Кофе			
03-янв	90		
20-янв	60		
Январь; Кофе	150	Январь: 250	Итого: 250
Месяц: Февраль			
Тип: Чай			
05-фев	70		
19-фев	80		
Февраль; Чай	150	Февраль: 150	Итого: 400
Тип: Кофе			
12-фев	150		
22-фев	100		
Февраль; Кофе	250	Февраль: 400	Итого: 650

Свойство «Сумма с накоплением» (RunningSum), пример инструкции Visual Basic)

В данном примере для свойства **Сумма с накоплением (RunningSum)** поля «Итого» задается значение 2 (Для всего).

```
Reports!Продажи!Итого.RunningSum = 2
```

Свойство Dirty

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDirtyC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDirtyX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproDirtyA"}

Свойство **Dirty** указывает, содержит ли текущая запись несохраненные изменения. Например, это свойство позволяет проверить, случайно или преднамеренно была изменена запись, и при необходимости, перейти на следующую запись без сохранения изменений.

Значения

Свойство **Dirty** может иметь следующие значения.

Значение	Описание	Visual Basic
True	Текущая запись была изменена.	**True** (–1)
False	Текущая запись не была изменена.	**False** (0)

Данное свойство доступно в режиме формы и в режиме таблицы.

Значение этого свойства может быть определено или считано с помощью макроса или в программе Visual Basic.

Дополнительные сведения

При сохранении записи свойство **Dirty** автоматически принимает значение **False**. Как только в запись будут внесены изменения, это свойство автоматически примет значение **True**.

Свойство Dirty, пример

В данном примере кнопка `Отменить` активизируется при изменении данных. Событие **После обновления (AfterUpdate)** поля приводит к вызову подпрограммы `UndoEdits( )`. Нажатие активизированной кнопки `Отменить` исходное значение поля с помощью свойства **OldValue**.

```
Sub UndoEdits()
   If Me.Dirty Then
      Me!Отменить.Enabled = True   ' Активизирует кнопку.
   Else
      Me!Отменить.Enabled = False  ' Отключает кнопку.
   End If
End Sub

Sub Отменить_Click()
   Dim ctlC As Control
      ' Для каждого элемента управления.
      For Each ctlC in Me.Controls
         If ctlC.ControlType = acTextBox Then
            ' Восстанавливаем исходное значение.
            ctlC.Value = ctlC.OldValue
         End If
      Next ctlC
End Sub
```

Свойство «Пустые строки» (AllowZeroLength)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproAllowZeroLengthC;daproAllowZeroLength;daproRequired"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowZeroLengthX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproAllowZeroLengthA"}

Свойство **Пустые строки (AllowZeroLength)** определяет, допускается ли ввод в данное поле пустых строк ("").

Примечание. Свойство **Пустые строки (AllowZeroLength)** определено только для полей таблиц с типом данных «Текстовый», «Поле MEMO» или «Гиперссылка».

Значения

Свойство **Пустые строки (AllowZeroLength)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Пустые строки являются допустимыми значениями.	**True** (–1)
Нет	(Значение по умолчанию). Пустые строки не являются допустимыми значениями.	**False** (0)

Значение данного свойства задается в окне свойств таблицы или в программе Visual Basic.

Примечание. Для доступа к свойству поля **Пустые строки (AllowZeroLength)** в программе Visual Basic следует использовать свойство **AllowZeroLength** объектов доступа к данным.

Дополнительные сведения

При задании значения «Да» для свойств **Пустые строки (AllowZeroLength)** и **Обязательное поле (Required)** Microsoft Access различает несуществующие данные (сохраняемые в виде пустых строк) и данные, которые существуют, но не известны (сохраняются в виде пустых (**Null**) значений).

В следующей таблице показано, к каким результатам приводит комбинирование значений свойств **Пустые строки (AllowZeroLength)** и **Обязательное поле (Required)**.

Пустые строки	Обязательное поле	Действия пользователя	Значение поля
Нет	Нет	Нажимает клавишу ENTER	**Null**
		Нажимает клавишу ПРОБЕЛ	**Null**
		Вводит пустую строку	(не допускается)
Да	Нет	Нажимает клавишу ENTER	**Null**
		Нажимает клавишу ПРОБЕЛ	**Null**
		Вводит пустую строку	Пустая строка
Нет	Да	Нажимает клавишу ENTER	(не допускается)
		Нажимает клавишу ПРОБЕЛ	(не допускается)
		Вводит пустую строку	(не допускается)
Да	Да	Нажимает клавишу ENTER	(не допускается)
		Нажимает клавишу ПРОБЕЛ	Пустая строка
		Вводит пустую строку	Пустая строка

Совет. Для различия пустых строк от значений **Null** используется свойство **Формат поля (Format).** При этом вместо пустых строк можно выводить строку «Пусто».

Свойства **Пустые строки (AllowZeroLength)** и **Обязательное поле (Required)** используются независимо друг от друга. Свойство **Обязательное поле** определяет, являются ли допустимыми пустые (**Null**) значения поля. Если для свойства **Пустые строки** задано значение

«Да», то пустые строки становятся допустимыми значениями данного поля вне зависимости от значения свойства **Обязательное поле**.

Свойство «Подпись» (Caption)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCaptionC;daproDescription"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproCaptionX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproCaptionA"}

Свойство **Подпись (Caption)** определяет текст, который выводится в подписях объектов в различных режимах.

- Подпись поля указывает текст, который выводится в подписях, присоединенных к элементам управления, создаваемых путем переноса с помощью мыши из списка полей. Этот текст также выводится для таблицы или запроса в заголовке столбца в режиме таблицы.
- Подпись формы указывает текст, который выводится в строке заголовка в режиме формы.
- Подпись отчета указывает заголовок отчета, выводящийся в режиме предварительного просмотра.
- Подпись кнопки и надписи указывает текст, который выводится в элементе управления.

Значения

Значением свойства **Подпись (Caption)** является строковое выражение, длиной до 2048 символов. Подписи форм или отчетов, которые не умещаются в строке заголовка, обрезаются.

Для элементов управления значение данного свойства задается в окне свойств. Для полей значение данного свойства задается в окне свойств в режиме конструктора таблицы или в окне запроса (в окне свойств списка полей). Кроме того, можно задать значения данного свойства в макросе или в программе Visual Basic.

Дополнительные сведения

Если пользователь не указывает текст подписи поля таблицы, то в качестве текста подписи элемента управления и заголовка столбца в режиме таблицы используется значение свойства **Имя поля (FieldName).** Если не задается значение подписи поля запроса, то в качестве подписи используется имя базового поля. Если подпись формы, кнопки или элемента управления-надписи не определена, то Microsoft Access присваивает объекту уникальное имя, определяемое типом объекта, например, «Форма1».

Если не указывается значение свойства **Подпись (Caption)** для элемента управления, создаваемого путем переноса с помощью мыши из списка полей, то значение свойства **Имя поля (FieldName)** копируется в свойство **Имя (Name)** и выводится в подписи элемента управления.

Примечание. Значением свойства **Подпись (Caption)** для элементов управления подпись и кнопка является отображаемый текст гиперссылки, если для этих элементов управления установлены свойства **Адрес гиперссылки (HyperlinkAddress)** или **Дополнительный адрес (HyperlinkSubAddress).**

Свойство **Подпись (Caption)** используется при определении назначенной клавиши для элемента управления подпись или кнопка. Для этого в подпись поля следует включить амперсанд (&) непосредственно перед символом, который указывает назначенную клавишу и выводиться в подписи с подчеркиванием. Для передачи фокуса данному элементу управления следует одновременно нажать назначенную клавишу и клавишу ALT.

Совет. Если символ амперсанда должен являться частью текста подписи, введите этот символ дважды (&&). Например, чтобы создать подпись «Save & exit», следует ввести в ячейку свойства **Подпись (Caption)** текст **Save && exit**.

Свойство «Тип данных» (DataType)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDataTypeC;damthCreateField;daproSize;daproType"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDataTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDataTypeA"}

Свойство **Тип данных (DataType)** определяет тип данных, сохраняемых в поле таблицы. В каждое поле допускается ввод данных только одного типа.

Значения

Свойство **Тип данных (DataType)** может иметь следующие значения.

Тип данных	Содержимое поля	Размер
Текстовый	(Значение по умолчанию). Текст или числа, не требующие проведения расчетов, например, номера телефонов.	Число символов, не превышающее минимальное из двух значений: 255 или значение свойства **Размер поля (FieldSize).** Microsoft Access не сохраняет пустые символы в неиспользуемой части поля.
Поле MEMO	Длинный текст или комбинация текста и чисел.	До 65535 символов. (Если поле MEMO обрабатывается через объекты доступа к данным (DAO) и содержит только текст и числа, а не двоичные данные, то его размер ограничивается размером базы данных).
Числовой	Числовые данные, используемые для проведения расчетов. Подробнее об использовании конкретных числовых типов см. в разделе справки для свойства **Размер поля (FieldSize)**.	1, 2, 4 или 8 байт (16 байт только для кода репликации).
Дата/время	Даты и время, относящиеся к годам с 100 по 9999, включительно.	8 байт.
Денежный	Денежные значения и числовые данные, используемые в математических расчетах, проводящихся с точностью до 15 знаков в целой и до 4 знаков в дробной части.	8 байт.
Счетчик	Уникальные последовательно возрастающие (на 1) или случайные числа, автоматически вводящиеся при добавлении каждой новой записи в таблицу. Значения полей типа счетчика обновлять нельзя. Подробнее см.	4 байт (16 байт, если для свойства **Размер поля (FieldSize)** задано значение кода репликации).

	в разделе справки для свойства **Новые значения (NewValues).**	
Логический	Логические значения, а также поля, которые могут содержать одно из двух возможных значений (True/False, Да/Нет).	1 бит.
Поле объекта OLE	Объект (например, электронная таблица Microsoft Excel, документ Microsoft Word, рисунок, звукозапись или другие данные в двоичном формате), связанный или внедренный в таблицу Microsoft Access.	До 1 Гбайт (ограничивается объемом диска).
Гиперссылка	Строка, состоящая из букв и цифр, и представляющая адрес гиперссылки. Адрес гиперссылки может состоять максимум из трех частей: *текст* – текст, выводимый в поле или в элементе управления; *адрес* – путь к файлу (в формате пути UNC) или странице (адрес URL). *дополнительный адрес* – смещение внутри файла или страницы. Чтобы вставить адрес гиперссылки в поле или в элемент управления, выберите команду **Гиперссылка** из меню **Вставка**. Для получения дополнительных сведений см. раздел Ввод адреса гиперссылки в режиме формы и в режиме таблицы.	Каждая из трех частей в типе Гиперссылка может содержать до 2048 символов.
Мастер подстановок	Создает поле, в котором предлагается выбор значений из списка, или из поля со списком, содержащего набор постоянных значений или значений из другой таблицы. Выбор этого параметра в списке в ячейке запускает мастера подстановок, который определяет тип поля.	Тот же размер, что и у ключевого поля, используемого в подстановке (обычно 4 байт).

Значение данного свойства может быть задано только в верхней половине окна таблицы в режиме конструктора.

В программах Visual Basic для определения или указания типа объекта следует использовать свойство **Type** объектов доступа к данным для определения типа данных поля перед добавлением его в семейство **Fields**.

Дополнительные сведения

Поля MEMO, гиперссылки и объекта OLE не допускают индексирования.

Совет. Денежный тип данных рекомендуется использовать для полей, в которых планируется

хранить числовые значения с одним-четырьмя знаками в дробной части. При обработке значений полей типа «С плавающей точкой (4 байт)» (**Single**) и «С плавающей точкой (8 байт)» (**Double**) выполняются вычисления с плавающей точкой. Для значений денежных полей используются более быстрые вычисления с фиксированной точкой.

Осторожно! Изменение типа поля после ввода данных в таблицу вызовет занимающее достаточно долгое время преобразование данных при сохранении таблицы. Несовместимость существующих данных с новым значением свойства **Тип данных (DataType)** может привести к потере данных.

Для того чтобы указать специальный формат или один из встроенных форматов отображения для поля с типом «Числовой», «Денежный», «Дата/время» или «Логический», следует определить значение свойства **Формат поля (Format).**

Дополнительные сведения по выбору типа данных поля содержатся в главе 4 «Переменные, константы и типы данных» руководства *Разработка приложений для Microsoft Access 97.*

Свойство «Число десятичных знаков» (DecimalPlaces)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDecimalPlacesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDecimalPlacesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDecimalPlacesA"}

Свойство **Число десятичных знаков (DecimalPlaces)** определяет число десятичных знаков, используемое при отображении чисел.

Значения

Свойство **Число десятичных знаков (DecimalPlaces)** может иметь следующие значения.

Значение	Описание	Visual Basic
Авто	(Значение по умолчанию). Числа изображаются в соответствие со значением свойства **Формат поля (Format).**	255
0-15	Справа от десятичного разделителя выводится указанное число десятичных знаков; цифры слева от десятичного разделителя изображаются согласно значению свойства **Формат поля (Format)**.	0-15

Для элементов управления поле и поле со списком значение этого свойства задается в окне свойств элемента управления, для полей таблиц - в окне свойств таблицы. Значение этого свойства можно также установить в окне свойств полей в режиме конструктора запроса.

Совет. В окне свойств таблицы следует установить свойство **Число десятичных знаков (DecimalPlaces)**. Присоединенный элемент управления, созданный в форме или отчете, наследует значение свойства **Число десятичных знаков**, установленное в базовой таблице или запросе. Поэтому нет необходимости определять это свойство отдельно для каждого созданного присоединенного объекта.

Для элементов управления значение этого свойства можно также определить в макросе или в программе Visual Basic.

Примечание. Свойство **Число десятичных знаков (DecimalPlaces)** не учитывается, если для свойства **Формат поля (Format)** выбрано значение «Основной».

Дополнительные сведения

Значение свойства **Число десятичных знаков (DecimalPlaces)** влияет только на число отображаемых десятичных знаков и не затрагивает число сохраняемых разрядов. Изменить число разрядов в сохраняемом значении позволяет свойство **Размер поля (FieldSize).** в режиме конструктора таблицы.

Свойство **Число десятичных знаков (DecimalPlaces)** позволяет отображать числа способом, отличным от указанного в свойстве **Формат поля (Format)** и от формата, в котором они сохраняются. Например, в денежном формате выводятся только два десятичных знака (5,35 р.). Чтобы увеличить число десятичных знаков до четырех, сохранив прочие атрибуты денежного формата (например, 5,3523 р.), необходимо задать для свойства **Число десятичных знаков** значение 4.

Свойство «Описание» (Description)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproDescriptionC;damthCreateProperty;daproDescription"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDescriptionX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDescriptionA"}

Свойство **Описание (Description)** определяет текст, содержащий описание объекта, выводящегося в окне базы данных, а также описание отдельных полей таблицы или запроса.

Значения

Текст описания объекта базы данных вводится в поле **Описание** в окне диалога, которое открывается командой **Свойства** в меню **Вид**. Описание таблицы или запроса можно также ввести в окно свойств таблицы или запроса. Описание объекта выводится сразу за его именем в окне базы данных при выборе команды **Таблица** в меню **Вид**.

Описание отдельных полей таблицы или запроса вводится в верхнюю половину окна режима конструктора таблицы или в окно свойств полей в окне запроса. Максимальная длина описания составляет 255 символов.

Для того чтобы в первый раз определить значение данного свойства в программе Visual Basic, следует с помощью метода **CreateProperty** создать свойство, определяемое в приложении.

Дополнительные сведения

Описание объекта выводится в столбце «Описание», если для окна базы данных используется режим таблицы.

При создании элементов управления путем переноса поля с помощью мыши из списка полей Microsoft Access копирует значение свойства **Описание (Description)** поля в свойство **Текст строки состояния (StatusBarText)** элемента управления.

Примечание. Для связанной таблицы Microsoft Access выводит в свойстве **Описание (Description)** информацию о подключении.

Свойство «Размер поля» (FieldSize)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFieldSizeC;daproFieldSize;daproSize"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproFieldSizeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproFieldSizeA"}

Свойство **Размер поля (FieldSize)** определяет максимальный размер данных, которые могут сохраняться в полях с типом данных Текстовый, Числовой или Счетчик.

Значения

Если свойство **Тип данных (DataType)** имеет значение «Текстовый», значением данного свойства должно быть целое число в диапазоне от 0 до 255. По умолчанию задается размер 50.

Если свойство **Тип данных (DataType)** имеет значение «Счетчик», то допустимыми являются значения свойства **Размер поля (FieldSize)** «Длинное целое» или «Код репликации».

Если свойство **Тип данных (DataType)** имеет значение «Числовой», то допустимыми являются следующие значения свойства **Размер поля (FieldSize)**.

Значение	Описание	Дробная часть	Размер
Байт	Числа от 0 до 255 (без дробной части).	Отсутствует	1 байт
Целое	Числа от -32 768 до 32 767 (без дробной части).	Отсутствует	2 байта
Длинное целое	(Значение по умолчанию). Числа от -2 147 483 648 до 2 147 483 647 (без дробной части).	Отсутствует	4 байта
С плавающей точкой (4 байт)	Числа от -3,402823E38 до –1,401298E–45 для отрицательных значений и от 1,401298E–45 до 3,402823E38 для положительных.	7	4 байта
С плавающей точкой (8 байт)	Числа от -1,79769313486232E308 до –4,94065645841247E для отрицательных значений, и от 1,79769313486231E308 до 4,94065645841247E–324 для положительных.	15	8 байт
Код репликации	Уникальный глобальный идентификатор (GUID).	Не определено	16 байт

Значение данного свойства можно задать в окне свойств таблицы.

Для того чтобы в программе Visual Basic прочитать или задать максимальный размер текстового поля, следует использовать свойство объектов доступа к данным **Size.** Для полей других типов значение свойства **Size** автоматически определяется значением свойства **Type.**

Примечание. Пользователь имеет возможность указать стандартные размеры текстовых и числовых полей в группе **Размеры полей по умолчанию** на вкладке **Таблицы/запросы** в диалоговом окне **Параметры**, которое открывается командой **Параметры** в меню **Сервис**.

Дополнительные сведения

Рекомендуется задавать минимально допустимое значение свойства **Размер поля (FieldSize)**,

поскольку обработка данных меньшего размера выполняется быстрее и требует меньше памяти.

Осторожно! Преобразование большего значения свойства **Размер поля (FieldSize)** к меньшему в таблице, которая уже содержит данные, может привести к потере данных. Например, при уменьшении размера текстового поля с 255 до 50 все значения, длина которых превышает 50 символов, будут усечены.

Данные в числовом поле, которые выходят за пределы диапазона, соответствующего новому размеру поля, округляются или заменяются пустыми (**Null**) значениями. Например, при замене значения «С плавающей точкой (4 байт)» на «Целое» дробные числа будут округлены до ближайшего целого числа, а значения вне диапазона от -32 768 до 32 767 будут преобразованы в пустые значения.

Невозможно отменить изменения данных, произошедших из-за модификации свойства **Размер поля (FieldSize)**, после его сохранения в режиме конструктора таблицы.

Совет. Рекомендуется использовать денежный тип данных для полей, в которых планируется хранить числовые значения с одним-четырьмя знаками в дробной части. При обработке числовых значений из полей типа «С плавающей точкой (4 байт)» (**Single**) и «С плавающей точкой (8 байт)» (**Double**) применяются вычисления с плавающей точкой. При обработке числовых значений из денежных полей используются более быстрые вычисления с фиксированной точкой.

Свойство «Пропуск пустых полей» (IgnoreNulls)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ.
пїЅпїЅпїЅпїЅпїЅпїЅS":"acproIgnoreNullsC;dacoIIndex;daobjIndex;daproIgnoreNulls;daproPrimary;daproUnique"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"acproIgnoreNullsX":1} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"acproIgnoreNullsA"}

Свойство **Пропуск пустых полей (IgnoreNulls)** определяет, следует ли включать в индекс поля, содержащие пустые (**Null**) значения.

Значения

Свойство **Пропуск пустых полей (IgnoreNulls)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Записи, содержащие пустые (**Null**) значения, не включаются в индекс.	**True** (–1)
Нет	(Значение по умолчанию). Записи, содержащие пустые (**Null**) значения, включаются в индекс.	**False** (0)

Значение данного свойства задается в окне индексов, которое открывается в режиме конструктора таблицы, или в программе Visual Basic.

Доступ к свойству индекса **Пропуск пустых полей (IgnoreNulls)** в программах Visual Basic обеспечивает свойство **IgnoreNulls** объектов доступа к данным.

Дополнительные сведения

Создание индекса для поля ускоряет выполнение поиска в этом поле таблицы и сортировку записей по этому полю. Если ожидается, что индексированное поле будет содержать пустые (**Null**) значения, следует задать значение «Да» для свойства индекса **Пропуск пустых полей (IgnoreNulls)**, тем самым уменьшив объем индекса.

Свойство «Индексированное поле» (Indexed)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acproIndexedC;damthCreateIndex;daproIgnoreNulls;daproPrimary"} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproIndexedX":1} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproIndexedA"}

Свойство **Индексированное поле (Indexed)** определяет индекс, создаваемый по одному полю. Индекс ускоряет выполнение запросов, в которых используются индексированные поля, и операции сортировки и группировки. Например, если часто выполняется поиск по полю "Фамилия" в таблице "Сотрудники", следует создать индекс для этого поля.

Значения

Свойство **Индексированное поле (Indexed)** может иметь следующие значения.

Значения	Описание
Нет	(Значение по умолчанию). Индекс не создается.
Да (Допускаются совпадения)	В индексе допускаются повторяющиеся значения.
Да (Совпадения не допускаются)	Повторяющиеся значения в индексе не допускаются.

Значение данного свойства можно задать только в окне свойств в режиме конструктора таблицы. Индекс по одному полю может быть определен путем установки свойства **Индексированное поле (Indexed)**. Кроме того, можно выбрать команду **Индексы** в меню **Вид** или нажать кнопку «Индексы» на панели инструментов. Будет открыто окно индексов.

После определения индекса по одному полю в окне индексов свойство **Индексированное поле (Indexed)** автоматически примет значение «Да».

Для создания индекса по одному полю в программе Visual Basic следует вызвать метод **CreateIndex** объектов доступа к данным.

Дополнительные сведения

Свойство **Indexed** используется для ускорения выполнения поиска и сортировки записей по одному полю таблицы. Индексированное поле может содержать как уникальные, так и повторяющиеся значения. Например, в таблице «Сотрудники» можно создать индекс по полю «КодСотрудника», которое содержит уникальные значения кода, или по полю «Фамилия», которое может содержать повторяющиеся значения.

Примечание. Не допускается создание индексов для полей MEMO, гиперссылок и объектов OLE.

Допускается создание произвольного количества индексов. Индексы создаются при сохранении макета таблицы и автоматически обновляются при вводе и изменении записей. Пользователь может в любое время добавить новые или удалить ненужные индексы в режиме конструктора таблицы.

Совет. Параметр **Автоиндекс при импорте/создании** на вкладке **Таблицы/запросы**, доступной по команде **Параметры** в меню **Сервис** позволяет указать слова, находящиеся в начале или в конце имени поля (такие как «ключ», «код» или «номер»), которые определяют поля индекса. При импорте данных из файлов, для полей, имена которых содержат указанные слова, Microsoft Access будет автоматически создавать индексы.

Если ключ таблицы состоит из одного поля, то Microsoft Access автоматически устанавливает для свойства **Индексированное поле (Indexed)** этого поля значение «Да (Совпадения не допускаются)».

Индексы, содержащие несколько полей, следует определять в окне индексов.

Свойство «Ключевое поле» (Primary)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅ":"acproPrimaryC;daobjIndex;daproPrimary;daproUnique"}
{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅ":"acproPrimaryX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproPrimaryA"}

Свойство **Ключевое поле (Primary)** определяет, является ли данный индекс ключом таблицы. Ключ однозначно определяет каждую запись в таблице; повторяющиеся значения ключа не допускаются.

Значения

Свойство **Ключевое поле (Primary)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Индекс является ключом.	**True** (–1)
Нет	Индекс не является ключом.	**False** (0)

Существуют три способа определения свойства **Ключевое поле (Primary).**

- В режиме конструктора таблицы выделите поле или поля, которые следует назначить ключевыми, а затем нажмите кнопку **Ключевое поле**  на панели инструментов.
- Откройте окно индексов, выделите или введите имя индекса в столбец **Индекс**, а затем в списке «Свойства индекса» выберите значение «Да» для свойства **Ключевое поле**.
- Определите ключ в программе Visual Basic. Доступ к свойству индекса **Ключевое поле (Primary)** осуществляется в программе Visual Basic с помощью свойства **Primary** объектов доступа к данным.

Дополнительные сведения

Microsoft Access автоматически создает индекс для ключа таблицы и использует его для поиска записей и объединения таблиц. Ключ не может содержать пустых и повторяющихся значений. Порядок полей определяет используемый в таблице по умолчанию порядок сортировки.

Если в новой таблице не определен ключ, то при сохранении таблицы Microsoft Access открывает диалоговое окно с приглашением создать ключ автоматически. При нажатии кнопки **Да** в таблицу добавляется поле счетчика (со значением «Последовательные» свойства **Новые значения (NewValues)**), которое определяется как ключевое. При нажатии кнопки **Нет** таблица сохраняется без определения ключа.

Таблицу, в которой не определен ключ, нельзя использовать при установлении межтабличных связей; кроме того, поиск и сортировка в такой таблице выполняются медленнее.

Свойство «Обязательное поле» (Required)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproRequiredC;daproAllowZeroLength;daproRequired"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproRequiredX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproRequiredA"}

Свойство **Обязательное поле (Required)** указывает, требует ли поле обязательного ввода значения. Если это свойство имеет значение «Да», то при вводе новой записи необходимо ввести значение в это поле или в любой присоединенный к нему элемент управления. Пустые (**Null**) значения в этом поле не допускаются. Например, можно потребовать, чтобы в элементе управления «Фамилия» в каждой записи обязательно выводилась какая-либо фамилия. Чтобы позволить ввод в поле пустых значений, недостаточно указать для свойства **Обязательное поле** значение «Нет». Если определено свойство **Условие на значение (ValidationRule),** оно должно иметь вид *условиеНаЗначение* Or Is Null.

Примечание. Свойство **Обязательное поле (Required)** не определено для полей с типом «Счетчик».

Значения

Свойство **Обязательное поле (Required)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Поле должно содержать значение.	**True** (–1)
Нет	(Значение по умолчанию). Поле не обязано содержать значение. Допускаются пустые значения поля.	**False** (0)

Это свойство может быть определено для всех полей таблицы (за исключением поля счетчика) в окне свойств таблицы или в программе Visual Basic.

Примечание. Доступ к свойству поля **Обязательное поле (Required)** осуществляется в программе Visual Basic с помощью свойства **Required** объектов доступа к данным.

Дополнительные сведения

Выполнение на уровне таблицы условий, налагаемых свойством **Обязательное поле (Required)**, обеспечивается ядром базы данных Microsoft Jet. При указанном значении «Да» поле, получившее фокус, должно принять или уже иметь введенные данные при вводе данных пользователем в таблицу (или в форму в режиме формы или в режиме таблицы, если таблица является базовой таблицей формы), а также при вводе в данное поле значения в макросе, в программе Visual Basic или при импорте данных в таблицу.

Свойства **Обязательное поле (Required)** и **Пустые строки (AllowZeroLength)** позволяют различать несуществующие данные (сохраняемые в поле в виде пустых строк ("")) и данные, которые, возможно, существуют, но не известны (сохраняемые в виде пустых (**Null**) значений). Если свойство **Пустые строки** имеет значение «Да», пустые строки являются допустимыми значениями данного поля вне зависимости от значения свойства **Обязательное поле**. Если для свойства **Обязательное поле** задано значение «Да», а для свойства **Пустые строки** значение «Нет», требуется ввод в поле любого непустого значения, причем пустые строки не допускаются.

Совет. Для различного отображения пустых строк и пустых значений поля может быть использована маска ввода. Например, в этом случае удобно отображать пустые строки с помощью значения «Отсутствует».

В следующей таблице показано, к каким результатам приводит комбинирование значений свойств **Пустые строки (AllowZeroLength)** и **Обязательное поле (Required)**.

Обязательное	Пустые	Действия пользователя	Значение поля

поле	**строки**		
Нет	Нет	Нажимает клавишу ENTER	**Null**
		Нажимает клавишу ПРОБЕЛ	**Null**
		Вводит пустую строку	(не допускается)
Нет	Да	Нажимает клавишу ENTER	**Null**
		Нажимает клавишу ПРОБЕЛ	**Null**
		Вводит пустую строку	Пустая строка
Да	Нет	Нажимает клавишу ENTER	(не допускается)
		Нажимает клавишу ПРОБЕЛ	(не допускается)
		Вводит пустую строку	(не допускается)
Да	Да	Нажимает клавишу ENTER	(не допускается)
		Нажимает клавишу ПРОБЕЛ	Пустая строка
		Вводит пустую строку	Пустая строка

Если значение «Да» свойства **Обязательное поле (Required)** задается для поля в таблице, уже содержащей данные, Microsoft Access выводит приглашение проверить, соответствуют ли эти данные новым условиям. Однако у пользователя имеется возможность потребовать обязательного ввода значений в новые записи и сохранить пустые значения в имеющихся записях.

Примечание. Чтобы при связывании таблиц наложить условия целостности данных, не допускающие пустых значений, следует задать значение «Да» для свойства **Обязательное поле (Required)** для поля внешнего ключа в подчиненной таблице. После этого ядро базы данных Jet перед созданием записи в подчиненной таблице будет проверять наличие связанной записи в главной таблице. Если поле внешнего ключа является частью ключа подчиненной таблицы, это свойство определять не обязательно, поскольку в ключевых полях пустые значения не допускаются.

Свойство «Имя поля» (FieldName)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFieldNameC;daproName"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFieldNameX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproFieldNameA"}

Свойство **Имя поля (FieldName)** определяет имя поля в таблице. Например, в поле «Фамилия» таблицы «Сотрудники» записываются фамилии сотрудников.

Значения

Введите имя, удовлетворяющее соглашениям об именах объектов Microsoft Access. Это имя не должно совпадать с именем другого поля в этой таблице.

Примечание. Необходимо избегать употребления имен полей, совпадающих с именами встроенных функций или свойств Microsoft Access. Например, имя «Name» совпадает с именем встроенного свойства **Name.**

Значение данного свойства можно задать только в верхней половине окна режима конструктора таблицы или в программе Visual Basic.

Дополнительные сведения

Microsoft Access различает поля по их именам. После того как имя поля задано в режиме конструктора таблицы, становится возможным его использование в выражениях, в процедурах Visual Basic и в инструкциях SQL.

Свойство «Уникальный индекс» (Unique)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ.
пїЅпїЅпїЅпїЅпїЅ":"acproUniqueC;dacoIIndex;daobjIndex;daproPrimary;daproUniqueDAO"} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acproUniqueX":1} {ewc
HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acproUniqueA"}

Свойство **Уникальный индекс (Unique)** определяет, требуется ли уникальность значений полей индекса таблицы.

Значения

Свойство **Уникальный индекс (Unique)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Индекс является уникальным.	**True** (–1)
Нет	Индекс не является уникальным.	**False** (0)

Значение данного свойства задается в окне индексов в режиме конструктора таблицы или в программе Visual Basic.

Примечание. Для доступа к свойству **Уникальный индекс (Unique)** в программе Visual Basic следует использовать свойство **Unique** объектов доступа к данным.

Дополнительные сведения

Уникальный индекс оптимизирует поиск записей. Он состоит из одного или нескольких полей, которые однозначно определят каждую запись в таблице. Если уникальный индекс состоит из одного поля, то это поле не должно содержать повторяющихся значений. Если уникальный индекс состоит из нескольких полей, то каждое поле может содержать повторяющиеся значения, однако, каждая комбинация этих значений должна быть уникальной. Для неуникальных индексов уникальность значений полей не является необходимой.

Индекс является ключом таблицы (первичным индексом), если его свойство **Ключевое поле** (**Primary**) имеет значение «Да». Каждая таблица может иметь только один первичный индекс.

Свойство «Тип элемента управления» (DisplayControl)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproDisplayControlC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproDisplayControlX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproDisplayControlA"}

Свойство **Тип элемента управления (DisplayControl)** используется в режиме конструктора таблицы для указания элемента управления, используемого по умолчанию для отображения поля.

Значения

Значение свойства **DisplayControl** задается в окне свойств таблицы в режиме конструктора на вкладке **Подстановка** на бланке свойств поля.

В данном свойстве содержится раскрывающийся список типов элементов управления, доступных для выбранного поля. Для полей с типами «Текстовый» или «Числовой» для данного свойства возможен выбор поля, списка или поля со списком. Для логических полей возможен выбор поля, поля со списком или флажка.

Дополнительные сведения

После выбора элемента управления, для которого определяется свойство **Тип элемента управления (DisplayControl),** на вкладке **Подстановка** выводятся все дополнительные свойства, необходимые для определения конфигурации элемента управления.

Совет. Для автоматического определения свойства **Тип элемента управления (DisplayControl)** и связанных с ним свойств можно использовать мастер подстановок Microsoft Access.

Значения данного свойства и относящиеся к нему типы элементов управления влияют на отображение поля как в режиме таблицы, так и в режиме формы. Поле отображается с помощью элемента управления и значений его свойств, определенных в режиме конструктора таблицы. После переноса с помощью мыши из списка полей в режиме конструктора формы поля, для которого определено свойство **DisplayControl,** Microsoft Access копирует соответствующие свойства в окно свойств элемента управления.

Свойство DatasheetCellsEffect

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDatasheetCellsEffectC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDatasheetCellsEffectX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDatasheetCellsEffectA;daobjQueryDef;daobjTableDef"}

Свойство **DatasheetCellsEffect** определяет использование при оформления ячеек таблицы специальных эффектов.

Примечание. Данное свойство применимо только к объектам в режиме таблицы.

Значения

Свойство **DatasheetCellsEffect** может иметь следующие значения.

Значение	Описание	Visual Basic	Значение
Обычное	(Значение по умолчанию). Специальные эффекты при оформлении ячеек таблицы не используются.	**acEffectNormal**	0
Приподнятое	Ячейки таблицы изображаются приподнятыми.	**acEffectRaised**	1
Утопленные	Ячейки таблицы изображаются утопленными.	**acEffectSunken**	2

Для задания значения данного свойства можно использовать кнопку **Оформление** на панели инструментов **Форматирование (режим таблицы),** макрос или программу Visual Basic.

Значение данного свойства также задается путем указания типа выделения ячейки в области **Оформление** диалогового окна **Вид сетки,** вызываемого командой **Ячейки** из меню **Формат.**

Используемые по умолчанию значения свойства **DatasheetCellsEffect** задаются на вкладке **Режим таблицы** диалогового окна **Параметры,** вызываемого командой **Параметры** из меню **Сервис.**

Дополнительные сведения

Данное свойство применяет выбранный эффект ко всей таблице.

Если для свойства **DatasheetCellsEffect** выбрано значение «Приподнятое» или «Утопленное», линии сетки будут выводиться в режиме таблицы вне зависимости от значения свойства **DatasheetGridlinesBehavior.**

В таблице приведены свойства, не определенные в семействе **Properties** для объектов **TableDef** и **QueryDef** до тех пор, пока они не будут добавлены при помощи метода **CreateProperty** или установлены при помощи панели инструментов **Форматирование (Режим таблицы).**

DatasheetFontItalic* **DatasheetForeColor***
DatasheetFontHeight* **DatasheetBackColor**
DatasheetFontName* **DatasheetGridlinesColor**
DatasheetFontUnderline* **DatasheetGridlinesBehavior**
DatasheetFontWeight* **DatasheetCellsEffect**

Примечание. При добавлении или установке свойства, помеченного в списке звездочкой, Microsoft Access автоматически добавляет все помеченные таким образом свойства в семейство **Properties.**

Свойство DatasheetGridlinesColor

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDatasheetGridlinesColorC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproDatasheetGridLinesColorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproDatasheetGridlinesColorA;daobjQueryDef;daobjTableDef"}

Свойство **DatasheetGridlinesColor** позволяет задать цвет линий сетки таблицы.

Примечание. Данное свойство применимо только к объектам в режиме таблицы.

Значения

Свойство **DatasheetGridlinesColor** определяется с помощью значения типа **Long Integer.** В программе Visual Basic для определения значения данного свойства допускается использование функций **RGB** и **QBColor.**

Значение свойства **DatasheetGridlinesColor** задается с помощью кнопки **Цвет линии/границы** ... с панели инструментов **Форматирование (Режим таблицы),** в макросе, или в программе Visual Basic.

Значение данного свойства также задается в диалоговом окне **Вид сетки,** вызываемом командой **Ячейки** из меню **Формат,** в котором допускается выбор допустимого цвета из выпадающего списка в области **Цвет линий сетки**.

Используемое по умолчанию значение свойства **DatasheetGridlinesColor** задается в области **Цвет по умолчанию** на вкладке **Режим таблицы** диалогового окна **Параметры**, вызываемого командой **Параметры** из меню **Сервис**.

Дополнительные сведения

Значение данного свойства присваивает выбранный цвет сетки для всей таблицы. Задание отдельных цветов для обрамления ячеек в режиме таблицы невозможно.

В таблице приведены свойства, не определенные в семействе **Properties** для объектов **TableDef** и **QueryDef** до тех пор, пока они не будут добавлены при помощи метода **CreateProperty** или установлены при помощи панели инструментов **Форматирование (Режим таблицы).**

DatasheetFontItalic*	**DatasheetForeColor***
DatasheetFontHeight*	**DatasheetBackColor**
DatasheetFontName*	**DatasheetGridlinesColor**
DatasheetFontUnderline*	**DatasheetGridlinesBehavior**
DatasheetFontWeight*	**DatasheetCellsEffect**

Примечание. При добавлении или установке свойства, помеченного в списке звездочкой, Microsoft Access автоматически добавляет все помеченные таким образом свойства в семейство **Properties.**

Свойство DatasheetGridlinesBehavior

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDatasheetGridlinesBehaviorC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproDatasheetGridLinesBehaviorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproDatasheetGridlinesBehaviorA;daobjQueryDef;daobjTableDef"}

Свойство **DatasheetGridlinesBehavior** определяет вывод линий сетки в режиме таблицы.

Примечание. Данное свойство применимо только к объектам в режиме таблицы.

Значения

Свойство **DatasheetGridlinesBehavior** может иметь следующие значения.

Значение	Описание	Visual Basic	Значение
Отсутствует	Линии сетки не выводятся.	**acGridlinesNone**	0
По горизонтали	Выводятся только горизонтальные линии сетки.	**acGridlinesHoriz**	1
По вертикали	Выводятся только вертикальные линии сетки.	**acGridlinesVert**	2
Все	(Значение по умолчанию). Выводятся горизонтальные и вертикальные линии сетки.	**acGridlinesBoth**	3

Значение данного свойства задается с помощью кнопки **Сетка** панели инструментов **Форматирование (Режим таблицы),** в макросе или в программе Visual Basic.

Значение данного свойства также задается путем выбора значений из области **Линии сетки** диалогового окна **Вид сетки**, вызываемого командой **Ячейки** из меню **Формат.**

Используемые по умолчанию значения свойства **DatasheetGridlinesBehavior** задаются в области **Линии сетки по умолчанию** на вкладке **Режим таблицы** диалогового окна **Параметры,** вызываемого командой **Параметры** из меню **Сервис.**

Дополнительные сведения

Результат изменения данного свойства становится видимым только при заданном для свойства **DatasheetCellsEffect** значении «Обычное».

В таблице приведены свойства, не определенные в семействе Properties для объектов **TableDef** и **QueryDef** до тех пор, пока они не будут добавлены при помощи метода **CreateProperty** или установлены при помощи панели инструментов **Форматирование (Режим таблицы).**

DatasheetFontItalic*	**DatasheetForeColor***
DatasheetFontHeight*	**DatasheetBackColor**
DatasheetFontName*	**DatasheetGridlinesColor**
DatasheetFontUnderline*	**DatasheetGridlinesBehavior**
DatasheetFontWeight*	**DatasheetCellsEffect**

Примечание. При добавлении или установке свойства, помеченного в списке звездочкой, Microsoft Access автоматически добавляет все помеченные таким образом свойства в

семейство **Properties.**

Свойство «Новые значения» (NewValues)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproNewValuesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproNewValuesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproNewValuesA"}

Свойство **Новые значения (NewValues)** определяет способ увеличения значения поля счетчика при добавлении в таблицу новых записей.

Примечание. Данное свойство применимо только к полям таблицы, имеющим тип **Счетчик.**

Значения

Свойство **Новые значения (NewValues)** может иметь следующие значения.

Значения	Описание
Последовательные	(Значение по умолчанию). Значение поля счетчика увеличивается на 1 в каждой новой записи.
Случайные	Поле счетчика в новой записи получает случайное значение типа **Long Integer.**

Значение данного свойства задается в окне свойств таблицы в режиме конструктора таблицы на вкладке **Общие** в области свойств поля.

Дополнительные сведения

При репликации базы данных для полей счетчика используется значение «Случайные», обеспечивающее уникальность номеров новых записей, добавляемых в разные реплики.

Свойства «Подписи с двоеточием» (AddColon), «Добавление подписи» (AutoLabel)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproAddColonAutoLabelC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproAddColonAutoLabelX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproAddColonAutoLabelA"}

- Свойство **Подписи с двоеточием (AddColon)** определяет автоматическое добавление двоеточия к тексту подписей для новых элементов управления.
- Свойство **Добавление подписи (AutoLabel)** определяет автоматическое создание подписи, связанной с новым элементом управления.

Значения

Свойство **Подписи с двоеточием (AddColon)** может иметь следующие значения.

Значение	Описание
Да	Для нового элемента управления после текста подписи автоматически добавляется двоеточие.
Нет	Автоматического добавления двоеточия не происходит.

Свойство **Добавление подписи (AutoLabel)** может иметь следующие значения.

Значение	Описание
Да	Новый элемент управления автоматически создается со связанной с ним подписью.
Нет	Автоматического создания связанных подписей не происходит.

Эти свойства могут быть определены только в окне стандартных свойств или при помощи метода Visual Basic **DefaultControl.**

Дополнительные сведения

Изменение стандартных свойств элемента управления влияет только на элементы управления, которые будут созданы в текущей форме или отчете. Для получения сведений о том, как изменить стандартные свойства элемента управления для всех новых форм и отчетов, созданных без помощи мастера, см. раздел Создание нового шаблона для форм и отчетов.

Свойства «Курсив» (FontItalic), «Подчеркнутый» (FontUnderline)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFontItalicUnderlineC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFontItalicUnderlineX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproFontItalicUnderlineA"}

Свойства **Курсив (FontItalic)** и **Подчеркнутый (FontUnderline)** позволяют задать соответственно наклонный и подчеркнутый стиль шрифта в следующих ситуациях:

- при выводе на экран или при печати элементов управления в формах и отчетах;
- при использовании в отчете метода **Print.**

Значения

Свойство **Курсив (FontItalic)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Добавляется атрибут наклонного шрифта.	**True** (–1)
Нет	(Значение по умолчанию). Атрибут наклонного шрифта не добавляется.	**False** (0)

Свойство **Подчеркнутый (FontUnderline)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Добавляется атрибут подчеркнутого шрифта.	**True**
Нет	(Значение по умолчанию). Атрибут подчеркнутого шрифта не добавляется.	**False**

Для элементов управления форм и отчетов значения данных свойств задаются в окне свойств, в макросе или в программе Visual Basic.

Использование данных свойств в отчетах допустимо только в процедуре обработки событий или в макросе, указанном в свойстве события **Печать (OnPrint)**.

Кроме того, значения данных свойств можно определить при помощи кнопок **Курсив** ... и **Подчеркнутый**

... на панели инструментов **Форматирование (Форма/отчет).**
Для элементов управления можно указать используемые по умолчанию значения этих свойств в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

Если на вкладке **Гиперссылки/HTML** диалогового окна **Параметры** установлен флажок **Подчеркивание,** то для полей, полей со списком, надписей и кнопок, содержащих гиперссылки, автоматически устанавливается свойство **Подчеркнутый (FontUnderline).** Для вызова данного окна используется команда **Параметры** из меню **Сервис.** При снятии гиперссылки с элемента управления (например, путем задания в качестве значения свойства присоединенного поля **Данные (ControlSource)** источника, не являющегося полем гиперссылки), автоматически восстанавливается стандартное значение свойства **Подчеркивание (FontUnderline).** Свойство **Подчеркивание (FontUnderline)** применимо только для кнопок, содержащих надпись, а не рисунок.

Свойства «Шрифт» (FontName), «Размер шрифта» (FontSize)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproFontNameSizeC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproFontNameSizeX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproFontNameSizeA"}

Свойства **Шрифт (FontName)** и **Размер шрифта (FontSize)** определяют соответственно размер шрифта и пункта текста в следующих ситуациях:

- при просмотре или выводе на печать элементов управления в форме или в отчете;
- при выводе отчета на печать с помощью метода **Print.**

Значения

Значение свойства **Шрифт (FontName)** задает имя шрифта, используемого при выводе текста.

Свойство **Размер шрифта (FontSize)** может иметь следующие значения.

Значение	Описание
8	8 пунктов (используется по умолчанию для всех отчетов и элементов управления, кроме кнопок).
10	10 пунктов (используется по умолчанию для кнопок).
другие размеры	Текст выводится шрифтом указанного размера.

Для элементов управления в формах и в отчетах значения данных свойств задаются в окне свойств, в макросе или в программе Visual Basic.

Кроме того значения этих свойств можно определить списка **Шрифт** ... и списка **Размер** ... на панели инструментов **Форматирование (форма/отчет).**

Для элементов управления можно указать используемые по умолчанию значения этих свойств в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Для отчетов значения данных свойств можно задать только в процедуре обработки события или в макросе, указанном в свойстве события **Печать (OnPrint).**

В программе Visual Basic значение свойства **Шрифт (FontName)** задается с помощью строкового выражения, содержащего имя шрифта, а значение свойства **Размер шрифта (FontSize)** с помощью числового выражения, равного нужному размеру. Диапазон значений свойства **Размер шрифта (FontSize)** от 1 до 127 включительно.

Дополнительные сведения

Доступность шрифтов зависит от системы и от принтера. Недоступные и неустановленные шрифты автоматически заменяются похожими.

Свойство «Насыщенность» (FontWeight)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproFontWeightC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproFontWeightX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproFontWeightA"}

Свойство **Насыщенность (FontWeight)** определяет ширину линии, которая используется для изображения и печати символов в элементе управления.

Значения

Свойство **Насыщенность (FontWeight)** может иметь следующие значения.

Значение	Visual Basic
Тонкий	100
Сверхсветлый	200
Светлый	300
Обычный	400
Средний	500
Плотный	600
Полужирный	700
Жирный	800
Сверхжирный	900

Значение данного свойства задается в окне свойств элемента управления, в макросе или в программе Visual Basic. Кроме того можно воспользоваться кнопкой **Полужирный** ... с панели инструментов **Форматирование (форма/отчет),** устанавливающей для свойства **Насыщенность (FontWeight)** значение полужирный (700).

Для элементов управления можно указать используемые по умолчанию значения этих свойств в окне стандартных свойств или с помощью метода Visual Basic **DefaultControl.**

Дополнительные сведения

В зависимости от параметров компьютера и принтера, экранные шрифты могут отличаться от шрифтов принтера. Например, шрифт с заданным для свойства **Насыщенность (FontWeight)** значением «Тонкий» может на экране выглядеть так же, как шрифт с атрибутом насыщенности «Обычный», но печататься более тонкими линиями.

Для задания полужирных линий для элемента управления или для отчета можно использовать свойство **FontBold,** доступное только в макросах и в программах Visual Basic. Свойство **FontBold** позволяет быстро установить для текста атрибут полужирного шрифта. Свойство **Насыщенность (FontWeight)** предоставляет лучший механизм управления толщиной линий при выводе текста на экран. Отношения между значениями двух этих свойств приведены в таблице.

Если	То тогда
FontBold = **False** (0)	**FontWeight** = Обычный (400)
FontBold = **True** (–1)	**FontWeight** = Полужирный (700)
FontWeight < 700	**FontBold** = **False**
FontWeight > = 700	**FontBold** = **True**

Свойства «Выравнивание подписи» (LabelAlign), «Выравнивание текста» (TextAlign)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproLabelTextAlignC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproLabelTextAlignX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproLabelTextAlignA":1}

- Свойство **Выравнивание подписи (LabelAlign)** позволяет указать тип выравнивания текста подписи, связанной с новым элементом управления.
- Свойство **Выравнивание текста (TextAlign)** позволяет указать тип выравнивания текста в новом элементе управления.

Значения

Свойство **Выравнивание подписи (LabelAlign)** может иметь следующие значения.

Значение	Описание
Обычное	(Значение по умолчанию). Текст подписи выравнивается по левому краю.
По левому краю	Текст подписи выравнивается по левому краю.
По центру	Текст подписи выравнивается по центру.
По правому краю	Текст подписи выравнивается по правому краю.

Значение свойства **Выравнивание подписи (LabelAlign)** можно задать в окне стандартных свойств элемента управления или при помощи метода Visual Basic **DefaultControl.**

Свойство **Выравнивание текста (TextAlign)** может иметь следующие значения.

Значение	Описание	Visual Basic
Обычное	(Значение по умолчанию). Текст выравнивается по левому краю; числа и даты выравниваются по правому краю.	0
По левому краю	Данные выравниваются по левому краю.	1
По центру	Данные выравниваются по центру.	2
По правому краю	Данные выравниваются по правому краю.	3

Значение свойства **Выравнивание текста (TextAlign)** задается в окне свойств элемента управления, в макросе или в программе Visual Basic.

Кроме того, задать значение свойства **Выравнивание текста (TextAlign)** можно при помощи кнопок **Выровнять по левому краю**  , **По центру**  и **Выровнять по правому краю**  на панели инструментов **Форматирование (форма/отчет).**

Значение по умолчанию для свойства **Выравнивание текста (TextAlign)** можно задать в окне стандартных свойств элемента управления или при помощи метода Visual Basic **DefaultControl.**

Дополнительные сведения

Пока для элементов управления установлено свойство **Добавление подписи (AutoLabel),** они создаются с присоединенной подписью по умолчанию. Изменение значения свойства **Выравнивание подписи (LabelAlign)** влияет только на элементы управления, которые будут созданы в текущей форме или отчете. Для получения сведений о том, как изменить

стандартные свойства элемента управления для всех новых форм и отчетов, созданных без помощи мастера, см. раздел Создание нового шаблона для форм и отчетов.

Свойства «Выравнивание подписи» (LabelAlign), «Выравнивание текста» (TextAlign) - Применение

Свойство **Выравнивание подписи (LabelAlign)**

Свойство **Выравнивание текста (TextAlign)**

Свойство FontBold

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFontBoldC":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFontBoldX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acproFontBoldA"}

Свойство **FontBold** задает использование полужирного шрифта в следующих ситуациях:

- при просмотре или выводе на печать элементов управления в форме или в отчете;
- при выводе отчета на печать с помощью метода **Print.**

Значения

Свойство **FontBold** может иметь следующие значения.

Значение	Описание
True (–1)	Текст выводится полужирным шрифтом.
False (0)	(Значение по умолчанию) Текст выводится без атрибута полужирного шрифта.

Значение свойства **FontBold** задается только в макросе и в программе Visual Basic.

Дополнительные сведения

Для применения свойства **FontBold** в отчете следует создать процедуру обработки события **Печать (Print).**

В зависимости от параметров компьютера и принтера, экранные шрифты могут отличаться от шрифтов принтера.

Свойство **Насыщенность (FontWeight)** может использоваться для задания толщины текста в элементах управления, в окне свойств которых оно доступно. Свойство **FontBold** позволяет быстро установить для текста атрибут полужирного шрифта. Свойство **Насыщенность (FontWeight)** предоставляет лучший механизм управления толщиной линий при выводе текста на экран. Отношения между значениями двух этих свойств приведены в таблице.

Если	То тогда
FontBold = **False**	**FontWeight** = Обычный (400)
FontBold = **True**	**FontWeight** = Полужирный (700)
FontWeight < 700	**FontBold** = **False**
FontWeight > = 700	**FontBold** = **True**

Свойство FontBold, пример

В следующей процедуре обработки события заголовок отчета и текущая дата печатаются полужирным шрифтом в позиции, определяемой значениями свойств **CurrentX** и **CurrentY**.

```
Private Sub ReportHeader0_Print(Cancel As Integer, PrintCount As Integer)
   Dim MyDate

   MyDate = Date
   Me.FontBold = True
   Me.Print("Отчет 'Управление продажами'")
' Заголовок отчета печатается полужирным шрифтом.
   Me.Print(MyDate)
End Sub
```

Свойство CountOfLines

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCountOfLinesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproCountOfLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproCountOfLinesA"}

Свойство **CountOfLines** возвращает значение типа **Long** и указывает число строк в стандартном модуле или в модуле класса.

Значения

Свойство **CountOfLines** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Нумерация строк в модуле начинается с 1.

Значение номера последней строки в модуле и является значением свойства **CountOfLines**.

Свойства CountOfLines и CountOfDeclarationLines, пример

В приведенном ниже примере подсчитано число строк в модуле и число строк описания для каждого стандартного модуля в семействе **Modules**. Семейство **Modules** содержит только модули, открытые в редакторе модулей.

```
Function ModuleLineTotal(strModuleName As String)
   Dim mdl As Module

   ' Открывает модуль, чтобы включить его в семейство модулей.
   DoCmd.OpenModule strModuleName
   ' Возврат ссылки на объект Module.
   Set mdl = Modules(strModuleName)
   ' Печать количества строк в модуле.
   Debug.Print "Количество строк: ", mdl.CountOfLines
   ' Печать количества строк описания.
   Debug.Print "Количество строк описания: ", _
      mdl.CountOfDeclarationLines
End Function
```

Свойство CountOfDeclarationLines

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCountOfDeclarationLinesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproCountOfDeclarationLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproCountOfDeclarationLinesA"}

Свойство **CountOfDeclarationLines** возвращает значение типа **Long** и указывает число строк в программе в разделе описаний в стандартном модуле или в модуле класса.

Значения

Свойство **CountOfDeclarationLines** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Нумерация строк в модуле начинается с 1.

Значение свойства **CountOfDeclarationLines** соответствует номеру последней строки в разделе описаний. Данное свойство используется для определения конца раздела описаний и начала тела модуля.

Свойство GUID

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproGUIDC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproGUIDX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproGUIDA"}

Свойство **GUID** объекта **Reference** возвращает код GUID, который определяет тип библиотеки в системном реестре Windows.

Значения

Свойство **GUID** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Каждому типу библиотек соответствует определенный код GUID, который хранится в системном реестре. При указании ссылки на библиотеку ее тип определяется с помощью кода GUID.

Для создания объекта **Reference** на основе типа библиотеки GUID можно использовать метод **AddFromGUID.**

Свойство IsBroken

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproIsBrokenC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproIsBrokenX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproIsBrokenA"}

Свойство **IsBroken** возвращает значение типа **Boolean** и определяет, указывает ли объект **Reference** на соответствующее значение в системном реестре Windows.

Значения

Свойство **IsBroken** доступно только в программах Visual Basic и допускает только чтение.

По умолчанию в свойстве **IsBroken** установлено значение **False** (0). В свойстве **IsBroken** будет установлено значение **True** (–1), только если объект **Reference** больше не указывает на соответствующую запись в системном реестре.

Дополнительные сведения

С помощью свойства **IsBroken** можно определить, был ли удален или перемещен в другую директорию файл, связанный с определенным объектом **Reference**.

Если для свойства **IsBroken** задано значение **True**, то при попытке обращения к свойствам **Имя (Name)** или **FullPath** возникает ошибка.

Свойство IsCompiled

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproIsCompiledC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproIsCompiledX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproIsCompiledA"}

Свойство **IsCompiled** возвращает значение типа **Boolean** и указывает, был ли скомпилирован проект Visual Basic.

Значения

Свойство **IsCompiled** доступно только в программах Visual Basic и допускает только чтение.

Свойство **IsCompiled** возвращает значение **True** (–1), если проект был скомпилирован.

Дополнительные сведения

Проект можно скомпилировать с помощью пользовательского интерфейса, если модуль находится в режиме конструктора. Для этого в меню **Отладка** нужно выбрать команду **Компилировать все модули** или **Компилировать и сохранить все модули**.

Для свойства **IsCompiled** объекта **Application** задается значение **False** (0), если проект не был до конца скомпилирован, если после компиляции модуль был добавлен, изменен или удален, а также если после компиляции модуль не был сохранен.

Свойство BuiltIn

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproBuiltInC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproBuiltInX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproBuiltInA"}

Свойство **BuiltIn** возвращает значение типа **Boolean** и определяет, указывает ли объект **Reference** на установленную по умолчанию ссылку, что необходимо для нормальной работы Microsoft Access.

Значения

Свойство **BuiltIn** доступно только в программах Visual Basic и допускает только чтение.

Свойство **BuiltIn** возвращает следующие значения.

Значение	Описание
True (–1)	Объект **Reference** указывает на используемую по умолчанию ссылку, которая не может быть удалена.
False (0)	Объект **Reference** указывает на ссылку, которая не используется по умолчанию и не является обязательной для нормальной работы Microsoft Access.

Дополнительные сведения

Используемые по умолчанию ссылки включают библиотеку объектов Microsoft Access 8.0 и библиотеку Visual Basic для приложений.

Свойство BuiltIn, пример

В приведенном ниже примере отображены используемые по умолчанию ссылки в семействе **References**:

```
Sub ReferenceBuiltInOnly()
   Dim ref As Reference

   ' Перечислять в семействе ссылок.
   For Each ref In References
      ' Проверка свойства BuiltIn.
      If ref.BuiltIn = True Then
         Debug.Print ref.Name
      End If
   Next ref
End Sub
```

Свойство FullPath

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproFullPathC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproFullPathX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproFullPathA"}

Свойство **FullPath** возвращает строку, содержащую путь и имя файла библиотеки типов, на которую указывает ссылка.

Значения

Свойство **FullPath** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Библиотеки типов находятся в файлах. В приведенной ниже таблице представлены расширения файлов, которые обычно содержат библиотеки типов.

Расширение файла	Тип файла
.olb, .tlb	Файл библиотеки типов
.mdb, .mda, .mde	База данных
.exe, .dll	Выполняемый файл
.ocx	Элемент ActiveX

Если для свойства **IsBroken** объекта **Reference** задано значение **True** (–1), то при обращении к свойству **FullPath** возникает ошибка.

Свойства FullPath, GUID, IsBroken, Major, Minor; пример

В приведенном ниже примере отображено значение свойств **FullPath**, **GUID**, **IsBroken**, **Major** и **Minor** для каждого объекта **Reference** в семействе **References.**

```
Sub ReferenceProperties()
   Dim ref As Reference

   ' Перечисление семейства ссылок.
   For Each ref In References
      ' Проверка свойства IsBroken.
      If ref.IsBroken = False Then
         Debug.Print "Имя: ", ref.Name
         Debug.Print "Путь: ", ref.FullPath
         Debug.Print "Версия: ", ref.Major & "." & ref.Minor
      Else
         Debug.Print "Коды GUID для нарушенных ссылок:"
         Debug.Print ref.GUID
      EndIf
   Next ref
End Sub
```

Свойство Lines

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproLinesC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproLinesX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproLinesA"}

Свойство **Lines** возвращает строку, в которой находится содержимое указанной строки или строк в стандартном модуле или в модуле класса.

Значения

объект.**Lines(***строка***,** *числоСтрок***)**

В свойстве **Lines** могут быть использованы следующие значения.

Значение	Описание
объект	Объект **Module.**
строка	Значение типа **Long,** указывающее номер первой строки для возврата.
числоСтрок	Значение типа **Long,** указывающее число строк для возврата.

Свойство **Lines** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Нумерация строк в модуле начинается с 1. Например, если в свойстве **Lines** аргумент *строка* имеет значение 1, и аргумент *числоСтрок* имеет значение 1, то свойство **Lines** возвращает строку, содержащую текст первой строки в модуле.

Для вставки строки текста в модуль используется метод **InsertLines.**

Свойство Major

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acproMajorC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproMajorX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproMajorA"}

Свойство **Major** объекта **Reference** возвращает значение типа **Long** и указывает номер основной версии приложения, на которое была установлена ссылка.

Значения

Свойство **Major** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Свойство **Major** возвращает значение, находящееся слева от разделяющей точки в номере версии. Например, если была установлена ссылка на приложение версии 2.5, то свойство **Major** вернет значение 2.

Свойство Minor

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproMinorC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproMinorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproMinorA"}

Свойство **Minor** объекта **Reference** возвращает значение типа **Long** и указывает дополнительный номер версии приложения, на которое установлена ссылка.

Значения

Свойство **Minor** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Свойство **Minor** возвращает значение, находящееся справа от разделяющей точки в номере версии. Например, если была установлена ссылка на приложение версии 2.5, то свойство **Minor** вернет значение 5.

Свойство ProcStartLine

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS.
nїSnїSnїSnїSnїS":"acproProcStartLineC;vastmFunction;vaSTMPropertyGet;vastmPropertyLet;vastmPropertySet;vastmSub"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acproProcStartLineX":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acproProcStartLineA"}

Свойство **ProcStartLine** возвращает значение типа **Long** и указывает строку, с которой начинается указанная процедура в стандартном модуле или в модуле класса. Процедура начинается с комментариев и констант компиляции, за которыми непосредственно следует описание процедуры, которое обозначено одной из следующих инструкций:

- инструкция **Sub**
- инструкция **Function**
- инструкция **Property Get**
- инструкция **Property Let**
- инструкция **Property Set**

Значения

объект.**ProcStartLine**(*имяПроцедуры*, *типПроцедуры*)

В свойстве **ProcStartLine** могут быть использованы следующие значения.

Значение	Описание
объект	Объект **Module.**
имяПроцедуры	Строковое выражение, указывающее имя процедуры в модуле.
типПроцедуры	Внутренняя константа, определяющая тип процедуры. Константа может иметь одно из следующих значений.

Константа	Описание
vbext_pk_Get	Процедура **Property Get**.
vbext_pk_Let	Процедура **Property Let**.
vbext_pk_Proc	Процедура **Sub** или **Function**.
vbext_pk_Set	Процедура **Property Set**.

Свойство **ProcStartLine** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Свойство **ProcStartLine** возвращает номер строки, с которой начинается процедура. Начало процедуры может содержать комментарии или константы компиляции, за которыми следует описание процедуры.

Для указания строки, с которой начинается описание процедуры, используется свойство **ProcBodyLine.** Это свойство возвращает номер строки, которая начинается с инструкций **Sub**, **Function**, **Property Get**, **Property Let** или **Property Set**.

Свойства **ProcStartLine** и **ProcBodyLine** могут иметь одинаковые значения, если в первой строке процедуры дано ее описание. Если же описание процедуры дано не в первой строке, то значение свойства **ProcBodyLine** будет превышать значение свойства **ProcStartLine**.

Примечания

- Чтобы облегчить определение начала процедуры, рекомендуется установить флажок **Разделитель процедур**. Если данный флажок установлен, то между концом одной процедуры и началом следующей проведена линия. Первая строка программы (или пустая строка), расположенная под разделителем процедур, является первой строкой следующей процедуры. Значение этой строки возвращается свойством **ProcStartLine**. Чтобы установить

флажок **Разделитель процедур,** в меню **Сервис** выберите команду **Параметры** и перейдите к вкладке **Модуль**. Прежде чем устанавливать флажок **Разделитель процедур,** необходимо установить флажок **Полный модуль**.

- Свойство **ProcStartLine** воспринимает процедуры **Sub** и **Function** одинаково, но различает типы процедуры **Property**.

Свойства ProcStartLine, ProcBodyLine, ProcCountLines, пример

В приведенном ниже примере производится печать данных об определенной процедуре в модуле в окне отладки.

```
Function ProcLineInfo(strModuleName As String, strProcName As String)
   Dim mdl As Module
   Dim lngStartLine As Long, lngBodyLine As Long
   Dim lngCount As Long, lngEndProc As Long

   ' Открывает определенный объект Module.
   DoCmd.OpenModule strModuleName
   ' Возвращает ссылку на объект Module.
   Set mdl = Modules(strModuleName)

   ' Считает строки в процедуре.
   lngCount = mdl.ProcCountLines(strProcName, vbext_pk_Proc)
   ' Определяет первую строку.
   lngStartLine = mdl.ProcStartLine(strProcName, vbext_pk_Proc)
   ' Определяет первую строку инструкций.
   lngBodyLine = mdl.ProcBodyLine(strProcName, vbext_pk_Proc)
   Debug.Print

   ' Печатает все строки процедуры, предшествующие первой строке
инструкций.
   Debug.Print "Строки, предшествующие процедуре" & strProcName & ": "
   Debug.Print mdl.Lines(lngStartLine, lngBodyLine - lngStartLine)

   ' Определяет номер последней строки в процедуре.
   lngEndProc = (lngBodyLine + lngCount - 1) - Abs(lngBodyLine -
lngStartLine)

   ' Печатает все строки тела процедуры.
   Debug.Print "Основные строки: "
   Debug.Print mdl.Lines(lngBodyLine, (lngEndProc - lngBodyLine) + 1)
End Function
```

Данную функцию можно вызвать из базы данных «Борей» с помощью следующей процедуры.

```
Sub GetProcInfo()
   ProcLineInfo "Служебные функции", "IsLoaded"
End Sub
```

Свойство ProcCountLines

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acproProcCountLinesC;vastmFunction;vaSTMPropertyGet;vastmPropertyLet;vastmPropertySet;vastmSub"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproProcCountLinesX":1} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproProcCountLinesA"}

Свойство **ProcCountLines** возвращает значение типа **Long,** содержащее число строк в определенной процедуре в стандартном модуле или в модуле класса. Процедура начинается с комментариев и констант компиляции, за которыми непосредственно следует описание процедуры, которое обозначено одной из следующих инструкций:

- инструкция **Sub**
- инструкция **Function**
- инструкция **Property Get**
- инструкция **Property Let**
- инструкция **Property Set**

Значения

объект.**ProcCountLines(***имяПроцедуры*, *типПроцедуры*)

В свойстве **ProcCountLines** могут быть использованы следующие значения.

Значение	Описание
объект	Объект **Module.**
имяПроцедуры	Строковое выражение, определяющее имя процедуры в модуле.
типПроцедуры	Внутренняя константа, указывающая тип процедуры. Константа может иметь одно из следующих значений.

Константа	Описание
vbext_pk_Get	Процедура **Property Get**.
Vbext_pk_Let	Процедура **Property Let**.
Vbext_pk_Proc	Процедура **Sub** или **Function**.
Vbext_pk_Set	Процедура **Property Set**.

Свойство **ProcCountLines** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Свойство **ProcCountLines** возвращает число строк в процедуре, начиная со строки, значение которой указано в свойстве **ProcStartLine,** и заканчивая последней строкой процедуры. Процедура может заканчиваться инструкциями `End Sub`, `End Function` или `End Property`.

Примечание. Свойство **ProcCountLines** воспринимает процедуры **Sub** и **Function** одинаково, но различает типы процедуры **Property**.

Свойство ProcBodyLine

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acproProcBodyLinesC;vastmFunction;vaSTMPropertyGet;vastmPropertyLet;vastmPropertySet;vastmSub"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproProcBodyLinesX":1} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproProcBodyLinesA"}

Свойство **ProcBodyLine** возвращает значение типа **Long,** содержащее номер строки, с которой начинается тело указанной процедуры в стандартном модуле или в модуле класса. Тело процедуры начинается с описания, которое обозначено одной из следующих инструкций:

- инструкция **Sub**
- инструкция **Function**
- инструкция **Property Get**
- инструкция **Property Let**
- инструкция **Property Set**

Значения

объект.**ProcBodyLine(***имяПроцедуры*, *типПроцедуры***)**

В свойстве **ProcBodyLine** могут быть использованы следующие значения.

Значения	**Описание**
объект	Объект **Module**.
имяПроцедуры	Строковое выражение, определяющее имя процедуры в модуле.
типПроцедуры	Внутренняя константа, указывающая тип процедуры. Константа может иметь следующие значения.

Константа	**Описание**
vbext_pk_Get	Процедура **Property Get**.
Vbext_pk_Let	Процедура **Property Let**.
Vbext_pk_Proc	Процедура **Sub** или **Function**.
Vbext_pk_Set	Процедура **Property Set**.

Свойство **ProcBodyLine** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Свойство **ProcBodyLine** возвращает число, определяющее номер строки, с которой начинается описание процедуры. Свойство же **ProcStartLine** возвращает число, указывающее номер строки, на которой заканчивается предыдущая и начинается следующая процедура. Все комментарии и константы компиляции, предшествующие описанию процедуры (телу процедуры), считаются частью процедуры, но свойство **ProcBodyLine** их игнорирует.

Примечание. Свойство **ProcBodyLine** воспринимает процедуры **Sub** и **Function** одинаково, но различает типы процедуры **Property**.

Свойство ProcOfLine

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS.
nïSnïSnïSnïSnïS":"acproProcOfLineC;vastmFunction;vaSTMPropertyGet;vastmPropertyLet;vastmPropertySet;vastmSub"}
{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproProcOfLineX":1} {ewc
HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproProcOfLineA"}

Свойство **ProcOfLine** возвращает строку, содержащую имя процедуры в стандартном модуле или в модуле класса, в которой находится указанная строка. Тело процедуры начинается с описания процедуры, которое обозначено одной из следующих инструкций:

- инструкция **Sub**
- инструкция **Function**
- инструкция **Property Get**
- инструкция **Property Let**
- инструкция **Property Set**

Значения

объект.**ProcOfLine(***строка*, *типПроцедуры***)**

Свойство **ProcOfLine** может иметь следующие значения.

Значения	Описание
объект	Объект **Module**.
строка	Значение типа **Long,** указывающее номер строки в модуле.
типПроцедуры	Значение типа **Long,** указывающее тип процедуры, содержащей строку, определенную аргументом *строка*. После определения значения свойства **ProcOfLine** аргумент *типПроцедуры* принимает значение внутренней константы, которая определяет тип процедуры, содержащей данную строку. Константа может иметь следующие значения.

Константа	Значение	Описание
vbext_pk_Get	3	Процедура **Property Get**.
Vbext_pk_Let	1	Процедура **Property Let**.
Vbext_pk_Proc	0	Процедура **Sub** или **Function**.
Vbext_pk_Set	2	Процедура **Property Set**.

Свойство **ProcOfLine** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

На любой введенный номер строки свойство **ProcOfLine** возвращает имя процедуры, содержащей данную строку. Так как за комментариями и константами компиляции непосредственно следует описание процедуры, то они считаются частью процедуры и для строки, находящейся вне тела процедуры, в свойстве **ProcOfLine** будет указано имя процедуры. Свойство **ProcStartLine** указывает строку, с которой начинается процедура; свойство **ProcBodyLine** указывает строку, с которой начинается описание (тело процедуры).

Аргумент *типПроцедуры* указывает, принадлежит ли данная строка процедуре **Sub** или **Function**, **Property Get**, **Property Let** или **Property Set**. Чтобы определить тип процедуры, содержащей данную строку, введите в свойство **ProcOfLine** переменную типа **Long** и проверьте значение данной переменной.

Примечание. Свойство **ProcBodyLine** воспринимает процедуры **Sub** и **Function** одинаково, но различает типы процедуры **Property**.

Свойство ProcOfLine, пример

В следующей процедуре функции перечислены все процедуры, содержащиеся в определенном модуле.

```
Function AllProcs(strModuleName As String)
   Dim mdl As Module
   Dim lngCount As Long, lngCountDecl As Long, lngI As Long
   Dim strProcName As String, astrProcNames() As String
   Dim intI As Integer, strMsg As String
   Dim lngR As Long

   ' Открывает указанный объект Module.
   DoCmd.OpenModule strModuleName
   ' Возвращает ссылку на объект Module.
   Set mdl = Modules(strModuleName)
   ' Подсчет строк в модуле.
   lngCount = mdl.CountOfLines
   ' Подсчет строк в разделе описаний в модуле.
   lngCountDecl = mdl.CountOfDeclarationLines
   ' Определяет имя первой процедуры.
   strProcName = mdl.ProcOfLine(lngCountDecl + 1, lngR)
   ' Настраивает переменную счетчика.
   intI = 0
   ' Повторное определение размерности массива.
   ReDim Preserve astrProcNames(intI)
   ' Запись имени первой процедуры в массив.
   astrProcNames(intI) = strProcName
   ' Определение имени процедуры для каждой строки, следующей за описанием.
   For lngI = lngCountDecl + 1 To lngCount
      ' Сравнение имени процедуры со значением свойства ProcOfLine.
      If strProcName <> mdl.ProcOfLine(lngI, lngR) Then
         ' Увеличивает значение счетчика на 1.
         intI = intI + 1
         strProcName = mdl.ProcOfLine(lngI, lngR)
         ReDim Preserve astrProcNames(intI)
         ' Записывает уникальные имена процедур в массив.
         astrProcNames(intI) = strProcName
      End If
   Next lngI
   strMsg = "Процедуры в модуле '" & strModuleName & "': " _
      & vbCrLf & vbCrLf
   For intI = 0 To UBound(astrProcNames)
      strMsg = strMsg & astrProcNames(intI) & vbCrLf
   Next intI
   ' Диалоговое окно, содержащее все процедуры в модуле.
   MsgBox strMsg
End Function
```

Вызвать данную функцию можно с помощью следующей процедуры.

```
Sub GetAllProcs()
   AllProcs "Служебные функции"
End Sub
```

Свойство Collection

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproCollectionC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproCollectionX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproCollectionA"}

Свойство **Collection** возвращает ссылку на семейство, содержащее объект.

Значения

Свойство **Collection** доступно только в программах Visual Basic и допускает только чтение.

Дополнительные сведения

Свойство **Collection** используется для доступа к семейству, содержащему объект. Например, свойство **Collection** для объекта **Reference** возвращает ссылку на объект в семействе **References.**

Свойство **Collection** аналогично свойству **Parent.**

Свойство Type

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproTypeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproTypeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproTypeA"}

Свойство **Type** используется для определения является ли модуль стандартным модулем или же модулем класса.

Значения

Свойство **Type** доступно только в программах Visual Basic и допускает только чтение.

Если модуль является стандартным, то свойство **Type** имеет значение **acStandardModule** или 0. Если же модуль является модулем класса, то свойство **Type** имеет значение **acClassModule** или 1.

Свойство Type, пример

В приведенном ниже примере определено, является ли объект **Module** стандартным модулем или же модулем класса.

```
Function CheckModuleType(strModuleName As String) As Integer
   Dim mdl As Module

   ' Открывает модуль, чтобы включить его в семейство Modules.
   DoCmd.OpenModule strModuleName
   ' Возвращает ссылку на объект Module.
   Set mdl = Modules(strModuleName)
   ' Проверяет свойство Type.
   If mdl.Type = acClassModule Then
      ' Вставляет комментарии.
      mdl.InsertLines 1, "' Модуль класса."
      CheckModuleType = acClassModule
   Else
      ' Вставляет комментарии.
      mdl.InsertLines 1, "' Стандартный модуль."
      CheckModuleType = acStandardModule
   End If
End Function
```

Свойство «Выравнивание рисунка» (PictureAlignment)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPictureAlignmentC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPictureAlignmentX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPictureAlignmentA"}

Свойство **Выравнивание рисунка (PictureAlignment)** определяет положение фонового рисунка в элементе управления-рисунке или в форме и отчете.

Значения

Свойство **Выравнивание рисунка (PictureAlignment)** может иметь следующие значения.

Значения	Описание	Visual Basic
Вверх влево	Рисунок помещается в верхний левый угол элемента управления, окна формы или страницы отчета.	0
Вверх вправо	Рисунок помещается в верхний правый угол элемента управления, окна формы или страницы отчета.	1
По центру	(Значение по умолчанию). Рисунок помещается в центр элемента управления, окна формы или страницы отчета.	2
Вниз влево	Рисунок помещается в нижний левый угол элемента управления, окна формы или страницы отчета.	3
Вниз вправо	Рисунок помещается в нижний правый угол элемента управления, окна формы или страницы отчета.	4
По центру формы	(Только для форм). Рисунок помещается по горизонтали в центре по ширине формы, а по вертикали по центру относительно полной высоты формы.	5

Значение свойства **Выравнивание рисунка (PictureAlignment)** задается в окне свойств формы или отчета, в макросе или в программе Visual Basic.

Значение по умолчанию для данного свойства может быть задано с помощью стандартного стиля элемента управления или с помощью метода **DefaultControl** из Visual Basic.

Значение данного свойства можно задать в любом режиме.

Дополнительные сведения

При значении «По центру формы» рисунок выравнивается по центру самой формы. При всех других значениях свойства **Выравнивание рисунка (PictureAlignment)** рисунок выравнивается относительно окна формы. Для того чтобы поместить рисунок в нужное место формы или разместить мозаику относительно формы, следует задать для свойства **Выравнивание рисунка (PictureAlignment)** значение «По центру формы».

Для отчетов рисунок всегда размещается относительно полной страницы, а не относительно реальных размеров отчета. Если размеры отчета меньше размеров полной страницы, и требуется поместить рисунок в положение, недоступное для значений свойства **Выравнивание рисунка (PictureAlignment)**, следует изменить положение элемента управления-рисунка.

Если для свойства **Мозаичное заполнение** (**PictureTiling**) задано значение «Да», мозаичное заполнение формы или отчета начинается с положения рисунка, определяемого значением свойства **Выравнивание рисунка (PictureAlignment)**.

Свойство PictureData

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPictureDataC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acproPictureDataX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acproPictureDataA"}

Свойство **PictureData** используется для копирования рисунка из формы, отчета или элемента управления в другой объект, поддерживающий свойство **Рисунок** (**Picture**).

Значения

В качестве значения свойства **PictureData** следует задать значение свойства **PictureData** исходного элемента управления: рисунка, кнопки, выключателя формы или отчета.

Значение данного свойства задается в программе Visual Basic.

Дополнительные сведения

Данное свойство позволяет вывести форму с разными фоновыми рисунками в зависимости от режима работы пользователя. Например, можно открывать форму «Клиенты» с одним фоновым рисунком для режима ввода данных и с другим для режима просмотра.

Кроме того, свойство **PictureData** используется вместе с событием **Таймер (Timer)** и свойством **Интервал таймера** (**TimerInterval**) для простой анимации рисунков в форме.

Свойство PictureData, пример

В следующем примере для создания в форме анимации порхающей бабочки используются три элемента управления-рисунка. Элемент управления `Скрытый1` содержит рисунок бабочки с поднятыми крыльями, а элемент управления `Скрытый2` - с опущенными. У обоих рисунков свойство **Вывод на экран (Visible)** имеет значение **False** (0). Свойство **Интервал таймера (TimerInterval)** получает значение 200. При каждом событии **Таймер (Timer)** изменяется рисунок, выводящийся в элементе управления `Видимый1`, для которого попеременно задается значение свойства **PictureData** одного из скрытых элементов управления. При этом видимый рисунок сдвигается на 200 твип вправо. Когда значение свойства **От левого края (Left)** превысит значение ширины формы, сохраняемое в общей переменной `gfrmWidth`, видимый рисунок перемещается на левый край формы. Переменная `gfrmWidth` получает значение `Me.Width` в процедуре обработки события открытия формы.

```
Private Sub Form_Timer()
   Static intPic As Integer
   Select Case intPic
      Case Is = 1
         Me!Видимый1.PictureData = Me!Hidden1.PictureData
      Case Is = 2
         Me!Видимый1.PictureData = Me!Hidden2.PictureData
      Case Else
   End Select
   If intPic = 2 Then intPic = 0
   intPic = intPic + 1
   If (Me!Видимый1.Left > gfrmWidth) Then Me!Visible1.Left = 0
   Me!Видимый1.Left = Me!Видимый1.Left + 200
End Sub
```

Свойство «Страницы с рисунком» (PicturePages)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPicturePagesC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPicturePagesX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPicturePagesA"}

Свойство **Страницы с рисунком (PicturePages)** определяет, на каких страницах отчета выводится рисунок.

Значения

Свойство **Страницы с рисунком (PicturePages)** может иметь следующие значения.

Значение	Описание	Visual Basic
Все страницы	(Значение по умолчанию). Рисунок выводится на всех страницах отчета.	0
Первая страница	Рисунок выводится только на первой странице отчета.	1
Первая страница	Рисунок в отчете не выводится.	2

Значение свойства **Страницы с рисунком (PicturePages)** задается в окне свойств отчета, в макросе или в программе Visual Basic.

Свойство «Масштабы рисунка» (PictureSizeMode)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPictureSizeModeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPictureSizeModeX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPictureSizeModeA"}

Свойство **Масштабы рисунка (PictureSizeMode)** определяет размеры рисунка в форме или отчете.

Значения

Свойство **Масштабы рисунка (PictureSizeMode)** может иметь следующие значения.

Значение	Описание	Visual Basic
Фрагмент	(Значение по умолчанию). Рисунок выводится с исходными размерами. Если размеры рисунка превышают размеры формы или отчета, то края обрезаются.	0
Вписать в рамку	Рисунок растягивается или сжимается по горизонтали и по вертикали до размеров формы; при этом возможно изменение пропорций.	1
По размеру рамки	Рисунок растягивается или сжимается с сохранением пропорций до максимально возможных размеров, при которых не происходит обрезания рисунка.	3

Значение свойства **Масштабы рисунка (PictureSizeMode)** задается в окне свойств формы или отчета, в макросе или в программе Visual Basic.

Дополнительные сведения

Если в свойстве **Рисунок** (**Picture**) формы или отчета указан маленький рисунок, то заданные для свойства **Масштабы рисунка (PictureSizeMode)** значения «Вписать в рамку» или «По размеру рамки» могут привести к существенному понижению разрешения деталей рисунка. Для мелких рисунков рекомендуется задавать мозаичное заполнение с помощью свойства **Мозаичное заполнение** (**PictureTiling**).

Свойство «Мозаичное заполнение» (PictureTiling)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acproPictureTilingC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acproPictureTilingX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acproPictureTilingA"}

Свойство **Мозаичное заполнение (PictureTiling)** определяет мозаичное заполнение элемента управления, окна формы, формы или страницы отчета фоновым рисунком.

Значения

Свойство **Мозаичное заполнение (PictureTiling)** может иметь следующие значения.

Значение	Описание	Visual Basic
Да	Мозаичное заполнение	**True** (–1)
Нет	Значение по умолчанию). Мозаичное заполнение не производится.	**False** (0)

Значение свойства **Мозаичное заполнение (PictureTiling)** задается в окне свойств объекта, в макросе или в программе Visual Basic.

Значение по умолчанию для данного свойства может быть задано с помощью стандартного стиля элемента управления или с помощью метода **DefaultControl** из Visual Basic.

Дополнительные сведения

Заданное для свойства **Мозаичное заполнение (PictureTiling)** значение «Да» позволяет создавать дополнительные графические эффекты. Выравнивание мозаики определяется значением свойства **Выравнивание рисунка (PictureAlignment).** Например, если свойство **Выравнивание рисунка (PictureAlignment)** имеет значение «Вверх влево», мозаичное заполнение начинается с верхнего левого угла элемента управления, окна формы или страницы отчета.

Примечание. При значении «По центру формы» свойства **Выравнивание рисунка (PictureAlignment)** и значении «Да» свойства **Мозаичное заполнение (PictureTiling)** производится мозаичное заполнение самой формы, а не окна формы.

Общие сведения о выражениях

Выражения являются основным средством выполнения многих операций Microsoft Access. Выражение представляет комбинацию символов – идентификаторов, операторов и значений – дающую определенный результат.

Например, следующее выражение позволяет вывести в элементе управления в форме или отчете сумму значений полей «СтоимостьДоставки» и «ОбщаяСуммаЗаказа»:

= [СтоимостьДоставки] + [ОбщаяСуммаЗаказа]

Приведем примеры операций, в которых используются выражения.

- Указание значения свойства, которое определяет вычисляемый элемент управления, определяет условие на значение или задает для поля значение по умолчанию.
- Указание условий отбора, создание вычисляемого поля и обновление записей в запросе или фильтре.
- Указание условий выполнения макрокоманды или набора макрокоманд в макросе и определение аргументов многих макрокоманд.
- Определение аргументов для многих функций, инструкций и методов в процедурах Visual Basic для приложений (VBA).
- Создание и изменение запросов SQL в режиме SQL окна запроса или определение свойств и аргументов с помощью инструкций SQL.

Для получения дополнительных сведений о создании выражений нажмите кнопку ....

Для просмотра примеров выражений нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acconWhatIsExpressionSW":1:"Foo":"Invisible"}

Создание выражения

При создании выражения необходимо объединить идентификаторы, значения и операторы таким образом, чтобы они давали определенный результат. С помощью выражений выполняются как простейшие арифметические действия, например, сложение чисел (1 + 1), так и сложные операции с данными, такие как проверка соответствия значения, введенного в поле «Индекс», стандарту страны, название которой создается в поле «Страна»:

= [Страна] In ("Франция", "Италия", "Испания") And Len([Индекс])<>5

Пользователь имеет возможность создавать выражения с помощью построителя выражений или самостоятельно создавать выражения, комбинирую компоненты выражений.

Предполагаемые действия

- Создание выражения с помощью построителя выражений.
- Создание выражения без помощи построителя выражений.
- Изучение выражений.
- Просмотр примеров выражений.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconCreatingExpressionsSW":1:"Foo":"Invisible"}

Создание выражения с помощью построителя выражений

1 Запустите построитель выражений.

 Инструкции

2 В левом нижнем поле построителя выберите папку, содержащую нужный элемент.

3 В нижнем среднем поле дважды щелкните элемент, чтобы вставить его в поле выражения, или выберите тип элементов.

4 Если выбран тип в нижнем левом поле, то значения будут отображаться в нижнем левом поле. Дважды щелкните на значении, чтобы вставить его в поле выражения.

Совет. Любая часть выражения может быть непосредственно введена в поле выражения.

5 Вставьте необходимые операторы в выражение. Для этого поместите указатель мыши в определенную позицию поля выражения и выберите одну из кнопок со знаками операций, расположенных в середине окна построителя.

6 Закончив создание выражения, нажмите кнопку **OK**.

Microsoft Access скопирует созданное выражение в ту позицию, из которой был вызван построитель выражений. Если в данной позиции уже содержится значение или если построитель выражений был вызван из окна модуля, в котором имеется выделенный текст, то исходное значение или выделенный текст будут заменены на новое выражение.

Для получения дополнительных сведений о построителе выражений нажмите кнопку  .

{ewc HLP95EN.DLL, DYNALINK, "nïSnïSnïSnïSnïS nïS Web nïSnïSnïS nïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïSnïS":"aclsaUsingExpressionBuilderW":1:"Foo":"Invisible"}

Запуск построителя выражений

Построитель выражений может быть вызван в тех случаях, когда пользователь должен ввести выражение, например, в окне свойств, в поле **Условие отбора** в бланке запроса, в окне макроса или в окне модуля.

Предполагаемые действия

 Запуск построителя выражений из окна свойств или из нижней половины окна макроса.
 Запуск построителя выражений из ячейки поля или условия в бланке запроса, из столбца условий в окне макроса или из окна модуля.

{ewc HLP95EN.DLL, DYNALINK, "nïSnïSnïSnïSnïS nïS Web nïSnïSnïS nïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS":"acdecStartExpressionBuilderSW":1:"Foo":"Invisible"}

Запуск построителя выражений из окна свойств или из нижней половины окна макроса

1. Выберите ячейку свойства или аргумента, в которую требуется ввести выражение.
2. Нажмите кнопку **Построить** ..., которая находится рядом с ячейкой свойства или аргумента.

   Если для данного свойства можно использовать несколько построителей, открывается окно диалога **Построители**. Выберите **Выражения** и нажмите кнопку **OK**.

Примечание. Если ячейка или аргумент для которой запущен построитель выражений уже содержал какое-то значение, то оно будет автоматически скопировано в поле выражения.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowStartExpressionBuilderPropertyMacroSW":1:"Foo":"Invisible"}

Запуск построителя выражений из бланка запроса, столбца условий в окне макроса или из окна модуля

Допускается вызов построителя выражений из полей **Условие отбора** и **Поле** в бланке запроса, из ячейки столбца условий в окне макроса или из окна модуля.

1 Установите указатель в позицию, в которую требуется ввести выражение, и нажмите правую кнопку мыши.

2 В контекстном меню выберите команду **Построить**.

Примечание. Если ячейка или столбец **Условие** из которого вызывается построитель выражений, уже содержит значение, это значение автоматически копируется в поле построителя выражений. Для того чтобы скопировать в построитель выражений текст из окна модуля, выделите копируемый текст перед вызовом построителя выражений.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowStartExpressionBuilderQueryMacroModuleSW":1:"Foo":"Invisible"}

Создание выражения без помощи построителя выражений

При создании выражения необходимо объединить идентификаторы, значения и операторы таким образом, чтобы они давали определенный результат. Например, следующее выражение увеличивает значение элемента управления «СтоимостьДоставки» в форме «Заказы» на 10 процентов:

= [Forms]![Заказы]![СтоимостьДоставки] * 1.1

В данном выражении:

- Forms![Заказы]![СтоимостьДоставки] является идентификатором ссылки на значение элемента управления «СтоимостьДоставки» в форме «Заказы».
- * является оператором умножения.
- 1,1 представляет число, на которое умножается значение элемента управления «СтоимостьДоставки».

В зависимости от требуемого результата, идентификаторы, операторы и значения при создании выражения комбинируются различными способами. С помощью выражений выполняются слияние строковых (текстовых) значений, сложение или умножение числовых значений, вызовы функций, ссылки на объекты и их значения, а также многие другие действия.

Некоторые выражения дают логические результаты ИСТИНА или ЛОЖЬ. Например, при вводе выражения в ячейку столбца **Условие** в строке макрокоманды эта макрокоманда будет выполнена только в том случае, если выражение истинно. Следующее выражение является истинным, когда поле «Страна» в форме «Сотрудники» содержит значение «Литва»:

Forms![Сотрудники]![Страна] = "Литва"

Для получения дополнительных сведений об использовании идентификаторов в выражениях нажмите кнопку  .

Для получения дополнительных сведений об операторах нажмите кнопку  .

Для получения дополнительных сведений об использовании значений в выражениях нажмите кнопку  .

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowCreatingExpressionWithoutBuilderSW":1:"Foo":"Invisible"}

Ссылки на объекты и их свойства в выражениях

Идентификаторы используются в выражениях для ссылок на объекты и их свойства. Например, допускаются ссылки на открытые формы, открытые отчеты, элементы управления в открытой форме или отчете, а также на любые свойства форм, отчетов или элементов управления. Следующий идентификатор определяет ссылку на свойство элемента управления **Вывод на экран (Visible)**:

Reports![Счет]![НазваниеПолучателя].Visible

Полный идентификатор объекта или свойства демонстрирует взаимосвязь между компонентами идентификатора. Ниже изложены пояснения к приведенному примеру.

- Компонент **Reports** определяет ссылку на семейство открытых отчетов базы данных. Microsoft Access автоматически создает семейства **Forms** и **Reports** для любой базы данных. В семейство **Forms** включаются все открытые формы, а в семейство **Reports** все открытые отчеты.
- Компонент **[Счет]** определяет ссылку на отчет с именем «Счет» в семействе **Reports**.
- Компонент **[НазваниеПолучателя]** определяет ссылку на элемент управления «НазваниеПолучателя» в отчете «Счет».
- Компонент **Visible** определяет ссылку на свойство **Вывод на экран (Visible)** элемента управления «НазваниеПолучателя».

Рекомендуется всегда ссылаться на объект или свойство с помощью полного идентификатора. В некоторых случаях полный идентификатор является обязательным. Например, необходимо использовать полный идентификатор для ссылок на элемент управления, не находящийся в текущей форме или отчете. Следующее выражение позволяет вывести в другой форме сумму значений элементов управления «СуммаЗаказа» и «СтоимостьДоставки» из формы «Заказы»:

= Forms![Заказы]![СуммаЗаказа] + Forms![Заказы]![СтоимостьДоставки]

Однако в некоторых случаях допускаются ссылки на элемент управления или его свойства без использования полного идентификатора.

- При ссылках на элемент управления в текущей форме или отчете нет необходимости указывать идентификатор формы или отчета. Например, для вывода суммы значений полей «СуммаЗаказа» и «СтоимостьДоставки» в другом поле в той же форме достаточно определить свойство **Данные (ControlSource)** этого поля с помощью выражения:

  = [СуммаЗаказа] + [СтоимостьДоставки]
- При ссылках на элемент управления в подчиненной форме или в подчиненном отчете нет необходимости указывать полный идентификатор формы или отчета с помощью свойства **Form** или **Report**. Например, допустимым является использование следующей ссылки на элемент управления «Количество» в подчиненной форме «Заказано»:

  Forms![Заказы]![Заказано]![Количество]

  Полный идентификатор элемента управления «Количество» имеет вид:

  Forms![Заказы]![Заказано].Form![Количество]
- В макросах или в аргументах макрокоманды нет необходимости указывать идентификаторы форм или отчетов, из которых запускается макрос. Например, если имя макроса задается в качестве значения свойства события формы, то в столбце **Условие** и в аргументах макрокоманды идентификатор формы или отчета в ссылках можно опустить.
- В процедурах Visual Basic для приложений (VBA) в ссылках на элементы управления в текущих формах или отчетах разрешается использовать ключевое слово **Me** вместо идентификаторов формы или отчета. Например, для присвоения суммы значений полей «СуммаЗаказа» и «СтоимостьДоставки» переменной `ценаЗаказа` в процедуре обработки события следует включить в процедуру следующую инструкцию:

  ```
  ценаЗаказа = Me![СуммаЗаказа] + Me![СтоимостьДоставки]
  ```

Примечания

- При запуске макроса или программы Visual Basic для приложений, которая содержит выражение, ссылающееся на форму или отчет, эта форма или отчет должны быть открыты.
- В процедурах Visual Basic поддерживается синтаксис ссылок на объект с помощью круглых скобок и кавычек (" ") вместо использования оператора !. Синтаксис с круглыми скобками является обязательным при использовании переменных в идентификаторах. Следующие идентификаторы являются эквивалентными:

```
Forms![Заказы]![ДатаРазмещения]
Forms("Заказы")("ДатаРазмещения")
```

Для получения дополнительных сведений об использовании операторов **!** и **.** (точка) в выражениях нажмите кнопку ....

Для получения дополнительных сведений о ссылках на формы, отчеты, подчиненные формы или подчиненные отчеты в выражениях нажмите кнопку ....

Для получения дополнительных сведений о ссылках на значения элементов управления или свойств в выражениях нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowIdentifiersExpressionsSW":1:"Foo":"Invisible"}

Использование значений в выражениях

Для того чтобы указать значение в выражении используют литералы, константы, функции и идентификаторы.

- Литералом называют значение в явном представлении, например, число, строковое значение или дату. Примерами значений в явном представлении могут служить "Москва", 100 и #1-янв-94#. Даты необходимо заключать в символы (#), а строковые значения в прямые кавычки (").
- Константа представляет не изменяющееся значение. **True**, **False** и **Null** являются примерами констант, автоматически определяемых в Microsoft Access. Пользователь имеет возможность определить собственные константы языка Visual Basic для приложений (VBA), которые могут быть использованы в процедурах Visual Basic.
- Функция возвращает значение, которое является результатом расчетов или выполнения других операций. В Microsoft Access определен ряд встроенных функций, например:
  - функция **Date** возвращает текущую дату;
  - функция **Sum** возвращает сумму набора значений поля;
  - функция **DLookup** возвращает значение указанного поля.

  В языке Visual Basic пользователь имеет возможность создавать собственные функции.
- Идентификатор представляет ссылку на значение поля, элемента управления или свойства. Например, следующий идентификатор определяет ссылку на значение свойства **Значение по умолчанию (DefaultValue)** элемента управления "ДатаРазмещения" в форме "Заказы":

  Forms![Заказы]![ДатаРазмещения].DefaultValue

Оператор слияния строк **&** позволяет объединить значение поля, элемента управления или свойства со строкой в явном представлении. Например, в следующем выражении выполняется слияние строки "[КодТипа] = " со значением элемента управления «КодТипа» в форме «Товары»:

"[КодТипа] = " & Forms![Товары]![КодТипа]

В некоторых случаях – например, в статистических функциях по подмножеству, таких как функция **DLookup** – значение должно заключаться в одинарные (') или в двойные (") кавычки. Простейшим способом добавления кавычек является включение символа одинарных кавычек в строку с последующим слиянием строки с другой строкой, содержащей единственный символ одинарных кавычек, как в следующем примере:

"[КодТипа] = ' " & Forms![Товары]![КодТипа] & " ' "

Существующие приложения Microsoft Access поддерживают использование операторов вертикальной черты (**| |**) вместо открывающих и закрывающих пар символов прямых кавычек (") и операторов **&** (слияния строк), например:

"[КодТипа] = '|Forms![Товары]![КодТипа]|' "

Однако использовать символы вертикальной черты не рекомендуется, поскольку в некоторых обстоятельствах они могут давать непредсказуемые результаты.

Для того чтобы получить в выражениях строку, заключенную в кавычки, следует либо использовать вложенные строки в одинарных кавычках, либо использовать тройные пары прямых кавычек. Например, следующие выражения являются эквивалентными:

Forms![Контакты]![Город].DefaultValue = " 'Рига' "

Forms![Контакты]![Город].DefaultValue = " " "Рига" " "

Для получения дополнительных сведений о константах нажмите кнопку ....

Для получения дополнительных сведений о создании функций пользователем нажмите кнопку ....

Для получения дополнительных сведений об использовании идентификаторов в выражениях нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconValuesExpressionsSW":1:"Foo":"Invisible"}

Примеры выражений

Выражения используются при выполнении многих операций Microsoft Access, в том числе, при создании вычисляемых элементов управления, определении условий в запросах и фильтрах, значений по умолчанию, условий на значение и условий в макросах. Часто простейшим способом создания выражения является поиск похожего выражения и его дальнейшее использование в качестве образца для создания требуемого выражения.

Выбор примеров

- Примеры выражений, используемых в формах и отчетах.
- Примеры выражений, используемых в запросах и фильтрах.
- Примеры выражений, определяющих значения по умолчанию.
- Примеры выражений, определяющих условия на значения.
- Примеры выражений для условий в макросах.
- Изучение выражений.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecExpressionExamplesSW":1:"Foo":"Invisible"}

Использование операторов ! и . (точка) в выражениях

Операторы **!** и **.** (точка) в идентификаторах указывают тип элемента, стоящего справа от оператора.

Оператор **!** указывает, что следующий за ним элемент является элементом, определяемым пользователем (элементом семейства). Например, с помощью оператора **!** определяют ссылку на открытую форму, на открытый отчет или элемент управления в открытой форме или отчете.

Идентификатор	Объект ссылки
Forms![Заказы]	Открытая форма «Заказы»
Reports![Счет]	Открытый отчет «Счет»
Forms![Заказы]![КодЗаказа]	Элемент управления «КодЗаказа» в открытой форме «Заказы»

Оператор **.** (точка) обычно указывает, что следующий за ним элемент определен в Microsoft Access. Например, оператор **.** (точка) используется для ссылок на свойства форм, отчетов и элементов управления.

Примечание. Допускается также использование оператора **.** (точка) для ссылок на значение поля в инструкции SQL, метод Visual Basic для приложений (VBA) или семейство. Например, идентификатор Forms![Заказы].Controls представляет ссылку на семейство **Controls** формы «Заказы». Однако, поскольку семейство **Controls** является семейством, используемым для форм и отчетов по умолчанию, обычно нет необходимости ссылаться на него в явном виде.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconUsingBangDotInExpressionsSW":1:"Foo":"Invisible"}

Ввод выражения в окно свойств, в бланк запроса или в аргумент макрокоманды.

При вводе в окно свойств, в поле бланка запроса или в ячейку аргумента макрокоманды выражения, размер которого превышает размер стандартной области ввода, существует возможность ввести выражение в окно **Область ввода.** Для того чтобы открыть окно **Область ввода**, нажмите клавиши SHIFT+F2, когда фокус находится в той позиции, в которую требуется ввести выражение.

При вводе выражения в окно свойств, в поле бланка запроса или в ячейку аргумента макрокоманды, Microsoft Access выполняет следующие действия.

- Анализ зависящих от региональных настроек имен функций, имен свойств и разделителей списков для национальных версий;
- Автоматическая вставка определенных символов при изменении положения фокуса. В зависимости от типа поля, в которое вводится выражение, автоматически добавляются:
  - прямые скобки ([ ]) вокруг имен форм, отчетов, полей и элементов управления;
  - символы (#) вокруг дат;
  - прямые кавычки вокруг (" ") строковых значений.

Примечание. Перед выражениями, определяющими вычисляемые элементы управления, всегда помещается знак равенства (**=**).

Имена функций и свойств в русской версии Microsoft Access

Ниже перечислены правила ввода имен функций или свойств в русской версии Microsoft Access.

- В русской версии имена функций не локализуются. Русские имена свойств присутствуют в окне свойств, в бланке запроса или в аргументах макрокоманд.
- При определении нескольких аргументов функции допускается использование указанного в региональных настройках символа разделителя списка. Символ разделителя списка выбирается на вкладке **Числа** в окне **Язык и стандарты** панели управления Windows. В русской версии по умолчанию в качестве разделителя списка выбирается символ точки с запятой (;).

Однако в программах на языке Visual Basic для приложений (VBA) необходимо использовать английские имена функций и свойств, а в качестве разделителя списка использовать запятую (,).

Ввод имен объектов

Имена таблиц, отчетов, запросов, полей и элементов управления заключаются в квадратные скобки ([ ]).

При вводе имен объектов в идентификаторы использование квадратных скобок является обязательным, если имя содержит пробелы или специальные символы, например, символ подчеркивания. Имена, не содержащие пробелы и специальные символы, можно вводить без квадратных скобок. Microsoft Access добавит квадратные скобки автоматически (с двумя исключениями, которые описаны ниже).

Например, для расчета суммы значений полей «СтоимостьДоставки» и «ОбщаяСуммаЗаказа» достаточно определить вычисляемое поле с помощью следующего выражения в ячейке свойства **Данные (ControlSource)**:

= СтоимостьДоставки + ОбщаяСуммаЗаказа

Microsoft Access автоматически отобразит это выражение следующим образом:

= [СтоимостьДоставки] + [ОбщаяСуммаЗаказа]

Примечание. В выражениях, определяющих свойство **Условие на значение (ValidationRule)** или вводящихся в ячейку строки **Условие отбора**, квадратные скобки вокруг имен не добавляются автоматически. При вводе имен объектов пользователь обязан ввести квадратные скобки самостоятельно. В противном случае Microsoft Access может воспринять вводящиеся имена как текстовые значения и добавить прямые кавычки.

Ввод значений даты/времени

Символы (#) вокруг элемента выражения указывают, что элемент представляет значение даты/времени. Microsoft Access автоматически оценивает значение, заключенное в символы (#), как значение даты/времени и позволяет вводить такие значения в любом поддерживаемом формате даты или времени.

Нет необходимости вводить символы (#) вокруг значений даты/времени в выражениях, определяющих условия на значение или условия отбора для полей, имеющих тип данных «Дата/время». Ввод значений допускается в любом поддерживаемом формате даты или времени. Microsoft Access автоматически добавит символы (#) вокруг введенного значения.

Значения отображаются в соответствии с настройками, заданными в окне **Язык и стандарты** панели управления Windows. Пользователь имеет возможность указать формат с помощью свойства **Формат поля (Format)**.

Ввод текста

Заключение элемента выражения в прямые кавычки (" ") указывает, что данные элемент представляет строковое значение.

При вводе строкового выражения в качестве условия на значение или условия отбора заключение значения в двойные кавычки не является обязательным, Microsoft Access добавит кавычки автоматически.

Например, если пользователь введет выражение **Рига**, Microsoft Access выведет выражение:

"Рига"

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconEnteringExpressionsW":1:"Foo":"Invisible"}

Ссылки на формы, отчеты, подчиненные формы или подчиненные отчеты в выражениях.

Для ссылки на открытую форму или отчет и на подчиненную форму или подчиненный отчет в выражении необходимо указать соответствующий идентификатор.

Ссылка на открытую форму или отчет

- Введите имя семейства **Forms** или **Reports**, за которым следует оператор **!** и имя формы или отчета. Например, следующий идентификатор определяет ссылку на форму «Заказы»:

  Forms![Заказы]

Ссылка на подчиненную форму или подчиненный отчет

Требуется ссылка на элемент управления подчиненную форму/отчет в главной форме или отчете, а затем с помощью свойства **Form** или **Report** подчиненного элемента управления указание реальной подчиненной формы или подчиненного отчета.

- Введите идентификатор формы, которая содержит подчиненную форму, далее введите имя элемента управления-подчиненной формы/отчета, оператор **.** (точка) и имя свойства **Form**. Например, следующий идентификатор определяет ссылку на подчиненную форму «Заказано» в форме «Заказы»:

  Forms![Заказы]![Заказано].Form

Примечания

- Для создания идентификатора для формы, отчета, подчиненной формы или подчиненного отчета можно использовать построитель выражений.
- При запуске макроса или программы Visual Basic для приложений, которая содержит выражение, ссылающееся на форму или отчет, эта форма или отчет должны быть открыты.

Для получения дополнительных сведений о создании выражений с помощью построителя выражений нажмите кнопку ....

Для получения дополнительных сведений об использовании идентификаторов в выражениях нажмите кнопку ....

Для получения дополнительных сведений о ссылках на значение элемента управления или свойства в выражении нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowReferFormReportSW":1:"Foo":"Invisible"}

Ссылки на значение элемента управления или свойства в выражении

Для ссылки в выражении на значение элемента управления или свойства необходимо ввести соответствующий идентификатор.

Ссылка на значение элемента управления

- Введите идентификатор для формы или отчета, за которым следует оператор **!** и имя элемента управления. Например, следующий идентификатор определяет ссылку на элемент управления «КодЗаказа» в форме «Заказы»:

  Forms![Заказы]![КодЗаказа]

  Совет. При ссылках на элемент управления в подчиненной форме или подчиненном отчете нет необходимости указывать полный идентификатор формы или отчета с помощью свойства **Form** или **Report**. Например, следующий идентификатор определяет ссылку на поле «Количество» в подчиненной форме «Заказано»:

  Forms![Заказы]![Заказано]![Количество]

Ссылка на значение свойства

- Введите идентификатор формы, отчета или элемента управления, к которым применяется свойство, далее введите оператор **.** (точка) и имя свойства. Например, следующий идентификатор определяет ссылку на свойство **Значение по умолчанию (DefaultValue)** поля «ДатаРазмещения» в форме «Заказы»:

  Forms![Заказы]![ДатаРазмещения].DefaultValue

Примечания

- Для создания идентификатора для формы, отчета, подчиненной формы или подчиненного отчета можно использовать построитель выражений.
- При запуске макроса или программы Visual Basic для приложений, которая содержит выражение, ссылающееся на форму или отчет, эта форма или отчет должны быть открыты.

Для получения дополнительных сведений о создании выражений с помощью построителя выражений нажмите кнопку ....

Для получения дополнительных сведений об использовании идентификаторов в выражениях нажмите кнопку ....

Для получения дополнительных сведений об использовании значений в выражениях нажмите кнопку ....

Для получения дополнительных сведений о ссылках на формы, отчеты, подчиненные формы или подчиненные отчеты в выражениях нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "nїSnїSnїSnїSnїS nїS Web nїSnїSnїS nїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїSnїS":"achowReferValueFieldControlPropSW":1:"Foo":"Invisible"}

Ссылки на разделы формы или отчета или на группы отчета в выражениях

- Введите идентификатор формы или отчета, далее введите оператор **.** (точка) и имя раздела или группы. Например, следующий идентификатор определяет ссылку на область данных формы «Заказы»:

  [Forms]![Заказы].ОбластьДанных

Примечания

- При запуске макроса или программы Visual Basic для приложений, которая содержит выражение, ссылающееся на форму или отчет, эта форма или отчет должны быть открыты.
- Допускаются также ссылки на разделы формы или отчета или на группы в отчете с помощью свойства **Section** с соответствующим индексом. Для получения дополнительных сведений о свойстве **Section** нажмите кнопку ....

Для получения дополнительных сведений об использовании идентификаторов в выражениях нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowReferSectionSW":1:"Foo":"Invisible"}

Ссылки на столбец списка в выражениях

Ссылки на столбец списка или поля со списком, содержащего несколько столбцов, делаются с помощью свойства **Column** элементов управления поле со списком или список.

- Введите идентификатор свойства **Column** с последующим индексом столбца в круглых скобках. Первому столбцу соответствует индекс 0, второму индекс 1 и т.д. Например, следующий идентификатор определяет ссылку на второй столбец списка «Страна» в форме «Клиенты»:

  [Forms]![Клиенты]![Страна].[Column](1)

Примечание. При запуске макроса или программы Visual Basic для приложений, которая содержит выражение, ссылающееся на форму или отчет, эта форма или отчет должны быть открыты.

Для получения дополнительных сведений о создании в элементов управления список и поле со списком, содержащих несколько столбцов, нажмите кнопку ....

Для получения дополнительных сведений о ссылках в выражениях на значение элемента управления или свойства нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowReferColumnInListSW":1:"Foo":"Invisible"}

Соглашения об именах полей, элементов управления и объектов.

В Microsoft Access действуют следующие ограничения на имена полей, элементов управления и объектов:

- имя должно содержать не более 64 символов;
- имя может включать любую комбинацию букв, цифр, пробелов и специальных символов за исключением точки (.), восклицательного знака (!), надстрочного символа (`) и прямых скобок ([ ]);
- не должно начинаться с символа пробела;
- не должно включать управляющие символы (с кодами ASCII от 0 до 31).

Хотя пробелы внутри имен полей, элементов управления и объектов являются допустимыми, в большинстве примеров в документации Microsoft Access имена полей записываются без пробелов. Пробелы в именах могут при некоторых обстоятельствах вызывать конфликты в программах Visual Basic.

Определяя имя для поля, элемента управления или объекта, полезно проверить, не совпадает ли это имя с именем свойства или другого элемента, используемого Microsoft Access (для русских имен такая ситуация может возникнуть при совпадении с именем свойства или функции, определяемых пользователем); в противном случае иногда могут возникать неверные результаты. Например, при ссылке на значение поля «Name» в таблице «NameInfo» с помощью синтаксиса NameInfo.Name будет возвращаться значение свойства таблицы **Имя (Name)**, а не значение поля «Name».

Простейшим способом обойти неопределенность в именах является использование оператора **!** вместо оператора **.** (точка) в ссылках на значения полей, элементов управления и объектов. Например, следующий идентификатор в явном виде определяет ссылку на значение поля «Name», а не на свойство **Имя (Name)**:

[NameInfo]![Name]

Для получения дополнительных сведений об использовании операторов **!** и **.** (точка) в выражениях нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecNamingConventionsSW":1:"Foo":"Invisible"}

Соглашения об именах процедур Visual Basic, переменных и констант.

В языке Visual Basic для приложений (VBA) действуют следующие ограничения на имена процедур, переменных и констант:

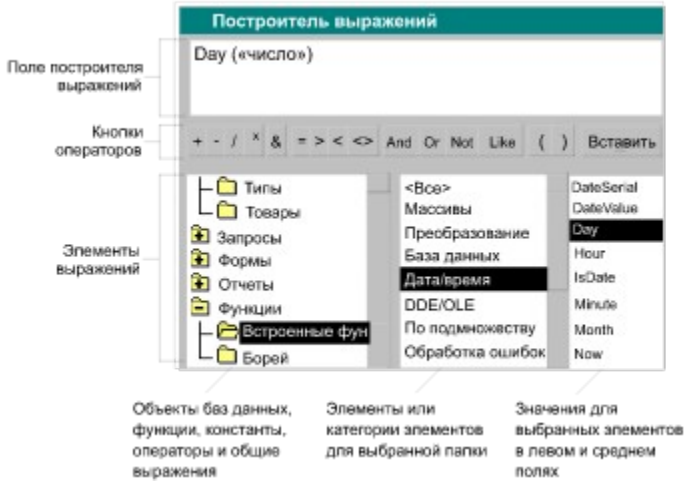
- имя должно содержать не более 255 символов;
- должно начинаться с буквы;
- может включать буквы, цифры и символы подчеркивания (_);
- не должно включать знаки препинания или пробелы;
- не должно совпадать с ключевыми словами Visual Basic.

Ключевыми словами называют слова, являющиеся элементами языка Visual Basic. В число ключевых слов входят имена инструкций (например, **If** и **Loop**), функций (например, **Len** и **Abs**) и операторов (например, **Or** и **Mod**).

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconVBNamingConventionsSW":1:"Foo":"Invisible"}

Общие сведения о построителе выражений

Построитель выражений состоит из трех разделов.



- В верхней части окна построителя расположено поле выражения. Ниже находится раздел, предназначенный для создания элементов выражения и их последующей вставки в поле выражения. Допускается непосредственный ввод части выражения в поле выражения.
- В средней части окна построителя находятся кнопки с часто используемыми операторами. При нажатии на одну из этих кнопок построитель вставит соответствующий оператор в текущую позицию поля выражения. Чтобы вывести полный список операторов, выберите папку **Операторы** в нижнем левом поле и нужный тип в среднем поле. В правом поле будут выведены все операторы выбранного типа.
- В нижней части окна построителя находятся три поля.
  - В левом поле выводятся папки, содержащие таблицы, запросы, формы, объекты базы данных, встроенные и определенные пользователем функции, константы, операторы и общие выражения.
  - В среднем поле задаются определенные элементы или типы элементов для папки, заданной в левом поле. Например, если выбрать в левом поле **Встроенные функции**, то в среднем поле появится список всех типов функций Microsoft Access.
  - В правом поле выводится список значений (если они существуют) для элементов, заданных левым и средним полями. Например, если выбрать в левом поле **Встроенные функции** и тип функции в среднем, то в правом поле будет выведен список всех встроенных функций выбранного типа.

Примечания

- Чтобы новые имена полей появились в построителе выражений, следует предварительно сохранить таблицу или запрос, содержащие такие поля.
- Если функция или объект не выводятся в нижней части построителя выражений, то это значит, что их использование недопустимо в позиции, из которой был вызван построитель. Например, нельзя ссылаться на другое поле или элемент управления в условии на значение для поля в режиме конструктора таблицы, таким образом папки **Таблицы**, **Запросы**, **Формы** и **Отчеты** не доступны при запуске построителя выражений из ячейки для свойства поля **Условие на значение (ValidationRule)** в режиме конструктора таблицы.
- При вставке идентификатора в выражение построитель вставляет только те его части, которые требуются в текущем контексте. Например, при запуске построителя выражений из окна свойств формы «Клиенты» и вставке идентификатора для свойства **Вывод на экран (Visible)** будет вставлено только имя свойства: **Visible**. При использовании данного выражения вне контекста формы необходимо включать полный идентификатор: Forms![Клиенты].Visible.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconAboutExpressionBuilderW":1:"Foo":"Invisible"}

Использование построителя выражений

Предполагаемые действия

- Общие сведения о построителе выражений.
- Запуск построителя выражений.
- Создание выражения без помощи построителя выражений.
- Просмотр примеров выражений.
- Изучение выражений.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ
пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecUseExpressionBuilderW":1:"Foo":"Invisible"}

Разрешение вопросов при работе с выражениями

Вопросы

Я не понимаю, как работает выражение. Требуется общий обзор.

Аргумент, определяющий условие в функции **DLookup** или в другой статистической функции по подмножеству, дает неверные результаты.

Неверные результаты при попытке объединить в выражении значения элементов управления, полей, переменных или строк.

Неверные результаты при попытке выполнить действия над полями, содержащими пустые значения.

Неверные результаты при попытке ввести значение даты в выражение.

Неверные результаты при ссылке на свойство или элемент управления подчиненной формы или отчета.

Неверные результаты при ссылке на столбец в списке или в поле со списком.

Как преобразовать текст так, чтобы первые буквы были прописными, а остальные строчными?

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"actrbUsingExpressionsW":1:"Foo":"Invisible"}

Неверные результаты при попытке объединить в выражении значения элементов управления, полей, переменных или строк.

Неверные результаты при попытке объединения (слияния) в выражении значений элементов управления, полей, переменных или строк могут возникать в следующих ситуациях.

- Ссылка на элемент управления или поле, имя которого совпадает с именем текущего элемента управления, в результате чего образуется циклическая ссылка. Измените имя поля или элемента управления.
- Ошибочное имя в ссылке на элемент управления или поле. Проверьте правильность имен.
- Ссылка на элемент управления или поле, имя которого совпадает с именем свойства. Для таких полей или элементов управления требуется полный идентификатор. Для получения дополнительных сведений об использовании идентификаторов в выражениях нажмите кнопку ....
- Ошибка при определении значения выражения из-за того, что некоторые компоненты выражения должны быть заключены в прямые скобки ([ ]) или в одинарные (') или прямые (") кавычки. Проверьте, правильно ли используются в выражении прямые скобки или кавычки.

Для получения дополнительных сведений о слиянии строковых значений нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbExprConcatenateStringsSW":1:"Foo":"Invisible"}

Неверные результаты при попытке выполнить действия над полями, содержащими пустые значения.

Результаты многих операций оказываются неверными, если в используемых выражениях имеются ссылки на поля, в которых содержатся пустые (**Null**) значения. Например, в таких случаях дают неверные результаты функции **DCount** или **DSum**.

Существуют несколько способов решения этой проблемы.

- Передавайте значения полей в выражение с помощью функции **Nz**. Если функция **Nz** принимает значение **Null**, она возвращает нуль или пустую строку. Для получения дополнительных сведений о функции **Nz** нажмите кнопку ....
- Организуйте проверку на пустые значения с помощью функций **Iif** или **IsNull** с возвращением подходящих значений вместо пустых. Для получения дополнительных сведений о функции **Iif** нажмите кнопку .... Для получения дополнительных сведений о функции **IsNull** нажмите кнопку ....

Для получения дополнительных сведений о свойствах полей, определяющих обработку пустых значений, нажмите кнопку ...

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbExprNullsEmptyStringsSW":1:"Foo":"Invisible"}

Неверные результаты при попытке ввести значение даты в выражение.

Неверные результаты иногда возникают при попытке ввести в выражение значение даты, не окружив его символами (#). Для получения дополнительных сведений о вводе выражений нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"actrbExprDateValuesSW":1:"Foo":"Invisible"}

Неверные результаты при ссылке на свойство или элемент управления подчиненной формы или отчета.

Неверные результаты при ссылке на свойство или элемент управления подчиненной формы или отчета могут возникнуть, если используется неправильный идентификатор.

Корректная ссылка на свойство подчиненной формы или отчета требует указания полного идентификатора формы или отчета с использованием свойств **Form** или **Report** элемента управления-подчиненная форма/отчет. Например, следующий идентификатор определяет ссылку на свойство **Вывод на экран (Visible)** подчиненной формы «Заказано»:

Forms![Заказы]![Заказано].Form.Visible

В данном примере:

- Forms![Заказы]![Заказано] является ссылкой на элемент управления, в котором выводится подчиненная форма.
- Forms![Заказы]![Заказано].Form является ссылкой на саму подчиненную форму. Указание свойства Form для ссылки на подчиненную форму или на ее свойства является обязательным.

В отличие от этого, для ссылки на элемент управления, содержащийся в подчиненной форме, или на свойства этого элемента управления ссылка на свойства **Form** или **Report** не требуется. Например, следующий идентификатор определяет ссылку на свойство **Вывод на экран (Visible)** элемента управления «Скидка» в подчиненной форме «Заказано»:

Forms![Заказы]![Заказано]![Скидка].Visible

Для получения дополнительных сведений о ссылках в выражениях на форму, отчет, подчиненную форму или подчиненный отчет нажмите кнопку ....

Для получения дополнительных сведений о ссылках в выражениях на значение элемента управления или свойства нажмите кнопку ....

Для получения дополнительных сведений об использовании идентификаторов в выражениях нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbExprSubformRefsSW":1:"Foo":"Invisible"}

Неверные результаты при ссылке на столбец в списке или в поле со списком.

Свойство **Column** используется для ссылок на столбец списка или поля со списком. При использовании свойства **Column** неверные результаты возникают в следующих случаях.

- Не указан индекс, определяющий в свойстве **Column** столбец, на который делается ссылка. Например, Column(0) определяет ссылку на первый столбец в списке или в поле со списком.
- Указан индекс, соответствующий неверному столбцу. Следует помнить, что индексы в свойстве **Column** отсчитываются от нуля. Column(0) определяет ссылку на первый столбец, Column(1) на второй столбец и т.д.
- Делается попытка числовых расчетов с использованием значения, возвращаемого в свойстве **Column**. Поскольку свойство Column возвращает текстовое значение, необходимо перед выполнением расчетов преобразовать это значение в числовое. Например, в следующем выражении перед операцией умножения функция **CCur** преобразует значение, возвращаемое в свойстве **Column**, в денежное:

  [Количество] * CCur([СписокЦен].Column(1))

Для получения дополнительных сведений о ссылках на столбцы списка нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"actrbExprColumnsSW":1:"Foo":"Invisible"}

Как преобразовать текст так, чтобы первые буквы были прописными, а остальные строчными?

Существует возможность преобразовать первую букву каждого слова в прописную с помощью функции **StrConv**.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbExprProperSW":1:"Foo":"Invisible"}

Использование клавиатуры в Microsoft Access

Для быстрого доступа к часто используемым командам и операторам применяются сочетания клавиш. Следующие разделы содержат описание сочетаний клавиш общих для всех продуктов входящих в комплект Microsoft Office и клавиши, применяемые только в Microsoft Access.

Предполагаемые действия

- Использование общих сочетаний клавиш в меню.
- Использование общих сочетаний клавиш на панели инструментов.
- Использование общих сочетаний клавиш в окнах и диалоговых окнах.
- Использование общих сочетаний клавиш в Office Assistant
- Использование общих сочетаний клавиш Microsoft Access.
- Работа в окне базы данных с помощью клавиатуры.
- Выделение текста или данных с помощью клавиатуры.
- Редактирование текста или данных с помощью клавиатуры.
- Перемещения в режиме таблицы с помощью клавиатуры.
- Перемещения в режиме формы с помощью клавиатуры.
- Использование клавиатуры в режиме предварительного просмотра.
- Использование клавиатуры в окне модуля и в окне отладки.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecUseFunctionKeysSW":1:"Foo":"Invisible"}

Использование общих сочетаний клавиш Microsoft Access

Вывод справки	Клавиши
Вывод окна помощника, содержания и предметного указателя справки Microsoft Access, контекстной справки о выделенном свойстве, элементе управления, макрокоманде, ключевом слове Visual Basic или об окне, а также справки в окне сообщения при наличии в нем кнопки «Справка».	F1
Для вывода раздела справки **Что это такое?** Нажмите SHIFT+F1, переместите курсор на команду меню, кнопку панели инструментов, элемент диалогового окна или области экрана и нажмите левую кнопку мыши.	SHIFT+F1

Открытие баз данных	Клавиши
Открытие новой базы данных	CTRL+T (N)
Открытие существующей базы данных	CTRL+Щ (O)

Печать и сохранение	Клавиши
Распечатка текущего или выбранного объекта	CTRL+3 (P)
Сохранение объекта базы данных	CTRL+Ы (S) или SHIFT+F12 или ALT+SHIFT+F2
Открытие диалогового окна **Сохранить как**	F12 или ALT+F2

Использование списка или поля со списком	Клавиши
Открытие списка в поле со списком	F4 или ALT+СТРЕЛКА ВНИЗ
Обновление содержимого подстановочного поля списка или поля со списком	F9
Переход на следующую строку	СТРЕЛКА ВНИЗ
Переход на следующую страницу	PAGE DOWN
Переход на предыдущую строку	СТРЕЛКА ВВЕРХ
Переход на предыдущую страницу	PAGE UP
Выход из списка или поля со списком	TAB

Поиск и замена текста или данных	Клавиши
Открытие диалогового окна **Поиск** (только для режима таблицы, режима формы и окна модуля)	CTRL+A (F)
Открытие диалогового окна **Замена** (только для режима таблицы, режима формы и окна модуля)	CTRL+P (H)
Поиск следующего вхождения текстовой строки, заданной в диалоговых окнах **Поиск** или **Замена**, когда эти окна закрыты (только режим формы и ражим таблицы)	SHIFT+F4

Режим конструктора	Клавиши
Переключение между режимом редактирования (с выведенным курсором) и режимом перемещения	F2
Выбор формы или отчета	CTRL+К (R)
Переключение в режим формы из режима конструктора формы или из модуля формы	F5
Переходы между верхней и нижней половиной окна (только в окнах таблицы, запроса и макроса, а также в окне **Расширенный фильтр**	F6
Добавление элемента управления в раздел(только в режиме конструктора формы и отчета) Для получения дополнительных сведений нажмите кнопку ...	SHIFT+ENTER

Операции с окнами	Клавиши
Перевод окна базы данных в верхний слой	F11 или ALT+F1
Циклические переходы между открытыми окнами	CTRL+F6
Восстановление свернутого окна, когда все окна свернуты	ENTER
Вывод оконного меню	ALT+ПРОБЕЛ
Закрытие активного окна	CTRL+Ц (W) или CTRL+F4

Прочие операции	Клавиши
Отображение полного адреса для выбранной гиперссылки	F2
Проверка орфографии	F7
Открытие окна **Область ввода** для ввода выражений и текста в поля малого размера	SHIFT+F2
Открытие окна свойств	ALT+ENTER
Выход из Microsoft Access, закрытие диалогового окна или окна свойств	ALT+F4
Вызов построителя	CTRL+F2
Переключение между меню пользователя и встроенным меню	CTRL+F11

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acrefFunctionKeysUsedGloballySW":1:"Foo":"Invisible"}

Работа в окне базы данных с помощью клавиатуры

Редактирование и перемещение по списку объектов	Клавиши
Переименовать выбранный объект	F2
Перемещение на одну строку вниз	СТРЕЛКА ВНИЗ
Перемещение на одно окно вниз	PAGE DOWN
Перемещение на последний объект	END
Перемещение на одну строку вверх	СТРЕЛКА ВВЕРХ
Перемещение на одно окно вверх	PAGE UP
Перемещение на первый объект	HOME

Перемещение и открытие объектов	Клавиши
Циклический переход по вкладкам каждого типа объекта	CTRL+TAB
Циклический переход по вкладкам каждого типа объекта справа налево	SHIFT+CTRL+TAB
Открытие выбранной таблицы или запроса в режиме таблицы или режиме формы	ENTER
Открытие выбранного отчета в режиме предварительного просмотра	ENTER
Запуск выбранного макроса	ENTER
Открытие выбранной таблицы, запроса, формы, макроса или модуля в режиме конструктора	CTRL+ENTER или ALT+B (D)

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acdecWorkDatabaseWindowUsingKeyboardSW":1:"Foo":"Invisible"}

Выделения текста или данных с помощью клавиатуры

Сдвиг границы выделения	Клавиши
На один символ вправо	SHIFT+СТРЕЛКА ВПРАВО
На одно слово вправо	CTRL+SHIFT+СТРЕЛКА ВПРАВО
На один символ влево	SHIFT+СТРЕЛКА ВЛЕВО
На одно слово влево	CTRL+SHIFT+СТРЕЛКА ВЛЕВО

Примечание. Для снятия выделения пользуйтесь клавишей со стрелкой противоположного направления.

Выделение текстового поля или записи	Клавиши
Выделение следующего поля	TAB
Переключение между режимом редактирования (с выведенным курсором) и режимом перемещения.	F2
Переключение из режима выбора текущей записи и первого поля текущей записи в режим перемещения	SHIFT+ПРОБЕЛ
Расширение границы выделения на предыдущую запись, если выбрана текущая	SHIFT+СТРЕЛКА ВВЕРХ
Расширение границы выделения на следующую запись, если выбрана текущая	SHIFT+СТРЕЛКА ВНИЗ
Выделение всех записей	CTRL+Ф (A) или CTRL+SHIFT+ПРОБЕЛ

Для получения дополнительных сведений по выделению полей или записей нажмите кнопку ...

Расширение границы выделения	Клавиши
Переход в режим выделения (в нижнем правом углу окна при этом появятся символы ВДЛ). В режиме выделения последовательные нажатия клавиши F8 расширяют область выделения на слово, поле, запись (только в режиме таблицы) и на все записи.	F8
Изменение границы выделенного фрагмента на соседние строки в режиме таблицы.	СТРЕЛКА ВЛЕВО или СТРЕЛКА ВПРАВО
Расширение границы выделения на соседние строки в режиме таблицы.	СТРЕЛКА ВВЕРХ, СТРЕЛКА ВНИЗ
Отмена предыдущего расширения области выделения.	SHIFT+F8
Отмена режима выделения.	ESC

Выделение столбца в режиме таблицы	Клавиши
Выделение текущего столбца или отмена выделения столбца (только в режиме перемещения	CTRL+ПРОБЕЛ

Выделение столбца справа, если выделен текущий столбец	СТРЕЛКА ВПРАВО
Выделение столбца слева, если выделен текущий столбец	СТРЕЛКА ВЛЕВО

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecSelectTextUsingKeyboardSW":1:"Foo":"Invisible"}

Редактирование текста и данных с помощью клавиатуры

Примечание. Если курсор невидим, нажмите клавишу F2, чтобы вывести его.

Перемещение курсора в поле	Клавиши
На один символ вправо	СТРЕЛКА ВПРАВО
На одно слово вправо	CTRL+СТРЕЛКА ВПРАВО
На один символ влево	СТРЕЛКА ВЛЕВО
На одно слово влево	CTRL+СТРЕЛКА ВЛЕВО
В конец поля, содержащего одну строку	END
В конец поля, содержащего несколько строк	CTRL+END
В начало поля, содержащего одну строку	HOME
В начало поля, содержащего несколько строк	CTRL+HOME

Копирование, перемещение и удаление текста	Клавиши
Копирование выделенного фрагмента в буфер обмена	CTRL+C (C)
Удаление выделенного фрагмента в буфер обмена	CTRL+Ч (X)
Вставка содержимого буфера обмена в позицию курсора	CTRL+М (V)
Удаление выделенного фрагмента или символа слева от позиции курсора	BACKSPACE
Удаление выделенного фрагмента или символа справа от позиции курсора	DEL

Отменяемое действие	Клавиши
Ввод с клавиатуры	CTRL+Я (Z) или ALT+BACKSPACE
Изменения текущего поля или текущей записи; если были изменены и поле, и запись, то первое нажатие клавиши ESC отменяет изменения в текущем поле, а второе нажатие изменения в текущей записи	ESC

Ввод данных в режиме таблицы или режиме формы	Клавиши
Вставка текущей даты	CTRL+; (точка с запятой)
Вставка текущего времени	CTRL+: (двоеточие)
Вставка в поле значения по умолчанию	CTRL+ALT+ПРОБЕЛ
Вставка в поле значения этого поля в предыдущей записи	CTRL+Э (' апостроф)
Добавление новой записи	CTRL++ (плюс)
Удаление текущей записи	CTRL+- (минус)
Сохранение изменений текущей записи	SHIFT+ENTER

Переключение состояния флажка или переключателя	ПРОБЕЛ
Вставка новой строки	CTRL+ENTER

Обновление данных в полях	**Клавиши**
Пересчет полей в активном окне	F9
Обновление базовых таблиц; в подчиненной форме обновляются только базовые таблицы подчиненной формы	SHIFT+F9
Обновление содержимого подстановочного поля списка или поля со списком	F9

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecEnterEditDataUsingKeyboardSW":1:"Foo":"Invisible"}

Использование сочетаний клавиш для перемещения в режиме таблицы

Переход к определенной записи	Клавиши
Переход в поле номера записи. Наберите номер записи и нажмите клавишу ENTER.	F5

Перемещение по полям и записям	Клавиши
Перемещение к следующему полю	TAB, ENTER или СТРЕЛКА ВПРАВО
Перемещение к последнему полю в текущей записи в режиме перемещения	END
Перемещение к предыдущему полю	SHIFT+TAB или СТРЕЛКА ВЛЕВО
Перемещение к первому полю в текущей записи в режиме перемещения	HOME
Перемещение к текущему полю в следующей записи	СТРЕЛКА ВНИЗ
Перемещение к текущему полю в последней записи в режиме перемещения	CTRL+СТРЕЛКА ВНИЗ
Перемещение к последнему полю в последней записи в режиме перемещения	CTRL+END
Перемещение к текущему полю в предыдущей записи	СТРЕЛКА ВНИЗ
Перемещение к текущему полю в первой записи в режиме перемещения	CTRL+СТРЕЛКА ВНИЗ
Перемещение к первому полю в первой записи в режиме перемещения	CTRL+HOME

Перемещение в другой экран данных	Клавиши
На один экран вниз	PAGE DOWN
На один экран вверх	PAGE UP
На один экран вправо	CTRL+PAGE DOWN
На один экран влево	CTRL+PAGE UP

Перемещение столбцов	Клавиши
Переход в режим сдвига (в нижнем правом углу окна появляются буквы СДВ)	CTRL+F8
Перемещение выделенного столбца на один столбец вправо в режиме сдвига	СТРЕЛКА ВПРАВО
Перемещение выделенного столбца на один столбец влево в режиме сдвига	СТРЕЛКА ВЛЕВО
Выключение режима сдвига	ESC

Для получения дополнительных сведений по перестановке столбцов нажмите кнопку

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecNavigateDatasheetViewUsingKeyboardSW":1:"Foo":"Invisible"}

Использование сочетаний клавиш для перемещения в режиме формы

Переход к определенной записи	Клавиши
Переход в поле номера записи. Набирите номер записи и нажмите клавишу ENTER	F5

Перемещение по полям и записям	Клавиши
Перемещение к следующему полю	TAB
Перемещение к предыдущему полю	SHIFT+TAB
Перемещение к последнему полю в текущей записи в режиме перемещения	END
Перемещение к последнему полю в последней записи в режиме перемещения	CTRL+END
Перемещение к первому полю в текущей записи в режиме перемещения	HOME
Перемещение к первому полю в первой записи в режиме перемещения	CTRL+HOME
Перемещение к текущему полю в следующей записи	CTRL+PAGE DOWN
Перемещение к текущему полю в предыдущей записи	CTRL+PAGE UP

Перемещение между разделами записи	Клавиши
Циклическое перемещение по разделам	F6
Циклическое перемещение по разделам в обратном порядке	SHIFT+F6

Перемещение в формах с несколькими страницами	Клавиши
Перемещение вниз на одну страницу; в конце записи - переход на ту же страницу следующей записи	PAGE DOWN
Перемещение вверх на одну страницу; в конце записи - переход на ту же страницу предыдущей записи	PAGE UP

Перемещение между основной формой и подформами	Клавиши
Ввод подформы из предшествующего поля в основной форме	TAB
Ввод подформы из последующего поля в основной форме	SHIFT+TAB
Выход из подформы и переход к следующему полю основной формы или к следующей записи	CTRL+TAB
Выход из подформы и переход к предыдущему полю основной формы или к предыдущей записи	CTRL+SHIFT+TAB

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ

пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acdecNavigateFormViewUsingKeyboardSW":1:"Foo":"Invisible"}

Использование клавиатуры в режиме предварительного просмотра и просмотра макета страницы

Диалоговые окна и операции с окнами	Клавиши
Открытие диалогового окна **Печать**	З (P) или CTRL+З (P)
Открытие диалогового окна **Страницы**	Ы (S)
Переключение масштаба страницы	Я (Z)
Выход из режима предварительного просмотра или просмотра макета страницы	С (C) или ESC

Просмотр разных страниц	Клавиши
Переход на поле номера страницы; далее следует ввести номер нужной страницы и нажать клавишу ENTER	F5
Переход на следующую страницу. (в режиме подгонки по размеру окна)	PAGE DOWN или СТРЕЛКА ВНИЗ
Переход на предыдущую страницу (в режиме подгонки по размеру окна)	PAGE UP или СТРЕЛКА ВВЕРХ

Перемещение в режиме предварительного просмотра и просмотра макета страницы	Клавиши
Прокрутка вниз с мелкими шагами	СТРЕЛКА ВНИЗ
Прокрутка вниз на один экран	PAGE DOWN
Переход в конец страницы	CTRL+СТРЕЛКА ВНИЗ
Прокрутка вверх с мелкими шагами	СТРЕЛКА ВВЕРХ
Прокрутка вверх на один экран	PAGE UP
Переход на начало страницы.	CTRL+СТРЕЛКА ВВЕРХ
Прокрутка вправо с мелкими шагами	СТРЕЛКА ВПРАВО
Прокрутка вправо до края страницы	END или CTRL+СТРЕЛКА ВПРАВО
Переход на правый нижний угол страницы	CTRL+END
Прокрутка влево с мелкими шагами	СТРЕЛКА ВЛЕВО
Прокрутка влево до края страницы	HOME или CTRL+СТРЕЛКА ВЛЕВО
Переход на левый верхний угол страницы	CTRL+HOME

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecUseKeyboardInPrintPreviewSW":1:"Foo":"Invisible"}

Использование клавиатуры в окне модуля и в окне отладки

Просмотр программы Visual Basic	Клавиши
Открытие средства просмотра модели объектов	F2
Открытие диалогового окна **Вызовы**	CTRL+Д (L)
Открытие описания процедуры, переменной или ключевого слова	SHIFT+F2
Возврат на последнюю позицию в процедуре, которая просматривалась перед текущей	CTRL+SHIFT+F2
Вывод на экран предыдущей процедуры	CTRL+СТРЕЛКА ВВЕРХ
Вывод на экран следующей процедуры	CTRL+СТРЕЛКА ВНИЗ
Вывод контекстной справки о ключевом слове, на котором находится курсор	F1
Установка фокуса на поле объекта	CTRL+F2
Установка фокуса на поле процедуры	CTRL+F2 (затем нажать TAB)
Вывод списка свойств и методов	CTRL+O (J)
Вывод списка констант	CTRL+SHIFT+O (J)
Ввод выбранного элемента в список	CTRL-ENTER или TAB
Ввод выбранного элемента и перемещение курсора на следующую строку	ENTER
Скрытие списка	ESC
Вывод кратких сведений	CTRL+Ш (I)
Вывод сведений о параметрах	CTRL+SHIFT+Ш (I)
Завершение набора ключевого слова Visual Basic	CTRL+ПРОБЕЛ

Редактирование программы Visual Basic	Клавиши
Выделение всего видимого текста программы в окне модуля	CTRL+Ф (A)
Поиск следующего вхождения образца, указанного в диалоговом окне **Поиск** или **Замена**, после закрытия диалогового окна	F3
Поиск предыдущего вхождения образца, указанного в диалоговом окне **Поиск** или **Замена**, после закрытия диалогового окна	SHIFT+F3
Поиск следующего вхождения образца.	CTRL+F3
Добавление отступа для выделенных строк	TAB или CTRL+Ь (M)
Удаление одного отступа для выделенных строк	SHIFT+TAB или CTRL+SHIFT+Ь (M)
Удаление текущей строки в буфер	CTRL+Н (Y)

Запуск и отладка программ на Visual Basic	Клавиши
Открытие окна отладки	CTRL+П (G)
Запуск подпрограммы без параметров,	F5

на которой находится курсор, из окна модуля	
Переходы между верхней и нижней половиной окна	F6
Шаг с заходом в процедуру	F8
Шаг с выполнением процедуры (перешагивание через процедуру)	SHIFT+F8
Шаг с выходом из процедуры	CTRL+SHIFT+F8
Запуск программы с ее последующей остановкой в строке, в которой находится курсор	CTRL+F8
Определение контрольного значения для выбранного выражения	SHIFT+F9
Установка и снятие точки останова в выбранной строке	F9
Снятие всех точек останова	CTRL+SHIFT+F9
Установка точки останова на следующей инструкции	CTRL+F9
Прерывание выполнения программы или макроса	CTRL+BREAK
Продолжение выполнения программы или макроса	F5
Сброс выполнения программы или макроса	SHIFT+F5
Переключение режима остановок на необрабатываемых ошибках и переход к следующей инструкции	ALT+F5
Переключение режима остановок в модулях класса с последующим продолжением выполнения	ALT+F8

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acrefUseKeyboardModuleWindowSW":1:"Foo":"Invisible"}

Пакет Office 97 ValuPack

На компакт диске Microsoft Office Professional Edition 97, с которого устанавливается Microsoft Access, помимо обычной программы установки содержится также пакет Office 97 ValuPack. В этот пакет включены дополнительные драйверы, файлы справки, документация и другие полезные программы и файлы.

Дополнительные сведения содержатся в файле справки пакета Office 97 ValuPack Help (Valupk8.hlp), который находится в папке ValuPack на компакт диске Office 97 или Microsoft Access 97 или на сетевом сервере с которого происходила установка. Этот файл справки содержит общие сведения и инструкции по установке для каждого компонента.

Примечание. Если установка Office 97 или Microsoft Access производилась не с компакт-диска, то пакет ValuPack можно получить по сети Web. Для этого выберите в меню **?** команду **Microsoft на Web** и подкоманду **Бесплатные материалы**.

В следующей таблице приведено краткое описание некоторых компонентов, входящих в пакет ValuPack, которые могут быть особенно полезны при работе с Microsoft Access 97.

Компонент	**Расположение**
Data Access Pak **Примечание.** Содержит три драйвера, не доступных при стандартной установке: Lotus 1-2-3, Paradox и Microsoft Exchange/OutLook	\ValuPack\DataAcc
Электронная версия книги Разработка приложений для Microsoft Access 97	\ValuPack\Access\Openbook.htm
Файлы справки по элементу просмотра Web ActiveX (входящему в комплект поставки Microsoft Internet Explorer версии 3.0)	\ValuPack\Access\WebHelp
Microsoft Internet Explorer версия 3.0	\ValuPack\IExplore
Личный Web-сервер	\ValuPack\PWebSrv
Мастер размещения в Web (Web Publishing Wizard)	\ValuPack\WebPost

События: совместная работа объектов базы данных

Что такое событие?

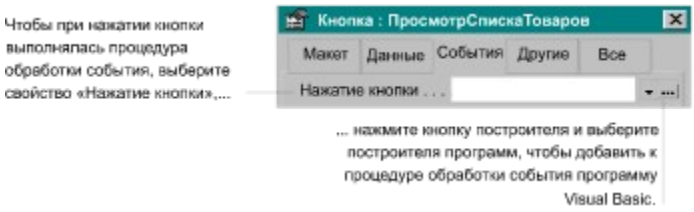
Событие - это определенное действие которое происходит над или возникает в определенном объекте. Microsoft Access реагирует на большое число различных событий: нажатие кнопки мыши, изменение данных, открытие или закрытие форм, и т. д. Обычно события возникают вследствие действий пользователя.

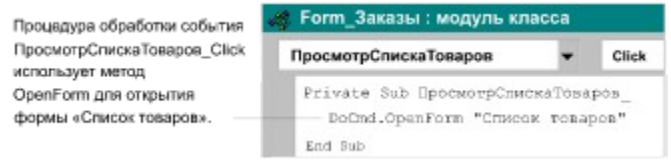
С помощью процедур обработки события или макроса возможно определение собственных откликов на события, происходящие в форме, отчете или элементе управления.

Предположим, что при нажатии кнопки **Сведения** в форме «Заказы» должна открываться форма «Товары». В следующем примере показано, как сделать это с помощью процедуры обработки события или с помощью макроса.

Отклик на событие «Нажатие кнопки» (Click) с помощью процедуры обработки события

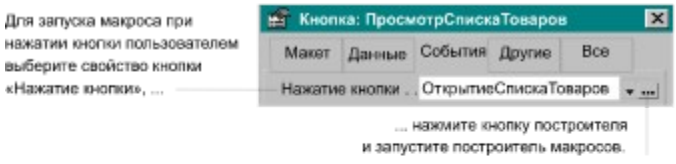
При создании процедуры обработки события для объекта в модуль формы или модуль отчета добавляется именованная заготовка процедуры обработки события для данного события и объекта. Остается только дописать код, определяющий желаемый отклик на событие в форме или отчете.

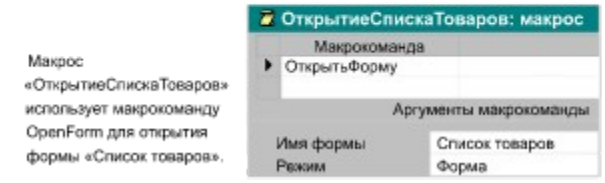




Отклик на событие «Нажатие кнопки» (Click) с помощью макроса

Чтобы в ответ на событие запустить макрос, следует открыть окно свойств для формы, отчета или элемента управления, найти соответствующее событию свойство и установить в качестве его значения вызов макроса.





Для получения полного списка событий нажмите кнопку ....

Более подробные сведения о запуске макроса и процедуре обработки события для формы, отчета или элемента управления можно получить, нажав кнопку ....

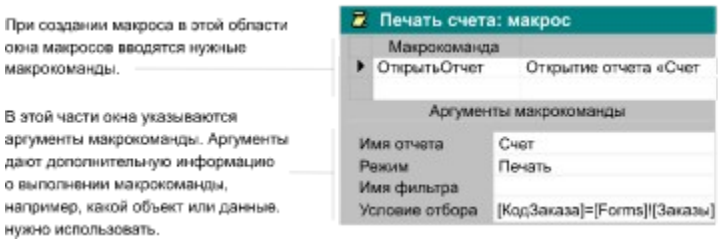
{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ

пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acconEventsMakingYourDatabaseObjectsWorkTogetherW":1:"Foo":"Invisible"}

Макросы: что это такое и как они работают

Что такое макрос?

Макросом называют набор из одной или более макрокоманд, выполняющих определенные операции, такие как открытие форм или печать отчетов. Макросы могут быть полезны для автоматизации часто выполняемых задач. Например, при нажатии пользователем кнопки можно запустить макрос, который распечатает отчет.



Макрос может быть как собственно макросом, состоящим из последовательности макрокоманд, так и группой макросов. В некоторых случаях для решения, должна ли в запущенном макросе выполняться определенная макрокоманда, может применяться условное выражение.

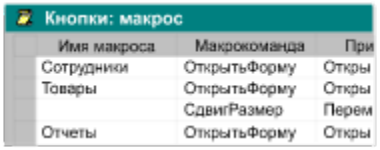
Последовательность макрокоманд

Следующий макрос состоит из серии макрокоманд. Эти макрокоманды выполняются каждый раз при запуске макроса. Для запуска макроса следует обратиться к имени макроса «Просмотр товаров».

Группа макросов

При наличии большого числа макросов, объединение родственных макросов в группы может упростить управление базой данных. Для просмотра имен макросов для выбранной группы макросов достаточно в окне макроса в меню **Вид** выбрать команду **Имена макросов**.

Например, следующая группа макросов с именем «Кнопки» состоит из трех родственных макросов: «Сотрудники», «Товары» и «Отчеты». В каждом макросе содержится макрокоманда **ОткрытьФорму (OpenForm)**, а в макросе «Товары» кроме того макрокоманда **СдвигРазмер (MoveSize)**.



Имя в столбце **Имя макроса** определяет каждый макрос. При запуске макроса в группе макросов, выполняется макрокоманда в столбце **Макрокоманда**, а также все следующие макрокоманды, в которых столбец **Имя макроса** пуст.

Для запуска макроса из группы макросов следует указать имя группы, а затем, через точку, имя макроса. В предыдущем примере для обращения к макросу «Сотрудники» в группе макросов «Кнопки» следовало использовать синтаксис **Кнопки.Сотрудники**.

Условные макрокоманды

Для вывода столбца **Условие** следует в окне макроса в меню **Вид** выбрать команду **Условия**. Следующий макрос запускает макрокоманды **Сообщение (MsgBox)** и **ОстановитьМакрос (StopMacro)** только в тех случаях, когда условие в столбце **Условие** истинно (когда поле «КодПоставщика» имеет

значение **Null**).

Более подробные сведения о создании макросов можно получить, нажав кнопку ... Более подробные сведения о запуске макросов можно получить, нажав кнопку

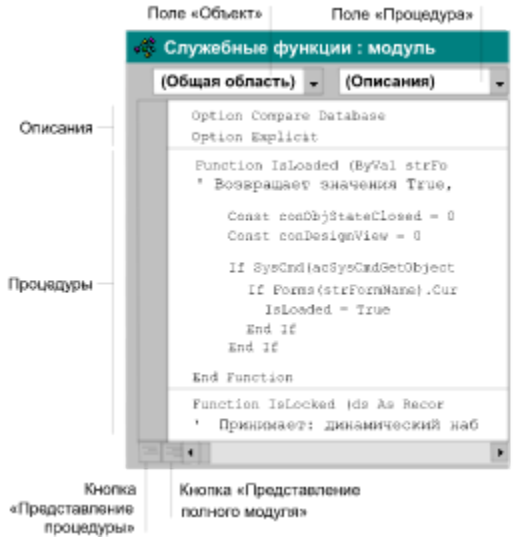
...

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconMacrosWhatTheyAreHowTheyWorkW":1:"Foo":"Invisible"}

Модули: что это такое и как они работают

Что такое модуль?

Модуль - это набор объявлений и процедур на языке Visual Basic для приложений, собранных в одну программную единицу.



Существует два основных типа модулей: модули класса и стандартные модули. Каждая процедура в модуле может быть либо процедурой-функцией **Function,** либо процедурой **Sub**.

Модули класса

Модули форм и модули отчетов являются модулями класса, связанными с определенной формой или отчетом. Они часто содержат процедуры обработки событий, запускаемых в ответ на событие в форме или отчете. Процедуры обработки событий используются для управления поведением формы или отчета и их откликом на события, например, такие как нажатие кнопки.

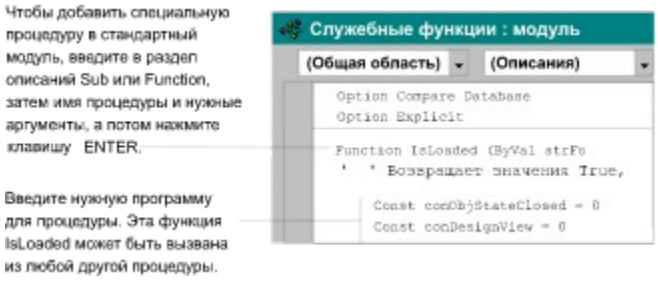
При создании первой процедуры обработки события для формы или отчета автоматически создается связанный с ней модуль формы или отчета. Для просмотра модуля для формы или отчета достаточно нажать кнопку **Программа** ... на панели инструментов в режиме конструктора.

В процедурах модулей форм и отчетов могут содержаться вызовы процедур, добавленных в стандартные модули.

В Microsoft Access 95 модули класса существуют только в связи с формой или отчетом. В Microsoft Access 97 модули класса могут также существовать независимо от них. Этот тип модуля класса выводится на вкладке **Модули** окна базы данных. Модуль класса на этой вкладке можно использовать для создания описания специального объекта. Более подробные сведения об использовании в программах этих модулей класса можно получить, нажав кнопку ...

Стандартные модули

В стандартных модулях содержатся общие процедуры, не связанные ни с каким объектом, а также часто используемые процедуры, которые могут быть запущены из любого окна базы данных.



Для просмотра списка стандартных модулей базы данных выберите вкладку **Модули** в окне базы данных. Формы, отчеты и стандартные модули выводятся также в окно просмотра объектов.

Более подробные сведения о процедурах можно получить, нажав кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconModulesWhatTheyAreHowTheyWorkW":1:"Foo":"Invisible"}

Запуск макроса или процедуры обработки для события в форме, отчете или элементе управления

Microsoft Access реагирует на события различных типов, возникающие в формах, отчетах или элементах управления, например, на нажатия кнопок мыши, изменение данных, а также на открытие или закрытие формы или отчета.

1. Откройте форму или отчет в режиме конструктора.
2. Создайте макрос или процедуру обработки события. Например, можно создать макрос или процедуру обработки события, выводящие сообщение при нажатии кнопки.
3. Введите в ячейку свойства соответствующего события формы, отчета или элемента управления имя макроса или выберите элемент **[Процедура обработки событий]**.

   Например, для вызова макроса, выводящего сообщение при нажатии кнопки, введите имя макроса в ячейку свойства **Нажатие кнопки (OnClick)**. Для использования процедуры обработки события создайте такую процедуру для события **Click** для кнопки и выберите в ячейке свойства этого события **[Процедура обработки событий]**.

Примечания

- Для получения дополнительных сведений о создании макроса нажмите кнопку ...
- Для получения дополнительных сведений по созданию процедуры обработки события нажмите кнопку ....
- Для получения дополнительных сведений о причинах и порядке возникновения событий нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowAttachMacroFormReportControlSW":1:"Foo":"Invisible"}

Причины и порядок возникновения событий

Каждое отдельное действие, например, переход с одного элемента управления на другой, может явиться причиной ряда других событий, возникающих в определенной последовательности. Понимание того, когда и в какой последовательности возникают события, весьма существенно, поскольку порядок событий определяет условия и очередность выполнения макросов и процедур обработки событий. Например, если созданы две процедуры обработки событий, которые должны выполняться в определенном порядке, пользователь должен быть уверен, что события, вызывающие эти процедуры, возникают в том же порядке.

Предполагаемые действия

- Изучение порядка событий для элементов управления в формах.
- Изучение порядка событий для записей в формах.
- Изучение порядка событий для форм и подформам.
- Изучение порядка событий клавиатуры и мыши.
- Изучение порядка событий в отчетах и в разделах отчетов.

Примечание. Для изучения порядка событий откройте форму «События» в демонстрационном приложении «Заказы». Для любого события, возникающего в форме «События», например, при открытии формы или при переходе на новую запись, результирующая цепочка событий в порядке их возникновения будет отражена в окне формы «ПротоколСобытий». Для получения справки по открытию приложения «Заказы» нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acconWhenDoEventsOccurSW":1:"Foo":"Invisible"}

Порядок событий элементов управления в формах

События элемента управления возникают в форме при переводе фокуса на элемент управления и при изменении или обновлении данных в элементе управления.

Перевод фокуса на элемент управления

При переводе фокуса в форму, в которой выводятся один или несколько активных элементов управления, или при переходе на другой элемент управления в той же форме, возникают события **Вход (Enter)** и **Получение фокуса (GotFocus)** в следующем порядке:

Вход ⇒ Получение фокуса

Когда пользователь открывает форму, эти события возникают после событий, связанных с открытием формы, таких как **Открытие (Open)**, **Включение (Activate)** или **Текущая запись Current)**, в такой последовательности:

Открытие (форма) ⇒ Включение (форма) ⇒ Текущая запись (форма) ⇒ Вход (элемент управления) ⇒ Получение фокуса (элемент управления)

При выводе фокуса из элемента управления в форме, например, при закрытии формы, в которой выводятся один или несколько активных элементов управления, или при переходе на другой элемент управления в той же форме, возникают события **Выход (Exit)** и **Потеря фокуса (LostFocus)**:

Выход ⇒ Потеря фокуса

Когда пользователь закрывает форму, события **Выход (Exit)** и **Потеря фокуса (LostFocus)** возникают перед событиями, связанными с закрытием формы, такими как **Выгрузка (Unload)**, **Отключение (Deactivate)** или **Закрытие (Close)**:

Выход (элемент управления) ⇒ Потеря фокуса (элемент управления) ⇒ Выгрузка (форма) ⇒ Отключение (форма) ⇒ Закрытие (форма)

Изменение и обновление данных в элементе управления

Когда пользователь вводит в элемент управления в форме новые данные или изменяет существующие, а после этого переводит фокус на другой элемент управления, события **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)** возникают в следующем порядке:

До обновления ⇒ После обновления

События **Выход (Exit)** и **Потеря фокуса (LostFocus)** для изменяемого элемента управления возникают после событий **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)**:

До обновления ⇒ После обновления ⇒ Выход ⇒ Потеря фокуса

Когда пользователь изменяет текст, содержащийся в поле, или содержимое поля в поле со списком, возникает событие **Изменение (Change)**. Данное событие возникает при любом изменении содержимого элемента управления, причем до перехода на другой элемент управления или на другую запись и следовательно, до возникновения событий **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)**. Если курсор находится в поле, то при нажатии любой клавиши, соответствующей символу, возникают следующие события:

Клавиша вниз ⇒ Нажатие клавиши ⇒ Изменение ⇒ Клавиша вверх

Событие **Отсутствие в списке (NotInList)** возникает, когда пользователь вводит в поле со списком значение, отсутствующее в списке, и предпринимает попытку перехода на другой элемент управления или другую запись. Это событие возникает после событий клавиатуры и события **Изменение (Change)** поля со списком, но до любых событий элемента управления или формы. Если для свойства поля со списком **Ограничиться списком (LimitToList)** задано значение «Да», то сразу после события **Отсутствие в списке (NotInList)** возникает событие формы **Ошибка (Error)**:

Клавиша вниз ⇒ Нажатие клавиши ⇒ Изменение ⇒ Клавиша вверх ⇒ Отсутствие в списке ⇒ Ошибка

Для получения дополнительных сведений о событиях нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acconOrderEventsControlsFormsSW":1:"Foo":"Invisible"}

Порядок событий записи в форме

События записи возникают в форме при переводе фокуса на запись, обновлении содержимого записи, удалении существующей записи или записей, а также при создании новой записи.

Перемещение фокуса по записям и обновление содержимого записей

При переводе фокуса на существующую запись в форме, вводе новых или изменении существующих данных в записи и при переходе на другую запись в форме возникает такая последовательность событий:

> Текущая запись (форма) ⇒ До обновления (форма) ⇒ После обновления (форма) ⇒ Текущая запись (форма)

При выходе из измененной записи, но до входа в новую запись, возникают события **Выход (Exit)** и **Потеря фокуса (LostFocus)** последнего элемента управления, имевшего фокус. Эти события возникают после событий формы **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)**:

> До обновления (форма) ⇒ После обновления (форма) ⇒ Выход (элемент управления) ⇒ Потеря фокуса (элемент управления) ⇒ Текущая запись (форма)

При переводе фокуса с одного элемента управления в форме на другой возникают события каждого элемента управления. Например, в различных ситуациях возникают такие последовательности событий:

- открытие формы и изменение данных в элементе управления:

  Текущая запись (форма) ⇒ Вход (элемент управления) ⇒ Получение фокуса (элемент управления) ⇒ До обновления (элемент управления) ⇒ После обновления (элемент управления)
- перевод фокуса на другой элемент управления:

  Вход (элемент управления 1) ⇒ Потеря фокуса (элемент управления 1) ⇒ Вход (элемент управления 2) ⇒ Получение фокуса (элемент управления 2)
- перевод фокуса на другую запись:

  До обновления (форма) ⇒ После обновления (форма) ⇒ Выход (элемент управления 2) ⇒ Потеря фокуса (элемент управления 2) ⇒ Текущая запись (форма)

Для получения дополнительных сведений о порядке событий элементов управления в формах нажмите кнопку ....

Удаление записей

Когда пользователь удаляет запись, в форме возникают следующие события, и Microsoft Access выводит окно диалога с приглашением подтвердить удаление:

> Удаление ⇒ До подтверждения Del ⇒ После подтверждения Del

Если пользователь отменяет событие **Удаление (Delete)**, то события **До подтверждения Del (BeforeDelConfirm)** и **После подтверждения Del (AfterDelConfirm)** не возникают и окно диалога не выводится.

Создание новой записи

При переводе фокуса на новую (пустую) запись в форме и создании новой записи путем ввода символов с клавиатуры, события возникают в следующем порядке:

> Текущая запись (форма) ⇒ Вход (элемент управления) ⇒ Получение фокуса (элемент управления) ⇒ До вставки (форма) ⇒ После вставки (форма)

События **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)** для элементов управления в форме и для новой записи возникают после события **До вставки (BeforeInsert)**, но до события **После вставки (AfterInsert)**.

Для получения дополнительных сведений о событиях нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "nïSnïSnïSnïSnïS nïS Web nïSnïSnïS nïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS":"acconOrderEventsRecordsFormsSW":1:"Foo":"Invisible"}

Порядок событий формы и подчиненной формы

События формы возникают при открытии и закрытии форм, при переходах между формами и при обработке данных в форме.

Открытие и закрытие формы

При открытии формы события возникают в следующем порядке:

Открытие ⇒ Загрузка ⇒ Изменение размера ⇒ Включение ⇒ Текущая запись

Кроме того, если все элементы управления в форме являются отключенными или скрытыми, событие **Получение фокуса (GotFocus)** возникает в форме после события **Включение (Activate)**, но до события **Текущая запись (Current)**.

При закрытии формы события возникают в следующем порядке:

Выгрузка ⇒ Отключение ⇒ Закрытие

Если форма не содержит активных элементов управления, событие **Потеря фокуса (LostFocus)** возникает в форме после события **Выгрузка (Unload)**, но до события **Отключение (Deactivate)**.

Переходы между формами

При переходе из открытого окна формы в окно другой открытой формы возникает событие **Отключение (Deactivate)** первой формы и событие **Включение (Activate)** второй формы:

Отключение (форма 1) ⇒ Включение (форма 2)

Событие формы **Отключение (Deactivate)** возникает также при переходе из окна формы в другое окно Microsoft Access. Однако событие **Отключение** не возникает при переходе в окно диалога, в окно формы со значением **Да** свойства **Всплывающее окно (PopUp)** или в окно другого приложения.

Примечание. При переходе в окно открытой формы событие **Открытие (Open)** не возникает, даже если переход в форму осуществляется с помощью макрокоманды **ОткрытьФорму (OpenForm).**

Обработка данных в форме

При перемещении по записям формы и изменении данных возникают события формы и события элементов управления. Например, при открытии формы возникает следующая цепочка событий:

Открытие (форма) ⇒ Загрузка (форма) ⇒ Изменение размера (форма) ⇒ Включение (форма) ⇒ Текущая запись (форма) ⇒ Вход (элемент управления) ⇒ Получение фокуса (элемент управления)

Аналогично, при закрытии формы возникают следующие события:

Выход (элемент управления) ⇒ Потеря фокуса (элемент управления) Выгрузка (форма) ⇒ Отключение (форма) ⇒ Закрытие (форма)

При изменении данных в элементе управления события **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate),** как для элемента управления, так и для формы, возникают до события **Выход (Exit).** Для получения дополнительных сведений о порядке событий элементов управления и событий формы нажмите кнопку ....

Работа с подчиненными формами

При открытии формы, содержащей подчиненную форму, подчиненная форма и ее записи загружаются до главной формы. Таким образом, события подчиненной формы и содержащихся в ней элементов управления, такие как **Открытие (Open), Текущая запись (Current), Вход (Enter)** и **Получение фокуса (GotFocus),** возникают до событий главной формы. Однако событие **Включение (Activate)** для подчиненных форм не возникает, поэтому при открытии формы, содержащей подчиненную форму, возникает только событие **Включение** главной формы.

Аналогично этому, при закрытии формы, содержащей подчиненную форму, подчиненная форма и ее записи выгружаются до главной формы. Событие **Отключение (Deactivate)** для подчиненных форм не возникает, поэтому при закрытии формы, содержащей подчиненную форму, возникает только событие **Отключение** главной формы. События элементов управления, формы и подчиненной формы возникают в следующем порядке.

1. События элементов управления в подчиненной форме, такие как **Выход (Exit)** и **Потеря фокуса (LostFocus).**
2. События элементов управления в главной форме (в том числе элемента управления самой

подчиненной формы).

3. События формы, такие как **Отключение (Deactivate)** и **Закрытие (Close).**
4. События подчиненной формы.

Для получения дополнительных сведений о событиях нажмите кнопку  .

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїSпїS":"acconOrderEventsFormsSW":1:"Foo":"Invisible"}

Порядок событий клавиатуры и мыши

События клавиатуры возникают в имеющих фокус формах или элементах управления в форме при нажатиях клавиш или при передаче команд клавиатуры. События мыши возникают для форм, разделов форм и элементов управления в формах при нажатии кнопок мыши, если в этот момент указатель установлен на форме, разделе формы или элементе управления.

События клавиатуры

Если элемент управления в форме имеет фокус, а пользователь нажимает и отпускает клавишу или выполняет макрокоманду **КомандыКлавиатуры (SendKeys),** возникает следующая цепочка событий:

Клавиша вниз ⇒ Нажатие клавиши ⇒ Клавиша вверх

При нажатии или отпускании клавиши или передаче команды клавиатуры, соответствующей символу из набора ANSI, всегда возникают события **Клавиша вниз (KeyDown), Нажатие клавиши (KeyPress)** и **Клавиша вверх (KeyUp)**. При нажатии и удерживании клавиши ANSI попеременно возникают события **Клавиша вниз, Нажатие клавиши, Клавиша вниз, Нажатие клавиши** и т.д. Это продолжается до отпускания клавиши; при отпускании клавиши возникает событие **Клавиша вверх**. Для просмотра набора символов ANSI нажмите кнопки ... и

...

.

При нажатии и отпускании клавиши, не являющуюся клавишей ANSI, возникают события **Клавиша вниз (KeyDown)** и **Клавиша вверх (KeyUp)**. Если пользователь нажимает и удерживает клавишу, не являющуюся клавишей ANSI, то до отпускания клавиши возникает и повторяется событие **Клавиша вниз**; при отпускании клавиши возникает событие **Клавиша вверх**.

Если нажатие клавиши вызывает другое событие элемента управления, то другое событие возникает после события **Нажатие клавиши (KeyPress),** но до события **Клавиша вверх (KeyUp**). Например, если нажатие клавиши приводит к изменению текста в поле, то событие **Изменение (Change)** возникает в следующей цепочке событий:

Клавиша вниз ⇒ Нажатие клавиши ⇒ Изменение ⇒ Клавиша вверх

Если нажатие клавиши вызывает перевод фокуса из одного элемента управления на другой, то событие **Клавиша вниз (KeyDown)** возникает для первого элемента управления, а события **Нажатие клавиши (KeyPress)** и **Клавиша вверх (KeyUp)** для второго. Например, если пользователь изменяет данные в элементе управления и нажимает клавишу TAB для перехода на следующий элемент управления, то возникают следующие события:

- первый элемент управления:

  Клавиша вниз ⇒ До обновления ⇒ После обновления ⇒ Выход ⇒ Потеря фокуса
- второй элемент управления:

  Вход ⇒ Получение фокуса ⇒ Нажатие клавиши ⇒ Клавиша вверх

События мыши

Если указатель установлен на элементе управления в форме, и пользователь нажимает и отпускает кнопку мыши, то возникает следующая цепочка событий элемента управления:

Кнопка вниз ⇒ Кнопка вверх ⇒ Нажатие кнопки

Если элемент управления имеет фокус, а пользователь устанавливает указатель на другой элемент управления и нажимает кнопку мыши, возникают следующие события:

- первый элемент управления:

  Выход ⇒ Потеря фокуса
- второй элемент управления:

  Вход ⇒ Получение фокуса ⇒ Кнопка вниз ⇒ Кнопка вверх ⇒ Нажатие кнопки

Если пользователь устанавливает указатель на элемент управления в другой записи и нажимает кнопку мыши, то событие формы **Текущая запись (Current)** возникает до события **Вход (Enter)** элемента управления.

Двойное нажатие кнопки мыши приводит к возникновению обоих событий **Двойное нажатие кнопки (DblClick)** и **Нажатие кнопки (Click).** Например, если указатель установлен на элементе управления, не являющемся кнопкой, то двойное нажатие кнопки мыши вызывает следующую цепочку событий элемента управления:

Кнопка вниз ⇒ Кнопка вверх ⇒ Нажатие кнопки ⇒ Двойное нажатие кнопки ⇒ Кнопка вверх

При двойном нажатии кнопки мыши на кнопке эти события возникают после второго события **Нажатие кнопки (Click)**.

Событие формы, раздела или элемента управления **Перемещение указателя (MouseMove)** возникает при перемещении указателя по форме, разделу или элементу управления. Это событие является независимым от других событий мыши.

Для получения дополнительных сведений о событиях клавиатуры нажмите кнопку ....

Для получения дополнительных сведений о событиях мыши нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acconOrderEventsKeyboardMouseSW":1:"Foo":"Invisible"}

Порядок событий отчета и разделов отчета

События отчета и разделов отчета возникают при открытии отчета для печати или предварительного просмотра, а также при закрытии отчета.

События отчета

При открытии отчета для печати или предварительного просмотра и последующем закрытии отчета или переходе в другое окно Microsoft Access события возникают в следующем порядке:

Открытие ⇒ Включение ⇒ Отключение ⇒ Закрытие

При переходе между двумя окнами открытых отчетов возникает событие **Отключение (Deactivate)** первого отчета и событие **Включение (Activate)** второго:

Отключение (отчет1) ⇒ Включение (отчет2)

Событие **Отключение (Deactivate)** отчета возникает также при переходе из окна отчета в другое окно Microsoft Access. Однако событие **Отключение** не возникает при переходе в окно диалога, в окно формы со значением **Да** свойства **Всплывающее окно (PopUp)** или в окно другого приложения.

При открытии отчета, созданного на основе запроса, Microsoft Access генерирует событие **Открытие (Open)** до выполнения базового запроса. Это делает возможным ввод в отчет условий отбора с помощью макроса или процедуры обработки события, которые запускаются в ответ на событие **Открытие**. Например, макрос или процедура обработки события позволяют открыть специальное окно диалога, через которое пользователь вводит в отчет условия отбора.

События разделов отчета

При печати или предварительном просмотре отчета события разделов отчета **Форматирование (Format)** и **Печать (Print)** возникают после событий отчета **Открытие (Open)** и **Включение (Activate)** и до событий отчета **Отключение (Deactivate)** и **Закрытие (Close)**:

Открытие (отчет) ⇒ Включение (отчет) ⇒ Форматирование (раздел отчета) ⇒ Печать (раздел отчета) ⇒ Отключение (отчет) ⇒ Закрытие (отчет)

Кроме того, на стадии форматирования отчета или после завершения форматирования, но до возникновения события **Печать**, могут возникнуть следующие события:

- событие **Возврат (Retreat)** возникает при возвращении в предыдущий раздел на стадии форматирования;
- событие **Отсутствие данных (NoData)** возникает при отсутствии выводящихся в отчете записей;
- событие **Страница (Page)** возникает после завершения форматирования, но до начала печати. Пользователь имеет возможность использовать данное события для изменения оформления отчета при печати.

Для получения дополнительных сведений о событиях нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconOrderEventsReportsSW":1:"Foo":"Invisible"}

Создание или изменение макросов

Предполагаемые действия

- Макросы: что это такое и как они работают
- Создание макроса
- Создание группы макросов

Создание макросов

Предполагаемые действия

- Создание макроса
- Создание группы макросов
- Добавление макрокоманды в макрос
- Изменение положения макрокоманды в макросе
- Ввод выражения в аргумент макрокоманды
- Использование условий в макросе
- Вывод на экран и скрытие имен макросов и условий в окне макроса

{ewc HLP95EN.DLL, DYNALINK, "Link to the Web or other sources":"acdecCreateMacroW":1:"Foo":"Invisible"}

Отладка макросов

Предполагаемые действия

Отладка макроса в пошаговом режиме
Разрешение вопросов при работе с макросами

{ewc HLP95EN.DLL, DYNALINK, "Link to the Web or other sources":"acdecDebugMacroW":1:"Foo":"Invisible"}

Создание или изменение процедур в модулях

Предполагаемые действия

- Модули: что это такое и как они работают
- Что такое процедура?

Создание своей первой процедуры

Предполагаемые действия

- Создание специальной функции
- Создание специальной процедуры Sub
- Создание процедуры обработки события
- Указание аргументов процедуры, инструкции или метода в программе Visual Basic

Настройка окна модуля

Предполагаемые действия

- Изменения параметров представления программ Visual Basic для приложений (VBA) в окне модуля
- Отключение проверки синтаксиса в программах Visual Basic для приложений (VBA)
- Отключение обработки ошибок в программах Visual Basic для приложений (VBA)
- Требование обязательного описания переменных в программах Visual Basic для приложений (VBA)

Отладка программ Visual Basic

Предполагаемые действия

- Отладка программ Visual Basic для приложений (VBA)
- Использование точек останова для временной остановки работы программы
- Выполнение программы Visual Basic для приложений (VBA) в пошаговом режиме
- Вывод значений переменных и выражений на панели проверки в окне отладки
- Установка контрольных выражений в программе Visual Basic для приложений (VBA)
- Трассировка вызовов процедур при отладке программ Visual Basic
- Вывод значений переменных и выражений на панели проверки в окне отладки

Вызов справки в окне модуля

Предполагаемые действия

- Вызов встроенной справки Visual Basic
- Копирование примеров программ Visual Basic из окна справки в окно модуля

Сведения о модулях в программах Visual Basic

Предполагаемые действия

- Модули: что это такое и как они работают
- Что такое процедура?
- Выполнение программы Visual Basic для приложений (VBA)
- Работа с данными и с объектами базы данных с помощью программ Visual Basic для приложений (VBA)

Работа с макросами в программах Visual Basic

Предполагаемые действия

- Преобразование макросов в процедуры Visual Basic для приложений (VBA)
- Выполнение макрокоманды в процедуре Visual Basic для приложений (VBA)
- Макрокоманды, не имеющие соответствующих методов объекта **DoCmd**, и эквивалентные им инструкции.

Работа с объектами в программах Visual Basic с помощью средства просмотра модели объектов

Предполагаемые действия

- Поиск метода или свойства в окне просмотра объектов
- Вставка метода или свойства из окна просмотра объектов в модуль

Использование макросов и программ Visual Basic

В Microsoft Access многие действия выполняются с помощью макросов или через интерфейс пользователя. Во многих других СУБД для решения тех же самых задач требуется программирование. Выбор между созданием макроса или разработкой программы Visual Basic для приложений обычно определяется требуемыми действиями.

В каких случаях следует создавать макрос?

Макрос является удобным средством выполнения простых задач, таких как открытие и закрытие форм, вывод на экран и скрытие панелей инструментов или запуск отчетов. Действия, связывающие различные объекты базы данных, выполняются легко и просто, поскольку пользователь не должен запоминать правила синтаксиса — все аргументы, требуемые каждой макрокомандой, выводятся в нижней половине окна макроса.

В дополнение к упомянутым простым действиям, макросы необходимо использовать для выполнения следующих задач.

- Определение общих назначенных клавиш.
- Выполнение макрокоманды или набора макрокоманд при открытии базы данных. Однако определенные действия, которые должны производиться при открытии базы данных, например, открытие формы, могут быть заданы в диалоговом окне **Параметры запуска**.

В каких случаях следует создавать программу Visual Basic?

Программы Visual Basic используют вместо макросов в случаях, когда необходимо:

- Упростить управление базой данных. Поскольку макросы являются объектами, существующими отдельно от использующих их форм и отчетов, поддержание базы данных, в которой реакция на события в формах и отчетах определяется многими макросами, становится достаточно затруднительным. В отличие от этого, процедуры обработки события Visual Basic являются встроенными в описания соответствующих форм и отчетов. При переносе формы или отчета из одной базы данных в другую встроенные процедуры обработки события автоматически переносятся вместе с формой или отчетом.
- Создавать собственные специализированные функции. В Microsoft Access определен ряд встроенных функций, например, функция **IPmt**, которая рассчитывает проценты по платежам. Пользователь имеет возможность использовать для проведения расчетов встроенные функции без необходимости разрабатывать сложные выражения. Однако язык Visual Basic позволяет пользователям создавать собственные функции как для решения задач, выходящих за рамки возможных для встроенных функций, так и для замены сложных выражений, содержащих встроенные функции. Кроме того, создаваемые пользователем функции используются для выполнения одинаковых операций над разными объектами.
- Скрывать сообщения об ошибках. Стандартные сообщения об ошибках Microsoft Access, выводящиеся при возникновении нештатных ситуаций во время работы пользователя с базой данных, могут оказаться малопонятными для пользователя, в особенности, для не имеющего большого опыта работы с Microsoft Access. Средства Visual Basic позволяют перехватывать ошибку при ее возникновении и либо выводить собственное сообщение об ошибке, либо предпринимать определенные действия.
- Создавать или обрабатывать объекты. В большинстве случаев удобнее создавать или изменять объекты в режиме конструктора. Однако в некоторых ситуациях приходится работать с описанием объекта в программе. Средства Visual Basic позволяют выполнять обработку любых объектов в базе данных и самой базы данных.
- Выполнять действия на системном уровне. Выполнение в макросе макрокоманды **ЗапускПриложения (RunApp)** позволяет запускать из собственного приложения другое приложение, работающее в среде Windows или MS-DOS, однако, это практически все, что можно сделать вне Microsoft Access из макроса. Средства Visual Basic позволяют проверять существование файлов, использовать механизм программирования объектов или динамического обмена данными (DDE) для связи с другими приложениями, работающими под управлением Windows, например, Microsoft Excel, а также вызывать функции из библиотек динамической компоновки (DLL) Windows.
- Обрабатывать записи по одной. Инструкции Visual Basic позволяют перебирать наборы записей по одной и выполнять определенные действия над отдельной записью. В отличие от этого, макросы позволяют работать только с целым набором записей.
- Передавать аргументы в специальные процедуры Visual Basic. Пользователь имеет возможность задать в окне макроса значения аргументов макрокоманды при создании макроса, однако, невозможно

изменить значения этих аргументов при выполнении макроса. В отличие от макросов, в программах Visual Basic допускается передача аргументов в программу при запуске программы или использование в качестве аргументов значений переменных. Поэтому использование программ Visual Basic дает более широкие возможности работы с данными.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconWhenShouldUseMacroWhenShouldUseVisualBasicCodSW":1:"Foo":"Invisible"}

Создание кнопки для печати текущей записи

1. Откройте форму в режиме конструктора.
2. Нажмите кнопку **Мастера** на панели элементов, если она еще не нажата.
3. На панели элементов выберите элемент **Кнопка**.
4. Установите указатель в том месте формы, куда требуется поместить кнопку, и нажмите кнопку мыши.
5. В первом окне диалога мастера выберите **Работа с записями** в списке **Категории** и **Печать записи** в списке **Действия**.
6. Выполняйте инструкции, выводящиеся в окна диалога мастера. В последнем окне диалога нажмите кнопку **Готово** для вывода кнопки в режиме конструктора.

Для просмотра процедуры обработки события, обеспечивающей работу кнопки, откройте окно свойств кнопки и нажмите кнопку построителя рядом с ячейкой свойства **Нажатие кнопки (OnClick)**.

Совет. Если требуется печатать текущую запись формы в отчете, следует вызвать макрокоманду **ОткрытьОтчет (OpenReport)** в процедуре обработки события кнопки **Нажатие кнопки (Click)**. Например, для печати текущей записи отчета с именем «Счет» добавьте в процедуру обработки события для кнопки из формы «Заказы» следующие инструкции Visual Basic:

```
Dim strDocName As String
Dim strFilter As String
strDocName = "Счет"
strFilter = "OrderID = Forms!Счет!КодЗаказа"
DoCmd.OpenReport DocName, acViewNormal, , strFilter
```

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconExamplePrintingCurrentRecordMacroSW":1:"Foo":"Invisible"}

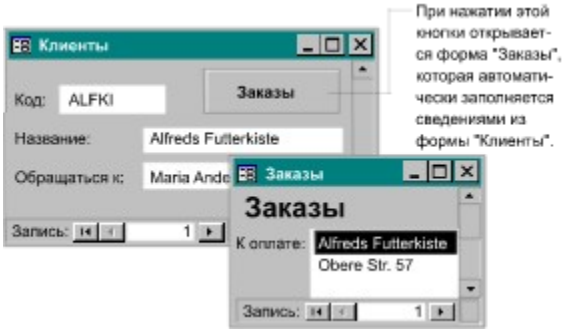
Задание значения элемента управления или свойства при возникновении события

Задание новых значений для элементов управления или свойств в ответ на события, возникающие в форме или отчете, позволяет организовать совместную работу с несколькими объектами.

Ниже приведено несколько примеров, в которых значения задаются с помощью макроса или процедуры обработки события.

Ввод в элемент управления в форме значения элемента управления из другой формы

В форму «Клиенты» помещается кнопка, которая открывает форму «Заказы» и автоматически вводит в заказ нужные данные из формы «Клиенты».



Обновление значения элемента управления в зависимости от значения другого элемента управления

Для элемента управления «ДатаОплаты» задается значение по умолчанию, равное дате исполнения заказа (значению элемента управления «ДатаИсполнения»). Для автоматического изменения значения поля «ДатаОплаты» при изменении значения поля «ДатаИсполнения» можно использовать макрос или процедуру обработки события.

Для получения дополнительных сведений о задании значения элемента управления с помощью значения другого элемента управления нажмите кнопку ....

Задание значения свойства формы или элемента управления в ответ на выбор пользователя или другое событие

Изменение значений свойств формы или элемента управления с помощью макроса или процедуры обработки события позволяет добиться следующих результатов.

- Вывести на экран или скрыть элемент управления в зависимости от значения другого элемента управления. Например, можно сделать элемент управления «ДатаОплаты» в форме «Заказы» скрытым, если заказ уже оплачен (если элемент управления «Оплачено» имеет значение «Да») или видимым, если заказ еще не оплачен.
- Отключить или заблокировать элемент управления в зависимости от значения другого элемента управления. Например, можно сделать недоступным поле «ВодительскиеПрава», если поле «Возраст» содержит значение, которое ниже минимального возраста, начиная с которого выдаются водительские права.
- Вывести разные наборы записей в зависимости от выбора пользователя. Например, можно вывести в форме «Клиенты» разные наборы записей, определяющиеся параметром, выбранным в группе параметров «Категории клиентов». Для этого следует указать в свойстве формы **Источник записей (RecordSource)** имя одного из нескольких запросов, в каждом из которых отбираются записи, соответствующие определенной категории клиентов.

Для получения дополнительных сведений о задании значения элемента управления в ответ на событие нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconSettingValuesMacrosSW":1:"Foo":"Invisible"}

Задание значения элемента управления на основании значения другого элемента управления

Для того чтобы задать значение элемента управления на основании значения другого элемента управления следует включить в макрос макрокоманду **ЗадатьЗначение (SetValue)** или использовать инструкцию присваивания в процедуре Visual Basic.

Предполагаемые действия

... Задание значения элемента управления на основании значения другого элемента управления с помощью макроса.

... Задание значения элемента управления на основании значения другого элемента управления с помощью программы Visual Basic. Applications

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowSetValueControlBasedValueAnotherControlSW":1:"Foo":"Invisible"}

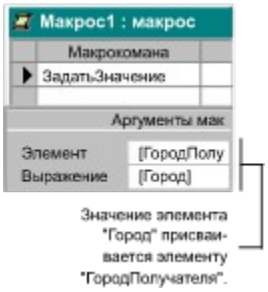
Задание значения элемента управления на основании значения другого элемента управления с помощью макроса

1 Создайте макрос.

Инструкции

2 В пустой строке бланка выберите макрокоманду **ЗадатьЗначение (SetValue)** в списке макрокоманд.

3 Укажите в аргументе **Элемент** идентификатор элемента управления, для которого задается значение, а в аргументе **Выражение** идентификатор элемента управления, значение которого присваивается первому элементу.



4 Нажмите кнопку **Сохранить** для записи макроса.

5 Выполните макрос.

Инструкции

Примечание. В зависимости от контекста для элементов управления могут потребоваться полные ссылки. Для получения дополнительных сведений по использованию идентификаторов в выражениях нажмите кнопку .

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowSetValueControlBasedValueAnotherControlMacroSW":1:"Foo":"Invisible"}

Задание значения элемента управления на основании значения другого элемента управления с помощью программы Visual Basic

1 Создайте процедуру Visual Basic.

... Инструкции

2 Добавьте в процедуру инструкцию присваивания, состоящую из идентификатора элемента управления, значение которого вы хотите задать, знака равенства (=) и идентификатора второго элемента управления, значение которого присваивается первому элементу.

- Для ссылки на значение элемента управления в текущей форме введите ключевое слово **Me** с последующим оператором **!** и именем элемента управления. Например, в следующей инструкции значение поля «Город» в текущей форме присваивается элементу управления «ГородПолучателя» в этой же форме:

  ```
  Me![ГородПолучателя] = Me![Город]
  ```

- Для ссылки на значение элемента управления в другой форме введите полный идентификатор элемента управления. Например, следующий идентификатор указывает ссылку на элемент управления «ГородПолучателя» в форме «ФормаДоставка»:

  ```
  Forms![ФормаДоставка]![ГородПолучателя]
  ```

3 Выполните процедуру.

... Инструкции

Для получения дополнительных сведений о ссылках на значение элемента управления нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowSetValueControlBasedValueAnotherControlVBSW":1:"Foo":"Invisible"}

Задание значения свойства в ответ на событие

Для того чтобы задать значение свойства в ответ на событие, например, нажатие кнопки, следует включить в макрос макрокоманду **ЗадатьЗначение (SetValue)** или использовать инструкцию присваивания в процедуре Visual Basic.

Предполагаемые действия

- Задание значения свойства в ответ на событие с помощью макроса.
- Задание значения свойства в ответ на событие с помощью программы Visual Basic.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїS":"achowSetValuePropertyBasedEventSW":1:"Foo":"Invisible"}

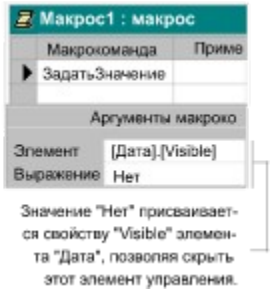
Задание значения свойства в ответ на событие с помощью макроса

1 Создайте макрос.

Инструкции

2 В пустой строке бланка выберите макрокоманду **ЗадатьЗначение (SetValue)** в списке макрокоманд.

3 Укажите в аргументе **Элемент** идентификатор свойства, значение которого требуется задать, а в аргументе **Выражение** значение, которое присваивается свойству.



4 Нажмите кнопку **Сохранить** для записи макроса.

5 Перейдите в форму или отчет, откройте окно свойств для объекта, в процедуре обработки у которого вызывается макрос, и введите имя макроса в ячейку свойства события, которое должно запускать макрос.

Для того чтобы увидеть примеры задания значения свойства в ответ на событие, нажмите кнопку .

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowSetValuePropertyBasedEventMacroSW":1:"Foo":"Invisible"}

Задание значения свойства в ответ на событие с помощью программы Visual Basic

1 Создайте процедуру обработки нужного события.

 Инструкции

2 Добавьте в процедуру инструкцию присваивания, состоящую из идентификатора свойства, значение которого требуется задать, знака равенства (=) и присваиваемого свойству значения.

- Для ссылки на свойство элемента управления в текущей форме введите ключевое слово **Me**, за которым следует оператор **!**, имя элемента управления, оператор **.** (точка) и имя свойства. Например, в следующей инструкции значение **False** присваивается свойству **Вывод на экран (Visible)** элемента управления «ДатаОплаты» в текущей форме:

  ```
  Me!ДатаОплаты.Visible = False
  ```

- Для ссылки на свойство элемента управления в другой форме введите полный идентификатор. Например, следующий идентификатор задает ссылку на свойство **Вывод на экран (Visible)** элемента управления «ДатаОплаты» в форме «ФормаДоставка»:

  ```
  Forms!ФормаДоставка!ДатаОплаты.Visible
  ```

Для получения дополнительных сведений о ссылках на свойства нажмите кнопку  .

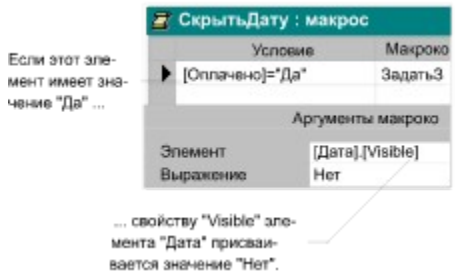
Для того чтобы увидеть примеры задания значения свойства в ответ на событие, нажмите кнопку  .

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowSetValuePropertyBasedEventVBSW":1:"Foo":"Invisible"}

Примеры задания значения свойства в ответ на событие

Для того чтобы задать значение свойства на основании значения в текущей записи, следует выполнить макрос или процедуру обработки события в ответ на событие формы **Текущая запись (Current).** Например, можно сделать элемент управления «ДатаОплаты» в форме «Заказы» скрытым, если заказ уже оплачен (если элемент управления «Оплачено» имеет значение «Да» или **True**).

Для запуска макроса укажите в значении свойства события формы **Текущая запись (OnCurrent)** имя макроса, который задает для свойства **Вывод на экран (Visible)** элемента управления «ДатаОплаты» значение «Нет», если элемент управления «Оплачено» имеет значение «Да».



Для выполнения тех же действий с помощью процедуры включите следующие инструкции Visual Basic в процедуру обработки события Form_Current:

```
If Me!Оплачено = True Then
     Me!ДатаОплаты.Visible = False
Else
     Me!ДатаОплаты.Visible = True
End If
```

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acconSetValuePropertyBasedEventExampleSW":1:"Foo":"Invisible"}

Синхронизация записей в двух формах

В некоторых случаях возникает необходимость одновременного просмотра связанных записей в двух формах. Например, при просмотре записей о клиенте в форме «Клиенты» полезно открыть форму «Заказы» и просмотреть список заказов, сделанных данным клиентом.

При переходах по записям в первой форме в таких случаях необходимо, чтобы во второй форме автоматически выводились связанные с ними записи. Например, при переходах по записям о поставщиках в форме «Поставщики» требуется вывести в форме «Список товаров» записи о товарах, поставляемых этими поставщиками.

Синхронизация записей, выводящихся в двух формах, осуществляется с помощью кнопки, которая открывает вторую форму и синхронизует ее с первой, или с помощью процедуры обработки события **Текущая запись (Current)**, в которой при переходах по записям первой формы выводятся связанные с ними записи во второй форме.

Совет. Для создания двух связанных форм применяется Мастер форм. На первом экране мастера выберите поля из двух различных таблиц. На втором экране установите переключатель **Связанные формы**. Мастер создаст две связанные формы, каждая из которых будет содержать поля из одной таблицы. В одной из форм будет создана кнопка, при нажатии которой вызывается другая форма и проводится синхронизация записей между этими формами.

Предполагаемые действия

- Создание кнопки для открытия и синхронизации формы.
- Вывод связанных записей в форме при переходах по записям в другой форме.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowSynchronizeRecordsBetweenTwoFormsMacrosSW":1:"Foo":"Invisible"}

Вывод связанных записей одной формы при переходах по записям в другой форме

При переходах по записям в форме вывод связанных записей во второй форме осуществляется с помощью макроса или процедуры обработки события в ответ на событие **Текущая запись (Current)** в первой форме

Предполагаемые действия

... Использование макроса для вывода связанных записей при переходах по записям в форме.
... Использование процедуры Visual Basic для вывода связанных записей при переходах по записям в форме.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowOnCurrentDisplayRelatedRecordsSW":1:"Foo":"Invisible"}

Использование макроса для вывода связанных записей при переходах по записям в форме

1. Откройте первую форму в режиме конструктора.
2. Создайте макрос.

... Инструкции

3. В пустой строке бланка выберите макрокоманду **ОткрытьФорму (OpenForm)** в списке макрокоманд.
4. Укажите в аргументе **Имя формы** имя второй формы, а в аргументе **Условие отбора** выражение, определяющее записи, выводящиеся во второй форме. Для просмотра примера нажмите кнопку ....
5. В режиме конструктора укажите имя макроса в свойстве **Текущая запись (OnCurrent)** первой формы.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowOnCurrentDisplayRelatedRecordsMacroSW":1:"Foo":"Invisible"}

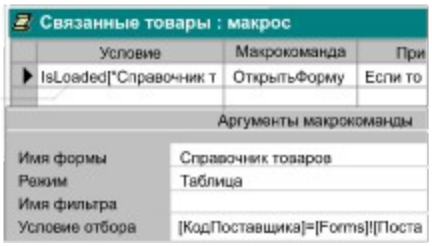
Пример использования макроса для вывода связанных записей при переходах по записям

При переходах по записям в форме «Поставщики» может возникнуть необходимость просмотреть в форме «Список товаров» записи о товарах, поставляемых каждым поставщиком. Такая задача решается с помощью макроса, в котором макрокоманда **ОткрытьФорму (OpenForm)** выполняется в ответ на событие **Текущая запись (Current)** в форме «Поставщики». Аргумент «Условие отбора», определяющий вывод в форме «Список товаров» записей, у которых значение поля «КодПоставщика» совпадает со значением поля «КодПоставщика» в форме «Поставщики», задается с помощью выражения:

КодПоставщика=Forms!Поставщики!КодПоставщика

Совет. Если не предполагается всегда держать форму «Список товаров» открытой при переходах по записям в форме «Поставщики», включите в макрос условное выражение, с помощью которого проверяется, открыта или нет форма «Список товаров».

Чтобы проверить, открыта ли форма "Справочник товаров", используется функция IsLoaded учебной базы данных "Борей".



После создания макроса укажите его имя в свойстве **Текущая запись (OnCurrent)** формы «Поставщики».

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowOnCurrentDisplayRelatedRecordsMacroExampleSW":1:"Foo":"Invisible"}

Использование процедуры Visual Basic для вывода связанных записей при переходах по записям в форме

1. Откройте первую форму в режиме конструктора.
2. Откройте процедуру обработки события Form_Current.


Инструкции

3. Включите в процедуру обработки события инструкции Visual Basic, указывающие вывод во второй форме записей, имеющих в связанном поле значения, совпадающие со значением соответствующего поля в первой форме.

Для того чтобы увидеть пример использования процедуры Visual Basic для вывода связанных записей при переходах по записям в форме, нажмите кнопку  .

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowOnCurrentDisplayRelatedRecordsVBSW":1:"Foo":"Invisible"}

Пример использования процедуры Visual Basic для вывода связанных записей при переходах по записям

При переходах по записям в форме «Поставщики» может возникнуть необходимость просмотреть в форме «Список товаров», записи о товарах, поставляемых каждым поставщиком. Для этого следует задать в процедуре обработки события `Form_Current` формы «Поставщики» значения свойств **Фильтр включен (FilterOn)** и **Фильтр (Filter)** формы «Список товаров»:

```
Private Sub Form_Current()

     ' Описывает переменную, в которой сохраняется предложение WHERE,
     ' определяющее отбираемые записи, и задает значение переменной.

     Dim strCond As String
     strCond = "КодПоставщика = Forms!Поставщики!КодПоставщика"

     ' Использует функцию IsLoaded из базы данных «Борей»
     ' для проверки, является ли форма «Список товаров»
     ' открытой, после чего определяет ее свойства.
     If IsLoaded("Список товаров") Then
          Forms![Список товаров].FilterOn = True
          Forms![Список товаров].Filter = strCond
     End If

End Sub
```

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowOnCurrentDisplayRelatedRecordsVBExampleSW":1:"Foo":"Invisible"}

Поиск записи по выбранному в списке значению

В форме имеется возможность создать список или поле со списком и организовать поиск записи по значению, выбранному в этом списке.

1 Откройте форму в режиме конструктора.

2 Нажмите кнопку **Мастера** на панели элементов, если эта кнопка еще не нажата.

3 На панели инструментов выберите **Список** или **Поле со списком** .

4 Установите указатель в том месте формы, куда требуется поместить список или поле со списком, и нажмите кнопку мыши.

5 В первом окне диалога мастера выберите параметр, определяющий поиск значений, выбираемых в списке или в поле со списком.

6 Выполняйте инструкции, выводящиеся в окна диалога мастера. В последнем окне диалога нажмите кнопку **Готово** для вывода списка или поля со списком в режиме конструктора.

Для просмотра процедуры обработки события, обеспечивающей работу списка или поля со списком, откройте окно свойств и нажмите кнопку построителя рядом с ячейкой свойства **После обновления (AfterUpdate)**.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowFindRecordBySelectingListSW":1:"Foo":"Invisible"}

Использование макросов и процедур обработки событий для проверки условий на значение

В большинстве случаев ограничения на данные, которые могут быть введены в элемент управления в форме, задаются с помощью свойства элемента управления **Условие на значение (ValidationRule)** или с помощью условий на значение полей или записей в базовой таблице, к которой присоединен элемент управления. Макросы и процедуры обработки события используют в тех случаях, когда требуется задать более сложные условия.

Ниже перечислены ситуации, в которых условия на значение задаются с помощью макросов или процедур обработки событий.

- В условия на значение включаются значения нескольких элементов управления в форме. Например, перед сохранением записи проверяется заполнение двух или трех полей.
- Для разных ошибок требуется вывести различные сообщения об ошибках. Например, одно сообщение выводится, когда введенное значение выходит за верхнюю границу допустимого диапазона, а другое, когда введенное значение меньше минимально допустимого.
- Пользователям предоставляется возможность нарушить условие на значение. В таких случаях обычно выводят предупреждение с приглашением подтвердить ввод данных.
- Выражения, определяющие условия на значения, содержат функции или ссылки на элементы управления в других формах.
- Условия на значение определяются для нескольких форм одновременно и требуется обеспечить возможность применения универсальных условий к разным формам.

Для получения дополнительных сведений о создании макросов нажмите кнопку ....

Для получения дополнительных сведений о разработке процедур обработки событий нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconWhyDoUseMacrosValidationSW":1:"Foo":"Invisible"}

Проверка условия на значение в макросе или в процедуре обработки события

Для проверки условия на значение, вводящегося в форму, следует создать макрос, который выполняется в ответ на событие в форме, или создать процедуру обработки события формы.

Предполагаемые действия

- Проверка условия на значение с помощью макроса.
- Проверка условия на значение с помощью процедуры обработки события.
- Дополнительные сведения о проверке условия на значение в макросе или в процедуре обработки события

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowValidateDataMacrosSW":1:"Foo":"Invisible"}

Проверка условия на значение с помощью макроса

1 Создайте макрос.

Инструкции

2 В окне макроса нажмите кнопку **Условия**  на панели инструментов.

3 Введите условие на значение в ячейку столбца **Условие** в пустой строке макрокоманды.

4 В ячейке столбца **Макрокоманда** выберите в списке макрокоманд макрокоманду, которая должна выполняться, если условие истинно.

5 Для всех других макрокоманд, которые должны выполняться, если условие истинно, введите многоточие (...) в ячейку столбца **Условие** и имя макрокоманды в ячейку столбца **Макрокоманда**. Например, обычно последней макрокомандой в наборе макрокоманд, выполняемых, если условие истинно, является макрокоманда **ОтменитьСобытие (CancelEvent)**.

6 Нажмите кнопку **Сохранить** для записи макроса.

7 Откройте форму в режиме конструктора.

8 Введите имя макроса в ячейку свойства события, которое должно запускать макрос. Например, для проверки условия при изменении данных в элементе управления следует указать имя макроса в качестве значения свойства **До обновления (BeforeUpdate)** элемента управления.

Совет. Для проверки условия на значение перед обновлением всей записи определите соответствующий макрос для свойства события **До обновления (BeforeUpdate)** формы, а не для события **До обновления** элемента управления.

Для того чтобы увидеть примеры проверки условий на значение с помощью макроса, нажмите кнопку .

Для получения дополнительных сведений о событиях, используемых при проверке условий на значения, нажмите кнопку .

Для получения дополнительных сведений о порядке проверки условий на значение элементов управления и полей нажмите кнопку .

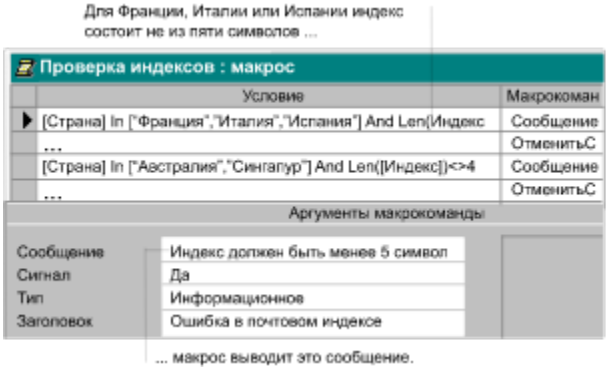
{ewc HLP95EN.DLL, DYNALINK, "nïSnïSnïSnïSnïS nïS Web nïSnïSnïS nïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïSnïS":"achowValidateDataMacrosMacroSW":1:"Foo":"Invisible"}

Примеры проверки условий на значение с помощью макроса

В некоторых ситуациях приходится использовать условия на значение, содержащие сложные выражения, а затем выполнять различные действия в зависимости от значений этих выражений. Например, в разных странах существуют различные стандарты для почтовых индексов, поэтому невозможно проверить правильность ввода в базу данных почтовых индексов с помощью простого условия.

В этом случае следует создать макрос, в котором определяется значение, введенное в поле «Страна», а затем проверяется соответствие значения, вводящегося в поле «Индекс», стандартам этой страны.

Вначале создайте макрос, выводящий различные сообщения в зависимости от значения поля «Страна» и длины значения поля «Индекс».



Затем откройте в режиме конструктора форму, содержащую элементы управления, и укажите имя макроса в свойстве события формы **До обновления (BeforeUpdate)**. Необходимо использовать свойство события **До обновления** формы, а не свойства событий элементов управления, поскольку в данном условии проверяются значения двух элементов управления.

Определенный таким образом макрос будет выполняться после завершения ввода новой записи, но до сохранения записи в таблице «Поставщики». При выполнении любого из двух условий ошибки будет выведено предупреждение, а запись в таблице «Поставщики» сохранена не будет.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconExampleMacroValidationSW":1:"Foo":"Invisible"}

Проверка условия на значение с помощью процедуры обработки события

1. Откройте форму в режиме конструктора.
2. Создайте процедуру обработки события, которое будет вызывать проверку вводящихся данных.

... Инструкции

   Например, для запуска процедуры проверки при обновлении данных в элементе управления создайте процедуру обработки события **До обновления (BeforeUpdate)** элемента управления.
3. Определите для процедуры обработки события программу Visual Basic, в которой задаются условия на значение и действия, выполняемые в зависимости от значений этих выражений.

Советы

- Обычно, полезным оказывается отмена события, которое запускает проверку условий, после завершения проверки. Для отмены большинства событий следует задать значение **True** для аргумента «Cancel» процедуры обработки события.
- Для проверки условия на значение перед обновлением всей записи определите соответствующую программу для процедуры обработки события **До обновления (BeforeUpdate)** формы, а не элемента управления.

Для того чтобы увидеть примеры проверки условий на значение с помощью процедуры обработки события нажмите кнопку ....

Для получения дополнительных сведений о событиях, используемых при проверке условий на значения, нажмите кнопку ....

Для получения дополнительных сведений о порядке проверки условий на значение элементов управления и полей нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowValidateDataMacrosVBSW":1:"Foo":"Invisible"}

Примеры проверки условий на значение с помощью процедуры обработки события

В некоторых ситуациях приходится использовать условия на значение, содержащие сложные выражения, а затем выполнять различные действия в зависимости от значений этих выражений. Например, в разных странах существуют различные стандарты для почтовых индексов, поэтому невозможно проверить правильность ввода в базу данных почтовых индексов с помощью простого условия.

В этом случае следует создать процедуру обработки события, в которой определяется значение, введенное в поле «Страна», а затем проверяется соответствие значения, вводящегося в поле «Индекс», стандартам этой страны.

Вначале откройте в режиме конструктора форму, содержащую элементы управления. Затем определите для процедуры обработки события **До обновления (BeforeUpdate)** следующую программу Visual Basic, выводящую различные сообщения в зависимости от значения поля «Страна» и длины значения поля «Индекс».

```
Private Sub Form_BeforeUpdate(Cancel As Integer)
     Select Case Me!Страна
          Case "Франция", "Италия", "Испания"
               If Len(Me!Индекс) <> 5 Then
                    MsgBox "Индекс должен содержать 5 символов."
                    Cancel = True
               End If
          Case "Австралия", "Сингапур"
               If Len(Me!Индекс) <> 4 Then
                    MsgBox "Индекс должен содержать 4 символа."
                    Cancel = True
               End If
     End Select
End Sub
```

Примечание. Программа Visual Basic определяется для процедуры обработки события **До обновления (BeforeUpdate)** формы, а не для событий элементов управления, поскольку в данном условии проверяются значения двух элементов управления.

Определенная таким образом программа будет выполняться после завершения ввода новой записи, но до сохранения записи в таблице «Поставщики». При выполнении любого из двух условий ошибки будет выведено предупреждение, а запись в таблице «Поставщики» сохранена не будет.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconExampleVBValidationSW":1:"Foo":"Invisible"}

События, используемые при проверке условий на значения.

Обычно, для запуска макросов или процедур обработки событий, проверяющих условия на значения, используют следующие события формы или элемента управления.

Событие формы	Проверка
До обновления (BeforeUpdate)	Перед сохранением новых или измененных данных в записи.
Удаление (OnDelete)	Перед удалением записи.

Событие элемента управления	Проверка
До обновления (BeforeUpdate)	Перед сохранением новых или измененных данных в элементе управления
Выход (OnExit)	Перед выходом из элемента управления.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconEventsMacroValidationSW":1:"Foo":"Invisible"}

Порядок проверки условий на значение элементов управления и полей

Условия на значения элементов управления и полей проверяются в следующем порядке:

1. Макрос или процедура обработки события (например, макрос или процедура обработки события, запускаемая в ответ на событие **До обновления (BeforeUpdate)** элемента управления или формы, содержащей элемент управления).
2. Свойство **Условие на значение (ValidationRule)** элемента управления.
3. Условия на значение поля, определенные в базовой таблице. Эти условия задаются в свойствах поля **Условие на значение (ValidationRule)**, **Обязательное поле (Required)** и **Пустые строки (AllowZeroLength)**.
4. Свойство **Условие на значение (ValidationRule)** базовой таблицы (данное свойство определяет условия для записей).

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconOrderValidationSW":1:"Foo":"Invisible"}

Печать отчета из формы

Для автоматизации процесса печати отчета следует создать макрос, в котором выполняются макрокоманды **ОткрытьОтчет (OpenReport)** и **Печать (PrintOut)**, или процедуру обработки события. Например, можно создать кнопку, печатающую отчет или определить для этой задачи специальную команду меню или сочетание клавиш.

Макрокоманду **ОткрытьОтчет (OpenReport)** необходимо использовать, если требуется произвести отбор выводящихся на печать записей или открыть отчет в режиме предварительного просмотра. При печати отчета после выполнения макрокоманды **ОткрытьОтчет** отчет печатается с использованием стандартных настроек, заданных в диалоговом окне **Печать**.

Макрокоманду **Печать (PrintOut)** используют для указания настроек печати перед выводом отчета на печать. Аргументы макрокоманды **Печать** соответствуют всем параметрам в диалоговом окне **Печать**.

Предполагаемые действия

- Создание кнопки для печати отчета.
- Добавление в макрос макрокоманды **ОткрытьОтчет (OpenReport)** или **Печать (PrintOut).**
- Выполнение макрокоманды **ОткрытьОтчет (OpenReport)** или **Печать (PrintOut)** в процедуре Visual Basic.
- Знакомство с макрокомандой **ОткрытьОтчет (OpenReport).**
- Знакомство с макрокомандой **Печать (PrintOut).**

{ewc HLP95EN.DLL, DYNALINK, "nїSnїSnїSnїSnїS nїS Web nїSnїSnїS nїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS":"acconOverviewPrintingReportMacroSW":1:"Foo":"Invisible"}

Принудительный перевод страницы в отчете при выполнении условия

Принудительный перевод страницы в отчете при выполнении условия осуществляется путем определения свойства **Вывод на экран (Visible)** элемента управления-конец страницы в макросе или в процедуре обработки события.

Предполагаемые действия

... Использование макроса для принудительного перевода страницы в отчете при выполнении условия.
... Использование процедуры Visual Basic для принудительного перевода страницы в отчете при выполнении условия.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowForcePageBreakIfConditionIsMetSW":1:"Foo":"Invisible"}

Использование макроса для принудительного перевода страницы в отчете при выполнении условия

1 Откройте отчет в режиме конструктора.

2 Нажмите кнопку **Конец страницы** ... на панели элементов, а затем установите указатель в том месте раздела отчета, в котором требуется перевести страницу, и нажмите кнопку мыши.

3 Нажмите кнопку **Свойства** ... на панели инструментов и задайте имя для элемента управления конец страницы в поле **Имя**.

4 Создайте макрос.

... Инструкции

5 В пустой строке бланка выберите макрокоманду **ЗадатьЗначение (SetValue)** в списке макрокоманд.

6 Укажите в аргументе **Элемент** идентификатор свойства **Вывод на экран (Visible)** элемента управления-конец страницы. Например, если данный элемент управления имеет имя «ПереводСтраницы», введите в ячейку аргумента **Элемент** выражение **[ПереводСтраницы].[Visible]**.

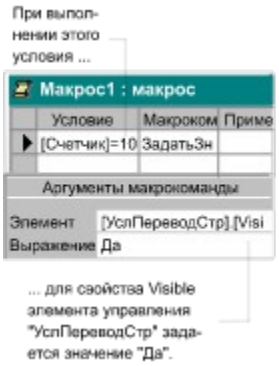
7 Введите в ячейку аргумента **Выражение** значение «Нет».

8 Нажмите кнопку **Сохранить** ... для записи макроса.

9 В режиме конструктора отчета укажите имя макроса в свойстве события **Форматирование (OnFormat)** верхнего колонтитула отчета.

В результате, элемент управления-конец страницы будет сделан скрытым при форматировании каждой страницы отчета, т.е. принудительного перевода страницы не будет.

10 Создайте второй макрос, который задает для свойства конца страницы **Вывод на экран (Visible)** значение «Да», когда истинным оказывается некоторое условное выражение.



11 Нажмите кнопку **Сохранить** ... для записи макроса.

12 Выберите раздел, в котором требуется выполнить перевод страницы, и укажите имя второго макроса в свойстве события **Форматирование (OnFormat)** этого раздела. Используя предыдущий пример по созданию конца страницы при превышении значения элемента управления «Счетчик» 10, укажите имя второго макроса в свойстве области данных **Форматирование (OnFormat)**.

Перевод страницы будет осуществляться при выполнении указанного условия. После перевода страницы макрос, связанный с верхним колонтитулом, снова сделает элемент управления-конец страницы скрытым до нового выполнения указанного условия.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowForcePageBreakIfConditionIsMetMacroSW":1:"Foo":"Invisible"}

Использование процедуры Visual Basic для принудительного перевода страницы в отчете при выполнении условия

1. Откройте отчет в режиме конструктора.
2. Нажмите кнопку **Конец страницы** ... на панели элементов, а затем установите указатель в том месте раздела отчета, в котором требуется перевести страницу, и нажмите кнопку мыши.
3. Откройте процедуру обработки события ВерхнийКолонтитул_Format.
   ... Инструкции
4. Включите в процедуру обработки события инструкцию присваивания, которая задает для свойства **Вывод на экран (Visible)** элемента управления конец страницы значение «Нет». Например, если элемент управления имеет имя «ПереводСтраницы», добавьте следующую инструкцию:

   ```
   Me![ПереводСтраницы].Visible = False
   ```

   Эта инструкция сделает элемент управления скрытым при форматировании каждой страницы, т.е. перевод страницы осуществляться не будет.
5. Включите в процедуру обработки события **Форматирование (Format)** раздела, в котором требуется выполнить перевод страницы, программу Visual Basic, которая задает для свойства **Вывод на экран (Visible)** элемента управления-конец страницы значение «Да» при выполнении определенного условия. Например, для того чтобы перевести страницу в области данных, когда элемент управления «Счетчик» имеет значение 10 (первые десять записей будут печататься на первой странице), добавьте в процедуру обработки события `ОбластьДанных_Format` следующую программу:

   ```
   If Me![Счетчик] = 10 Then
        Me![ПереводСтраницы].Visible = True
   End If
   ```

Перевод страницы будет осуществляться при выполнении указанного условия. После перевода страницы процедура обработки события, связанная с верхним колонтитулом снова сделает элемент управления-конец страницы скрытым до нового выполнения указанного условия.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowForcePageBreakIfConditionIsMetVBSW":1:"Foo":"Invisible"}

Вывод на экран или скрытие предупреждений

Пользователь имеет возможность отключить на время выполнения макроса или процедуры Visual Basic вывод модальных окон сообщений с предупреждениями Microsoft Access или снова включить их с помощью макрокоманды **УстановитьСообщения (SetWarnings)**, вызываемой в макросе или в процедуре.

Предполагаемые действия


Вывод на экран или скрытие предупреждений с помощью макроса.


Вывод на экран или скрытие предупреждений с помощью программы Visual Basic.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowHideWarningMessagesWhenRunMacroSW":1:"Foo":"Invisible"}

Вывод на экран или скрытие предупреждений с помощью макроса

1 Откройте существующий макрос в окне макроса.

Инструкции

2 В той строке макроса, начиная с которой требуется прекратить или возобновить вывод модальных окон предупреждающих сообщений, поместите макрокоманду **УстановитьСообщения (SetWarnings).**

Инструкции

3 Для временного прекращения вывода предупреждений задайте для аргумента **Включить сообщения** значение «Нет». Для возобновления вывода сообщений задайте для аргумента **Включить сообщения** значение «Да».

4 Нажмите кнопку **Сохранить** для записи макроса.

5 Выполните макрос.

Инструкции

После прекращения выполнения макроса вывод предупреждений будет восстановлен.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowHideWarningMessagesWhenRunMacroMacroSW":1:"Foo":"Invisible"}

Вывод на экран или скрытие предупреждений с помощью программы Visual Basic

1. Откройте процедуру **Function** или **Sub** в окне модуля.
2. В той строке процедуры, начиная с которой требуется прекратить или возобновить вывод модальных окон предупреждающих сообщений, используйте метод **SetWarnings** для вызова макрокоманды **УстановитьСообщения (SetWarnings).**

... Инструкции

3. Для временного прекращения вывода предупреждений задайте для аргумента «Включить сообщения» значение **False**. Для возобновления вывода сообщений задайте для аргумента «Включить сообщения» значение **True**.
4. Выполните процедуру.

... Инструкции

После прекращения выполнения процедуры вывод предупреждений будет восстановлен.

Для получения дополнительных сведений по методу **SetWarnings** нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowHideWarningMessagesWhenRunMacroVBSW":1:"Foo":"Invisible"}

Перемещение по элементам управления, записям и страницам формы

Для автоматического перехода на элемент управления, запись или на страницу форму в ответ на событие в форме следует создать макрос или процедуру обработки события, в которых выполняется макрокоманда **КЭлементуУправления (GoToControl)**, **НаСтраницу (GoToPage)** или **НаЗапись (GoToRecord)**.

Предполагаемые действия

Перемещение по элементам управления, записям и страницам формы с помощью макроса.
Перемещение по элементам управления, записям и страницам формы с помощью программы Visual Basic.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowNavigateBetweenControlsPagesRecordsMacrosSW":1:"Foo":"Invisible"}

Перемещение по элементам управления, записям и страницам формы с помощью макроса

1 Создайте макрос.

... Инструкции

2 Выполните одно из следующих действий:

- для перемещения фокуса на элемент управления или поле в текущей записи выберите макрокоманду **КЭлементуУправления (GoToControl)** в списке макрокоманд в пустой строке макрокоманды и укажите в аргументе **Имя элемента** имя нужного элемента управления;
- чтобы сделать конкретную запись текущей выберите макрокоманду **НаЗапись (GoToRecord)** в списке макрокоманд в пустой строке макрокоманды и укажите аргументы макрокоманды;
- для перевода фокуса в активной форме на первый элемент управления на конкретной странице формы выберите макрокоманду **НаСтраницу (GoToPage)** в списке макрокоманд в пустой строке макрокоманды и укажите аргументы макрокоманды.

3 Нажмите кнопку **Сохранить** ... для записи макроса.

4 Выполните макрос.

... Инструкции

Для получения дополнительных сведений о макрокоманде **КЭлементуУправления (GoToControl)** нажмите кнопку ....

Для получения дополнительных сведений о макрокоманде **НаЗапись (GoToRecord)** нажмите кнопку ....

Для получения дополнительных сведений о макрокоманде **НаСтраницу (GoToPage)** нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowNavigateBetweenControlsPagesRecordsMacrosMacroSW":1:"Foo" :"Invisible"}

Перемещение по элементам управления, записям и страницам формы с помощью программы Visual Basic

1 Откройте в окне модуля процедуру обработки события.

... Инструкции

Например, для выполнения программы в ответ на нажатие кнопки откройте процедуру обработки события кнопки **Нажатие кнопки (Click)**.

2 Выполните одно из следующих действий:

- для перемещения фокуса на элемент управления или поле в текущей записи выполните в процедуре метод **GoToControl**, указав в аргументе *ИмяЭлемента* имя нужного элемента управления;

  ... Инструкции

- чтобы сделать конкретную запись текущей или для создания новой записи используйте метод **GoToRecord** для выполнннения макрокоманды **НаЗапись (GoToRecord)** в процедуре;
- для перевода фокуса в активной форме на первый элемент управления на конкретной странице формы используйте метод **GoToPage** для выполнннения макрокоманды **НаСтраницу (GoToPage)** в процедуре.

3 Выполните процедуру.

... Инструкции

Совет. В процедурах Visual Basic для приложений для поиска записей и переходов по записям используются также объекты **Recordset**, группы методов **Move** и **Find** и метод **Seek**.

Для получения дополнительных сведений о макрокоманде **КЭлементуУправления (GoToControl)** нажмите кнопку ....

Для получения дополнительных сведений о макрокоманде **НаЗапись (GoToRecord)** нажмите кнопку ....

Для получения дополнительных сведений о макрокоманде **НаСтраницу (GoToPage)** нажмите кнопку ....

Для получения дополнительных сведений о объектах **Recordset** и определенных для них методах нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowNavigateBetweenControlsPagesRecordsMacrosVBSW":1:"Foo":"Invisible"}

Автоматизация импорта, экспорта и связывания данных.

В зависимости от типа данных, с которыми проводится работа, для автоматизации импорта, экспорта или связывания данных необходимо создать макрос или процедуру Visual Basic, в которых выполняются макрокоманды **ПреобразоватьБазуДанных (TransferDatabase)**, **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** или **ПреобразоватьТекст (TransferText)**.

Предполагаемые действия


Автоматизация импорта, экспорта и связывания данных с помощью макроса.


Автоматизация импорта, экспорта и связывания данных с помощью программы Visual Basic

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowAutomateImportingExportingDataMacroSW"":1:"Foo":"Invisible"}

Автоматизация импорта, экспорта и связывания данных с помощью макроса.

1 Создайте макрос.

Инструкции

2 Выполните одно из следующих действий:

- для выполнения операции импорта, экспорта или связывания данных между текущей базой данных Microsoft Access и другой базой данных выберите макрокоманду **ПреобразоватьБазуДанных (TransferDatabase)** в списке макрокоманд в пустой строке макрокоманды;
- для выполнения операции импорта, экспорта или связывания данных между текущей базой данных Microsoft Access и файлом электронной таблицы выберите макрокоманду **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** в списке макрокоманд в пустой строке макрокоманды;
- для выполнения операции импорта, экспорта или связывания данных между текущей базой данных Microsoft Access и текстовым файлом выберите макрокоманду **ПреобразоватьТекст (TransferText)** в списке макрокоманд в пустой строке макрокоманды;

3 Задайте значения аргументов макрокоманды.

4 Нажмите кнопку **Сохранить** для записи макроса.

5 Выполните макрос.

Инструкции

Для получения дополнительных сведений о макрокоманде **ПреобразоватьБазуДанных (TransferDatabase)** нажмите кнопку .

Для получения дополнительных сведений о макрокоманде **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** нажмите кнопку .

Для получения дополнительных сведений о макрокоманде **ПреобразоватьТекст (TransferText)** нажмите кнопку .

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowAutomateImportingExportingDataMacroMacroSW":1:"Foo":"Invisible"}

Автоматизация импорта, экспорта и связывания данных с помощью программы Visual Basic.

1 Создайте процедуру Visual Basic.

Инструкции

2 Выполните одно из следующих действий:

- Для выполнения операции импорта, экспорта или связывания данных между текущей базой данных Microsoft Access и другой базой данных используйте метод **TransferDatabase**, который выполнит макрокоманду **ПреобразоватьБазуДанных (TransferDatabase)** в процедуре.

  Инструкции

- Для выполнения операции импорта, экспорта или связывания данных между текущей базой данных Microsoft Access и файлом электронной таблицы используйте метод **TransferSpreadsheet**, который выполнит макрокоманду **ПреобразоватьЭлектроннуюТаблицу (TransferSpreadsheet)** в процедуре.
- Для выполнения операции импорта, экспорта или связывания данных между текущей базой данных Microsoft Access и текстовым файлом используйте метод **TransferText**, который выполнит макрокоманду **ПреобразоватьТекст (TransferText)** в процедуре.

3 Выполните процедуру.

Инструкции

Для получения дополнительных сведений о методе **TransferDatabase** нажмите кнопку .

Для получения дополнительных сведений о методе **TransferSpreadsheet** нажмите кнопку .

Для получения дополнительных сведений о методе **TransferText** нажмите кнопку .

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowAutomateImportingExportingDataMacroVBSW":1:"Foo":"Invisible"}

Автоматический вывод в форме обновленных записей

Для автоматического вывода в форме последних записей следует выполнить операции обновления экрана или обновления данных.

При работе с общей базой данных в сети удобно создать кнопку, с помощью которой на экран выводится последняя версия существующих данных.

Для обновления набора записей, выводящихся в форме, следует выполнить макрокоманду **Обновление (Requery)** в макросе или вызвать метод **Requery** в процедуре Visual Basic.

Предполагаемые действия

- Создание кнопки для обновления данных в форме.
- Автоматический вывод в форме обновленных записей с помощью макроса.
- Автоматический вывод в форме обновленных записей с помощью программы Visual Basic.

{ewc HLP95EN.DLL, DYNALINK, "nïSnïSnïSnïSnïS nïS Web nïSnïSnïS nïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïSnïS":"achowAutomaticallyViewLatestRecordsFormMacroSW":1:"Foo":"Invisible" }

Автоматический вывод в форме обновленных записей с помощью макроса

1 Создайте макрос.

Инструкции

2 В пустой строке бланка выберите макрокоманду **Обновление (Requery)** в списке макрокоманд.

Для обновления записей, выводящихся в элементе управления, присоединенном к таблице или запросу, укажите в аргументе **Имя элемента** имя элемента управления. Для обновления всей формы оставьте аргумент **Имя элемента** пустым.

3 Нажмите кнопку **Сохранить** для записи макроса.

4 Выполните макрос.

Инструкции

Для получения дополнительных сведений о макрокоманде **Обновление (Requery)** нажмите кнопку .

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"achowAutomaticallyViewLatestRecordsFormMacroMacroSW":1:"Foo":"Invisible"}

Автоматический вывод в форме обновленных записей с помощью программы Visual Basic

1 Создайте процедуру Visual Basic.

Инструкции

2 Добавьте метод **Requery** в процедуру, определенную для формы или элемента управления, записи в которых требуется обновить. Например, для вывода в поле со списком «КодТипа» последних данных включите в процедуру следующую конструкцию вызова метода:

```
КодТипа.Requery
```

3 Выполните процедуру.

Инструкции

Для получения дополнительных сведений о методе **Requery** нажмите кнопку ....

Для того чтобы увидеть пример обновления списка, обеспечивающего вывод в нем последних данных, нажмите кнопку ...

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowAutomaticallyViewLatestRecordsFormMacroVBSW":1:"Foo":"Invisible"}

Вывод окна сообщения

При необходимости вывести короткое сообщение, например, предупреждение для пользователей, следует использовать встроенное окно диалога, которое называют окном сообщений. Окно сообщения выводится на экран с помощью макрокоманды **Сообщение (MsgBox)** в макросе или функции **MsgBox** в процедуре Visual Basic.

Предполагаемые действия

- Вывод окна сообщения с помощью макроса.
- Вывод окна сообщения с помощью программы Visual Basic.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowDisplayMessageBoxW":1:"Foo":"Invisible"}

Вывод окна сообщения с помощью макроса

1 Создайте макрос.

Инструкции

2 В пустой строке бланка выберите макрокоманду **Сообщение (MsgBox)** в списке макрокоманд и укажите аргументы макрокоманды.

3 Нажмите кнопку **Сохранить** для записи макроса.

4 Выполните макрос.

Инструкции

Для получения дополнительных сведений о макрокоманде **Сообщение (MsgBox)** нажмите кнопку .

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowDisplayMessageBoxMacroW":1:"Foo":"Invisible"}

Вывод окна сообщения с помощью программы Visual Basic

1 Создайте процедуру Visual Basic.

Инструкции

2 Добавьте в процедуру инструкцию, присваивающую переменной значение, возвращаемое функцией **MsgBox**. Значение, возвращаемое функцией, определяется кнопкой, которую пользователь нажимает в окне сообщения. В дальнейшем в процедуре можно определить различные действия, выполняемые в зависимости от возвращаемого значения.

Например, следующая инструкция вызывает функцию **MsgBox**, выводящую сообщение, и присваивает возвращаемое значение переменной `RetValue`:

```
RetValue = MsgBox("Продолжить?", vbOKCancel)
```

3 Выполните процедуру.

Инструкции

Для получения дополнительных сведений о функции **MsgBox** нажмите кнопку .

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowDisplayMessageBoxVBW":1:"Foo":"Invisible"}

Сохранение объекта базы данных с помощью макроса или процедуры обработки события

Для автоматического сохранения объекта базы данных следует создать макрос или процедуру обработки события, в которых выполняется макрокоманда **Сохранить (Save)**.

Предполагаемые действия

- Сохранение объекта базы данных с помощью макроса.
- Сохранение объекта базы данных с помощью процедуры обработки события.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowSaveDatabaseObjectMacroSW":1:"Foo":"Invisible"}

Сохранение объекта базы данных с помощью макроса

1 Создайте макрос.

Инструкции

2 В пустой строке бланка выберите макрокоманду **Сохранить (Save)** в списке макрокоманд и укажите в аргументах **Тип объекта** и **Имя объекта** тип и имя сохраняемого объекта.

3 Нажмите кнопку **Сохранить** для записи макроса.

4 Выполните макрос.

Инструкции

Например, для создания кнопки, сохраняющей форму, укажите имя макроса в свойстве события кнопки **Нажатие кнопки (OnClick)**.

Для получения дополнительных сведений о макрокоманде **Сохранить (Save)** нажмите кнопку .

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowSaveDatabaseObjectMacroMacroSW":1:"Foo":"Invisible"}

Сохранение объекта базы данных с помощью процедуры обработки события

1. Откройте в окне модуля процедуру обработки события.

Инструкции

Например, для создания кнопки, сохраняющей форму, откройте процедуру обработки события кнопки **Нажатие кнопки (Click)**.

2. Используйте метод **Save**, который выполнит макрокоманду **Сохранить (Save)** в процедуре, указав в аргументах ТипОбъекта и ИмяОбъекта тип и имя сохраняемого объекта.

Инструкции

3. Выполните процедуру.

Инструкции

Для получения дополнительных сведений о макрокоманде **Сохранить (Save)** нажмите кнопку .

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowSaveDatabaseObjectMacroVBSW":1:"Foo":"Invisible"}

Выполнение макрокоманд при открытии базы данных

Специальный макрос AutoExec позволяет автоматически выполнить макрокоманду или набор макрокоманд при открытии базы данных. В процессе открытия базы данных Microsoft Access проводит поиск макроса с этим именем и, если такой макрос существует, автоматически запускает его.

1 Создайте макрос, содержащий макрокоманды, которые требуется выполнить при открытии базы данных.

 Инструкции

2 Сохраните макрос под именем AutoExec.

При следующем открытии базы данных Microsoft Access автоматически запустит этот макрос.

Примечания

- Если требуется открыть базу данных, не выполняя при этом макрос AutoExec, открывайте базу данных при нажатой клавише SHIFT.
- Другие элементы окна диалога **Параметры запуска** (меню **Сервис**) позволяют указать другие параметры, применяемые при запуске базы данных или приложения. Для получения дополнительных сведений нажмите кнопку  .

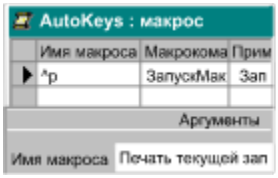
{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowRunMacroWhenMyDatabaseFirstOpensSW":1:"Foo":"Invisible"}

Назначение макрокоманды или набора макрокоманд на клавишу

Допускается связывание макрокоманды или набора макрокоманд с конкретной клавишей или сочетанием клавиш с помощью специальной группы макросов AutoKeys. После этого при нажатии клавиши или сочетания клавиш Microsoft Access будет выполнять данную макрокоманду.

1. В окне базы данных выберите вкладку **Макросы**.
2. Нажмите кнопку **Создать**.
3. Нажмите кнопку **Имена макросов**  на панели инструментов.
4. Укажите в ячейке столбца **Имя макроса** клавишу или сочетание клавиш, с которыми связывается макрокоманда или набор макрокоманд.

   Для получения дополнительных сведений о сочетаниях клавиш, используемых для назначения макросов на клавиатуру, нажмите кнопку ....
5. Введите в макрос макрокоманду или набор макрокоманд, которые должны выполняться при нажатии указанной клавиши или сочетания клавиш. Например, можно выбрать макрокоманду **ЗапускМакроса (RunMacro)**, которая будет запускать макрос «Печать текущей записи» при нажатии клавиш CTRL+П.



6. Повторите шаги 4 и 5 для каждого нового определения сочетания клавиш.
7. Сохраните группу макросов под именем AutoKeys.

Новые сочетания клавиш вступят в действие сразу после сохранения макроса и будут использоваться при следующем открытии базы данных.

Примечание. Если макрокоманда или набор макрокоманд связывается с сочетанием клавиш, которое уже используется в Microsoft Access (например, CTRL+C - сочетание клавиш для команды **Копировать**), то новое назначение макрокоманд на это сочетание клавиш заменит стандартное назначение команд Microsoft Access.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowAssignMacroKeySW":1:"Foo":"Invisible"}

Сочетания клавиш в макросе AutoKeys

В следующей таблице представлены сочетания клавиш, используемые для назначения клавиш в группе макросов AutoKeys. Допустимые сочетания клавиш являются подмножеством синтаксиса инструкции Visual Basic **SendKeys**. Для получения дополнительных сведений о синтаксисе инструкции **SendKeys** нажмите кнопку ....

Инструкция SendKeys	Сочетание клавиш
^A или ^4	CTRL+Любая буква или цифра
{F1}	Любая функциональная клавиша
^{F1}	CTRL+Любая функциональная клавиша
+{F1}	SHIFT+Любая функциональная клавиша
{INSERT}	INS
^{INSERT}	CTRL+INS
+{INSERT}	SHIFT+INS
{DELETE} or {DEL}	DEL
^{DELETE} or ^{DEL}	CTRL+DEL
+{DELETE} or +{DEL}	SHIFT+DEL

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acrefAutokeysKeyCombinationsSW":1:"Foo":"Invisible"}

Что такое процедура?

В языке Visual Basic для приложений (VBA) замкнутыми программными единицами являются процедуры. Процедура содержит набор инструкций и методов, с помощью которых выполняются действия или рассчитывается значение. Например, в следующей процедуре обработки события метод **OpenForm** открывает форму «Заказы»:

```
Private Sub ОткрытиеФормыЗаказы_Click()

     DoCmd.OpenForm "Заказы"

End Sub
```

Существуют процедуры двух типов:

- Процедура-подпрограмма **Sub**, аналогично инструкциям Visual Basic, выполняет действие или набор действий, но не возвращает значение. Пользователь имеет возможность создавать процедуры **Sub** самостоятельно или использовать процедуры обработки событий, определенные в Microsoft Access.

  Каждая форма или отчет в базе данных имеют встроенный модуль формы или модуль отчета содержащий встроенные процедуры обработки событий, которые выполняются в ответ на события, возникающие в форме или отчете и в элементах управления в форме или отчете. После того как Microsoft Access определит, что событие возникло в форме, отчете или в элементе управления, автоматически запускается процедура обработки события, имя которой образуется как комбинация имен объекта и события. Например, процедура обработки события позволяет открыть другую форму при нажатии кнопки в форме.

  Совет. При создании объекта с помощью мастера часто автоматически создаются процедуры обработки событий для данного объекта. Для того чтобы понять, как работают процедуры, полезно ознакомиться с такими процедурами.
- Процедура-функция **Function** (часто такие процедуры называют просто функциями) возвращает значение, например, полученное в результате расчетов. Visual Basic включает ряд встроенных функций; например, функция **Now** возвращает текущее значение даты и времени. В дополнение к встроенным функциям, пользователь имеет возможность самостоятельно создавать функции, которые называют специальными функциями.

  Функции, возвращающие значения, могут использоваться в выражениях. Выражения, содержащие функции, широко используются в Microsoft Access, например, в аргументах инструкций или в методов Visual Basic, при указании значений свойств и при определении условий в запросах или фильтрах.

Ниже приводится пример процедуры **Function** с именем «НачалоСледующегоМесяца», которая возвращает дату первого дня следующего месяца:

```
Function НачалоСледующегоМесяца()

     НачалоСледующегоМесяца = _
          DateSerial(Year(Now), Month(Now) + 1, 1)

End Function
```

Данная функция состоит из единственной инструкции присвоения, в которой результат выражения (стоящего справа от знака равенства [=]) присваивается функции «НачалоСледующегоМесяца» (имя которой стоит слева от знака равенства). Результат рассчитывается с помощью встроенных функций Visual Basic **DateSerial**, **Year**, **Now** и **Month**.

После создания функции ее можно использовать в Microsoft Access в любых выражениях. Например, чтобы сделать дату первого дня следующего месяца значением, которое поле получает по умолчанию, следует в окне свойств определить свойство этого поля **Значение по умолчанию (DefaultValue)** с помощью следующего выражения:

```
=НачалоСледующегоМесяца()
```

Примечание. Для того чтобы использовать функцию в качестве значения свойства, ее необходимо поместить в модуль формы или отчета или в стандартный модуль. Нельзя использовать функцию в модуле класса, не закрепленном за формой или отчетом в виде значения свойства формы или отчета.

Процедуры **Sub** и **Function** могут принимать аргументы. Для получения дополнительных сведений об

использовании аргументов в Visual Basic нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS
пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acconWhatIsProcedureW":1:"Foo":"Invisible"}

Создание специальных процедур Visual Basic для приложений

Имеется возможность расширить базу данных путем включения процедур языка Visual Basic, настраивая при этом совместную работу таблиц, форм, отчетов и запросов. Существует несколько типов процедур. Пользователь имеет возможность создавать собственные процедуры обработки событий, добавляя программы в стандартные шаблоны процедур в модулях форм и модулях отчета. Кроме того, пользователь может создавать собственные процедуры **Function** или процедуры **Sub** в стандартном модуле, в модуле класса (который включает модуль формы и отчета).

Процедуры, содержащиеся в стандартном модуле, могут быть вызваны из выражений, из процедур обработки событий, из процедур в других стандартных модулях или из макросов. Процедуры обработки событий автоматически вызываются в ответ на события, такие как нажатия кнопок мыши, возникающие в форме, отчете или элементах управления.

Предполагаемые действия

- Создание специальной функции
- Создание специальной процедуры **Sub**
- Создание процедуры обработки события

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowCreateVBProcedureSW":1:"Foo":"Invisible"}

Создание специальной функции

1. Откройте новый модуль.
   - Чтобы открыть новый стандартный модуль, выберите вкладку **Модули** в окне базы данных и нажмите кнопку **Создать**. Для того чтобы открыть существующий стандартный модуль, выберите вкладку **Модули**, далее модуль, который следует открыть, и нажмите кнопку **Конструктор**.
   - Чтобы открыть модуль формы или модуль отчета, откройте форму или отчет в режиме конструктора и нажмите кнопку  на панели инструментов.
   - Чтобы открыть модуль класса, который не связан с формой или отчетом в окне базы данных, выберите **Модуль** в меню **Вставка**. Для того чтобы открыть существующий модуль класса, выберите вкладку **Модули**, далее модуль, который следует открыть, и нажмите кнопку **Конструктор**.
2. Объявите функцию, введя инструкцию **Function**.
3. Укажите все аргументы функции в круглых скобках вслед за именем функции. Например, следующая инструкция описания функции **IsLoaded** указывает строковый аргумент *strFormName* :

   ```
   Function IsLoaded(strFormName As String) As Boolean
   ```

3. Добавьте текст программы на языке Visual Basic для приложений (VBA), выполняющий операции, которые должна выполнять функция.

Для получения дополнительных сведений об инструкции **Function** нажмите кнопку  .

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowWriteUDFSW":1:"Foo":"Invisible"}

Создание специальной процедуры Sub

1. Откройте новый модуль.
   - Чтобы открыть новый стандартный модуль, выберите вкладку **Модули** в окне базы данных и нажмите кнопку **Создать**. Для того чтобы открыть существующий стандартный модуль, выберите вкладку **Модули**, далее модуль, который следует открыть, и нажмите кнопку **Конструктор**.
   - Чтобы открыть модуль формы или модуль отчета, откройте форму или отчет в режиме конструктора и нажмите кнопку  на панели инструментов.
   - Чтобы открыть модуль класса, который не связан с формой или отчетом, в окне базы данных выберите **Модуль** в меню **Вставка**. Для того чтобы открыть существующий модуль класса, выберите вкладку **Модули**, далее модуль, который следует открыть, и нажмите кнопку **Конструктор**.
2. Объявите процедуру, введя инструкцию **Sub**.
3. Введите имя процедуры и все аргументы в круглых скобках. Например, следующая инструкция описания процедуры **Sub** с именем ShowEvent указывает строковый аргумент EventName:

   ```
   Sub ShowEvent(EventName As String)
   ```
4. Добавьте текст программы на языке Visual Basic для приложений (VBA), выполняющей операции, которые должна выполнять процедура.

Для получения дополнительных сведений об инструкции **Sub** нажмите кнопку . Для получения дополнительных сведений о написании процедуры **Sub** нажмите кнопку

 .

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowWriteSubProcedureSW":1:"Foo":"Invisible"}

Создание процедуры обработки события

Microsoft Access позволяет легко создать процедуру обработки событий. При указании для свойства события формы, отчета или элемента управления значения «[Процедура обработки события]» Microsoft Access создает заготовку процедуры обработки события. Пользователю остается только задать действия, выполняемые в ответ на определенные события в форме, отчете или элементе управления. Например, имеется возможность запустить программу при нажатии пользователем кнопки или при изменении данных.

1. Откройте форму или отчет в режиме конструктора формы или режиме конструктора отчета.
2. Выведите окно свойств для формы, отчета или для раздела или элемента управления формы или отчета.
3. Выберите вкладку **События**
4. Выберите свойство события, в ответ на которое должна выполняться процедура. Например, для того, чтобы открыть процедуру обработки события **Изменение (Change)**, выберите свойство **Изменение (OnChange)**.
5. Нажмите кнопку построителя  рядом с ячейкой свойства, чтобы открыть окно диалога **Построители**.
6. Выберите двойным нажатием кнопки мыши элемент «Программы» для получения в модуле формы или модуле отчета инструкций **Sub** и **End Sub**. Эти инструкции описывают, т.е. определяют процедуру обработки события.

   Microsoft Access автоматически описывает в модуле формы или отчета процедуры обработки события для каждого объекта и события с использованием ключевого слова **Private**, указывающим, что доступ к этой процедуре возможен только из других процедур в том же модуле.
7. Добавьте в процедуру программу, которая должна выполняться в ответ на событие. Например, для подачи звукового сигнала через динамик компьютера при изменении данных в поле «Название» включите инструкцию **Beep** в процедуру обработки события Название_Change:

```
Private Sub Название_Change()

   Beep

End Sub
```

   Процедура обработки события выполняется каждый раз при возникновении этого события для объекта.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowWriteEventProcedureSW":1:"Foo":"Invisible"}

Выполнение программы Visual Basic для приложений (VBA)

В Microsoft Access для выполнения программы Visual Basic следует запустить содержащую эту программу процедуру **Sub** или процедуру **Function**. Программа записывается внутри процедуры как набор инструкций и методов, с помощью которых выполняются требуемые операции или рассчитываются значения.

Процедуры сохраняются в объектах базы данных, которые называются модулями. Сами модули не являются выполняемыми объектами; содержащиеся в модулях процедуры выполняются в ответ на события или вызываются из выражений, макросов или других процедур.

В Microsoft Access существуют следующие способы запуска программ Visual Basic:

- Включение программы в процедуру обработки события..

  Например, программу, выполняющуюся при открытии формы, включают в процедуру обработки события **Нажатие кнопки (Click)** для кнопки, при нажатии которой будет открываться эта форма. Для получения дополнительных сведений об изменении процедуры обработки события нажмите кнопку ... .
- Вызов функции в выражении или на панели проверки окна отладки.

  Например, функции применяются в выражениях, определяющих вычисляемые поля в формах, отчетах или запросах. Выражения используются для указания условий в запросах и фильтрах, а также в макросах и макрокомандах, в инструкциях и методах Visual Basic и в инструкциях SQL. Для получения дополнительных сведений о создании специальных функций нажмите кнопку ....
- Вызов процедуры **Sub** в другой процедуре или в окне отладки с панели проверки.

  Например, в процедуру **Sub**, которая вызывается в других процедурах, включают подпрограмму, выполняющую операции, которые требуется выполнять в разных процедурах. Вместо того чтобы включать одну и ту же программу Visual Basic в каждую процедуру, удобно один раз создать процедуру-подпрограмму, а затем вызывать ее по мере необходимости. Для получения дополнительных сведений о создании процедуры **Sub** нажмите кнопку ....
- Выбор команды **Продолжить** в меню **Запуск** для запуска процедуры без аргументов.

  В окне модуля поместите курсор в процедуру которую нужно запустить. Выберите **Продолжить** в меню **Запуск**.
- Выполнение макрокоманды **ЗапускПрограммы (RunCode)** в макросе.

  Макрокоманда **ЗапускПрограммы (RunCode)**, выполняемая в макросе, вызывает встроенную функцию Visual Basic или функцию, созданную пользователем. (Для запуска процедуры **Sub** следует создать функцию, которая вызывает процедуру **Sub**, и с помощью макрокоманды **ЗапускПрограммы (RunCode)** вызвать эту функцию). Для получения дополнительных сведений о макрокоманде **ЗапускПрограммы (RunCode)** нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowRunVBCodeSW":1:"Foo":"Invisible"}

Выполнение макрокоманды в процедуре Visual Basic для приложений (VBA)

В Microsoft Access определен особый объект **DoCmd**, предназначенный для выполнения макрокоманд в процедурах Visual Basic. Для выполнения макрокоманды в программе следует включить метод объекта **DoCmd** в процедуру. Большинству макрокоманд соответствуют методы **DoCmd**, имена которых совпадают с английскими именами макрокоманд.

- Для вызова методов объекта **DoCmd** в процедурах используется следующий синтаксис:

  DoCmd.ИмяМакрокоманды [аргументы]

имяМакрокоманды представляет английское имя макрокоманды. Необязательные *аргументы* представляют аргументы макрокоманды, если таковые существуют.

Например, для создания процедуры, в которой будет выполняться макрокоманда **ОткрытьФорму (OpenForm)**, следует вызвать в процедуре метод **OpenForm** объекта **DoCmd**. Выполнение следующей конструкции эквивалентно выполнению макрокоманды **ОткрытьФорму**; в результате будет открыта форма «Новый товар»:

```
DoCmd.OpenForm "Новый товар"
```

У нескольких макрокоманд нет соответствующих им методов объекта **DoCmd**, хотя у некоторых из них имеются эквивалентные инструкции или функции Visual Basic. Для того чтобы увидеть список этих макрокоманд и эквивалентных инструкций Visual Basic, нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowRunActionsVBSW":1:"Foo":"Invisible"}

Макрокоманды, не имеющие соответствующих методов объекта DoCmd, и эквивалентные им инструкции

Макрокоманда	Инструкция Visual Basic
ДобавитьМеню (AddMenu)	Отсутствует
Сообщение (MsgBox)	Функция и инструкция **MsgBox**
ЗапускПриложения (RunApp)	Функция **Shell**
ЗапускПрограммы (RunCode)	Вызов процедуры (инструкция **Call**)
КомандыКлавиатуры (SendKeys)	Инструкция **SendKeys**
ЗадатьЗначение (SetValue)	Инструкция присвоения (**Let**)
ОстановитьВсеМакросы (StopAllMacros)	Инструкции **Stop** или **End**
ОстановитьМакрос (StopMacro)	Инструкции **Exit Sub** или **Exit Function**

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acrefActionsNoDoCmdMethodSW":1:"Foo":"Invisible"}

Вызов встроенной справки Visual Basic

Существует ряд способов получить ответы на вопросы о языке Visual Basic для приложений (VBA), а также о его ключевых словах. Ниже приводится описание полезных приемов обращения к справочной системе.

Примечание. Если программа установки Microsoft Access выполняется с параметром **Обычная**, то не устанавливаются разделы встроенной справки для компонентов языка, общих для Microsoft Access и других приложений. Например, не устанавливаются разделы справки для некоторых функций и инструкций Visual Basic. Для установки всех компонентов справки языка Visual Basic, снова запустите программу установки, выберите параметр компонентов справки, нажмите кнопку **Состав** и выберите «Справочник по программированию».

Существуют следующие средства доступа к встроенной справочной системе Visual Basic:

- **Нажатие клавиши F1 в окне модуля.** При работе в окне модуля имеется возможность прямого вызова справки о методе, свойстве, функции, инструкции или объекте. Выберите ключевое слово в строке программы и нажмите клавишу F1.
- **Нажатие клавиши F1 в окне просмотра объектов.** В окне просмотра объектов выводится список методов и свойств, определенных для каждого объекта. Для получения дополнительных сведений о работе в окне просмотра объектов нажмите кнопку  .
- **Вызов помощника**. Для вызова помощника нужно нажать кнопку **Помощник** на панели инструментов, или, если помощник уже на экране, но закрыт другими объектами, выберите его с помощью мыши. Помощник позволяет легко найти необходимую информацию.

  Примечания

  - Когда активным является окно модуля, помощник по умолчанию выводит только разделы справки по программированию. Для поиска в справке по программированию и в справке по продукту щелкните правой кнопкой окно помощника, выберите в контекстном меню команду **Параметры** и установите флажок **Просматривать две справочные системы при работе с VBA**.
  - Если активным является любое другое окно, помощник выводит только разделы справки по продукту. Справка о макрокомандах и свойствах Microsoft Access доступна в справочной системе продукта и в справке по программированию.
- Использование вкладок **Содержание**, **Предметный указатель** и **Поиск**. Окно справочной системы содержит несколько вкладок, обеспечивающих разные способы поиска справки. Для вывода этого окна следует выбрать команду **Вызов справки** в меню **?**.
  - Выберите вкладку **Содержание** и раскройте двойным нажатием кнопки мыши книжку **Справочник по программированию Microsoft Access**. Содержимое этого раздела сгруппировано по следующим категориям: макрокоманды, события, функции, объекты и семейства, методы, свойства, инструкции. Для того чтобы увидеть список разделов справки, относящихся к каждой категории, раскройте соответствующий подраздел двойным нажатием кнопки мыши.

    Для печати раздела справки или полного набора разделов выделите раздел или разделы и нажмите кнопку **Печать**.
  - На вкладке **Предметный указатель** проводится тематический поиск нужных разделов справки. В верхнее поле вводятся первые буквы, по которым проводится поиск заголовков тем в указателе. Для того чтобы вывести на экран список разделов справки, следует выбрать заголовок в указателе и нажать кнопку **Вывести**.

    Примечание. Если несколько разделов справки имеют одинаковые имена, то для второго, третьего и пр. разделов выводится имя программного продукта или того компонента продукта, к которому относится раздел. Для первого раздела продукт или компонент не указывается.
  - Вкладка **Поиск** позволяет провести полномасштабный поиск конкретных слов или фраз в справочной системе.

Советы

- Для переключения между окном приложения и окном справки используйте клавиши ALT+TAB или панель задач Windows. После выхода из активного окна справочной системы можно вернуться к тому же разделу справки.
- Команды контекстного меню позволяют копировать выделенный текст, изменять размер шрифта, с которым выводится справка, выводит раздел на печать или удерживать окно справки поверх других окон.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowGetOnlineHelpSW":1:"Foo":"Invisible"}

Копирование примеров программ Visual Basic из окна справки в окно модуля

1. С помощью правой кнопки мыши выделите в окне справки текст, который требуется скопировать.
2. Выберите в контекстном меню команду **Копировать**.
3. Переключитесь в окно Microsoft Access.
4. Вставьте скопированный пример в собственный модуль.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ
пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowCopyVBExamplesSW":1:"Foo":"Invisible"}

Работа с объектами Visual Basic для приложений (VBA) в окне просмотра объектов

Окно просмотра объектов позволяет посматривать объекты, доступные для использования в Microsoft Access и других приложениях, поддерживающих язык Visual Basic для приложений (VBA). Кроме того, в окне просмотра объектов выводятся методы и свойства каждого объекта. Выбрав метод или свойство, пользователь имеет возможность вставить соответствующую синтаксическую конструкцию в активный модуль.

Предполагаемые действия

- Поиск метода или свойства в окне просмотра объектов
- Вставка метода или свойства из окна просмотра объектов в активный модуль Visual Basic

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowUserObjectBrowserSW":1:"Foo":"Invisible"}

Поиск метода или свойства в окне просмотра объектов

1. Откройте любой модуль.
2. Нажмите кнопку **Просмотр объектов**  на панели инструментов.
3. В поле со списком **Библиотеки/Базы данных** выберите **Access** или другую библиотеку.
4. В списке **Модули/Классы** выберите нужный объект.
5. В списке **Методы/Свойства** выберите метод или свойство.

Примечания

- Для получения информации о классах, методах, событиях или свойствах нажмите кнопку **Справка**  на панели инструментов окна просмотра объектов.
- Для просмотра объектов из библиотеки объектов Microsoft Office 8.0, например, панелей команд, следует установить ссылку на эту библиотеку. В окне модуля выберите команду **Ссылки** в меню **Сервис**и установите флажок **Microsoft Office 8.0 Object Library**.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowLocateMethodPropertyOBSW":1:"Foo":"Invisible"}

Вставка метода или свойства из окна просмотра объектов в модуль

1. Выберите в модуле место, в которое требуется вставить метод или свойство.
2. Найдите метод или свойство в окне просмотра объектов.
3. Нажмите кнопку **Копировать**  на панели инструментов окна просмотра объектов
4. Щелкните в модуле место, куда требуется вставить копируемый элемент.
5. Нажмите кнопку **Вставить**  на панели инструментов.

Синтаксическая конструкция, показанная на нижней панели окна просмотра объектов, будет вставлена в модуль.

Примечание. После вставки синтаксической конструкции для процедуры **Sub** необходимо удалить скобки, окружающие прототипы в синтаксической конструкции.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowPasteSyntaxOBSW":1:"Foo":"Invisible"}

Изменения параметров представления программ Visual Basic для приложений (VBA) в окне модуля

1 Выберите в меню **Сервис** команду **Параметры**.

Подсказка

2 Выберите вкладку **Модуль**.

3 Выберите нужные параметры.

При установке параметров диалоговом окне **Параметры** имеется возможность изменить цвет, шрифт, размер окна модуля. Так же можно определить как показывать процедуры: внутри модуля или каждую отдельно. Возможно установление дополнительных параметров для работы в окне модуля.

Для получения дополнительных сведений о параметрах нажмите кнопку ... в строке заголовка окна диалога **Параметры**, а затем выберите нужный параметр.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowChangeVBCodeDisplaySW":1:"Foo":"Invisible"}

Отключение проверки синтаксиса в программах Visual Basic для приложений (VBA)

Visual Basic автоматически проверяет синтаксис программ, которые вводятся в окно модуля Для отключения проверки синтаксиса выполните следующие действия:

1. Выберите в меню **Сервис** команду **Параметры**.

... Подсказка

2. Выберите вкладку **Модуль**.
3. В группе **Программирование** снимите флажок **Проверка синтаксиса**.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowDisableSyntaxCheckingSW":1:"Foo":"Invisible"}

Отключение обработки ошибок в программах Visual Basic для приложений (VBA)

Если добавить в процедуру Visual Basic инструкции **On Error**, то при возникновении ошибки автоматически осуществляется переход на подпрограмму обработки ошибок. Однако в некоторых обстоятельствах, например, при отладке процедуры, требуется отключить обработку ошибок.

1 Выберите в меню **Сервис** команду **Параметры**.

Подсказка

2 Выберите вкладку **Другие**.

3 В группе **Перехват ошибок: Останов** установите флажок **При любой ошибке**.

После установки этого флажка в случае возникновения ошибки автоматически устанавливается режим останова, вне зависимости от существования подпрограмм обработки ошибок.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowIgnoreErrorHandlingSW":1:"Foo":"Invisible"}

Требование обязательного описания переменных в программах Visual Basic для приложений (VBA)

Visual Basic не требует обязательного явного описания переменной перед ее использованием в процедуре. Если используется переменная, не описанная в явном виде, Visual Basic неявно описывает ее как переменную типа **Variant**.

Несмотря на удобство использования неявно описанных переменных, в некоторых случаях наличие таких переменных может привести к ошибкам, которые трудно обнаружить. Для того чтобы наложить требование обязательного описания переменных, используемых в процедурах, выполните следующий действия:

1 Выберите в меню **Сервис** команду **Параметры**.


Подсказка

2 Выберите вкладку **Модуль**.

3 В группе **Программирование** установите флажок **Явное описание переменных**.

После установки этого флажка Microsoft Access автоматически включает инструкцию **Option Explicit** в раздел описаний любого нового модуля в базе данных, в том числе в новые модули создаваемых форм и отчетов. Для получения дополнительных сведений об инструкции **Option Explicit** нажмите кнопку  .

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowRequereDeclarationsSW":1:"Foo":"Invisible"}

Указание аргумента командной строки для использования в программе Visual Basic

При запуске Microsoft Access параметр командной строки **/Cmd** позволяет указать аргумент командной строки, который может быть передан в программу Visual Basic для приложений (VBA). Для того чтобы задать значение данного аргумента, выполните следующие действия:

1. Выберите в меню **Сервис** команду **Параметры**.

... Подсказка

2. Выберите вкладку **Модуль**
3. В группе **Только для текущей базы данных** введите значение аргумента в поле **Параметры командной строки**.

   Для получения дополнительных сведений о параметре командной строки **/Cmd** нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowSpecifyCommandLineArgumentSW":1:"Foo":"Invisible"}

Изменение режима компиляции программ Visual Basic для приложений (VBA)

Предполагаемые действия

- Дополнительная информация об отладке
- Использование точек останова для временной остановки работы программы
- Пошаговая отладка программы
- Просмотр значений переменных и выражений в окне отладки
- Установка контрольных значений переменных или выражений
- Отладка вызовов процедур Visual Basic
- Показать величину переменной на панели окна отладки
- Показать величину переменной или выражения в окне модуля когда программа временно остановлена

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecDebugVBCodeSW":1:"Foo":"Invisible"}

Использование точек останова для временной остановки работы программы

Прерыванием называют состояние, когда программа Visual Basic остается загруженной, но после выполнения одной инструкции следующая инструкция не выполняется. Для прерывания программы Visual Basic следует установить в программе точку останова..

1 В окне модуля переведите курсор в строку, в которой нет точки останова.

2 Нажмите кнопку  на панели инструментов или нажмите клавишу F9.

Для того чтобы снять точку останова, переведите курсор в строку, в которой поставлена точка останова, и нажмите кнопку  на панели инструментов или нажмите клавишу F9. Для продолжения выполнения программы выберите **Продолжить** в меню **Запуск**.

Примечание. Для прерывания программы можно также добавить в процедуру инструкцию **Stop** или нажать клавиши CTRL+BREAK во время выполнения программы.

{ewc HLP95EN.DLL, DYNALINK, "nïSnïSnïSnïSnïS nïS Web nïS nïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïSnïS":"achowSuspendExecutionSW":1:"Foo":"Invisible"}

Выполнение программы Visual Basic для приложений (VBA) в пошаговом режиме

Выполнение программы Visual Basic в пошаговом режиме позволяет определить, в каком месте программы возникает ошибка. Такой режим позволяет видеть результаты выполнения каждой программной строки.

1 Прервите выполнение программы.

Инструкции

Microsoft Access выводит строку программы, на которой было прервано выполнение.

2 Выполните одно из следующих действий.

- Для выполнения каждой строки программы по отдельности, в том числе и строк процедур, вызываемых из других процедур (с заходом в процедуры), нажмите кнопку **Шаг с заходом** на панели инструментов.
- Для выполнения каждой строки программы по отдельности, но с выполнением вызываемых процедур (с обходом процедур), нажмите кнопку **Шаг с обходом** на панели инструментов.
- Для выполнения программы с ее начала и до строки, предшествующей той, в которой находится курсор, с последующим прерыванием для перехода в пошаговый режим (выполнение до текущей позиции), выберите в меню **Отладка** команду **Выполнить до текущей позиции**.
- Для выполнения остатка текущей процедуры с последующим переходом на следующую строку в предыдущей процедуре на дереве вызовов, нажмите кнопку **Шаг с выходом** на панели инструментов.

Примечание. Пройти процедуру в пошаговом режиме можно без прерывания программы. В окне модуля установите курсор в нужной процедуре и нажмите кнопку **Шаг с заходом** на панели инструментов.

Пользователь имеет возможность попеременно использовать все три разновидности пошагового режима в зависимости от того, какие части программы требуется проверить.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowStepThroughVBCodeSW":1:"Foo":"Invisible"}

Вывод значений переменных и выражений на панели проверки в окне отладки

Панель проверки окна отладки используется для проверки результатов выполнения строк программ Visual Basic для приложений (VBA). Панель проверки позволяет выводить значения элемента управления, поля или свойства; отображать результат выражения.; а также присваивать новое значение переменной, полю или свойству. Панель проверки используется аналогично странице черновика, на которой проверяются инструкции, методы, и процедуры **Sub**.

1 Для того чтобы сделать панель проверки доступной в определенном месте программы, прервите в этом месте выполнение программы.


Инструкции

2 Нажмите кнопку на панели инструментов.

3 Введите в окно отладки инструкцию, метод или вызов процедур **Function** и **Sub** и нажмите клавишу ENTER.

Советы

- Для того чтобы вывести на панель проверки результат выражения, вызовите метод **Print** объекта **Debug** и укажите выражение. Для краткости можно заменить вызов метода **Print** символом (**?**). Например, Visual Basic выведет результат, возвращаемый специальной функцией «НачалоСледующегоМесяца», при вводе на панель проверки любой из следующих строк:

  ```
  Debug.Print НачалоСледующегоМесяца()

  ? НачалоСледующегоМесяца()
  ```

  Кроме того, допускается включение вызовов метода **Print** объекта **Debug** в программу Visual Basic для записи результатов значений или выражений на панель проверки во время выполнения программы. Эти результаты станут доступными для просмотра после завершения выполнения программы.
- Панель проверки может быть открыта в любом окне Microsoft Access нажатием клавиш CTRL+G.

{ewc HLP95EN.DLL, DYNALINK, "nїSnїSnїSnїSnїS nїS Web nїS nїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїSnїS":"achowUseImmediatePaneSW":1:"Foo":"Invisible"}

Установка контрольных выражений в программе Visual Basic для приложений (VBA)

После прерывания выполнения программы станут доступными для просмотра результаты контрольных выражений, выбранных на панели контрольных значений в окне отладки. Кроме того, возможен прямой просмотр значений любых выражений, выбираемых пользователем.

Предлолагаемые действия

- Добавление контрольного выражения в окне отладки
- Просмотр контрольных значений в контрольном окне

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowSetWatchSW":1:"Foo":"Invisible"}

Добавление контрольного выражения на панель контрольных значений окна отладки.

1. Откройте процедуру в окне **Модуля**.
2. Выберите в меню **Отладка** команду **Добавить контрольное значение**.
3. Если в окне модуля имеется выделенное выражение, оно автоматически вводится в поле **Выражение** в диалоговом окне. Если выделенного выражения нет, то контрольное выражение определяет пользователь. В качестве контрольного выражения можно указать переменную, свойство, функцию или другое правильное выражение.

   Можно не вводить выражение, а перетащить его из окна модуля.
4. Для того чтобы указать диапазон программ, в которых будет определяться значение контрольного выражения, выберите модуль или процедуру в группе **Контекст**.

   Примечание. Выбирайте самый узкий диапазон, отвечающий вашим интересам. Выбор всех процедур или всех модулей может замедлить выполнение программы.
5. Для того чтобы определить реакцию системы на значение контрольного выражения, выберите нужный параметр в группе **Тип контрольного значения**:
   - Для вывода значения контрольного выражения выберите параметр **Контрольное выражение**.
   - Для прерывания программы, если выражение имеет значение **True**, выберите параметр **Останов, если значение истинно**.
   - Для прерывания программы в момент изменения значения выражения выберите параметр **Останов при изменении значения**.

При выполнении программы в окне отладки будет отображаться значение заданного выражения. Для того чтобы изменить контрольное выражение, выделите выражение на панели контрольных выражений и выберите в меню **Отладка** команду **Изменить контрольное значение**. Для удаления контрольного выражения нажмите кнопку **Удалить** в диалоговом окне **Изменение контрольного значения**.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowAddWatchExpressionWatchPaneSW":1:"Foo":"Invisible"}

Просмотр контрольных значений при отладке программы Visual Basic для приложений (VBA)

1 Прервите выполнение программы Visual Basic.

Инструкции

2 Выделите выражение, значение которого требуется просмотреть.

3 Нажмите кнопку  на панели инструментов.

Открывается окно диалога **Контрольное значение**, позволяющее увидеть значение выделенного выражения и добавить его на панель контрольных значений окна отладки.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowInstantWatchSW":1:"Foo":"Invisible"}

Трассировка вызовов процедур при отладке программ Visual Basic

После прерывания выполнения программы Visual Basic для приложений (VBA) при отладке программы пользователь имеет возможность открыть окно диалога **Вызовы**, содержащее список процедур, которые были запущены, но не завершены.

1 Прервите выполнение программы Visual Basic.

Инструкции

2 Нажмите кнопку  на панели инструментов.

Первой в списке выводится последняя вызванная процедура, далее следует процедура, вызванная перед ней и т.д. Для вывода инструкции, которая вызывает следующую процедуру в списке, нажмите кнопку **Показать**.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowTraceCallsSW":1:"Foo":"Invisible"}

Работа с данными и с объектами базы данных с помощью программ Visual Basic для приложений (VBA)

Наряду с обработкой данных и объектов базы данных с помощью интерфейса пользователя Microsoft Access, пользователь имеет возможность определить процедуры которые автоматически создают, изменяют и удаляют данные и объекты. Например, можно создать процедуру, помещающую подпись в форму или изменяющую цвет поля в форме.

В процедурах допускаются прямые ссылки на данные и объекты или описание объектных переменных представляющих данные и объекты. После описания объектной переменной и присвоения ей значения эта переменная может быть использована везде, где используется объект. Значения объектных переменных изменяются так же, как значения других переменных.

Например, следующая процедура **Sub** создает таблицу «Старые счета», содержащую единственное поле «КодЗаказа»:

```
Sub CreateTable ()

     ' Описывает переменные.
     Dim dbs As Database, tbl As TableDef, fld As Field

     ' Присваивает объектной переменной
     ' значение текущей базы данных.
     Set dbs = CurrentDB

     ' Создает таблицу и поле и присваивает их
     ' объектным переменным.
     Set tbl = dbs.CreateTableDef("Старые счета ")
     Set fld = tbl.CreateField("КодЗаказа", dbText)

     ' Добавляет поле в таблицу
     ' и таблицу в базу данных.
     tbl.Fields.Append fld
     dbs.TableDefs.Append tbl
     dbs.TableDefs.Refresh

End Sub
```

В ядре базы данных Microsoft Jet определены объекты, такие как таблицы, запросы, связи и индексы, с помощью которых выполняются операции сохранения и упорядочения данных в базах данных Microsoft Access. Такие объекты называют объектами доступа к данным. (DAO). Программы Visual Basic, в которых используются объекты доступа к данным, могут применяться для совместной работы с другими приложениями, такими как Microsoft Excel, использующими ядро базы данных Jet.

Кроме того, в Microsoft Access определен ряд объектов, предназначенных непосредственно для обработки данных, таких как формы, отчеты и элементы управления.

Для получения дополнительных сведений о работе с объектами доступа к данным нажмите кнопку  или см. главу 5 «Объекты и семейства» книги Разработка приложений для Microsoft Access 97.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acconWorkWithDataSW":1:"Foo":"Invisible"}

Преобразование макросов в процедуры Visual Basic для приложений (VBA)

Microsoft Access позволяет автоматически преобразовывать макросы в процедуры обработки событий или в модули, которые выполняют эквивалентные действия с помощью программ Visual Basic. Допускается преобразование макросов, определенных в форме или отчете, или преобразование общих макросов, не связанных с конкретными формами и отчетами.

Предполагаемые действия


Преобразование макросов в форме report

Преобразование общих макросов

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅS пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅS пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅS":"achowConvertMacrosToVBSW":1:"Foo":"Invisible"}

Преобразование макросов в форме или отчете в процедуры Visual Basic для приложений (VBA)

1 В режиме конструктора формы или в режиме конструктора отчета

2 Выберите в меню **Сервис** команду **Макросы** и подкоманду **Преобразовать макросы**.

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїSпїS":"achowConvertMacrosFormReportSW":1:"Foo":"Invisible"}

Преобразование общих макросов в процедуры Visual Basic для приложений (VBA)

1. В окне базы данных выберите вкладку **Макросы**.
2. Выберите макрос.
3. Выберите в меню **Файл** команду **Сохранить как/Экспорт**.
4. В диалоговом окне **Сохранение** выберите параметр **В виде модуля Visual Basic**.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowConvertGlobalMacrosSW":1:"Foo":"Invisible"}

Использование аргументов в программах Visual Basic для приложений (VBA)

Аргументы используются для передачи необходимых данных в процедуру, инструкцию или метод. Для того чтобы иметь возможности передавать информацию в вызываемую процедуру, следует при создании процедуры **Function** или **Sub** описать ее аргументы.

При вызове процедуры, имеющей аргументы, необходимо указать значения этих аргументов. Аргументы имеются у ряда инструкций и методов.

Предполагаемые действия

- Создание специальной функции
- Создание процедуры **Sub**
- Указание аргументов процедуры, инструкции или метода

{ewc HLP95EN.DLL, DYNALINK, "nïSnïSnïSnïSnïS nïS Web nïS nïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acdecUseArgumentsVBSW":1:"Foo":"Invisible"}

Указание аргументов процедуры, инструкции или метода в программе Visual Basic

- Аргументы помещаются сразу после имени процедуры, инструкции или метода. Для процедуры **Function** или для метода, который возвращает значение, необходимо заключать аргументы в круглые скобки. Если задаются несколько аргументов, то необходимо разделять их запятыми.

Например, в следующей конструкции метод **Close** объекта **DoCmd** закрывает форму «Новый товар». Данный метод принимает два аргумента, константу `acForm` и имя `"Новый товар"`:

```
DoCmd.Close acForm, "Новый товар"
```

Этот метод не возвращает значение, поэтому аргументы не заключаются в скобки.

Совет. Допускается вставка в программы Visual Basic для приложений (VBA) синтаксических конструкций, содержащих имена методов и свойств со всеми необходимыми аргументами, из окна просмотра объектов. При этом вставляются именованные аргументы, значения которых пользователь может задавать в произвольном порядке. В программах допускается использование именованных аргументов или их замена на стандартные аргументы. Для получения дополнительных сведений о работе в окне просмотра объектов нажмите кнопку ... Для получения дополнительных сведений об именованных аргументах нажмите кнопку
...

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"achowSetArgumentsVBSW":1:"Foo":"Invisible"}

Отладка программ Visual Basic для приложений (VBA)

Отладкой называют процесс поиска и исправления ошибок в программах. Существуют ошибки трех типов:

- Ошибки при компиляции возникают в неверно составленных программных конструкциях. Примерами таких ошибок могут служить неполные пары инструкций (например, **If** и **End If** или **For** и **Next**) или ошибки, нарушающие правила языка Visual Basic (например, ошибочно записанные ключевые слова, пропущенные разделители или неверные типы данных).

  К ошибкам при компиляции относятся также ошибки синтаксиса, являющиеся результатом нарушения правил грамматики или пунктуации. Примерами таких ошибок могут служить неполные пары скобок или неверное количество аргументов, переданных в функцию.
- Ошибки при выполнении возникают на стадии выполнения программы. К ошибкам при выполнении относятся недопустимые операции, такие как деление на нуль или запись в несуществующий файл.
- Логическими ошибками называют ошибки, которые не мешают выполнению приложения, но приводят к неверным результатам.

Чтобы помочь пользователю обнаруживать ошибки любых типов и контролировать ход выполнения программ, в Visual Basic имеются средства отладки, позволяющие выполнять программы в пошаговом режиме, следить за изменением значений переменных или выражений и выполнять трассировку (регистрировать вызовы) процедур.

Примечание. Большинство средств отладки Microsoft Access становятся доступными после прерывания выполнения программы. Для получения дополнительных сведений нажмите кнопку

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconResolveErrorsInVisualBasicCodeW":1:"Foo":"Invisible"}

Использование окна модуля, окна отладки и средства просмотра модели объектов

Разделы

Создание модуля класса, связанного с формой или отчетом

1. В окне базы данных или в окне модуля выберите команду **Модуль класса** в меню **Вставка**.
   Появится пустой модуль класса.
2. Добавьте в модуль необходимые процедуры и описания.
3. Для сохранения модуля нажмите кнопку **Сохранить**  на панели инструментов, и задайте имя модуля в диалоговом окне **Сохранение**.

После первого сохранения модуля класса его имя появляется на вкладке **Модули** окна базы данных. Чтобы открыть модуль нажмите кнопку **Конструктор**.

Для получения сведений по использованию модулей классов нажмите кнопку  . См. также главу 2 «Введение в Visual Basic» и главу 5 «Объекты и семейства» книги *Разработка приложений для Microsoft Access 97*.

Просмотр информации по синтаксису в окне модуля

1. Выберите команду **Параметры** в меню **Сервис**.
2. Выберите вкладку **Модуль**.
3. Убедитесь, что в группе **Программирование** в окне **Параметры** установлен флажок **Краткие сведения**.

Если флажок **Краткие сведения** установлен, то в окне модуля Microsoft Access отображается информация по синтаксису. При вводе имени существующего метода или процедуры после пробела или открывающейся скобки автоматически возникает подсказка с данными по синтаксису этого метода или процедуры, например, отображаются необходимые аргументы.

Просмотр значения переменной или выражения в окне модуля приостановленной программы

1. Выберите команду **Параметры** в меню **Сервис**.
2. Выберите вкладку **Модуль**.
3. Убедитесь, что в группе **Программирование** в окне **Параметры** установлен флажок **Подсказки значений данных**.

Если флажок **Подсказки значений данных** установлен, то при остановке программы возможен просмотр значений переменных и выражений. Для просмотра текущего значения достаточно выбрать переменную или выражение с помощью мыши.

Просмотр значений переменных в области локальных переменных окна отладки

1 Приостановите выполнение программы, написанной на языке Visual Basic для приложений.

Инструкции

2 Нажмите кнопку **Окно отладки** на панели инструментов.

В верхней половине окна **Отладка** на вкладке **Локальные** будут автоматически выводится имя, текущее значение и тип всех переменных и объектов текущей процедуры. Значения на вкладке **Локальные** обновляются при каждой остановке программы. Для изменения значения переменной в области локальных переменных необходимо выбрать существующее значение и ввести новое.

Просмотр списка подходящих объектов, свойств и методов в окне модуля

1. Выберите команду **Параметры** в меню **Сервис**.
2. Выберите вкладку **Модуль**.
3. Убедитесь, что в группе **Программирование** в окне **Параметры** установлен флажок **Список компонентов**.

Если установлен флажок **Список компонентов**, то включается режим помощи в завершении инструкций при вводе программы на языке Visual Basic для приложений. При вводе имени объекта, следующего за открывающей скобкой или за пробелом, отображается список соответствующих объектов, свойств и методов, которые могут следовать за этим именем. Для ввода элемента списка в текущую строку программы используются следующие клавиши.

Клавиша	Действие
TAB	Вводит элемент
ПРОБЕЛ	Вводит элемент через пробел
ENTER	Вводит элемент и перемещает курсор на следующую строку

Примечание. Чтобы убрать список, нажмите клавишу ESC.

Разрешение вопросов по работе с модулями Visual Basic

Вопросы

- Как определить, в каком месте в программе Visual Basic для приложений возникает ошибка?
- Аргумент условие функции **DLookup** или другой статистической функции по подмножеству дает неверные результаты.
- При выполнении программы Microsoft Access выводит ненужное сообщение.
- Как с помощью Visual Basic добавить новую запись в таблицу?
- Процедура дает неверные результаты при вызове одного из методов **Find**.
- Как проверить работу процедуры **Function** или **Sub**?
- Получено сообщение «Процедура Sub или Function не определена» при попытке вызвать функцию IsLoaded.
- Получено сообщение «Метод или компонент данных не найден» при ссылке в программе на поле в таблице.
- Как с помощью Visual Basic для приложений изменить базовую инструкцию SQL запроса?

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbModuleSetWarningsSW":1:"Foo":"Invisible"}

При выполнении программы Microsoft Access выводит ненужное сообщение

Для того чтобы временно отключить вывод предупреждений и других сообщений на время выполнения программы Visual Basic для приложений, следует с помощью метода **SetWarnings** выполнить макрокоманду **УстановитьСообщения (SetWarnings)**. Повторный вызов метода **SetWarnings** позволяет снова включить режим вывода сообщений. Для получения дополнительных сведений нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbModuleSetWarningsSW":1:"Foo":"Invisible"}

Добавление новой записи в таблицу с помощью Visual Basic для приложений

Новая запись добавляется в программе в таблицу или в результирующий набор записей с помощью методов **AddNew** и **Update** объекта **Recordset**. Для получения дополнительных сведений нажмите кнопку ... или см. главу 9 «Работа с полями и записями» книги *Разработка приложений для Microsoft Access 97.*

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbModuleAddNewRecordSW":1:"Foo":"Invisible"}

Процедура дает неверные результаты при вызове одного из методов Find

Методы **FindFirst**, **FindLast**, **FindNext** и **FindPrevious** могут давать неверные результаты по одной из следующих причин.

- Неверный синтаксис ссылок на значение элемента управления или свойства в аргументе *условие*. Для получения дополнительных сведений о ссылках на значения элементов управления нажмите кнопку ....
- Неверный синтаксис в операции, используемой в аргументе *условие* для слияния значения поля, элемента управления или свойства со строковым значением. Для получения дополнительных сведений об использовании значений в выражениях нажмите кнопку ....

Методы **FindRecord** и **FindNext** могут также давать неверные результаты в том случае, когда аргумент *текущееПоле* метода **FindRecord** определен с помощью константы **acCurrent**. Перед вызовом метода **FindRecord** или **FindNext** переведите фокус на нужный элемент управления с помощью метода **GoToControl**.

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbModuleFindMethodsSW":1:"Foo":"Invisible"}

Проверка работы процедуры Function или Sub

Выводите промежуточные результаты процедуры **Function** или **Sub** на панель проверки окна отладки. Например, для вывода значения, возвращаемого специальной функцией «ПервыйДеньМесяца», введите вопросительный знак, пробел, имя функции и все необходимые аргументы в скобках:

```
? ПервыйДеньМесяца(Now)
```

Для получения дополнительных сведений нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbModuleRunProcedureSW":1:"Foo":"Invisible"}

Получено сообщение «Процедура Sub или Function не определена» при попытке вызвать функцию IsLoaded

Функция IsLoaded является примером процедуры **Function**. Эта функция, определенная в демонстрационной базе данных «Борей», не является встроенной функцией Visual Basic. Для того чтобы иметь возможность вызывать функцию IsLoaded в собственной базе данных, откройте базу данных «Борей», скопируйте функцию в буфер и вставьте ее в модуль в собственной базе данных.

Для того чтобы получить справку об открытии базы данных «Борей», нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbModuleIsLoadedSW":1:"Foo":"Invisible"}

Получено сообщение «Метод или компонент данных не найден» при ссылке в программе на поле в таблице

Microsoft Access возвращает эту ошибку, если в ссылке на поле таблицы используется оператор **.** (точка) вместо оператора **!** (восклицательный знак). Например, данная ошибка возникнет в последней строке следующей программы Visual Basic для приложений:

```
Dim dbsCurrent As Database
Dim tblTest As TableDef
Dim fldTest As Field

Set dbsCurrent = CurrentDb
Set tblTest = dbsCurrent.TableDefs("Типы")
Set fldTest = tblTest.КодТипа
```

Для того чтобы исправить ошибку, замените в программе оператор **.** (точка) на оператор **!** или сошлитесь на поле как на элемент семейства **Fields**. Любая из следующих конструкций является правильной, хотя первая будет выполняться несколько быстрее:

```
Set fldTest = tblTest!КодТипа

Set fldTest = tblTest.Fields("КодТипа")
```

Для получения дополнительных сведений об использовании операторов **.** (точка) и **!** нажмите кнопку ....

{ewc HLP95EN.DLL, DYNALINK, "nïSnïSnïSnïSnïS nïS Web nïSnïSnïS nïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS":"actrbModuleReferTableFieldSW":1:"Foo":"Invisible"}

Изменение базовой инструкции SQL запроса с помощью Visual Basic для приложений

Для того чтобы изменить базовую инструкцию SQL запроса следует использовать свойство **SQL** объекта **QueryDef**. Например, следующая программа Visual Basic для приложений определяет инструкцию SQL запроса «СписокСотрудников», указывающую отбор всех записей из таблицы «Сотрудники»:

```
Dim dbsCurrent As Database
Dim qryTest As QueryDef

Set dbsCurrent = CurrentDb
Set qryTest = dbsCurrent.QueryDefs("СписокСотрудников")
qryTest.SQL = "SELECT * FROM Сотрудники;"
```

Для получения дополнительных сведений о свойстве **SQL** нажмите кнопку  .

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"actrbModuleChangeSQLSW":1:"Foo":"Invisible"}

Оптимизация программ Visual Basic для приложений

Обычно, лучшим способом ускорить выполнение программы является выбор более эффективного алгоритма. Однако полезно помнить о следующих приемах, повышающих быстродействие.

- Заранее компилируйте все программы в базе данных. В окне модуля выберите в меню **Отладка** команду **Компилировать и сохранить все модули**.

  Примечание. После переименования файла базы данных следует перекомпилировать программы.
- Описывайте все переменные в явном виде. Для того чтобы наложить требование явного описания всех переменных перед их использованием в процедурах, установите флажок **Явное описание переменных** на вкладке **Модуль** диалогового окна **Параметры**. Для получения дополнительных сведений нажмите кнопку ...
- Используйте наиболее подходящие типы переменных. Например, описывайте переменную, представляющую форму с типом **Form**, а не с типом **Object** или **Variant**.
- Используйте переменные для ссылок на свойства, элементы управления и объекты доступа к данным. Если требуется сослаться несколько раз на значение свойства элемента управления в форме, а также на объект доступа к данным или на его свойство, создайте объектные переменные и используйте эти переменные вместо полных идентификаторов.
- Пользуйтесь ключевым словом **Me** для ссылок на форму в процедуре обработки события.
- Не злоупотребляйте использованием функции **IIf**. Старайтесь не пользоваться функцией **IIf**, если расчет любого из входящих в эту функцию выражений занимает много времени.
- Разумно используйте функции, обрабатывающие строковые значения. Пользуйтесь функциями типа **$** (например, функцией **Str$**) при работе со строковыми значениями и функциями, не принадлежащими к типу **$**, при работе с типом **Variant** или другими типами данных.
- По возможности, используйте типы данных **Integer** или **Long** в математических расчетах.
- Для экономии памяти используйте динамические массивы и инструкции **Erase** или **ReDim**.
- По возможности, используйте константы.
- Для ускорения переходов по записям пользуйтесь закладками вместо метода **FindNext** или других способов перехода на конкретную запись.
- Применяйте методы **FindRecord** и **FindNext** к индексированным полям. Для индексированных полей эти методы выполняются значительно быстрее.
- Оптимизируйте структуру модулей. Visual Basic загружает модули по требованию — это означает, что модуль загружается в память только при вызове в программе процедуры из этого модуля. Размещение взаимосвязанных процедур в одном модуле позволяет снизить время, требующееся на загрузку.
- Уничтожайте ненужные программы и не используемые переменные. По мере развития приложения могут появиться процедуры, которые больше не вызываются, или описанные переменные, которые больше не используются. Регулярно просматривайте ваши программы и удаляйте устаревшие блоки.

Более подробное описание см. в главе 13 «Оптимизация приложений» книги *Разработка приложений для Microsoft Access 97.*

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconOptimizingVBSW":1:"Foo":"Invisible"}

Содержание пакета «Microsoft Office 97, издание для разработчиков»

Пакет «Microsoft Office 97, издание для разработчиков (ODE)» включает усовершенствованные инструменты, позволяющие разработчикам создавать и распространять специальные приложения.

В пакет ODE входят:

Компакт-диск Microsoft Office 97, профессиональное издание

Компакт-диск инструментов Microsoft Office 97, издание для разработчиков

- Лицензия на бесплатное распространение и инструменты (включая приложение Microsoft Graph), которые позволяют распространять копии специальных приложений Microsoft Office для произвольного количества пользователей.
- Расширенный мастер по установке, с помощью которого можно создавать диски и упаковывать файлы для распространяемых приложений.
- Новая версия диспетчера репликации, который позволяет регистрировать изменения по репликам, определять какие объекты базы данных являются реплицируемыми, просматривать все реплики в наборе, управлять несколькими наборами реплик одновременно.
- Компонент Microsoft Access Source Code Control для поддержки нескольких разработчиков. (Microsoft Visual SourceSafe продается отдельно).
- Элементы ActiveX, которые используются для построения разносторонних приложений и для дублирования функций Microsoft Windows 95, включая новые элементы управления для приложений Интернета.
- Windows API Viewer, который содержит описания и константы, используемые в интерфейсе прикладного программирования Windows 95 (API). Предусмотрено копирование и восстановление этих описаний и констант в модули Visual Basic.
- Windows Help Workshop для разработки системы помощи для приложений.
- Компонент Microsoft Office Developer Sampler со сведениями из базы данных Microsoft Developer's Network, связанными с разработкой приложений для Office.

Печатная документация

- *Microsoft Office 97 Programmer's Guide*
- *Разработка приложений для Microsoft Access 97*

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acconADTContentsSW":1:"Foo":"Invisible"}

Ограничение данных в подмножестве записей

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"achowSpecifyCriteriaDomainAggC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"achowSpecifyCriteriaDomainAggX":1}

При работе с данными часто возникает необходимость ограничить набор отбираемых записей. Например, указание условий отбора требуется при использовании статистических функций по подмножеству. Кроме того, условия требуется задать при вызове одного из методов **Find** объекта **Recordset,** при определении свойства **Фильтр (Filter)** для формы или при создании инструкции SQL. Несмотря на то, что синтаксис для всех перечисленных операций различный, при указании условий отбора записей применяются одни и те же приемы.

Например, статистическая функция по подмножеству **DSum** позволяет найти общую сумму стоимости доставки для всех заказов в таблице «Заказы». Для создания вычисляемого поля в документе следует ввести в ячейку свойства **Данные (ControlSource)** объекта-поля такое выражение:

```
=DSum("[СтоимостьДоставки]", "Заказы")
```

Если указать аргумент *условие*, то вычисления будут выполнены для подмножества, отбираемого в наборе, который был задан в аргументе *наборЗаписей*. Например, в следующем выражении функция **DSum** используется для расчета стоимости доставки всех заказов в таблице "Заказы", доставляемых в Крым.

```
=DSum("[СтоимостьДоставки]", "Заказы", "[ОбластьПолучателя] = 'Крым'")
```

Если для статистической функции по подмножеству указан аргумент *условие*, то вначале Microsoft Access находит значение каждого содержащегося в этом аргументе выражения, после чего формируется строковое выражение. Затем строковое выражение передается в функцию. Строковое выражение эквивалентно предложению WHERE в инструкции SQL, но без ключевого слова WHERE.

Условия отбора могут быть заданы с помощью числовых или строковых значений, а также с помощью значений даты/времени. Вне зависимости от типа выражения, с помощью которого задаются условия отбора, аргумент *условие* всегда перед передачей в статистическую функцию по подмножеству преобразуется в строку. При этом пользователь должен обеспечить, чтобы после определения значения каждого из выражений, образующих аргумент *условие*, все компоненты были слиты в строку, заключенную в прямые кавычки (").

При конструировании строки, задающей условия отбора, необходимо соблюдать определенные правила.

Особенности указания условий для статистической функции по подмножеству обсуждаются в следующих разделах:

Числовые значения в условиях отбора

Строковые значения в условиях отбора

Значения даты и времени в условиях отбора

Изменение условия числового значения из элемента управления в форме

Изменение условия строкового значения из элемента управления в форме

Изменение условия значения даты и времени из элемента управления в форме

Указание нескольких полей в условиях отбора

Числовые значения в условиях отбора

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"achowSpecifyNumericCriteriaC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"achowSpecifyNumericCriteriaX":1}

Для того чтобы указать числовое значение в аргументе *условие*, необходимо ввести это значение как часть строкового выражения.

Предположим, что нужно найти в таблице «Сотрудники» фамилию сотрудника по известному коду. Для этого требуется указать в аргументе *условие* функции **DLookup** значение поля « КодСотрудника». Следующая конструкция возвращает фамилию сотрудника по значению кода 7 в поле «КодСотрудника».

```
=DLookup("[Фамилия]", "Сотрудники", "[КодСотрудника] = 7")
```

Строковые значения в условиях отбора

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"achowSpecifyTextualCriteriaC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"achowSpecifyTextualCriteriaX":1}

Для того чтобы указать текстовое значение в аргументе *условие*, необходимо ввести это значение как часть строкового выражения. Данное текстовое значение необходимо заключить в одинарные кавычки (').

Примечание. Одинарные кавычки позволяют ввести в аргументе *условие* строку внутри строки.

Предположим, что для поиска в таблице «Сотрудники» первого сотрудника с заданной фамилией используется метод **FindFirst** объектов доступа к данным (DAO). При этом аргумент *условие* может быть построен как показано в следующем примере. В этом примере указатель текущей записи перемещается на первую запись, в которой фамилия сотрудника - Иванов. Заметьте, что строковое значение `Иванов` заключено в одинарные кавычки, а вся строка, включая аргумент *условие*, должна быть заключена в двойные кавычки (").

```
Dim dbs As Database, rst As Recordset

Set dbs = CurrentDb
Set rst = dbs.OpenRecordset("Сотрудники", dbOpenDynaset)
rst.FindFirst "[Фамилия] = 'Иванов'"
```

Значения даты и времени в условиях отбора

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"achowSpecifyDateTimeCriteriaC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"achowSpecifyDateTimeCriteriaX":1}

Для того чтобы указать значение даты или времени в аргументе *условие*, необходимо ввести это значение как часть строкового выражения. Данное значение должно быть окружено символами номера (#).

Примечание. Символы номера (#) позволяют ввести в аргументе *условие* значение даты или времени внутри строки.

Предположим, что требуется создать условие отбора для формы «Сотрудники», согласно которому будут выводиться только сотрудники, родившиеся после 1 января 1960 г. Аргумент *условие* для свойства формы **Фильтр (Filter)** может быть задан как это показано в следующем примере. Обратите внимание на размещение символов номера.

```
Forms!Сотрудники.Filter = "[ДатаРождения] >= #1-1-60#"
```

Изменение условия числового значения из элемента управления в форме

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅS":"achowSpecifyNumCriteriaControlFormC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅS":"achowSpecifyNumCriteriaControlFormX":1}

Если требуется предоставить пользователю возможность изменять аргумент *условие* статистической функции по подмножеству, то следует указать, что значение аргумента принимается из элемента управления в форме. Например, можно указать, что аргумент *условие* принимает значение из поля в форме, в котором выводится значение поля «КодСотрудника» из таблицы «Сотрудники».

Для ввода в условие отбора числового значения из элемента управления в форме следует включить в аргумент *условие* выражение, содержащее ссылку на этот элемент управления. Это выражение должно быть выделено в строковом выражении таким образом, чтобы его значение определялось отдельно, а затем объединялось с остальной частью строкового выражения перед выполнением необходимого действия.

Предположим, что для поиска в таблице «Сотрудники» фамилии сотрудника по известному коду используется функция **DLookup.** В следующем примере условие отбора определяется по текущему значению объекта-поля «КодСотрудника» в форме «Заказы». Значение выражения, содержащего ссылку на элемент управления, определяется при каждом вызове функции, поэтому аргумент, определяющий условие отбора, изменяется в соответствие с текущим значением элемента управления в форме.

```
=DLookup("[Фамилия]", "Сотрудники", "[КодСотрудника] = " & Forms!Заказы!
КодСотрудника)
```

Если текущим значением объекта-поля «КодСотрудника» является 7, то аргумент *условие* будет передан в функцию **DLookup** в следующем виде:

```
"[КодСотрудника] = 7"
```

Совет. При вводе сложных выражений в аргумент *условие* разбивайте выражение на компоненты и проверяйте каждый из них по отдельности в окне отладки. После этого объединяйте компоненты по одному, каждый раз проверяя правильность результирующего выражения.

Допускается также указание числовых значений в аргументе *условие* с помощью переменных. Переменная должна быть выделена в строковом выражении таким образом, чтобы ее значение определялось отдельно, а затем объединялось с остальной частью строкового выражения.

Следующая конструкция демонстрирует определение аргумента *условие* с помощью переменной.

```
Dim intNum As Integer, varResult As Variant
intNum = 7
varResult = DLookup("[Фамилия]", "Сотрудники", _
   "[КодСотрудника] = " & intNum)
```

Изменение условия строкового значения из элемента управления в форме

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"achowSpecifyTextCriteriaControlFormC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"achowSpecifyTextCriteriaControlFormX":1}

Если требуется предоставить пользователю возможность изменять аргумент *условие* статистической функции по подмножеству, то следует указать, что значение аргумента принимается из элемента управления в форме. Например, можно указать, что аргумент *условие* принимает значение из списка, содержащего фамилии сотрудников из таблицы «Сотрудники».

Для ввода в условие отбора строкового значения из элемента управления в форме следует включить в аргумент *условие* выражение, содержащее ссылку на этот элемент управления. Это выражение должно быть выделено в строковом выражении таким образом, чтобы его значение определялось отдельно, а затем объединялось с остальной частью строкового выражения перед выполнением необходимой операции.

Кроме заключения полного строкового выражения в двойные кавычки (") необходимо заключать вводящееся текстовое значение в одинарные кавычки ('). Одинарные кавычки следует вводить в начале и в конце выражения, содержащего ссылку на элемент управления в форме.

Примечание. Одинарные кавычки позволяют ввести в аргументе *условие* строку внутри строки.

В следующем примере в таблице «Сотрудники» проводится поиск фамилии, совпадающей со значением элемента управления (списка) «Фамилия» в форме «Сотрудники», и возвращается значение поля «Область». Условие отбора определяется текущим значением, выбранным в списке. Обратите внимание на способ включения апострофов в выражение.

```
=DLookup("[Область]", "Сотрудники", "[Фамилия] = '" & Forms!Сотрудники!
Фамилия & "'")
```

Если текущим значением объекта-списка является `Королев`, то после вычисления выражения и объединения строк аргумент *условие* будет передан в функцию **DLookup** в следующем виде:

```
"[Фамилия] = 'Королев'"
```

Заметьте, что вся строка, включая аргумент *условие* должна быть заключена в двойные кавычки.

Совет. При вводе сложных выражений в аргумент *условие* разбивайте выражение на компоненты и проверяйте каждый из них по отдельности в окне отладки. После этого объединяйте компоненты по одному, каждый раз проверяя правильность результирующего выражения.

Допускается также указание текстовых значений в аргументе *условие* с помощью переменных. Переменная должна быть выделена в строковом выражении таким образом, чтобы ее значение определялось отдельно, а затем объединялось с остальной частью строкового выражения. Текстовая строка должна быть заключена в одинарные или двойные кавычки

Следующая конструкция демонстрирует определение аргумента *условие* с помощью переменной, представляющей строковое значение:

```
Dim strLastName As String, varResult As Variant
strLastName = "Королев"
varResult = DLookup("[КодСотрудника]", "Сотрудники", "[Фамилия] = '" &
strLastName & "'")
```

Изменение условия значения даты и времени из элемента управления в форме

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"achowSpecifyDateTimeCriteriaControlFormC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"achowSpecifyDateTimeCriteriaControlFormX":1}

Если требуется предоставить пользователю возможность изменять аргумент *условие* статистической функции по подмножеству, то следует указать, что значение аргумента принимается из элемента управления в форме. Например, можно указать, что аргумент *условие* принимает значение из поля со списком, в котором выводится значение поля «ДатаРазмещения» из таблицы «Заказы».

Для ввода в условие отбора значения даты или времени из элемента управления в форме следует включить в аргумент *условие* выражение, содержащее ссылку на этот элемент управления. Это выражение должно быть выделено в строковом выражении таким образом, чтобы его значение определялось отдельно, а затем объединялось с остальной частью строкового выражения перед выполнением необходимой операции.

Кроме заключения полного строкового выражения в прямые кавычки (") необходимо окружать вводимое значение даты или времени символами #. Эти символы следует вводить в начале и в конце выражения, содержащего ссылку на элемент управления в форме.

Примечание. Символы # позволяют ввести в аргументе *условие* значение даты или времени внутри строки.

В следующем примере задается значение свойства формы **Фильтр (Filter)** на основе условия отбора из элемента управления «ДатаНайма». Обратите внимание на размещение символов #.

```
Forms!Сотрудники.Filter = "[ДатаНайма] >= #" & Forms!Сотрудники!ДатаНайма &
"#"
Forms!Сотрудники.FilterOn = True
```

Если для элемента управления «ДатаНайма» будет задано значение `5-1-92`, то свойству **Фильтр (Filter)** в качестве аргумента *условие* будет передана следующая строка:

```
"[ДатаНайма] >= #5-1-92#"
```

Совет. При вводе сложных выражений в аргумент *условие* разбивайте выражение на компоненты и проверяйте каждый из них по отдельности в окне отладки. После этого объединяйте компоненты по одному, каждый раз проверяя правильность результирующего выражения.

Допускается также указание значений даты или времени в аргументе *условие* с помощью переменных. Переменная должна быть выделена в строковом выражении таким образом, чтобы ее значение определялось отдельно, а затем объединялось с остальной частью строкового выражения. Значения даты или времени должны выделяться с помощью символов #.

Следующая конструкция демонстрирует определение аргумента *условие* с помощью переменной, представляющей значение даты:

```
Dim datHireDate As Date
datHireDate = #5-1-92#
Forms!Сотрудники.Filter = "[ДатаНайма] >= #" & _
   datHireDate & "#"
```

Указание нескольких полей в условиях отбора

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"achowSpecifyMultFieldsCriteriaC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"achowSpecifyMultFieldsCriteriaX":1}

Допускается определение аргумента *условие* статистической функции по подмножеству на основании значений нескольких полей.

Для указания нескольких полей в условиях отбора необходимо составить строковое выражение для каждого поля и правильно объединить эти выражения с тем, чтобы они образовали допустимое предложение WHERE в инструкции SQL. В предложение WHERE инструкции SQL, включающем несколько полей, выражения для отдельных полей объединяются с помощью ключевых слов: **AND**, **OR** или **NOT**. Результатом полного выражения должна быть строка, содержащее одно из этих ключевых слов.

Предположим, например, что необходимо установить значение для свойства **Фильтр (Filter)** формы «Сотрудники» для вывода записей, ограниченных двумя наборами условий. В следующем примере выбираются только записи сотрудников с должностью «Менеджер по продажам», нанятых на работу после 1 января 1993 г.

```
Dim datHireDate As Date, strTitle As String
datHireDate = #1/1/93#
strTitle = "Менеджер по продажам"
Forms!Сотрудники.Filter = "[ДатаНайма] >= #" & _
   datHireDate & "# AND [Должность] = '" & strTitle & "'"
Forms!Сотрудники.FilterOn = True
```

Аргумент *условие* будет передан в следующем виде:

```
"[ДатаНайма] >= #1-1-93# AND [Должность] = 'Менеджер по продажам'"
```

Совет. При вводе сложных выражений в аргумент *условие* разбивайте выражение на компоненты и проверяйте каждый из них по отдельности в окне отладки. После этого объединяйте компоненты по одному, каждый раз проверяя правильность результирующего выражения.

Использование кавычек в строковых выражениях

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"achowUsingQuotesInStringsC;damthFindFirst"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"achowUsingQuotesInStringsX":1}

При объединении строк часто возникает необходимость включить сроку или строковую переменную как часть другой строки. Обычно такие ситуации возникают при следующих действиях:

- указание условий отбора для статистических функций по подмножеству;.
- указание условий отбора в методах **Find;**
- указание условий отбора для свойства формы **Фильтр (Filter);**
- создание инструкций SQL.

В каждом из этих случаев Microsoft Access должен передать строку в ядро базы данных Microsoft Jet. Например, при указании аргумента *условие* для статистической функции по подмножеству требуется определить значение каждой переменной, объединить их в строку, а затем передать полную строку условий отбора в ядро базы данных Jet.

При включении в условия числовой переменной Microsoft Access просто определяет значение переменной и сливает его со строкой. Если переменная имеет строковое значение, то результирующая строка условий должна содержать вложенную строку. Строка, вложенная в другую строку, должна быть выделена с помощью ограничителей строковых значений. В противном случае ядро Jet не сможет распознать, какая часть переданной строки условий является подстрокой, которую следует использовать.

Ограничители строковых значений не являются частью строковой переменной и должны отдельно включаться в строку аргумента *условие*. Допускается использование как одинарных кавычек ('), так и прямых кавычек ("). Существуют три способа создания строки *условие*. Их результатом является строка одного из следующих типов.

```
"[Фамилия] = 'Гуськов'"
```

– или –

```
"[Фамилия] = "Гуськов""
```

Включение одинарных кавычек

Символы одинарных кавычек (апострофов) включаются в строку *условие* таким образом, чтобы строковое значение после объединения с полной строкой оказалось заключенным в одинарные кавычки. Предположим, например, что в аргумент *условие* требуется включить значение, представляемое строковой переменной `strФамилия`. В этом случае допускается указание аргумента *условие* следующим образом:

```
"[Фамилия] = '" & strФамилия & "'"
```

После определения значения переменной `strФамилия` и его объединения со строкой *условие* аргумент *условие* принимает вид:

```
"[Фамилия] = 'Гуськов'"
```

Примечание. Данный синтаксис не допускает присутствия апострофов (') в значении строковой переменной. Если в значении переменной содержится апостроф, возникает ошибка при выполнении. В этом случае следует использовать другой синтаксис, описанный ниже.

Включение прямых кавычек

Символы прямых (двойных) кавычек включаются в строку *условие* таким образом, чтобы строковое значения после объединения с полной строкой оказалось заключенным в прямые кавычки. Внутри строки следует включать два символа прямых кавычек для представления одного символа прямых кавычек. В этом случае указание аргумента *условие* образуется

следующим образом.

```
"[Фамилия] = """ & strФамилия & """"
```

После определения значения переменной `строкаФамилия` и его объединения со строкой *условие* два идущих подряд символа прямых кавычек заменяются одним, в результате чего аргумент *условие* принимает вид:

```
"[Фамилия] = 'Гуськов'"
```

Данный синтаксис является несколько более сложным, чем в предыдущем примере, но в нем допускается включение в аргумент *условие* строк, содержащих апострофы. Данный синтаксис обеспечивает также возможность создания строк, содержащих вложенные строки нескольких уровней.

Включение переменной, представляющей символы кавычек

Удобным приемом является создание строковой переменной, представляющей символ прямых кавычек, и использование этой переменной при создании строки условий отбора. В кодировке ANSI символ прямых кавычек представляется функцией Chr$(34); это значение можно присвоить переменной `strКавычки`. Определенная таким образом переменная включается в условие отбора следующим образом:

```
"[Фамилия] = " & strКавычки & strФамилия & strКавычки
```

После определения значений переменных и их объединения в строке *условие* аргумент принимает вид:

```
"[Фамилия] = "Гуськов""
```

Спецификации Microsoft Access

Предполагаемые действия

- Просмотр спецификаций баз данных
- Просмотр спецификаций таблиц
- Просмотр спецификаций запросов
- Просмотр спецификаций форм и отчетов
- Просмотр спецификаций макросов

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acdecLookUpMicrosoftAccessSpecificationsSW":1:"Foo":"Invisible"}

Спецификации баз данных

Атрибут	Максимальное число
Размер файла базы данных (.mdb)	1 Гбайт. Хотя база данных может включать присоединенные таблицы, ее общий размер ограничивается только доступным объемом на диске.
Число объектов в базе данных	32768
Модули (включая формы и отчеты для которых свойство **Наличие модуля (HasModule)** принимает значение **True**)	1024
Число символов в имени объекта	64
Число символов в пароле	14
Число символов в имени пользователя или группы	20
Число одновременно работающих пользователей	255

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acrefDatabaseSpecificationsSW":1:"Foo":"Invisible"}

Спецификации таблиц

Атрибут	Максимальное число
Число символов в имени таблицы	64
Число символов в имени поля	64
Количество полей в таблице	255
Количество открытых таблиц	1024. Реальное число может быть меньше, так как Microsoft Access открывает также внутренние таблицы.
Размер таблицы	1 Гбайт
Число символов в текстовом поле	255
Число символов в поле Memo	65535 при вводе данных через интерфейс пользователя; 1 Гбайт при вводе данных программным путем.
Размер объекта OLE	1 Гбайт
Количество индексов в таблице	32
Количество полей в индексе	10
Число символов в сообщении об ошибке в строке состояния	255
Число символов в выражении для условия на значение	2048
Число символов в описании таблицы или поля	255
Число символов в записи (не считая поля MEMO и поля объектов OLE)	2000
Число символов в выражении для значения свойства	255

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acrefTableSpecificationsSW":1:"Foo":"Invisible"}

Спецификации запросов

Атрибут	Максимальное число
Число наложенных связей	32 на таблицу минус число индексов, построенных по полям или комбинациям полей, не участвующих в сортировке.
Количество таблиц в запросе	32
Число полей в наборе	255
Размер набора записей	1 Гбайт
Предел сортировки	255 символов в одном или нескольких полях
Число уровней вложенных запросов	50
Число символов в ячейке в бланке запроса	1024
Число символов в параметрическом запросе	255
Количество операторов AND в предложениях WHERE или HAVING	40
Число символов в инструкции SQL	примерно 64000

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acrefQuerySpecificationsSW":1:"Foo":"Invisible"}

Спецификации форм и отчетов

Атрибут	Максимальное число
Число символов в надписи	2048
Число символов в поле	65535
Ширина формы или отчета	22 дюйма (55,87 см)
Высота раздела	22 дюйма (55,87 см)
Суммарная высота всех разделов плюс заголовки и примечания разделов (в режиме конструктора)	200 дюймов (508 см)
Число уровней вложенности форм или отчетов	3
Количество полей или выражений, используемых для сортировки или группировки в отчете.	10
Количество областей заголовков и примечаний в отчете	1 заголовок/примечание отчета; 1 верхний/нижний колонтитул; 10 заголовков/примечаний групп
Количество печатных страниц в отчете	65536
Количество элементов управления и разделов, которые могут быть добавлены в форму или отчет за время их существования	754

{ewc HLP95EN.DLL, DYNALINK, "пїSпїSпїSпїSпїS пїS Web пїSпїSпїS пїSпїSпїSпїSпїSпїS пїSпїSпїSпїSпїSпїSпїSпїSпїS":"acrefFormReportSpecificationsSW":1:"Foo":"Invisible"}

Спецификации макросов

Атрибут	Максимальное число
Количество макрокоманд в макросе	999
Число символов в условии	255
Число символов в комментарии	255
Число символов в аргументе макрокоманды	255

{ewc HLP95EN.DLL, DYNALINK, "пїЅпїЅпїЅпїЅпїЅ пїЅ Web пїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acrefMacroSpecificationsSW":1:"Foo":"Invisible"}

Функция DDE

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDDEFuncC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDDEFuncX":1}

Функция **DDE** позволяет открыть сеанс динамического обмена данными (DDE) с другим приложением, направить требование на прием данных из этого приложения и вывести полученные данные в элементе управления в форме или отчете.

Например, указание функции **DDE** в свойстве **Данные (ControlSource)** поля в форме позволяет вывести в этом поле содержимое конкретной ячейки электронной таблицы Microsoft Excel.

Синтаксис

DDE(*приложение, документ, раздел***)**

Функция **DDE** использует следующие аргументы.

Аргументы	Описание
приложение	Строковое выражение, которое определяет приложение, участвующее в сеансе DDE. Обычно, для приложений, работающих в среде Microsoft Windows, аргумент *приложение* задает имя файла .EXE (без расширения .EXE). Например, для открытия канала связи DDE с Microsoft Excel следует указать **"Excel"** в аргументе *приложение*.
документ	Строковое выражение, содержащее имя документа, принимаемое *приложением*. В аргументе *документ* обычно указывается документ или файл данных. За списком поддерживаемых приложением имен документов следует обращаться к документации данного приложения.
раздел	Строковое выражение, содержащее имя раздела данных, принимаемое *приложением*. За списком поддерживаемых приложением имен разделов данных следует обращаться к документации данного приложения.

Дополнительные сведения

При вызове функции **DDE** делается попытка открыть сеанс связи DDE с указанным *приложением*, передать соответствующее имя *документа* и требование на прием данных, указанных в аргументе *раздел*. При успешном выполнении функция **DDE** возвращает строку, содержащую затребованные данные.

В требовании на прием данных из Microsoft Excel аргумент *раздел* должен содержать идентификатор адреса ячейки (строки и столбца, например, `"R1C1"`) или имя диапазона ячеек. В следующем примере функция **DDE** передает требование на прием содержимого ячейки, находящейся в первой строке и первом столбце электронной таблицы Microsoft Excel. В ячейку **Данные (ControlSource)** в окне свойств поля вводится следующее выражение:

```
=DDE("Excel", "Лист1", "R1C1")
```

Функция **DDE** используется только для указания свойства **Данные (ControlSource)** поля, группы, флажка или поля со списком. Не допускается вызов функции **DDE** в инструкциях Visual Basic.

После ввода функции **DDE** элемент управления становится нередактируемым в режиме формы и предварительного просмотра. Например, после ввода функции **DDE** в поле это поле становится нередактируемым. Содержимое этого поля следует редактировать в другом приложении. Поскольку свойство **Данные (ControlSource)** становится в режимах формы и предварительного просмотра доступным только для чтения, все изменения должны вноситься

в элемент управления в режиме конструктора.

Максимальное количество одновременно открываемых сеансов DDE определяется настройками Windows, а также памятью и ресурсами компьютера. Если попытка открыть сеанс оказалась неудачной из-за того, что *приложение* не запущено, не распознается *документ* или превышено максимально допустимое число сеансов, функция **DDE** возвращает значение **Null.**

Примечание. В конфигурации другого приложения может быть указано, что запросы на открытие сеанса DDE игнорируются. В этом случае функция **DDE** возвращает **Null**. Аналогично, в настройках Microsoft Access можно указать игнорирование запросов на открытие сеансов из других приложений: для этого выберите в меню **Сервис** команду **Параметры**, выберите вкладку **Другие** и в группе **Операции DDE** установите флажок **Пропуск команд DDE**.

Совет. Чтобы работать с объектами другого приложения из Microsoft Access, следует использовать механизм программирования объектов.

В следующей таблице описано использование функции **DDE** с разными элементами управления.

Элемент управления	**Описание**
Поле	Аргумент *раздел* может представлять текст или числа. Если *раздел* представляет несколько элементов данных, например, имя диапазона, содержащего несколько ячеек электронной таблицы Microsoft Excel, то функция **DDE** возвращает содержимое первого элемента. Функция **DDE** позволяет вывести в поле содержимое ячейки электронной таблицы.
Поле со списком	Функция **DDE** заполняет список данными, указанными в аргументе *раздел*. Не допускается ввод данных в поле. Функция **DDE** позволяет вывести в поле со списком список значений, сохраняемых в электронной таблице Microsoft Excel.
Группа параметров	Свойство **Значение параметра (OptionValue)** каждого из переключателей в группе имеет числовое значение. Обычно первый переключатель имеет значение 1, второй - значение 2 и т.д. Числовое значение, возвращаемое функцией **DDE**, определяет, какой из переключателей будет выбран. Например, если функция **DDE** возвращает 2, будет включен второй переключатель. Если функция **DDE** возвращает значение, не соответствующее ни одному из свойств **Значение параметра (OptionValue)**, не будет выбран ни один из переключателей. Если *раздел* представляет несколько элементов данных, например, имя диапазона, содержащего несколько ячеек электронной таблицы Microsoft Excel, то функция **DDE** возвращает содержимое первого элемента.
Флажок	Если функция **DDE** возвращает 0, флажок будет снят. Если функция **DDE** возвращает любое ненулевое значение, например, 1 или –1, флажок будет установлен. Если *раздел* представляет несколько элементов данных, например, имя диапазона, содержащего несколько ячеек электронной таблицы Microsoft Excel, состояние флажка становится неопределенным.

Функция DDESend

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDDESendC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acstmDDESendX":1}

Функция **DDESend** позволяет открыть сеанс динамического обмена данными (DDE) с другим приложением и передать в это приложение данные из элемента управления в форме или отчете.

Например, функция **DDESend** вводится в ячейку свойства **Данные (ControlSource)** поля, чтобы передать данные из этого поля в ячейку электронной таблицы Microsoft Excel.

Синтаксис

DDESend(*приложение***,** *документ***,** *раздел***,** *данные***)**

Функция **DDESend** использует следующие аргументы.

Аргумент	Описание
приложение	Строковое выражение, которое определяет приложение, участвующее в сеансе DDE. Обычно, для приложений, работающих в среде Microsoft Windows, аргумент *приложение* задает имя файла .EXE (без расширения .EXE). Например, для открытия канала связи DDE с Microsoft Excel следует указать **"Excel"** в аргументе *приложение*.
документ	Строковое выражение, содержащее имя документа, принимаемое *приложением*. В аргументе *документ* обычно указывается документ или файл данных. За списком поддерживаемых приложением имен документов следует обращаться к документации данного приложения.
раздел	Строковое выражение, содержащее имя раздела данных, принимаемое *приложением*. За списком поддерживаемых приложением имен разделов данных следует обращаться к документации данного приложения.
данные	Строка или выражение, содержащие данные, передаваемые в *приложение*.

Дополнительные сведения

При вызове функции **DDESend** делается попытка открыть сеанс связи DDE с указанным *приложением*, передать соответствующее имя *документа* и указать *раздел*, который должен принять *данные*. Например, если аргумент *приложение* определяет Microsoft Excel, *документ* может иметь вид `"Лист1"`, а *раздел* представлять идентификатор адреса ячейки (строки и столбца, например, `"R1C1"`) или имя диапазона ячеек.

Аргумент *данные* определяет передаваемые данные. Этот аргумент может содержать строку, например, `"Отчет подготовил В.Сидоров"` или выражение, включающее функцию как часть результирующей строки, например, `"Отчет подготовлен " & Date()`. Если в аргументе *раздел* указывается несколько адресов элементов, например, имя диапазона ячеек электронной таблицы Microsoft Excel, функция **DDESend** передает *данные* в первый элемент.

В следующем примере функция **DDESend** передает строку `"Некий текст"` в первую ячейку электронной таблицы Microsoft Excel. Данное выражение может быть введено в ячейку **Данные (ControlSource)** в окне свойств поля в следующем виде:

```
=DDESend("Excel", "Лист1", "R1C1", "Некий текст")
```

Предположим теперь, что требуется передать данные из присоединенного элемента управления, находящегося в форме Microsoft Access, в ячейку электронной таблицы Microsoft

Excel. В свойстве **Данные (ControlSource)** связанного элемента управления уже указано имя поля или выражения. В этом случае следует создать еще одно поле или поле со списком, задать для него в свойстве **Данные** выражение, включающее функцию **DDESend**, и указать в аргументе *данные* имя связанного элемента управления. Например, если связанным является поле "Фамилия", то в ячейку **Данные** второго поля следует ввести такое выражение:

```
=DDESend("Excel", "Лист1", "R1C1", [Фамилия])
```

В качестве промежуточного элемента управления необходимо использовать поле или поле со списком. Не допускается указание имени связанного элемента управления в аргументе *данные* для флажка или группы переключателей.

Функция **DDESend** используется только для указания свойства **Данные (ControlSource)** поля, группы, флажка или поля со списком. Не допускается вызов функции **DDESend** в инструкциях Visual Basic.

После ввода функции **DDESend** элемент управления становится нередактируемым в режиме формы и предварительного просмотра. Поскольку свойство **Данные (ControlSource)** становится в режимах формы и предварительного просмотра доступным только для чтения, все изменения должны вноситься в элемент управления в режиме конструктора.

Максимальное количество одновременно открываемых сеансов DDE определяется настройками Windows, а также памятью и ресурсами компьютера. Если попытка открыть сеанс оказалась неудачной из-за того, что *приложение* не запущено, не распознается *документ* или превышено максимально допустимое число сеансов, функция **DDESend** возвращает значение **Null.**

Примечание. В конфигурации другого приложения может быть указано, что запросы на открытие сеанса DDE игнорируются. В этом случае функция **DDESend** возвращает **Null**. Аналогично, в настройках Microsoft Access можно указать игнорирование запросов на открытие сеансов из других приложений: для этого выберите в меню **Сервис** команду **Параметры**, выберите вкладку **Другие** и в группе **Операции DDE** установите флажок **Пропуск команд DDE**.

Совет. Для того чтобы работать с объектами другого приложения из Microsoft Access следует использовать механизм программирования объектов..

В следующей таблице описано использование функции **DDESend** с разными элементами управления.

Элемент управления	**Описание**
Поле или поле со списком	Так как в режимах формы и предварительного просмотра поле выводится пустым, для его свойства **Вывод на экран (Visible)** можно задать значение **False.** Аргумент *данные* содержит ссылку на другой элемент управления. В следующем примере демонстрируется передача содержимого поля «Фамилия» в электронную таблицу Microsoft Excel. `=DDESend("Excel", "Лист1", "R1C1", [Фамилия])`
Группа параметров	Ни один из переключателей в группе не является выбранным в режимах формы и предварительного просмотра, поэтому для свойства **Вывод на экран (Visible)** группы (и содержащихся в ней переключателей) можно задать значение **False**. Аргумент *данные* должен содержать числовое значение, например, `"2"`. Если значение аргумента *данные* не является числовым, функция **DDESend** не передает информацию и состояние принимающего *элемента* не изменяется.
Флажок	Состояние флажка в режиме формы и предварительного

просмотра является неопределенным, поэтому для его свойства **Вывод на экран (Visible)** можно задать значение **False**.

Аргумент *данные* должен содержать числовое значение, например, `"2"`. Если значение аргумента *данные* не является числовым, функция **DDESend** не передает информацию и состояние принимающего *элемента* не изменяется.

Использование Microsoft Access в роли сервера DDE

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"achowDDEUsingAsServerC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"achowDDEUsingAsServerX":1}

Microsoft Access поддерживает протокол динамического обмена данными (DDE, dynamic data exchange), выполняя роль как приложения, принимающего данные (клиента), так и источника данных (сервера). Например, база данных Microsoft Access может быть сервером для приложения Microsoft Word, выступающего в роли клиента, принимающего данные по каналу связи DDE.

Совет. Для того чтобы работать с объектами другого приложения из Microsoft Access, следует использовать механизм программирования объектов..

Сеанс связи DDE между клиентом и сервером открывается для конкретного документа. Документ может быть либо файлом данных в формате, поддерживаемом приложением-сервером, либо являться системным документом, содержащим информацию о самом приложении-сервере. Во время сеанса связи, открытого для конкретного документа, возможна передача данных только из раздела данных, связанного с этим документом.

Предположим, например, что выполняется приложение Microsoft Word для Windows, в документ которого требуется вставить данные из конкретной базы данных Microsoft Access. Сеанс связи DDE с Microsoft Access начинается с открытия канала связи DDE с помощью функции **DDEInitiate** и указания в качестве документа имени файла базы данных. После этого становится возможной передача данных в Microsoft Word по этому каналу.

В качестве сервера DDE Microsoft Access поддерживает следующие документы:

- документ System (системный);
- имя файла базы данных (документ *базаДанных*);
- имя таблицы (документ *имяТаблицы*);
- имя запроса (документ *имяЗапроса*);
- инструкция SQL Microsoft Access (документ *инструкцияSQL*).

После открытия сеанса обмена данными по протоколу DDE для передачи команд из приложения-клиента в приложение сервер используют инструкцию **DDEExecute.** В роли сервера DDE Microsoft Access принимает как допустимую любую из перечисленных ниже команд.

- Имя макроса в текущей базе данных.
- Любая макрокоманда, которая может быть выполнена в программе Visual Basic с помощью одного из методов объекта **DoCmd.**
- Макрокоманды **OpenDatabase** и **CloseDatabase**, которые используются только в инструкциях DDE. (Примеры использования этих макрокоманд приведены ниже в данном разделе).

Примечание. При указании имени макрокоманды в инструкции **DDEExecute** сама макрокоманда и все ее аргументы должны в соответствие с синтаксисом объекта **DoCmd** заключаться в прямые скобки ([ ]). Приложения, поддерживающие протокол DDE, не допускают использования внутренних констант в операциях DDE. Строковые аргументы должны заключаться в прямые кавычки (""), если в строке содержится запятая. В остальных случаях кавычки не являются обязательными.

Вызванная в приложении-клиенте функция **DDERequest** передает требование на пересылку текстовых данных с приложения-сервера по открытому каналу DDE. Для пересылки данных из приложения-клиента в приложение-сервер следует вызвать функцию **DDEPoke.** После того как передача данных закончится следует вызвать в приложении-клиенте функцию **DDETerminate** для закрытия текущего канала обмена данными DDE или функцию **DDETerminateAll** для закрытия всех открытых каналов.

Примечание. После того как в приложении-клиенте закончится прием данных по каналу DDE, необходимо закрыть данный канал в этом приложении для экономии ресурсов памяти.

В следующем примере демонстрируется создание макроса на языке Visual Basic приложения Word для Windows, использующего Microsoft Access в качестве сервера DDE. (Для выполнения данного примера необходимо предварительно запустить Microsoft Access).

```
Sub AccessDDE()
   Dim intChan1 As Integer, intChan2 As Integer
   Dim strQueryData As String

   ' Используем документ System и открываем базу данных Борей.mdb.
   ' Необходимо открыть базу данных перед использованием других документов
DDE.
   intChan1 = DDEInitiate("MSAccess", "System")
   ' Данный путь следует заменить на реальный путь
   ' к базе данных Борей.mdb.
   DDEExecute intChan1, "[OpenDatabase C:\Access\Samples\Борей.mdb]"

   ' Считываем все данные из запроса «Десять самых дорогих товаров».
   intChan2 = DDEInitiate("MSAccess", "Борей.mdb;" _
      & "QUERY Десять самых дорогих товаров")
   strQueryData = DDERequest(intChan2, "All")
   DDETerminate intChan2

   ' Закрываем базу данных.
   DDEExecute intChan1, "[CloseDatabase]"
   DDETerminate intChan1

   ' Печатаем полученные данные в окне отладки.
   Debug.Print strQueryData
End Sub
```

Более подробное описание использования Microsoft Access в качестве клиента DDE можно найти в главе 11 «Взаимодействие с другими приложениями» книги *Разработка приложений для Microsoft Access 97.*

Ниже приводится описание допустимых имен документов DDE, поддерживаемых Microsoft Access:

Документ System

System является стандартным именем документа для всех приложений, работающих в среде Microsoft Windows. Данное имя позволяет получать сведения о других именах документов, поддерживаемых приложением. Для того чтобы получить доступ к этой информации необходимо сначала вызвать функцию **DDEInitiate**, указав `"System"` в аргументе *документ*, после чего выполнить инструкции **DDERequest** с одним из следующих значений аргумента *раздел*.

Раздел	Результат
SysItems	Список имен разделов, входящих в документ System в Microsoft Access.
Formats	Список форматов, в которых Microsoft Access может копировать данные в буфер обмена.
Status	Состояние сеанса обмена: "Busy" (занято) или "Ready" (свободно).
Topics	Список всех открытых баз данных.

В следующем примере демонстрируется использование функций **DDEInitiate** и **DDERequest** с документом System.

```
' В программе Visual Basic открывается сеанс обмена DDE с Microsoft Access.
Dim intChan1 As Integer, strResults As String
intChan1 = DDEInitiate("MSAccess", "System")
' Запрашивается список документов, поддерживаемых документом System.
strResults = DDERequest(intChan1, "SysItems")
' Вызывается макрокоманда OpenDatabase, открывающая базу данных Борей.mdb.
' Данный путь следует заменить на реальный путь
' к базе данных Борей.mdb.
DDEExecute intChan1, "[OpenDatabase C:\Access\Samples\Борей.mdb]"
```

Документ *базаДанных*

Документ *базаДанных* представляет имя файла существующей базы данных. Допускается непосредственное указание имени базы данных (например, Борей) или указание имени с путем и расширением .mdb (C:\Msoffice\Access\Samples\Борей.mdb). После открытия сеанса обмена DDE с базой данных можно затребовать список всех объектов этой базы данных.

Примечание. Не допускается использование протокола DDE для запросов к файлу рабочей группы Microsoft Access.

Документ *базаДанных* поддерживает следующие имена разделов.

Раздел	Результат
TableList	Список таблиц.
QueryList	Список запросов.
FormList	Список форм.
ReportList	Список отчетов.
MacroList	Список макросов.
ModuleList	Список модулей.

В следующем примере демонстрируется открытие в программе Visual Basic формы «Сотрудники» из базы данных «Борей» (Борей.mdb).

```
' в программе Visual Basic открывается сеанс обмена DDE
' с базой данных Борей.
' База данных должна быть уже открыта.
intChan2 = DDEInitiate("MSAccess", "Борей")
' Запрашиваем список форм из базы данных Борей.mdb.
strResponse = DDERequest(intChan2, "FormList")
' Для открытия формы «Сотрудники» вызываем макрокоманду ОткрытьФорму
(OpenForm) с необходимыми аргументами.
DDEExecute intChan2, "[OpenForm Сотрудники,0,,,1,0]"
```

Документы TABLE *имяТаблицы*, QUERY *имяЗапроса* и SQL *инструкцияSQL*

Для данных документов требуется следующий синтаксис:

имяБазыДанных; **TABLE** *имяТаблицы*
имяБазыДанных; **QUERY** *имяЗапроса*
имяБазыДанных; **SQL [***инструкцияSQL***]**

Аргумент	Описание
имяБазыДанных	Имя базы данных, к которой принадлежат таблица или запрос, или к которой относится инструкция SQL. После имени базы данных необходимо помещать точку с запятой (;). Допускается непосредственное указание имени базы данных (например, Борей)

или указание имени с путем и расширением .mdb (C:\Msoffice\Access\Samples\Борей.mdb).

имяТаблицы Имя существующей таблицы.

ИмяЗапроса Имя существующего запроса.

ИнструкцияSQL Допустимая инструкция SQL длиной до 256 символов, заканчивающаяся точкой с запятой. Для создания инструкции SQL с длиной, превышающей 256 символов, следует опустить данный аргумент и взамен использовать несколько инструкций **DDEPoke**.

Например, в следующей программе Visual Basic инструкция SQL строится с помощью инструкций **DDEPoke**, после чего запрашиваются результаты данного запроса.

```
intChan1 = DDEInitiate("MSAccess", "Борей;SQL")
DDEPoke intChan1, "SQLText", "SELECT *"
DDEPoke intChan1, "SQLText", " FROM Заказы"
DDEPoke intChan1, "SQLText", " WHERE [Стоимость
доставки] > 100;"
strResponse = DDERequest(intChan1, "NextRow")
DDETerminate intChan1
```

В следующей таблице перечислены допустимые имена разделов для документов TABLE *имяТаблицы*, QUERY *имяЗапроса* и SQL *инструкцияSQL.*

Раздел	**Результат**
All	Все данные из таблицы, включая имена полей.
Data	Содержимое строк данных без имен полей.
FieldNames	Одна строка с именами полей.
FieldNames;T	Список из двух строк, содержащий имена полей (первая строка) и их типы данных (вторая строка). Ниже приводится список возвращаемых значений и соответствующие им типы данных:.

Значение	**Тип данных**
0	Неверный тип данных.
1	**True**/**False** (не пустое **Null** значение)
2	Однобайтовое целое без знака.
3	Двухбайтовое целое со знаком (**Integer**).
4	Четырехбайтовое целое со знаком (**Long**).
5	Восьмибайтовое число со знаком (**Currency**).
6	Четырехбайтовое число с плавающей точкой (**Single**).
7	Восьмибайтовое число с плавающей точкой двойной точности (**Double**)
8	Дата/время.
9	Двоичные данные, не более 256 байт.
10	Текст в кодировке ANSI без учета регистра символов, не более 256 байт(**Text**).
11	Длинное двоичное значение (объект OLE).
12	Длинное текстовое значение (тип Memo).

NextRow	Данные из следующей записи таблицы или запроса. При открытии канала NextRow возвращает содержимое первой записи. Если текущая запись является последней, вызов

	NextRow приводит к ошибке.
PrevRow	Данные из предыдущей записи таблицы или запроса. Если PrevRow является первым вызовом на новом канале, возвращается содержимое последней записи таблицы или запроса. Если текущей является первая запись, вызов PrevRow приводит к ошибке.
FirstRow	Данные из первой записи таблицы или запроса.
LastRow	Данные из последней записи таблицы или запроса.
FieldCount	Число полей в таблице или запросе.
SQLText	Инструкция SQL, представляющая таблицу или запрос. Для таблиц данный аргумент возвращает инструкцию SQL вида "SELECT * FROM *имяТаблицы*;".

SQLText;*n*	Инструкция SQL, представляющая таблицу или запрос, разбитая на части по *n* символов, где *n* является целым числом, не превышающим 256. Предположим, например, что запрос представляется следующей инструкцией SQL:
	`"SELECT * FROM Заказы;"`
	Если указать раздел в виде "SQLText;7", то возвращаются следующие части инструкции, разделяемые символами табуляции:
	`"SELECT "`
	`"* FROM "`
	`"Заказы;"`

В следующем примере демонстрируется использование механизма DDE в программе Visual Basic для получения данных из таблицы, содержащихся в учебной базе данных «Борей», и вставки этих данных в текстовый файл.

```
Sub NorthwindDDE
   Dim intChan1 As Integer, intChan2 As Integer, intChan3 As Integer
   Dim strResp1 As Variant, strResp2 As Variant, strResp3 As Variant

   ' В модуле Visual Basic запрашиваем данные из таблицы «Типы»,
   ' запроса «Каталог» и таблицы «Заказы» в базе данных Борей.mdb.
   ' База данных должна быть уже открыта.
   intChan1 = DDEInitiate("MSAccess", "Борей;TABLE Типы")
   intChan2 = DDEInitiate("MSAccess", "Борей;QUERY Каталог")
   intChan3 = DDEInitiate("MSAccess", "Борей;SQL SELECT * " _
      & "FROM Заказы " _
      & "WHERE КодЗаказа > 10050;")

   strResp1 = DDERequest(intChan1, "All")
   strResp2 = DDERequest(intChan2, "FieldNames;T")
   strResp3 = DDERequest(intChan3, "FieldNames;T")
   DDETerminate intChan1
   DDETerminate intChan2
   DDETerminate intChan3

   ' Вставляем данные в текстовый файл.
   Open "C:\Данные.TXT" For Append As #1
   Print #1, strResp1
   Print #1, strResp2
   Print #1, strResp3
   Close #1
```

```
End Sub
```

Функции и инструкции DDE

Инструкция DDEExecute

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acstmDDEExecuteC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acstmDDEExecuteX":1}

Инструкция **DDEExecute** передает по открытому каналу динамического обмена данными (DDE) команду из приложения-клиента в приложение-сервер.

Предположим, например, что канал DDE открыт в Microsoft Access для передачи текста из электронной таблицы Microsoft Excel в базу данных Microsoft Access. Инструкция **DDEExecute** позволяет передать в Microsoft Excel команду **New (Создать)** для создания новой электронной таблицы. В этом случае Microsoft Access является приложением-клиентом, а Microsoft Excel приложением-сервером.

Синтаксис

DDEExecute *канал*, *команда*

Инструкция **DDEExecute** использует следующие аргументы.

Аргумент	Описание
канал	Номер канала, представленный длинным целым числом, который возвращается функцией **DDEInitiate.**
команда	Строковое выражение, задающее команду, принимаемую приложением-сервером. Список допустимых команд см. в документации приложения-сервера.

Дополнительные сведения

Допустимые значения аргумента *команда* зависят от выбранного приложения и имени документа, указанного при открытии данного *канала*. Выполнение данной инструкции приводит к ошибке, если значение аргумента *канал* не является целым числом, соответствующим открытому каналу, или если указанная команда не может быть выполнена в другом приложении.

В программах Visual Basic инструкцию **DDEExecute** можно использовать только для передачи команд в другие приложения. Сведения о передаче команд из других приложений в Microsoft Access приведены в разделе Использование Microsoft Access в роли сервера DDE.

Совет. Для того чтобы работать с объектами другого приложения из Microsoft Access, следует использовать механизм программирования объектов.

Пример динамического обмена данными (DDE)

Данная программа открывает сеанс обмена по протоколу DDE с Microsoft Excel, помещает значения в ячейки первой строки новой электронной таблицы и создает диаграмму по этим значениям. Вначале инструкция **DDEInitiate** открывает канал связи DDE. Затем в инструкции **DDEExecute** в Microsoft Excel передается команда открытия новой электронной таблицы. В инструкции **DDERequest** в Microsoft Excel запрашивается имя созданной электронной таблицы. После этого открывается новый канал и инструкция **DDEPoke** передает в Microsoft Excel данные, по которым строится диаграмма. И наконец, инструкция **DDETerminate** закрывает канал связи DDE с Microsoft Excel, а инструкция **DDETerminateAll** закрывает все активные каналы DDE.

```
Sub ExcelDDE()
   Dim intI As Integer, intChan1 As Integer
   Dim strTopics As String, strResp As String, strSheetName As String

   On Error Resume Next                          ' Обработка ошибок.

   intChan1 = DDEInitiate("Excel", "System")     ' Открытие сеанса обмена.
   If Err Then                                   ' При возникновении ошибки
      Err = 0                                    ' электронная таблица не
запускается. Сброс ошибки
      Shell "C:\Excel\Excel.exe", 1              ' и запуск.
      If Err Then Exit Sub                       ' Выход при повторной
ошибке.
      ' Открытие сеанса обмена.
      intChan1 = DDEInitiate("Excel", "System")
   End If

   ' Создание новой электронной таблицы.
   DDEExecute intChan1, "[New(1)]"
   ' Считываем список имен документов, имя таблицы.
   strTopics = DDERequest(intChan1, "Selection")
   strSheetName = Left(strTopics, InStr(1, strTopics, "!") - 1)
   ' Закрываем канал DDE.
   DDETerminate intChan1
   ' Открываем канал обмена с новой электронной таблицей.
   intChan1 = DDEInitiate("Excel", strSheetName)
   For intI = 1 To 10                            ' Заносим значения.
      DDEPoke intChan1, "R1C" & intI, intI       ' в первую строку.
   Next intI
   ' Строим диаграмму.
   DDEExecute intChan1, "[Select(""R1C1:R1C10"")][New(2,2)]"
   ' Закрываем все каналы обмена.
   DDETerminateAll
End Sub
```

Функция DDEInitiate

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDDEInitiateC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDDEInitiateX":1}

Функция **DDEInitiate** позволяет открыть сеанс динамического обмена данными (DDE) с другим приложением. Функция **DDEInitiate** открывает канал связи DDE, обеспечивающий передачу данных между сервером DDE и приложением-клиентом.

Например, для передачи данных из электронной таблицы Microsoft Excel в базу данных Microsoft Access следует открыть канал связи между двумя приложениями с помощью функции **DDEInitiate**. В этом случае Microsoft Access будет выполнять роль приложения-клиента, а Excel приложения-сервера.

Синтаксис

DDEInitiate(*приложение***,** *документ***)**

Функция **DDEInitiate** использует следующие аргументы.

Аргументы	Описание
приложение	Строковое выражение, которое определяет приложение, участвующее в сеансе DDE. Обычно, для приложений, работающих в среде Microsoft Windows, таких как Microsoft Excel, аргумент *приложение* задает имя файла .exe (без расширения .exe).
документ	Строковое выражение, содержащее имя документа, принимаемое *приложением*. За списком поддерживаемых приложением имен документов следует обращаться к документации данного приложения.

Дополнительные сведения

При успешном выполнении функция **DDEInitiate** открывает сеанс связи DDE с указанными *приложением* и *документом* и возвращает значение типа **Long.** Данное число представляет уникальный номер канала, определяющий канал, по которому будет проводиться обмен данными. Этот номер канала будет использоваться в аргументах других функций и инструкций DDE.

Если приложение еще не запущено, а также если запущенное приложение не принимает аргумент *документ* или не поддерживает протокол DDE, функция **DDEInitiate** возвращает ошибку при выполнении.

Допустимые значения аргумента *документ* определяются *приложением*. В приложениях, использующих документы или файлы данных, допустимыми значениями данного аргумента обычно являются имена этих файлов.

Примечание. Максимально возможное число одновременно открытых каналов определяется настройками Windows, а также системной памятью и ресурсами компьютера. Если канал не используется, то для экономии ресурсов следует закрыть его с помощью инструкций **DDETerminate** или **DDETerminateAll.**

Совет. Для того чтобы работать с объектами другого приложения из Microsoft Access, следует использовать механизм программирования объектов.

Инструкция DDEPoke

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acstmDDEPokeC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acstmDDEPokeX":1}

Инструкция **DDEPoke** позволяет передать текст из приложения-клиента в приложение-сервер по открытому каналу динамического обмена данными (DDE).

Например, если открыт канал связи DDE между Microsoft Access и Microsoft Excel, инструкция **DDEPoke** позволяет передать текст из базы данных Microsoft Access в электронную таблицу Microsoft Excel. В этом случае Microsoft Access будет выполнять роль приложения-клиента, а Excel приложения-сервера.

Синтаксис

DDEPoke *канал*, *раздел*, *данные*

Инструкция **DDEPoke** использует следующие аргументы.

Аргумент	Описание
канал	Номер канала, целое значение, возвращаемое функцией **DDEInitiate.**
раздел	Строковое выражение, содержащее имя раздела данных, принимаемое приложением, заданным в функции **DDEInitiate**. За списком поддерживаемых приложением имен разделов данных следует обращаться к документации данного приложения.
данные	Строка, содержащая данные, передаваемые в приложение.

Дополнительные сведения

Допустимые значения аргумента *раздел* определяются именами приложения и документа, указанными при открытии *канала*. Например, *разделом* может быть диапазон ячеек электронной таблицы Microsoft Excel.

Строка *данные* должна передаваться в текстовом формате. Другие форматы при передаче данных не поддерживаются. Например, в аргументе *данные* можно передать число для заполнения указанного диапазона электронной таблицы Excel.

Аргумент *канал* должен представлять целое число, совпадающее с номером открытого канала. При указании другого значения, а также если приложение-получатель не распознает или не принимает указанные данные, возникает ошибка при выполнении.

Совет. Для того чтобы работать с объектами другого приложения из Microsoft Access, следует использовать механизм программирования объектов.

Функция DDERequest

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDDERequestC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDDERequestX":1}

Функция **DDERequest** передает в приложение-сервер по открытому каналу динамического обмена данными (DDE) требование на прием данных из указанного раздела.

Например, если открыт канал связи DDE между Microsoft Access и Microsoft Excel, функция **DDERequest** позволяет передать текст из электронной таблицы Microsoft Excel в базу данных Microsoft Access. В этом случае Microsoft Access будет выполнять роль приложения-клиента, а Excel приложения-сервера.

Синтаксис

DDERequest(*канал***,** *раздел***)**

Функция **DDERequest** использует следующие аргументы.

Аргумент	**Описание**
канал	Номер канала. Целое значение, возвращаемое функцией **DDEInitiate.**
раздел	Строковое выражение, содержащее имя раздела данных, принимаемое приложением, определенным при вызове функции **DDEInitiate**. За списком поддерживаемых приложением имен разделов данных следует обращаться к документации данного приложения.

Дополнительные сведения

Аргумент *канал* задает номер используемого канала связи DDE, а аргумент *раздел* определяет данные, загружаемые из приложения сервера. Допустимые значения аргумента *раздел* определяются именами приложения и документа, указанными при открытии *канала*. Например, разделом может быть диапазон ячеек электронной таблицы Microsoft Excel.

При успешном выполнении функция **DDERequest** возвращает значение типа **Variant,** в виде строки, содержащей затребованные данные.

Допускается прием данных только в обычном текстовом формате. Прием рисунков или текста в другом формате не поддерживается.

Аргумент *канал* должен представлять целое число, совпадающее с номером открытого канала. При указании другого значения, а также при невозможности передачи затребованных данных, возникает ошибка при выполнении.

Совет. Для того чтобы работать с объектами другого приложения из Microsoft Access, следует использовать программирования объектов.

Инструкция DDETerminate

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acstmDDETerminateC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acstmDDETerminateX":1}

Инструкция **DDETerminate** закрывает указанный канал динамического обмена данными (DDE).

Например, если открыт канал связи DDE между Microsoft Access и Microsoft Excel, инструкция **DDETerminate** позволяет закрыть канал после завершения передачи данных.

Синтаксис

DDETerminate *канал*

Дополнительные сведения

Аргумент *канал* задает номер канала. Значение номера канала возвращается функцией **DDEInitiate** при открытии канала. Если указанное в аргументе *канал* значение не представляет номер открытого канала, возникает ошибка при выполнении.

После закрытия канала вызов любой функции или инструкции DDE с номером этого канала приводит к ошибке при выполнении.

Вызов инструкции **DDETerminate** не оказывает влияния на активизированные связи DDE, задаваемые с помощью выражений для полей в формах или отчетах.

Совет. Для того чтобы работать с объектами другого приложения из Microsoft Access, следует использовать механизм программирования объектов.

Инструкция DDETerminateAll

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acstmDDETerminateAllC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acstmDDETerminateAllX":1}

Инструкция **DDETerminateAll** закрывает все открытые каналы динамического обмена данными (DDE).

Например, если были открыты два канала связи DDE между Microsoft Excel и Microsoft Access, один для загрузки системной информации о Microsoft Excel и второй для передачи данных, то инструкция **DDETerminateAll** позволяет закрыть оба канала одновременно.

Синтаксис

DDETerminateAll

Дополнительные сведения

Вызов инструкции **DDETerminateAll** эквивалентен выполнению инструкции **DDETerminate** для каждого открытого канала. Аналогично инструкции **DDETerminate**, инструкция **DDETerminateAll** не оказывает влияния на активизированные связи DDE, задаваемые с помощью выражений для полей в формах или отчетах.

Если открытых каналов связи DDE нет, вызов **DDETerminateAll** приводит к ошибке при выполнении.

Советы

- При прерывании процедуры, осуществляющей обмен по протоколу DDE, некоторые каналы могу случайно оказаться открытыми. Для экономии системных ресурсов рекомендуется вызывать инструкцию **DDETerminateAll** в программе или из панели проверки окна отладки во время отладки программ, в которых выполняются операции DDE.
- Для того чтобы работать с объектами другого приложения из Microsoft Access, следует использовать механизм программирования объектов.

Статистические функции по подмножеству

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acidxDomainRefC"}

Статистические функции возвращают информацию о наборах (подмножестве) записей. Например, с помощью статистических функций подсчитывают число записей в конкретном наборе или определят среднее по значениям конкретного поля.

Существуют статистические функции двух типов, статистические функции по подмножеству и статистические функции SQL, выполняющие одни и те же действия, но использующиеся в различных ситуациях. Статистические функции SQL могут быть включены в инструкции SQL, но не вызываются в инструкциях Visual Basic. Статистические функции по подмножеству, напротив, допускают вызов непосредственно в инструкциях Visual Basic. Кроме того, они могут быть включены в инструкции SQL, однако, использование статистических функций SQL обычно оказывается более эффективным.

При выполнении расчетов в программах необходимо использовать статистические функции по подмножеству. Кроме того, статистические функции по подмножеству используют в выражениях в запросах для указания условий отбора, обновления значений или создания вычисляемых полей. В вычисляемых элементах управления форм или отчетов допускается использование как статистических функций SQL, так и статистических функции по подмножеству.

Существуют следующие статистические функции по подмножеству:

Функция **DAvg**

Функция **DCount**

Функция **DLookup**

Функции **DFirst**, **DLast**

Функции **DMin**, **DMax**

Функции **DStDev**, **DStDevP**

Функция **DSum**

Функции **DVar**, **DVarP**

Функция DAvg

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDAvgC;dasqlAvg":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDAvgX":1}

Функция **DAvg** возвращает среднее для набора значений, принадлежащих указанному набору (подмножеству) записей. Функцию **DAvg** используют в макросах или в программах на Visual Basic, в выражениях в запросах, а также для определения вычисляемого элемента управления.

Например, включение функции **DAvg** в строку условий запроса на выборку позволяет отобрать записи, в которых значения конкретного поля превышает среднее для данного поля. В вычисляемом поле функция **DAvg** позволяет вывести рядом со значением величины нового заказа среднее по предыдущим заказам.

Синтаксис

DAvg(*выражение*, *набор*[, *условие*]**)**

Функция **DAvg** использует следующие аргументы.

Аргументы	Описание
выражение	Выражение, определяющее поле, которое содержит усредняемые данные. Данный аргумент может задаваться строковым выражением, определяющим поле в таблице или запросе, или представлять выражение, задающее выполнение вычислений над данными, содержащимися в поле. Допускается использование в аргументе *выражение* имени поля в таблице или элемента управления в форме, константы, а также встроенной или определяемой пользователем функции. Не допускается использование в аргументе *выражение* других статистических функций по подмножеству или статистических функций SQL.
набор	Строковое выражение, определяющее набор записей, образующих подмножество. Может представлять имя таблицы или запроса.
условие	Необязательное строковое выражение, ограничивающее диапазон данных, для которых рассчитывается среднее значение. Например, аргумент *условие* часто является эквивалентом предложения WHERE инструкции SQL, но без ключевого слова WHERE. Если аргумент *условие* опущен, **DAvg** выполняет расчеты над полем, заданным в аргументе *выражение,* для всего набора записей. Любое поле, указанное в аргументе *условие*, должно принадлежать подмножеству, заданному аргументом *набор*; в противном случае функция **DAvg** возвращает значение **Null.**

Дополнительные сведения

Записи, содержащие в указанном поле пустые (**Null**) значения, в расчет среднего значения не включаются.

При любом использовании функции **DAvg** в макросе или в модуле, в выражении в запросе или в вычисляемом элементе управления необходимо обеспечить правильное составление аргумента *условие*.

Допускается ввод функции **DAvg** в строку «Условие отбора» бланка запроса. Например, предположим, что требуется просмотреть список всех товаров, заказы на которые превышают среднее значение. В этом случае следует создать запрос по таблицам «Заказы», «Заказано» и «Товары», включить в бланк запроса поля «Марка» и «Количество» и ввести следующее

выражение в ячейку строки **Условие отбора** под полем «Количество».

```
>DAvg("[Количество]", "Заказано")
```

Функция **DAvg** может быть включена в выражение для вычисляемого поля в запросе, а также в строку **Обновление** запроса на обновление.

Примечание. В выражениях для вычисляемых полей в итоговых запросах используют как функцию **Davg**, так и функцию **Avg**. При использовании функции **DAvg** расчеты средних значение выполнятся до группировки данных. При использовании функции **Avg** расчеты средних значение выполнятся после группировки данных.

Функцию **DAvg** используют в вычисляемом элементе управления, если необходимо указать условия отбора, ограничивающие диапазон усредняемых данных. Например, для вывода средней стоимости доставки заказов в Крым, следует ввести в ячейку свойства **Данные** (**ControlSource**) вычисляемого поля следующее выражение.

```
=DAvg("[СтоимостьДоставки]", "Заказы", "[ОбластьПолучателя] = 'Крым'")
```

Если требуется просто найти среднее для всех записей в наборе, определяемом аргументом *набор*, следует использовать функцию **Avg**.

Допускается использование функции **DAvg** в модуле или макросе или в вычисляемом элементе управления в форме в случае, когда поле, для которого проводится усреднение, не принадлежит к базовому источнику записей формы. Например, предположим, что базовой таблицей формы является таблица «Заказы», и что требуется вывести в форме для сравнения среднее значение поля «Количество» из таблицы «Заказано» для заказов конкретного клиента. Функция **DAvg** позволяет выполнить такие расчеты и вывести результаты в форме.

Советы

- При использовании функции **DAvg** в вычисляемом поле это поле обычно помещают в область заголовка или примечаний формы, чтобы не пересчитывать поле при каждом переходе к новой записи.
- Если поле, для которого построено *выражение*, имеет числовой тип данных, то функция **DAvg** возвращает значение типа **Double**. Для повышения производительности рекомендуется применять функцию преобразования типа данных при определении вычисляемого поля с помощью функции **DAvg**.
- Хотя функция **DAvg** позволяет определять среднее для значений полей во внешней таблице, обычно более эффективным приемом оказывается создание запроса, содержащего все необходимые поля, с последующим созданием формы или отчета на базе этого запроса.

Примечание. Несохраненные изменения записей, входящих в *набор*, не учитываются данной функцией. Если требуется обеспечить учет в функции **DAvg** измененных значений, необходимо сначала сохранить изменения с помощью команды **Сохранить запись** из меню **Записи**, перевести фокус на другу запись или вызвать метод **Update.**

Функция DAvg, примеры

В данном примере возвращается средняя стоимость доставки для заказов, отправленных в Литву после 1 января 1996 г. Аргумент *набор* определяет таблицу «Заказы». Аргумент *условие* ограничивает подмножество записей теми, которые имеют в поле «СтранаПолучателя» значение «Литва», а в поле «ДатаИсполнения» значения до 1.01.96. Отметим, что условия для разных полей в строке *условие* объединятся ключевым словом **AND**. Функция **DAvg** выполняет расчеты над записями, удовлетворяющими обоим условиям одновременно.

```
Dim dblX As Double
dblX = DAvg("[СтоимостьДоставки]", "Заказы", _
   "[СтранаПолучателя] = 'Литва' AND [ДатаИсполнения] >= #1-1-96#")
```

В следующем примере при расчете среднего значения условия отбора задаются с помощью переменной `strCountry`. Отметим использование одинарных кавычек (') для выделения подстроки в строке условий. Здесь условия для разных полей соединятся ключевым словом **OR**. В расчетах будут отобраны все записи, удовлетворяющие одному из условий или обоим.

```
Dim dblX As Double, strCountry As String
strCountry = "Литва"
dblX = DAvg("[СтоимостьДоставки]", "Заказы", _
   "[СтранаПолучателя] = '" & strCountry & "'OR [ДатаИсполнения] >=
#1.01.96#")
```

Функция DCount

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDCountC;dasqlCount":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDCountX":1}

Функция **DCount** возвращает число записей в указанном наборе (подмножестве) записей. Функцию **DCount** используют в программах на Visual Basic, в макросах, в выражениях в запросах, а также для определения вычисляемого элемента управления.

Например, функция **DCount** в модуле позволяет подсчитать число записей в таблице «Заказы», соответствующих заказам, сделанным в определенный день.

Синтаксис

DCount(*выражение*, *набор*[, *условие*]**)**

Функция **DCount** использует следующие аргументы.

Аргументы	Описание
выражение	Выражение, определяющее поле, для которого производится подсчет значений. Данный аргумент может задаваться строковым выражением, определяющим поле в таблице или запросе, или представлять выражение, задающее выполнение вычислений над данными, содержащимися в поле. Допускается использование в аргументе выражение имени поля в таблице или элемента управления в форме, константы, а также встроенной или определяемой пользователем функции. Не допускается использование в аргументе выражение других статистических функций по подмножеству или статистических функций SQL.
набор	Строковое выражение, определяющее набор записей, образующих подмножество. Может представлять имя таблицы или запроса.
условие	Необязательное строковое выражение, ограничивающее диапазон данных, для которых подсчитывается число значений. Например, аргумент *условие* часто является эквивалентом предложения WHERE инструкции SQL, но без ключевого слова WHERE. Если аргумент *условие* опущен, **DCount** выполняет расчеты над полем, заданным в аргументе *выражение,* для всего набора записей. Любое поле, указанное в аргументе *условие*, должно принадлежать подмножеству, заданному аргументом *набор*; в противном случае функция **DCount** возвращает значение **Null.**

Дополнительные сведения

Функцию **DCount** используют для подсчета количества записей в подмножестве, когда не требуется использовать конкретные значения. Хотя в аргументе *выражение* можно указать любые расчеты, **DCount** всегда возвращает число записей. Значение аргумента *выражение* является недоступным.

При любом использовании функции **DCount** в макросе или в модуле, в выражении в запросе или в вычисляемом элементе управления необходимо обеспечить правильное составление аргумента *условие*.

Функцию **DCount** используют в вычисляемом элементе управления, если необходимо указать условия отбора, ограничивающие диапазон подсчитываемых записей. Например, для вывода числа заказов, отправляемых в Крым, следует ввести в ячейку свойства **Данные** (**ControlSource**) вычисляемого поля следующее выражение.

```
= DCount("[КодЗаказа]", "Заказы", "[ОбластьПолучателя] = 'Крым'")
```

Если требуется просто подсчитать число записей в наборе, определяемом аргументом *набор*, следует использовать функцию **Count.**

Совет. Функция **Count** оптимизирована для быстрого подсчета числа записей в запросах. В запросах вместо функции **DCount** лучше использовать функцию **Count** и задавать условия отбора в бланке запроса. Функцию **DCount** рекомендуется использовать для подсчета записей в программах модулей, в макросах или в вычисляемых полях.

Допускается использование функции **DCount** для подсчета записей, содержащих конкретное поле, в случае, когда поле, для которого проводится подсчет значений, не принадлежит к базовому источнику записей формы. Например, в форме, у которой базовой таблицей является таблица «Товары», можно вывести число заказов, содержащихся в таблице «Заказы».

Функция **DCount** не подсчитывает записи, содержащей пустые (**Null**) значения в поле, задаваемом аргументом *выражение*, за исключением случая, когда в аргументе *выражение* задан подстановочный знак звездочка (*). При указании данного подстановочного знака **DCount** подсчитывает полное число записей, в том числе содержащих значения типа **Null**. В следующей конструкции подсчитывается полное число записей в таблице «Заказы».

```
intX = DCount("*", "Заказы")
```

Если в аргументе *набор* указана таблица с определенным ключевым полем, то для подсчета всех записей в таблице достаточно указать в аргументе *выражение* ключевое поле, поскольку ключевое поле не может содержать значения типа **Null**.

Если в аргументе *выражение* задаются несколько полей, то их имена следует объединять с помощью оператора конкатенации, представляемого символом амперсанд (&) или знаком плюс (+). Если имена полей разделятся амперсандом (&), **DCount** подсчитывает все записи в перечисленных полях. Если используется знак плюс (+), то **DCount** возвращает число записей во всех перечисленных полях с непустыми значениями. В следующем примере демонстрируются результаты применение этих операторов для поля, содержащего данные во всех записях («НазваниеПолучателя»), и поля, содержащего пустые значения («ОбластьПолучателя»):

```
intW = DCount("[НазваниеПолучателя]", "Заказы")        ' Возвращает 831.
intX = DCount("[ОбластьПолучателя]", "Заказы")              ' Возвращает
323.
intY = DCount("[НазваниеПолучателя] + [ОбластьПолучателя]", "Заказы")    '
Возвращает 323.
intZ = DCount("[НазваниеПолучателя] & [ОбластьПолучателя]", "Заказы")    '
Возвращает 831.
```

Примечание. При объединении строк рекомендуется использовать оператор амперсанд. Старайтесь использовать оператор плюс только в числовых выражения. В строковые выражения включайте его только для явного указания возможности использования пустых значений.

Несохраненные изменения записей, входящих в *набор*, не учитываются данной функцией. Если требуется обеспечить учет в функции **DCount** измененных значений, необходимо сначала сохранить изменения с помощью команды **Сохранить запись** из меню **Записи**, перевести фокус на другу запись или вызвать метод **Update.**

Функция DCount, примеры

В данном примере возвращается количество заказов, отправленных в Литву до 6 июня 1995 г. Аргумент *набор* определяет таблицу «Заказы». Аргумент *условие* ограничивает подмножество записей теми, которые имеют в поле «СтранаПолучателя» значение «Литва», а в поле «ДатаИсполнения» значения до 6.06.95.

```
intX = DCount("[ДатаИсполнения]", "Заказы", _
   "[СтранаПолучателя] = 'Литва' AND [ДатаИсполнения] > #6.06.95#")
```

В следующем примере аргумент *условие* включает текущее значение поля со списком «СтранаПолучателя», в котором выводятся значения поля «СтранаПолучателя» из таблицы «Заказы». Обратите внимание, что ссылка на элемент управления не заключается в кавычки, отмечающие строковые значения внутри строки условий. Это обеспечивает определение текущего значения элемента управления при каждом вызове функции **DCount**.

```
intX = DCount("[ДатаИсполнения]", "Заказы", "[СтранаПолучателя] = '" _
   & Forms!Заказы![СтранаПолучателя] & "'AND [ДатаИсполнения] > #6.06.95#")
```

Функции DFirst, DLast

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDFirstLastC;dasqlFirstLast":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDFirstLastX":1}

Функции **DFirst** и **DLast** используют для возвращения значений из случайно выбранных записей определенного поля в таблице или запросе. Эти функции применяются в макросах, модулях, выражениях в запросах, или в вычисляемых элементах управления формы или отчета.

Синтаксис

DFirst(*выражение*, *набор*[, *условие*]**)**
DLast(*выражение*, *набор*[, *условие*]**)**

Функции **DFirst** и **DLast** используют следующие аргументы.

Аргументы	**Описание**
выражение	Выражение, определяющее поле, в котором производится поиск первого или последнего значения. Данный аргумент может задаваться строковым выражением, определяющим поле в таблице или запросе, или представлять выражение, задающее выполнение вычислений над данными, содержащимися в поле. Допускается использование в аргументе выражение имени поля в таблице или элемента управления в форме, константы, а также встроенной или определяемой пользователем функции. Не допускается использование в аргументе выражение других статистических функций по подмножеству или статистических функций SQL.
набор	Строковое выражение, определяющее набор записей, образующих подмножество.
условие	Необязательное строковое выражение, ограничивающее диапазон данных, для которых определяется значение первого или последнего поля. Например, аргумент *условие* часто является эквивалентом предложения WHERE инструкции SQL, но без ключевого слова WHERE. Если аргумент условие опущен, **DFirst** и **DLast** выполняют действия над полем, заданным в аргументе *выражение*, для всего набора записей. Любое поле, указанное в аргументе условие, должно принадлежать подмножеству, заданному аргументом набор; в противном случае функции **Dfirst** и **DLast** возвращают значение **Null.**

Дополнительные сведения

Примечание. Если требуется возвратить первую или последнюю запись из набора (подмножества) записей, необходимо создать запрос с сортировкой либо по возрастанию, либо по убыванию и указать для свойства **Набор значений (TopValues)** значение 1. Более подробное описание см. в разделе справки для свойства **Набор значений** (**TopValues**). В программе Visual Basic для возвращения первой или последней записи из набора записей следует создать объект **Recordset** и вызвать метод **MoveFirst** или **MoveLast.**

Функция DLookup

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDLookupC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDLookupX":1}

Функция **DLookup** возвращает значение конкретного поля в указанном наборе (подмножестве) записей. Функцию **DLookup** используют в программах на Visual Basic, макросах, в выражениях в запросах, а также для определения вычисляемого элемента управления в форме или отчете.

Допускается использование функции **DLookup** для вывода значения поля в случае, когда это поле не принадлежит к базовому источнику записей формы или отчета. Предположим, например, что базовой таблицей формы является таблица «Заказано». В форме выводятся поля «КодЗаказа», «КодТовара», «Цена», «Количество» и «Скидка». Однако поле «Марка» находится в другой таблице, «Товары». В этом случае функция **DLookup** позволяет создать в этой форме вычисляемое поле, в котором будет выводиться значение поля «Марка».

Синтаксис

DLookup(*выражение*, *набор*[, *условие*]**)**

Функция **DLookup** использует следующие аргументы.

Аргументы	Описание
выражение	Выражение, определяющее нужное поле. Данный аргумент может задаваться строковым выражением, определяющим поле в таблице или запросе, или представлять выражение, задающее выполнение вычислений над данными, содержащимися в поле. Допускается использование в аргументе *выражение* имени поля в таблице или элемента управления в форме, константы, а также встроенной или определяемой пользователем функции. Не допускается использование в аргументе выражение других статистических функций по подмножеству или статистических функций SQL.
набор	Строковое выражение, определяющее набор записей, образующих подмножество. Может представлять имя таблицы или запроса.
условие	Необязательное строковое выражение, ограничивающее диапазон данных, в которых производится поиск значений. Например, аргумент *условие* часто является эквивалентом предложения WHERE инструкции SQL, но без ключевого слова WHERE. Если аргумент *условие* опущен, **DLookup** выполняет действия над полем, заданным в аргументе *выражение,* для всего набора записей. Любое поле, указанное в аргументе *условие*, должно принадлежать подмножеству, заданному аргументом *набор*; в противном случае функция **DLookup** возвращает значение **Null.**

Дополнительные сведения

DLookup возвращает значение поля или выражения, определенного в аргументе *выражение*. Выбираемые в таблице или запросе значения принадлежат подмножеству записей, определяемых аргументом *набор* и удовлетворяющих условиям отбора, задаваемым в аргументе *условие*.

Если ни одна из записей набора не удовлетворяет аргументу *условие* или *набор* не содержит записей, функция **DLookup** возвращает значение **Null**.

Если указанным условиям удовлетворят несколько полей, **DLookup** возвращает значение

первого найденного поля. Рекомендуется указывать условия, обеспечивающие уникальность значения, возвращаемого функцией **DLookup**. Одним из способов обеспечения уникальности возвращаемых значений является указание условий для ключевого поля, такого как поле «КодСотрудника» в следующем примере:

```
Dim varX As Variant
varX = DLookup("[Фамилия]", "Сотрудники", "[КодСотрудника] = 1")
```

При любом использовании функции **DLookup** в макросе или в модуле, в выражении в запросе или в вычисляемом элементе управления необходимо обеспечить правильное составление аргумента *условие*.

Функция **DLookup** может быть включена в строку «Условие отбора» бланка запроса, в выражение для вычисляемого поля, а также в строку **Обновление** запроса запроса на обновление.

Допускается использование функции **DLookup** в вычисляемом элементе управления в форме или отчете в случае, когда поле, для которого проводится поиск значения, не принадлежит к базовому источнику записей формы. Например, предположим, что в форме «Заказы» создано поле с именем «КодТовара», в котором выводится значение поля «КодТовара» из таблицы «Заказы», и что требуется одновременно вывести значение поля «Марка» из таблицы «Товары», соответствующее выведенному коду. Для этого следует создать в форме другое поле и ввести для него следующее выражение в ячейку свойства **Данные** (**ControlSource**).

```
=DLookup("[Марка]", "Товары", "[КодТовара] =" & Forms![Заказы]!КодТовара)
```

Советы

- Хотя функция **DLookup** позволят проводить поиск значений во внешней таблице, обычно более эффективным приемом оказывается создание запроса, содержащего все необходимые поля, с последующим созданием формы или отчета на базе этого запроса.
- Поиск значений во внешней таблице может быть также проведен с помощью мастера подстановок.

Примечание. Несохраненные изменения записей, входящих в *набор*, не учитываются данной функцией. Если требуется обеспечить учет в функции **DLookup** измененных значений, необходимо сначала сохранить изменения с помощью команды **Сохранить запись** из меню **Записи**, перевести фокус на другу запись или вызвать метод **Update**.

Функция DLookup, примеры

В данном примере возвращается название организации из поля «Название», удовлетворяющего аргументу *условие*. Набор записей отбирается в таблице «Поставщики». Аргумент *условие* указывает запись со значением 1 в поле «КодПоставщика».

```
Dim varX As Variant
varX = DLookup("[Название]", "Поставщики", "[КодПоставщика] = 1")
```

В следующем примере код поставщика выбирается из поля «ИмяПоля» в форме «ИмяФормы». Отметим, что ссылка на элемент управления помещается вне кавычек, определяющих строку условий. Это обеспечивает считывание текущего значения элемента управления при каждом вызове функции **DLookup**.

```
Dim varX As Variant
varX = DLookup("[Название]", "Поставщики", "[КодПоставщика] = " _
   & Forms!ИмяФормы!ИмяПоля)
```

В следующем примере для указания условий используется переменная `intSearch`.

```
Dim intSearch As Integer, varX As Variant
intSearch = 1
varX = DLookup("[Название]", "Поставщики", _
   "[КодПоставщика] = " & intSearch)
```

Функции DMin, DMax

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDMinDMaxC;dasqlMinMax":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acfctDMinDMaxX":1}

Функции **DMin** и **DMax** возвращают минимальное и максимальное значения поля в указанном наборе (подмножестве) записей. Функции **DMin** и **DMax** используют в макросах или программах на Visual Basic, в выражениях в запросах, а также для определения вычисляемого элемента управления.

Например, с помощью функций **DMin** и **DMax** определяют вычисляемые поля в отчете, в которых выводятся минимальная и максимальная суммы заказов конкретного клиента. Функцию **DMin** можно включить в запросе в выражение, с помощью которого будут отбираться заказы со скидкой, превышающей минимально допустимую.

Синтаксис

DMin(*выражение*, *набор*[, *условие*]**)**
DMax(*выражение*, *набор*[, *условие*]**)**

Функции **DMin** и **DMax** используют следующие аргументы.

Аргументы	Описание
выражение	Выражение, определяющее нужное поле. Данный аргумент может задаваться строковым выражением, определяющим поле в таблице или запросе, или представлять выражение, задающее выполнение вычислений над данными, содержащимися в поле. Допускается использование в аргументе *выражение* имени поля в таблице или элемента управления в форме, константы, а также встроенной или определяемой пользователем функции. Не допускается использование в аргументе выражение других статистических функций по подмножеству или статистических функций SQL.
набор	Строковое выражение, определяющее набор записей, образующих подмножество. Может представлять имя таблицы или запроса.
условие	Необязательное строковое выражение, ограничивающее диапазон данных, для которых определяется минимальное или максимальное значение поля. Например, аргумент *условие* часто является эквивалентом предложения WHERE инструкции SQL, но без ключевого слова WHERE. Если аргумент *условие* опущен, **DMin** и **DMax** выполнят действия над полем, заданным в аргументе *выражение,* для всего набора записей. Любое поле, указанное в аргументе *условие*, должно принадлежать подмножеству, заданному аргументом *набор*; в противном случае функции **DMin** и **DMax** возвращают значение **Null.**

Дополнительные сведения

Минимальное и максимальное значение поля отбираются в соответствие с условиями, указанными в аргументе *условие*. Если *выражение* определяет числовые данные, **DMin** и **DMax** возвращают числовые значения. Если *выражение* определяет строковые значения, то возвращается строка, являющаяся первой или последней в алфавитном порядке.

Пустые (**Null**) значения поля, определяемого аргументом *выражение*, игнорируются. Однако, если ни одна из записей набора не удовлетворяет аргументу *условие* или *набор* не содержит записей, функции **DMin** и **DMax** возвращают значение **Null**.

При любом использовании функций **DMin** или **DMax** в макросе. функции или в модуле, в выражении в запросе или в вычисляемом элементе управления необходимо обеспечить правильное составление аргумента *условие*.

Функции **DMin** и **DMax** могут быть включены в строку **Условие отбора** бланка запроса, в выражение для вычисляемого поля, а также в строку **Обновление** запроса на обновление.

Примечание. В выражениях для вычисляемых полей в итоговых запросах используют как функции **DMin** и **DMax**, так и функции **Min** и **Max.** При использовании функции **DMin** или **DMax** значения находятся до группировки данных. При использовании функции **Min** или **Max** сначала выполняется группировка данных, а потом определятся минимальное или максимальное значения.

Функцию **DMin** или **DMax** используют в вычисляемом элементе управления, если необходимо указать условия отбора, ограничивающие диапазон данных. Например, для вывода максимальной стоимости доставки заказа в Крым следует ввести для поля в форме в ячейку свойства **Данные** (**ControlSource**) следующее выражение.

```
=DMax("[СтоимостьДоставки]", "Заказы", "[ОбластьПолучателя] = 'Крым'")
```

Если требуется просто найти минимальное или максимальное значение поля для всех записей в наборе, определяемом аргументом *набор*, следует использовать функцию **Min** или **Max**.

Допускается использование функции **DMin** или **DMax** в модуле или макросе или в вычисляемом элементе управления в форме в случае, когда поле, для которого проводится поиск первого или последнего значения, не принадлежит к базовому источнику записей формы.

Совет. Хотя функции **DMin** или **DMax** позволяют проводить поиск значений во внешней таблице, обычно более эффективным приемом оказывается создание запроса, содержащего все необходимые поля, с последующим созданием формы или отчета на базе этого запроса.

Примечание. Несохраненные изменения записей, входящих в *набор* не учитываются данными функциями. Если требуется обеспечить учет в функции **DMax** или **DMin** измененных значений, необходимо сначала сохранить изменения с помощью команды **Сохранить запись** из меню **Записи**, перевести фокус на другу запись или вызвать метод **Update.**

Функции DMin, DMax, примеры

В данном примере выводятся минимальное и максимальное значения из поля «Стоимость доставки» для заказов, доставляемых в Литву. Аргумент *набор* определяет таблицу «Заказы». Аргумент *условие* задает отбор записей, имеющих в поле «СтранаПолучателя» значение «Литва».

```
Dim curX As Currency, curY As Currency
curX = DMin("[СтоимостьДоставки]", "Заказы", "[СтранаПолучателя] =
'Литва'")
curY = DMax("[СтоимостьДоставки]", "Заказы", "[СтранаПолучателя] =
'Литва'")
```

В следующем примере в аргумент *условие* включается текущее значение из списка «ДатаРазмещения», в котором выводятся значения поля «ДатаРазмещения» из таблицы «Заказы». Отметим, что ссылка на элемент управления помещается вне кавычек, определяющих строку условий. Это обеспечивает считывание текущего значения элемента управления при каждом вызове функции **DMax**.

```
Dim curX As Currency
curX = DMax("[СтоимостьДоставки]", "Заказы", "[ДатаРазмещения] = #" _
   & Forms!Заказы!ДатаРазмещения & "#")
```

В следующем примере для указания условий используется переменная `dteOrderDate`. Обратите внимание, что символы номера (#) включатся в строку условий, поэтому после слияния строк эти символы будут окружать дату.

```
Dim dteOrderDate As Date, curX As Currency
dteOrderDate = #3/30/95#
curX = DMin("[СтоимостьДоставки]", "Заказы",
   "[ДатаРазмещения] = #" & dteOrderDate & "#")
```

Функции DStDev, DStDevP

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDStDevC;dasqlDStDevP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acfctDStDevX":1}

Функции **DStDev** и **DStDevP** возвращаю значение среднеквадратичного отклонения для выборки или ограниченной выборки значений, содержащихся в указанном наборе записей (подмножестве). Функции **DStDev** и **DStDevP** используют в макросах, программах на Visual Basic, в выражениях в запросах, а также для определения вычисляемого элемента управления в форме или отчете.

DStDevP возвращает смещенное значение среднеквадратичного отклонения, а **DStDev** несмещенное значение (ограниченная выборка).

Например, функцию **DStDev** можно использовать в модуле для расчета среднеквадратичного отклонения для студенческих экзаменационных оценок.

Синтаксис

DStDev(*выражение*, *набор*[, *условие*]**)**
DStDevP(*выражение*, *набор*[, *условие*]**)**

Функции **DStDev** и **DStDevP** используют следующие аргументы.

Аргументы	Описание
выражение	Выражение, определяющее нужное поле. Данный аргумент может задаваться строковым выражением, определяющим поле в таблице или запросе, или представлять выражение, задающее выполнение вычислений над данными, содержащимися в поле. Допускается использование в аргументе *выражение* имени поля в таблице или элемента управления в форме, константы, а также встроенной или определяемой пользователем функции. Не допускается использование в аргументе выражение других статистических функций по подмножеству или статистических функций SQL.
набор	Строковое выражение, определяющее набор записей, образующих подмножество. Может представлять имя таблицы или запроса.
условие	Необязательное строковое выражение, ограничивающее диапазон данных, для которых определяется среднеквадратичное отклонение. Например, аргумент *условие* часто является эквивалентом предложения WHERE инструкции SQL, но без ключевого слова WHERE. Если аргумент *условие* опущен, **DStDev** и **DStDevP** выполнят действия над полем, заданным в аргументе *выражение,* для всего набора записей. Любое поле, указанное в аргументе *условие*, должно принадлежать подмножеству, заданному аргументом *набор*; в противном случае функции **DStDev** и **DStDevP** возвращают значение **Null.**

Дополнительные сведения

Если аргумент *набор* определяет меньше двух записей или если меньше двух записей удовлетворят аргументу *условие*, **DStDev** и **DStDevP** возвращают пустое (**Null**) значение, показывающее, что расчет среднеквадратичного отклонения невозможен.

При любом использовании функций **DStDev** или **DStDevP** в макросе, функции или в модуле, в

выражении в запросе или в вычисляемом элементе управления необходимо обеспечить правильное составление аргумента *условие*.

Функции **DStDev** и **DStDevP** могут быть включены в строку «Условие отбора» бланка запроса на выборку. Например, они позволят создать запрос по таблицам «Заказы» и «Товары», в котором будет выводиться список всех товаров, для которых стоимость доставки превышает сумму средней стоимости и среднеквадратичного отклонения стоимости доставки. В этом случае в ячейку строки условий для поля «СтоимостьДоставки» следует ввести такое выражение:

```
>(DStDev("[СтоимостьДоставки]", "Заказы") + DAvg("[СтоимостьДоставки]",
"Заказы"))
```

Функции **DStDev** и **DStDevP** могут быть также включены в выражение для вычисляемого поля и в строку **Обновление** запроса на обновление.

Примечание. В выражениях для вычисляемых полей в итоговых запросах используют как функции **DStDev** и **DStDevP**, так и функции **StDev и StDevP.** При использовании функции **DStDev** или **DStDevP** значения находятся до группировки данных. При использовании функции **StDev** или **StDevP** сначала выполняется группировка данных, а потом определятся значения среднеквадратичного отклонения.

Функцию **DStDev** или **DStDevP** используют в вычисляемом элементе управления, если необходимо указать условия отбора, ограничивающие диапазон данных. Например, для вывода среднеквадратичного отклонения стоимости доставки заказов в Крым следует ввести для поля в форме в ячейку свойства **Данные** (**ControlSource**) следующее выражение:

```
=DStDev("[СтоимостьДоставки]", "Заказы", "[ОбластьПолучателя] = 'Крым'")
```

Если требуется просто найти значение среднеквадратичного отклонения для всех записей в наборе, определяемом аргументом *набор*, следует использовать функцию **StDev** или **StDevP**.

Совет. Если поле, для которого построено *выражение*, имеет числовой тип данных, то функции **DStDev** и **DStDevP** возвращают значение типа **Double.** Для повышения производительности рекомендуется применять функцию преобразования типа данных при определении вычисляемого поля с помощью функции **DStDev** или **DStDevP**.

Примечание. Несохраненные изменения записей, входящих в *набор* не учитываются данными функциями. Если требуется обеспечить учет в функции **DStDev** или **DStDevP** измененных значений, необходимо сначала сохранить изменения с помощью команды **Сохранить запись** из меню **Записи**, перевести фокус на другу запись или вызвать метод **Update**.

Функции DStDev, DStDevP, примеры

В данном примере рассчитываются значения смещенного и несмещенного среднеквадратичного отклонения для стоимости доставки заказов в Литву. Аргумент *набор* определяет таблицу «Заказы». Аргумент *условие* задает отбор записей, имеющих в поле «СтранаПолучателя» значение «Литва».

```
Dim dblX As Double, dblY As Double
'Несмещенное значение.
dblX = DStDev("[СтоимостьДоставки]", "Заказы", "[СтранаПолучателя] =
'Литва'")
' Смещенное значение.
dblY = DStDevP("[СтоимостьДоставки]", "Заказы", "[СтранаПолучателя] =
'Литва'")
```

В следующем примере те же расчеты выполнятся с помощью переменной `strCountry`, задающей страну в строке условий. Отметим, что одинарные кавычки (') включатся в строковое выражение, поэтому после слияния строк строковая константа «Литва» будет заключена в одинарные кавычки.

```
Dim strCountry As String, dblX As Double, dblY As Double
strCountry = "Литва"
dblX = DStDev("[СтоимостьДоставки]", "Заказы", _
   "[СтранаПолучателя] = '" & strCountry & "'")
dblY = DStDevP("[СтоимостьДоставки]", "Заказы", _
   "[СтранаПолучателя] = '" & strCountry & "'")
```

Функция DSum

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctDSumC;dasqlSum":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDSumX":1}

Функция **DSum** возвращает сумму набора значений в указанном наборе (подмножестве) записей. Функцию **DSum** используют в макросах, в программах на Visual Basic, в выражениях в запросах, а также для определения вычисляемого элемента управления.

Например, функция **DSum** позволяет подсчитать в запросе общую сумму продаж, совершенных конкретным сотрудником за указанный период времени, или создать вычисляемое поле, в котором выводится сумма с накоплением для продаж конкретного товара.

Синтаксис

DSum(*выражение*, *набор*[, *условие*]**)**

Функция **DSum** использует следующие аргументы

Аргументы	**Описание**
выражение	Выражение, определяющее нужное поле. Данный аргумент может задаваться строковым выражением, определяющим поле в таблице или запросе, или представлять выражение, задающее выполнение вычислений над данными, содержащимися в поле. Допускается использование в аргументе *выражение* имени поля в таблице или элемента управления в форме, константы, а также встроенной или определяемой пользователем функции. Не допускается использование в аргументе выражение других статистических функций по подмножеству или статистических функций SQL.
набор	Строковое выражение, определяющее набор записей, образующих подмножество. Может представлять имя таблицы или запроса.
условие	Необязательное строковое выражение, ограничивающее диапазон данных, для которых производится суммирование значений. Например, аргумент *условие* часто является эквивалентом предложения WHERE инструкции SQL, но без ключевого слова WHERE. Если аргумент *условие* опущен, **DSum** выполняет расчеты над полем, заданным в аргументе *выражение,* для всего набора записей. Любое поле, указанное в аргументе *условие*, должно принадлежать подмножеству, заданному аргументом *набор*; в противном случае функция **DSum** возвращает значение **Null.**

Дополнительные сведения

Если отсутствуют записи, удовлетворяющие указанным условиям, или если набор не содержит записей, функция **DSum** возвращает значение **Null**.

При любом использовании функции **DSum** в макросе или в модуле, в выражении в запросе или в вычисляемом элементе управления необходимо обеспечить правильное составление аргумента *условие*.

Функция **DSum** может быть включена в строку **Условие отбора** бланка запроса, в выражение для вычисляемого поля, а также в строку **Обновление** запроса на обновление.

Примечание. В выражениях для вычисляемых полей в итоговых запросах используют как функцию **DSum**, так и функцию **Sum.** При использовании функции **DSum** суммирование

выполняется до группировки данных. При использовании функции **Sum**, сначала выполняется группировка данных, а потом их суммирование.

Допускается использование функции **DSum** для вывода суммы значений поля, не принадлежащего к базовому источнику записей формы или отчета. Например, предположим, что имеется форма, в которой выводятся данные о конкретном товаре. В этом случае можно создать в форме вычисляемое поле и выводить в нем с помощью **DSum** сумму с накоплением для величин продаж данного товара.

Совет. Если поле, для которого требуется рассчитывать сумму с накоплением, принадлежит к базовой таблице отчета, то для определения вычисляемого поля следует использовать свойство **Сумма с накоплением** (**RunningSum**). Функцию **DSum** используют для расчета суммы с накоплением в формах.

Примечание. Несохраненные изменения записей, входящих в *набор*, не учитываются данной функцией. Если требуется обеспечить учет в функции **DSum** измененных значений, необходимо сначала сохранить изменения с помощью команды **Сохранить запись** из меню **Записи**, перевести фокус на другу запись или вызвать метод **Update**.

Функция DSum, примеры

В данном примере подсчитывается общая стоимость доставки заказов, отправленных в Литву. Аргумент *набор* определяет таблицу «Заказы». Аргумент *условие* ограничивает подмножество записей теми, которые имеют в поле «СтранаПолучателя» значение «Литва»

```
Dim curX As Currency
curX = DSum("[СтоимостьДоставки]", "Заказы", "[СтранаПолучателя] =
'Литва'")
```

В следующем примере выполнятся расчеты с двумя условиями. Отметим, что одинарные кавычки (') и символы номера (#) включатся в строковое выражение, поэтому после слияния строк строковая константа будет заключена в одинарные кавычки, а дата в символы (#).

```
Dim curX As Currency
curX = DSum("[СтоимостьДоставки]", "Заказы", _
   "[СтранаПолучателя] = 'Литва' AND [ДатаИсполнения] > #1-1-95#")
```

Допускается ввод статистических функций по подмножеству в строку **Обновление** запроса на обновление. Предположим, например, что требуется отслеживать текущие данные по продажам товаров, перечисленных в таблице «Товары». В этом случае можно добавить в таблицу «Товары» новое поле с именем «Продано» и заполнять его с помощью запроса на обновление. Для этого следует создать новый запрос по таблице «Товары», выбрать команду **Обновление** в меню **Запрос**, добавить поле «Продано» в бланк запроса и ввести для него в строку **Обновление** следующее выражение:

```
DSum("[Количество]*[Цена]", "Заказано", "[КодТовара] = " & [КодТовара])
```

При выполнении этого запроса по данным из таблицы «Заказано» будет подсчитываться общий объем продаж для каждого товара. Суммарные значения для каждого товара будут включаться в таблицу «Товары».

Функции DVar, DVarP

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDVarC;dasqlDVarP"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctDVarX":1}

Функции **DVar** и **DVarP** возвращают значение дисперсии для выборки или ограниченной выборки значений, содержащихся в указанном наборе записей (подмножестве). Функции **DVar** и **DVarP** используют в макросах в программах на Visual Basic, в выражениях в запросах, а также для определения вычисляемого элемента управления в форме или отчете.

DVarP возвращает значение смещенной, а **DVar** несмещенной дисперсии.

Например, функцию **DVar** можно использовать в модуле для расчета разброса студенческих экзаменационных оценок.

Синтаксис

DVar(*выражение*, *набор*[, *условие*]**)**
DVarP(*выражение*, *набор*[, *условие*]**)**

Функции **DVar** и **DVarP** используют следующие аргументы.

Аргументы	Описание
выражение	Выражение, определяющее нужное поле. Данный аргумент может задаваться строковым выражением, определяющим поле в таблице или запросе, или представлять выражение, задающее выполнение вычислений над данными, содержащимися в поле. Допускается использование в аргументе *выражение* имени поля в таблице или элемента управления в форме, константы, а также встроенной или определяемой пользователем функции. Не допускается использование в аргументе выражение других статистических функций по подмножеству или статистических функций SQL.
набор	Строковое выражение, определяющее набор записей, образующих подмножество. Может представлять имя таблицы или запроса.
условие	Необязательное строковое выражение, ограничивающее диапазон данных, для которых определяется дисперсия. Например, аргумент *условие* часто является эквивалентом предложения WHERE инструкции SQL, но без ключевого слова WHERE. Если аргумент *условие* опущен, **DVar** и **DVarP** выполнят действия над полем, заданным в аргументе *выражение,* для всего набора записей. Любое поле, указанное в аргументе *условие*, должно принадлежать подмножеству, заданному аргументом *набор*; в противном случае функции **DVar** и **DVarP** возвращают значение **Null.**

Дополнительные сведения

Если аргумент *набор* определяет меньше двух записей или если меньше двух записей удовлетворят аргументу *условие*, **DVar** и **DVarP** возвращают пустое (**Null**) значение, показывающее, что расчет дисперсии невозможен.

При любом использовании функций **DVar** или **DVarP** в макросе, функции или в модуле, в выражении в запросе или в вычисляемом элементе управления необходимо обеспечить правильное составление аргумента *условие*.

Функции **DVar** и **DVarP** могут быть включены в строку **Условие отбора** бланка запроса на выборку; в выражение для вычисляемого поля или в строку **Обновление** запроса на

обновление.

Примечание. В выражениях для вычисляемых полей в итоговых запросах используют как функции **DVar** и **DVarP**, так и функции **Var** или **VarP.** При использовании функции **DVar** или **DVarP** значения находятся до группировки данных. При использовании функции **Var** или **VarP** сначала выполняется группировка данных, а потом определяются значения дисперсии.

Функцию **DVar** или **DVarP** используют в вычисляемом элементе управления, если необходимо указать условия отбора, ограничивающие диапазон данных. Например, для вывода значения дисперсии для стоимости доставки заказов в Крым следует ввести для поля в форме в ячейку свойства **Данные** (**ControlSource**) следующее выражение:

```
=DVar("[СтоимостьДоставки]", "Заказы", "[ОбластьПолучателя] = 'Крым'")
```

Если требуется просто найти значение дисперсии для всех записей в наборе, определяемом аргументом *набор*, следует использовать функцию **Var** или **VarP**.

Примечание. Несохраненные изменения записей, входящих в *набор* не учитываются данными функциями. Если требуется обеспечить учет в функции **DVar** или **DVarP** измененных значений, необходимо сначала сохранить изменения с помощью команды **Сохранить запись** из меню **Записи**, перевести фокус на другу запись или вызвать метод **Update**.

Функции DVar, DVarP, примеры

В данном примере рассчитываются значения смещенной и несмещенной дисперсии для стоимости доставки заказов в Литву. Аргумент *набор* определяет таблицу «Заказы». Аргумент *условие* задает отбор записей, имеющих в поле «СтранаПолучателя» значение «Литва».

```
Dim dblX As Double, dblY As Double
' Несмещенная дисперсия.
dblX = DVar("[СтоимостьДоставки]", "Заказы", "[СтранаПолучателя] =
'Литва'")
' Смещенная дисперсия.
dblY = DVarP("[СтоимостьДоставки]", "Заказы", "[СтранаПолучателя] =
'Литва'")
```

В следующем примере те же расчеты выполнятся с помощью переменной `strCountry`, задающей страну в строке условий. Отметим, что одинарные кавычки (') включатся в строковое выражение, поэтому после слияния строк строковая константа `Литва` будет заключена в одинарные кавычки.

```
Dim strCountry As String, dblX As Double
strCountry = "Литва"
dblX = DVar("[СтоимостьДоставки]", "Заказы", "[СтранаПолучателя] = '" _
    & strCountry & "'")
```

Использование значений полей в статистических функциях по подмножеству

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"achowFieldCalcDomainC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"achowFieldCalcDomainX":1}

Строковое выражение для аргумента *выражение* в статистических функциях по подмножеству позволяет выполнять расчеты с использованием значений полей. Например, допускается расчет процентных долей (скидки или налога) путем деления значения поля на определенное число.

В следующей таблице приведены примеры действий над значениями полей из таблиц «Заказы» и «Заказано».

Действие	**Пример**
Сложение числа со значением поля	"[СтоимостьДоставки] + 5"
Вычитание числа из значения поля	"[СтоимостьДоставки] - 5"
Умножение значения поля на число	"[СтоимостьДоставки] * 2"
Деление значения поля на число	"[СтоимостьДоставки] / 2"
Сложение значений полей	"[НаСкладе] + [Заказано]"
Вычитание значений полей	"[МинимальныйЗапас] - [НаСкладе]"

Статистические функции по подмножеству обычно используют в макросе или в модуле, в вычисляемом элементе управления в форме или отчете, а также в условиях отбора в запросе.

В следующем примере рассчитывается средняя величина скидки для всех заказов в таблице «Заказано». Для каждой записи величина скидки определяется путем перемножения значений полей «Цена» и «Скидка», после чего рассчитывается среднее по всей таблице. Данное выражение допускает включение в процедуру модуля:

```
Dim dblX As Double
dblX = DAvg("[Цена] * [Скидка]", "[Заказано]")
```

События «До обновления» (BeforeUpdate), «После обновления» (AfterUpdate)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtBeforeAfterUpdateC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtBeforeAfterUpdateX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtBeforeAfterUpdateMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtBeforeAfterUpdateEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS":"acevtBeforeAfterUpdateA"}

- Событие **До обновления (BeforeUpdate)** возникает перед обновлением изменяемых данных элемента управления или записи.
- Событие **После обновления (AfterUpdate)** возникает после обновления изменяемых данных в элемента управления или записи.

Примечания

- Изменение данных в элементе управления с помощью программы Visual Basic или макроса, содержащего макрокоманду **ЗадатьЗначение (SetValue)**, не приводит к возникновению событий **До обновления** и **После обновления** для данного элемента управления. Однако при последующем переходе на другую запись или при сохранении текущей записи эти события возникают для формы.
- События **До обновления** и **После обновления** определены только для элементов управления в формах и не определены для элементов управления в отчетах.
- Эти события определены только для групп переключателей. Они не определены для переключателей, флажков или выключателей в группах.

Дополнительные сведения

Для выполнения макроса или процедуры обработки событий, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **До обновления (BeforeUpdate)** или **После обновления (AfterUpdate)**.

События **До обновления** и **После обновления** возникают при обновлении элемента управления или записи. При потере фокуса элементом управления в некоторой записи или при нажатии пользователем клавиш ENTER или TAB изменяемые данные для этого элемента управления обновляются. Если же сама запись теряет фокус или если пользователь выбирает команду **Сохранить запись** в меню **Записи**, вся запись целиком обновляется, и данные сохраняются в базе данных.

Когда пользователь вводит новые или изменяет существующие данные для элемента управления в форме, а затем переходит на другую запись или сохраняет текущую запись с помощью команды **Сохранить запись** в меню **Записи**, события **До обновления** и **После обновления** для формы возникают сразу после событий **До обновления** и **После обновления** для этого элемента управления. При переходе на другую запись сразу после этих событий возникают события **Выход (Exit)** и **Потеря фокуса (LostFocus)** для данного элемента управления, за ними следует событие **Текущая запись (Current)** для записи, на которую осуществлен переход, а затем события **Вход (Enter)** и **Получение фокуса (GotFocus)** для первого элемента управления в этой записи. Чтобы запустить макросы или процедуры обработки событий **До обновления** и **После обновления**, не запуская при этом макросы или процедуры обработки событий **Выход** и **Потеря фокуса**, следует сохранить данную запись с помощью команды **Сохранить запись** в меню **Записи**.

Макросы и процедуры обработки событий **До обновления** и **После обновления** запускаются только при изменении пользователем данных в элементе управления. При изменении значения в вычисляемом элементе управления эти события не возникают. Макросы и процедуры обработки событий **До обновления** и **После обновления** для формы запускаются только при изменении пользователем данных в одном или нескольких элементах управления в записи.

Событие **До обновления** может использоваться в формах для отмены обновления записи перед переходом на другую запись.

Свойство **OldValue** для присоединенных элементов управления получает обновленное значение только после возникновения события **После обновления** для формы. Даже когда пользователь вводит в такой элемент управления новое значение, значение свойства **OldValue** изменяется, только если сохранены данные (обновлена запись). Если пользователь отменяет обновление, значение свойства **OldValue** заменяет существующее значение в данном элементе управления.

Событие **До обновления** часто используется для проверки соответствия данных условиям на значение, в особенности, в сложных случаях типа следующих:

- проверяются условия для нескольких значений в форме;
- при вводе различных данных выводятся разные сообщения об ошибке;
- допускается нарушение пользователем условий на значение;
- условия содержат ссылки на элементы управления в других формах или включают определяемые пользователем функции.

Примечание. Для проверки простых или не очень сложных условий на значение, таких как требование на обязательный ввод значения в поле или проверка значений нескольких элементов управления в форме, используется свойство **Условие на значение (ValidationRule)** для элементов управления, а также свойства **Условие на значение** и **Обязательное поле (Required)** для полей и записей в таблицах.

События «До обновления» (BeforeUpdate), «После обновления» (AfterUpdate) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtBeforeAfterUpdateMOC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtBeforeAfterUpdateMOX":1}

Для выполнения макроса, связанного с событиями **До обновления (BeforeUpdate)** или **После обновления (AfterUpdate)**, следует указать имя данного макроса в качестве значения свойства **До обновления (BeforeUpdate)** или **После обновления (AfterUpdate)**. Microsoft Access запускает макрос, указанный в свойстве **До обновления** или **После обновления** для элемента управления или формы, непосредственно до или после обновления данных в элементе управления или в текущей записи формы. Если содержимое элемента управления или записи не обновляется, то макрос выполнен не будет.

Дополнительные сведения

Макросы, связанные с событием **До обновления**, используются, если надо объединить все условия на значения для формы в одну группу макросов, или если выражение, определяющее условия на значения, применяется сразу к нескольким формам.

Предположим, что в приложении используется форма, в которой пользователь вводит название страны в поле «Страна» и соответствующий почтовый индекс в поле «Индекс». Применение условных выражений в макросе позволяет организовать проверку соответствия данных, введенных в эти поля, и вывести сообщение, если значение в поле «Индекс» не соответствует стандарту данной страны.

Макрос, связанный с событием **После обновления**, может быть использован для вывода новой страницы в форме или для перевода фокуса на определенный элемент управления или на запись. Например, после ввода пользователем в форме значения в поле «Категория» вызов макрокоманды **НаСтраницу (GoToPage)** в макросе, связанном с событием **После обновления**, позволяет перевести фокус на страницу формы, содержащую элементы управления, предназначенные для отображения указанной категории товаров.

Допускается использование макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе события **До обновления** для отмены операции обновления. Если создан макрос для обработки события **До обновления** элемента управления, то макрокоманда **ОтменитьСобытие** возвращает фокус на этот элемент управления, а введенные пользователем в этот элемент управления данные остаются неизмененными.

Если макрос связан с событием **До обновления** формы (в этом случае макрос выполняется при каждом обновлении записи), то макрокоманда **ОтменитьСобытие** возвращает фокус на запись.

Не допускается использование макрокоманды **ОтменитьСобытие** в макросе, связанном с событием **После обновления**.

События «До обновления» (BeforeUpdate), «После обновления» (AfterUpdate) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtBeforeAfterUpdateEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtBeforeAfterUpdateEPX":1}

Для создания процедуры обработки событий, которая выполняется при возникновении события **До обновления (BeforeUpdate)** или **После обновления (AfterUpdate)**, следует выбрать в качестве значения свойства **До обновления (BeforeUpdate)** или **После обновления (AfterUpdate)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_BeforeUpdate(Cancel **As Integer)**
Private Sub *имяЭлемента***_BeforeUpdate(**Cancel **As Integer)**
Private Sub Form_AfterUpdate( )
Private Sub *имяЭлемента***_AfterUpdate( )**

Процедура обработки события **До обновления** использует следующие аргументы.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **До обновления (BeforeUpdate)**.
Cancel	Определяет возникновение события **До обновления**. Заданное для аргумента «Cancel» значение **True** (–1) отменяет событие **До обновления**.

Процедура обработки события **После обновления** использует следующий аргумент.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **После обновления (AfterUpdate)**.

Дополнительные сведения

Отменить событие **После обновления** невозможно.

Процедура обработки события **После обновления** может быть использована для вывода новой страницы в форме или для перевода фокуса на определенный элемент управления или на запись. Например, после ввода пользователем в форме значения в поле «Категория» вызов метода **GoToPage** в процедуре обработки события **После обновления** позволяет перевести фокус на страницу формы, содержащую элементы управления, предназначенные для отображения указанной категории товаров.

События «До обновления» (BeforeUpdate), «После обновления» (AfterUpdate) - Пример процедуры обработки события

В данном примере демонстрируется использование процедуры обработки события **До обновления (BeforeUpdate)** для проверки на совпадающие названия товаров. После того как пользователь вводит название товара в поле «Марка», введенное значение сравнивается со значениями поля «Марка» в таблице «Товары». Если в таблице обнаруживается совпадающее значение, на экран выводится сообщение о том, что сведения о товаре уже занесены в базу данных.

Для проверки данного примера следует поместить процедуру обработки события в форму «Товары», которая содержит поле с именем «Марка».

```
Private Sub Марка_BeforeUpdate(Cancel As Integer)
   If(Not IsNull(DLookup("[Марка]", "Товары", _
         "[Марка] ='" & Me!Марка & "'"))) Then
      MsgBox "Сведения об этом товаре уже содержатся в базе данных."
      Cancel = True
      Me!Марка.Undo
   End If
End Sub
```

Событие «Нажатие кнопки» (Click)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtClickC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtClickX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtClickMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtClickEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtClickA"}

Событие **Нажатие кнопки (Click)** возникает, когда пользователь нажимает и отпускает кнопку мыши при установленном на объекте указателе.

Примечания

- Событие **Нажатие кнопки** определено для форм, разделовформ и элементов управления в формах. Оно не определено для элементов управления в отчетах.
- Это событие не определено для флажков,переключателей или выключателей в группах переключателей. Оно определено только для самих групп переключателей.
- Данное событие не определено для надписи, присоединенной к другому элементу управления, например, для подписи поля. Оно определено только для «самостоятельных» надписей. Выбор присоединенной надписи с помощью мыши имеет тот же эффект, что и выбор элемента управления, к которому присоединена надпись. При этом возникает обычная цепочка событий элемента управления, а не события присоединенной надписи.
- Данное событие определено для элементов управления, содержащих гиперссылки.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с данным событием, следует указать имя этого макроса или элемент [Процедура обработки события] в качестве значения свойства **Нажатие кнопки (OnClick)**.

Для формы данное событие возникает при нажатии кнопки мыши, когда указатель расположен на пустой области формы или на области выделения записи в форме.

Для элемента управления данное событие возникает при следующих действиях пользователя:

- Нажатие левой кнопки мыши при установленном на элементе управления указателе. Нажатие правой или средней кнопки мыши не приводит к возникновению данного события.
- Нажатие левой кнопки мыши при указателе, установленном на элементе управления, содержащем гиперссылку. Нажатие правой или средней кнопки мыши не приводит к возникновению данного события. Когда пользователь помещает указатель мыши на элемент управления, содержащий гиперссылку, указатель принимает вид «руки». При нажатии кнопки мыши гиперссылка активизируется, и возникает событие **Нажатие кнопки**.
- Выбор элемента из спискаили поля со списком с помощью мыши или с помощью клавиш перемещения курсора с последующим нажатием клавиши ENTER.
- Нажатие клавиши ПРОБЕЛ, когда фокуснаходится на кнопке, флажке, переключателе или выключателе.
- Нажатие клавиши ENTER в форме, которая содержит кнопку с заданным для свойства **По умолчанию (Default)** значением «Да».
- Нажатие клавиши ESC в форме, которая содержит кнопку с заданным для свойства **Отмена (Cancel)** значением «Да».
- Нажатие клавиши доступа.Например, если для свойства кнопки **Подпись (Caption)** задано значение «&Пуск», то нажатие клавиш Alt+П вызывает событие **Нажатие кнопки**.

Наиболее часто событие **Нажатие кнопки** используется для процедуры обработки события или макроса, связанных с кнопкой, при нажатии которой выполняются команды и другие подобные действия. Для других элементов управления с помощью этого события инициируются действия, выполняемые в ответ на одно из описанных выше действий.

Следующее утверждение справедливо только для кнопок: Microsoft Access запускает макрос или процедуру обработки события, указанные в качестве значения свойства **Нажатие кнопки (OnClick)**, при выборе кнопки с помощью клавиши ENTER или с помощью клавиши доступа. Макрос или процедура обработки события выполняется только один раз. Если требуется, чтобы макрос или процедура выполнялись неоднократно, пока кнопка остается нажатой, то следует задать значение «Да» для свойства кнопки **Автоматический повтор (AutoRepeat)**. Для других типов элементов управления событие **Нажатие кнопки** возникает только при нажатии кнопки мыши.

Для кнопки событие **Нажатие кнопки** возникает при выборе данной кнопки пользователем. Кроме того, если фокус не находился на выбираемой пользователем кнопке, то перед событием **Нажатие кнопки** возникают события кнопки **Вход (Enter)** и **Получение фокуса (GotFocus).**

Для элементов управления двойное нажатие кнопки мыши вызывает как событие **Двойное нажатие кнопки (DblClick)**, так и событие **Нажатие кнопки**. Для кнопок двойное нажатие вызывает следующую цепочку событий:

Кнопка вниз ⇒ Кнопка вверх ⇒ Нажатие кнопки ⇒ Двойное нажатие кнопки ⇒ Нажатие кнопки

Применение макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе обработки события **Двойное нажатие кнопки** позволяет отменить второе событие **Нажатие кнопки**. Дополнительные сведения содержатся в разделе справки о событии **Двойное нажатие кнопки**.

Для группы параметров событие **Нажатие кнопки** возникает после того, как пользователь изменяет значение входящего в группу элемента управления, выбирая этот элемент с помощью мыши. Например, если пользователь выбирает переключатель, выключатель или флажок в группе, событие группы **Нажатие кнопки** возникает после событий **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)** для данной группы.

Совет. При необходимости различать нажатие левой, правой и средней кнопок мыши следует использовать события **Кнопка вниз (MouseDown)** и **Кнопка вверх (MouseUp)**.

Событие «Нажатие кнопки» (Click) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtClickMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtClickMOX":1}

Для выполнения макроса, связанного с событием **Нажатие кнопки (Click)**, следует указать имя данного макроса в качестве значения свойства **Нажатие кнопки (OnClick)**.

Дополнительные сведения

Макросы, связанные с событием **Нажатие кнопки**, часто применяются для выполнения определенных действий при нажатии кнопок формы. Например, при нажатии одной кнопки формы пользователь может вывести на печать отчет по записям формы. При нажатии другой кнопки открывается связанная по смыслу форма.

Не допускается выполнение макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Нажатие кнопки**.

Событие «Нажатие кнопки» (Click) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtClickEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtClickEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Нажатие кнопки (Click)**, следует выбрать в качестве значения свойства **Нажатие кнопки (OnClick)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_Click( )
Private Sub *имяЭлемента***_Click( )**

Процедура обработки события **Нажатие кнопки** использует следующий аргумент.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Нажатие кнопки (Click)**.

Дополнительные сведения

При двойном нажатии кнопки мыши на списке возникает следующая цепочка событий:

Кнопка вниз ⇒ Кнопка вверх ⇒ Нажатие кнопки ⇒ Двойное нажатие кнопки

Аналогичные цепочки событий возникают и для других элементов управления. При создании процедур обработки таких связанных событий необходимо обеспечить отсутствие конфликтов в текстах соответствующих программ. (Например, процедура обработки события **Двойное нажатие кнопки** не должна содержать инструкции, отменяющие действия, которые выполняются в процедуре обработки события **Нажатие кнопки**.

Отменить событие **Нажатие кнопки** невозможно.

Событие «Нажатие кнопки» (Click) - Пример процедуры обработки события

В данном примере приведена процедура обработки события **Нажатие кнопки (Click)** для флажка **ТолькоЧтение**. Данная процедура обработки события устанавливает значение свойств **Доступ (Enabled)** и **Блокировка (Locked)** поля «Сумма» для другого элемента управления в форме. При выборе флажка с помощью мыши процедура обработки события проверяет, устанавливается или снимается флажок, а затем задает значения свойств поля, соответственно, разрешающие или запрещающие внесение изменений.

Для проверки работы данного примера следует поместить процедуру обработки события в форму, которая содержит флажок с именем «ТолькоЧтение» и поле «Сумма».

```
Private Sub ТолькоЧтение_Click()
   With Me!Сумма
      If Me!ТолькоЧтение = True Then     ' Если флажок установлен,
         .Enabled = False                ' запрещает редактирование.
         .Locked = True
      Else                               ' Если флажок снят,
         .Enabled = True                 ' включает редактирование.
         .Locked = False
      End If
   End With
End Sub
```

Событие «Двойное нажатие кнопки» (DblClick)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtDblClickC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtDblClickX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acevtDblClickMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS":"acevtDblClickEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acevtDblClickA"}

Событие **Двойное нажатие кнопки (DblClick)** возникает, когда пользователь дважды быстро нажимает и отпускает левую кнопку мыши при установленном на объекте указателе. Время выполнения этого действия не должно превышать предельного времени, заданного в системе для двойных нажатий кнопки мыши.

Для формыданное событие возникает при двойном нажатии кнопки мыши, когда указатель расположен на пустой области формы или на области выделения записи в форме. Для элемента управления в режиме формы оно возникает при двойном нажатии кнопки мыши, когда указатель расположен на элементе управления или на присоединенной к нему подписи. Событие **Двойное нажатие кнопки** возникает после двойного нажатия кнопки мыши пользователем, но до выполнения связанного с данным событием действия (например до того, как Microsoft Access выделит в полеслово, в котором находится курсор.

Примечания

- Событие **Двойное нажатие кнопки** определено для форм, разделовформ и элементов управления в формах. Оно не определено для элементов управления в отчетах.
- Это событие не определено для флажков,переключателей или выключателей в группах переключателей. Оно определено только для самих групп переключателей.
- Данное событие не определено для надписи, присоединенной к другому элементу управления, например, для подписи поля. Оно определено только для «самостоятельных» надписей. Двойное нажатие кнопки мыши для присоединенной надписи имеет тот же эффект, что и для элемента управления, к которому присоединена надпись. При этом возникает обычная цепочка событий элемента управления, а не события присоединенной надписи.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с данным событием, следует указать имя этого макроса или элемент [Процедура обработки события] в качестве значения свойства **Двойное нажатие кнопки (OnDblClick)**.

Для элементов управления результат двойного нажатия кнопки мыши зависит от типа элемента управления. Например, если указатель расположен на слове в поле, то двойное нажатие кнопки мыши приводит к выделению всего слова. Если указатель расположен на элементе управления, содержащем объект OLE, то двойное нажатие кнопки мыши приводит к запуску приложения, в котором был создан этот объект OLE, и открывает объект для редактирования.

Если время, в течение которого происходит двойное нажатие кнопки мыши, превышает предельное время, заданное в системе, то форма, раздел или элемент управления воспринимают данное действие как два независимых события **Нажатие кнопки (Click)**, а не как единое событие **Двойное нажатие кнопки (DblClick)**. Предельное время для двойных нажатий кнопки мыши определяется настройкой параметра **Скорость двойного нажатия** на вкладке **Кнопки мыши** в разделе **Мышь** на панели управления Windows.

Запуск макроса или процедуры обработки события **Двойное нажатие кнопки** позволяет открывать окно или документ при выборе соответствующего значка двойным нажатием кнопки мыши.

Двойное нажатие кнопки мыши приводит к возникновению в элементе управления обоих событий, **Нажатие кнопки** и **Двойное нажатие кнопки**. Если фокус не находился на элементе управления в момент двойного нажатия кнопки мыши, то перед событиями **Нажатие кнопки** и

Двойное нажатие кнопки возникают события элемента управления **Вход (Enter)** и **Получение фокуса (GotFocus).**

В объектах, принимающих события мыши, возникает следующая цепочка событий:

Кнопка вниз ⇒ Кнопка вверх ⇒ Нажатие кнопки ⇒ Двойное нажатие кнопки

При двойном нажатии кнопки мыши при указателе, помещенном на кнопке,возникает следующая цепочка событий:

Кнопка вниз ⇒ Кнопка вверх ⇒ Нажатие кнопки ⇒ Двойное нажатие кнопки ⇒ Кнопка вверх ⇒ Нажатие кнопки

Второе нажатие кнопки мыши может и не иметь последствий (например, если в ответ на первое событие **Нажатие кнопки** запускается макрос или процедура обработки события, открывающие модальное диалоговое окно). Чтобы предотвратить запуск макроса или процедуры обработки события в ответ на второе событие **Нажатие кнопки**, следует включить макрокоманду **ОтменитьСобытие (CancelEvent)** в макрос события **Двойное нажатие кнопки**, или использовать аргумент «Cancel» в процедуре обработки события **Двойное нажатие кнопки**. Вообще говоря, двойное нажатие кнопки мыши при указателе, помещенном на кнопке, не рекомендуется.

При двойном нажатии кнопки мыши на любом элементе управления, отличном от кнопки, второе событие **Нажатие кнопки** не возникает.

Событие «Двойное нажатие кнопки» (DblClick) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtDblClickMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtDblClickMOX":1}

Для выполнения макроса, связанного с событием **Двойное нажатие кнопки (DblClick)**, следует указать имя данного макроса в качестве значения свойства **Двойное нажатие кнопки (OnDblClick)**.

Дополнительные сведения

Запуск макроса при возникновении события **Двойное нажатие кнопки** позволяет изменить действие, выполняемое элементом управления по умолчанию в ответ на двойное нажатие кнопки мыши. Например, можно запустить макрос перед открытием сервера OLE в ответ на двойное нажатие кнопки мыши при указателе, установленном на внедренном объекте. Такой макрос позволяет вывести на экран сообщение, предоставляющее пользователю возможность выбора между выводом объекта на печать и редактированием объекта. Вывод объекта на печать может быть осуществлен с помощью созданной пользователем функции.

Событие **Двойное нажатие кнопки** используется также для закрытия диалогового окна после выбора нужного элемента управления с помощью двойного нажатия кнопки мыши, вместо обычного нажатия кнопки «OK». В этом случае в макрос, связанный с событием **Двойное нажатие кнопки**, включается макрокоманда **Закрыть (Close)** для диалогового окна. Вместо закрытия диалогового окна макрос позволяет скрыть его, задавая значение «Нет» для свойства диалогового окна **Вывод на экран (Visible)** с помощью макрокоманды **ЗадатьЗначение (SetValue)**.

Для отмены действия, выполняемого элементом управления по умолчанию в ответ на двойное нажатие кнопки мыши, следует вызвать макрокоманду **ОтменитьСобытие (CancelEvent)** в макросе события **Двойное нажатие кнопки**. Например, если в описанной выше ситуации пользователь выбрал вывод объекта OLE на печать вместо редактирования объекта, то в макросе следует сначала вызвать созданную пользователем функцию для печати объекта, а затем отменить открытие сервера OLE с помощью макрокоманды **ОтменитьСобытие**.

Событие «Двойное нажатие кнопки» (DblClick) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtDblClickEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtDblClickEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Двойное нажатие кнопки (DblClick)**, следует выбрать в качестве значения свойства **Двойное нажатие кнопки (OnDblClick)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_DblClick(Cancel **As Integer)**
Private Sub *имяЭлемента***_DblClick(**Cancel **As Integer)**

Процедура обработки события **Двойное нажатие кнопки** использует следующие аргументы.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Двойное нажатие кнопки (DblClick)**.
Cancel	Определяет возникновение события **Двойное нажатие кнопки**. Заданное для аргумента «Cancel» значение **True** (–1) отменяет событие **Двойное нажатие кнопки**.

Дополнительные сведения

Запуск процедуры обработки события **Двойное нажатие кнопки** позволяет изменить действие, выполняемое элементом управления по умолчанию в ответ на двойное нажатие кнопки мыши. Например, можно запустить процедуру обработки события перед открытием сервера OLE в ответ на двойное нажатие кнопки мыши при указателе, установленном на внедренном объекте. Такая процедура обработки события позволяет вывести на экран сообщение, предоставляющее пользователю возможность выбора между выводом объекта на печать и редактированием объекта. Вывод объекта на печать может быть осуществлен с помощью созданной пользователем функции. Если выбран вывод объекта OLE на печать вместо его редактирования, то следует сначала вывести объект на печать, а затем с помощью аргумента «Cancel» отменить открытие сервера OLE.

При создании процедур обработки событий **Кнопка вниз (MouseDown)**, **Кнопка вверх (MouseUp)**, **Нажатие кнопки (Click)** и **Двойное нажатие кнопки** необходимо обеспечить отсутствие конфликтов в текстах соответствующих программ. (Например, процедура обработки события **Нажатие кнопки** не должна содержать инструкции, отменяющие действия, которые выполняются в процедуре обработки события **Двойное нажатие кнопки**.)

При двойном нажатии кнопки мыши при указателе, помещенном на кнопке,возникает следующая цепочка событий:

Кнопка вниз ⇒ Кнопка вверх ⇒ Нажатие кнопки ⇒ Двойное нажатие кнопки ⇒ Кнопка вверх ⇒ Нажатие кнопки

Чтобы предотвратить возникновение второго события **Нажатие кнопки**, следует задать для аргумента «Cancel» значение **True** (–1) или вызвать метод **CancelEvent** объекта **DoCmd** в процедуре обработки события **Двойное нажатие кнопки**. Если второе событие **Нажатие кнопки** не будет отменено, то программа процедуры обработки события **Нажатие кнопки** будет выполнена вторично. В следующем примере отменяется второе событие **Нажатие кнопки**:

```
Private Sub CmdButton_DblClick(Cancel As Integer)
   Cancel = True     ' Отменяет второе событие "Нажатие кнопки".
End Sub
```

Событие «Двойное нажатие кнопки» (DblClick) - Пример процедуры обработки события

В данном примере демонстрируется использование процедуры обработки события **Двойное нажатие кнопки (DblClick)** для открытия формы, в которой на экран выводятся записи из таблицы, являющейся источником строк для поля со списком. При двойном нажатии кнопки мыши для поля со списком «Продавец» формы «Заказы» выводится форма «Сотрудники» с записью о сотруднике, данные которого выведены в поле со списком.

Для проверки работы данного примера следует добавить следующую процедуру обработки события в форму «Заказы», которая содержит поле со списком «КодСотрудника». Источником строк для данного поля со списком должна быть та же самая таблица, которая является источником строк для формы «Сотрудники» (или запрос на основе этой таблицы).

```
Private Sub КодСотрудника_DblClick(Cancel As Integer)
   DoCmd.OpenForm "Сотрудники", , , _
      "КодСотрудника = Forms!Заказы!КодСотрудника"
End Sub
```

События «Вход» (Enter), «Выход» (Exit)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtEnterExitC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtEnterExitX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtEnterExitMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtEnterExitEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acevtEnterExitA"}

- Событие **Вход (Enter)** возникает непосредственно перед тем, как элемент управления получит фокус от другого элемента управления в той же форме.
- Событие **Выход (Exit)** возникает непосредственно перед тем, как элемент управления передаст фокус другому элементу управления в той же форме.

Примечания

- События **Вход** и **Выход** определены только для элементов управления в формах и не определены для элементов управления в отчетах.
- Эти события не определены для флажков,переключателей или выключателей в группах переключателей. Они определены только для самих групп переключателей.

Дополнительные сведения

Для выполнения макроса или процедуры обработки событий, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Вход (OnEnter)** или **Выход (OnExit)**.

Событие **Вход** возникает перед переходом фокуса на определенный элемент управления, поэтому макросы или процедуры обработки события **Вход** удобны для вывода на экран каких-либо сведений, относящихся к этому элементу управления; примером может служить вывод в окне сообщений или в небольшой форме информации о данных, которые обычно содержит этот элемент управления, или инструкций по его использованию.

Событие **Вход** возникает до события **Получение фокуса (GotFocus).** Событие **Выход** возникает до события **Потеря фокуса (LostFocus)**.

В отличие от событий **Получение фокуса** и **Потеря фокуса**, события **Вход** и **Выход** происходят не во всех случаях, когда форма получает или теряет фокус. Предположим, что пользователь выбирает флажок в форме, а затем переходит в окно отчета с помощью нажатия кнопки мыши. При выборе флажка возникают события **Вход** и **Получение фокуса**. При переходе к отчету возникает только событие флажка **Потеря фокуса**. Событие **Выход** не возникает (поскольку фокус передается другому окну). Если после этого пользователь опять выберет с помощью мыши флажок формы для того, чтобы вывести форму на передний план, то возникнет только событие **Получение фокуса**. Событие **Вход** при этом не возникает (поскольку этот элемент управления имел фокус в последний момент, когда форма была активной). Событие **Выход** возникнет только при выборе другого элемента управления в той же форме.

Если фокус переходит в другую форму на элемент управления, который не имел фокуса, когда форма была активной, то возникают события **Выход** и **Потеря фокуса** для последнего активного элемента управления в этой форме с последующими событиями **Вход** и **Получение фокуса** для элемента управления, получающего фокус.

При переводе с помощью мыши фокуса с элемента управления в главной форме на элемент управления в подчиненной форме (который не имел фокуса, когда подчиненная форма была активной) возникают следующие события:

Выход (для элемента управления в главной форме)

⇓

Потеря фокуса (для элемента управления в главной форме)

⇓

Вход (для элемента управления в подчиненной форме)

⇓

Выход (для элемента управления в подчиненной форме, имевшего фокус)

⇓

Потеря фокуса (для элемента управления в подчиненной форме, имевшего фокус)

⇓

Вход (для элемента управления в подчиненной форме, получающего фокус)

⇓

Получение фокуса (для элемента управления в подчиненной форме, получающего фокус)

При переходе в подчиненной форме на элемент управления, который последним имел в ней фокус, для этого элемента возникают не два события **Вход** и **Получение фокуса**, а только одно событие **Вход**. Когда фокус переводится с элемента управления в подчиненной форме на элемент управления в главной форме, для элемента управления в подчиненной форме возникают не два события **Выход** и **Потеря фокуса**, а только одно событие **Выход**; при этом для элемента управления в главной форме возникают события **Вход** и **Получение фокуса**.

Примечание. Часто фокус переводится на другой элемент управления с помощью мыши или с помощью клавиатуры, например, нажатием клавиши TAB. При этом наряду с описанными в данном разделе событиями возникают события мыши или события клавиатуры.

События «Вход» (Enter), «Выход» (Exit) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtEnterExitMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtEnterExitMOX":1}

Для выполнения макроса, связанного с событием **Вход (Enter)** или **Выход (Exit)**, следует указать имя данного макроса в качестве значения свойства **Вход (OnEnter)** или **Выход (OnExit)**.

Дополнительные сведения

При возникновении событий **Вход** или **Выход** запуск макроса, содержащего макрокоманду **КЭлементуУправления (GoToControl)** позволяет изменить последовательность перехода в зависимости от текущего значения элемента управления. Например, если пользователь выберет пункт «Холост» в списке «СемейноеПоложение», можно исключить поле «ИмяСупруги» из последовательности переходов.

Для отмены выхода из элемента управления следует вызвать макрокоманду **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Выход**. Вызов макрокоманды **ОтменитьСобытие** в макросе, связанном с событием **Вход**, невозможен. Однако допускается использование макрокоманды **КЭлементуУправления** макросе, связанном с событием **Вход**, для перевода фокуса на другой элемент управления.

События «Вход» (Enter), «Выход» (Exit) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtEnterExitEPC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtEnterExitEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Вход (Enter)** или **Выход (Exit)**, следует выбрать в качестве значения свойства **Вход (OnEnter)** или **Выход (OnExit)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub *имяЭлемента*_**Enter( )**
Private Sub *имяЭлемента*_**Exit(**Cancel **As Integer)**

Процедура обработки события **Вход** использует следующий аргумент.

Аргумент	**Описание**
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Вход (Enter)**.

Процедура обработки события **Выход** использует следующие аргументы.

Аргумент	**Описание**
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Выход (Exit)**.
Cancel	Определяет возникновение события **Выход**. Заданное для аргумента «Cancel» значение **True** (–1) отменяет событие **Выход**.

Дополнительные сведения

Отмена события **Вход** не допускается.

События «Вход» (Enter), «Выход» (Exit) - Пример процедур обработки событий

В данном примере две процедуры обработки событий связываются с полем «Фамилия». Процедура обработки события **Вход (Enter)** выводит сообщение, подсказывающее пользователю, какие данные следует ввести в поле. Процедура обработки события **Выход (Exit)** выводит диалоговое окно, в котором пользователю предлагается сохранить изменения перед передачей фокуса другому элементу управления. Если пользователь нажимает кнопку **Отмена**, то аргументу «Cancel» присваивается значение **True** (–1). При этом фокус переводится на другой элемент управления без сохранения изменений. Если пользователь нажимает кнопку **OK**, то изменения сохраняются, после чего фокус переводится на другой элемент управления.

Для проверки работы данного примера следует поместить процедуру обработки события в форму, которая содержит поле «Фамилия».

```
Private Sub Фамилия_Enter()
   MsgBox "Введите фамилию."
End Sub

Private Sub Фамилия_Exit(Cancel As Integer)
   Dim strMsg As String

   strMsg = "Введена фамилия '" & Me!Фамилия & "' ." & _
      vbCrLf & "Это правильно?"
   If MsgBox(strMsg, vbYesNo) = vbNo Then
      Cancel = True        ' Отменяет выход.
   Else
      Exit Sub             ' Выход с сохранением изменений.
   End If
End Sub
```

Событие «Форматирование» (Format)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtFormatC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtFormatX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acevtFormatMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS":"acevtFormatEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acevtFormatA"}

Событие **Форматирование (Format)** возникает после того, как Microsoft Access определяет, какие данные входят в разделотчета, но до форматирования раздела для печати или предварительного просмотра.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с данным событием, следует указать имя этого макроса или элемент [Процедура обработки события] в качестве значения свойства **Форматирование (OnFormat)**

Событие **Форматирование** возникает для каждого раздела отчета. Это позволяет создавать сложные отчеты, содержащие результаты вычислений с использованием данных из всех разделов, включая и разделы, которые не будут напечатаны.

Для области данных отчета событие **Форматирование** возникает для каждой записи раздела непосредственно перед тем, как Microsoft Access форматирует содержимое этой записи. Макросы или процедуры обработки события **Форматирование** имеют доступ к данным только в текущей записи.

Для заголовков групп отчета событие **Форматирование** возникает для каждой новой группы. Макросы или процедуры обработки события **Форматирование** имеют доступ к данным заголовка группы и к содержимому первой записи области данных. Для примечаний групп отчета событие **Форматирование** возникает для каждой новой группы. Макросы или процедуры обработки события **Форматирование** имеют доступ к данным в примечании группы и к содержимому последней записи области данных.

Выполнение макроса или процедуры обработки события при возникновении события **Форматирование** позволяет использовать данные из текущей записи для внесения в отчет изменений, влияющих на макет страницы отчета. Например, можно вывести или скрыть поздравительное сообщение рядом с данными об итогах месячных продаж продавца в отчете о продажах, в зависимости от суммы продаж. После того как элемент управления будет выведен или скрыт, Microsoft Access проведет форматирование раздела с использованием значений свойств описания макета, таких как **Расширение (CanGrow)**, **Сжатие (CanShrink)**, **Не выводить повторы (HideDuplicates)**, **Не разрывать (KeepTogether)** и **Вывод на экран (Visible)**.

Для изменений, которые не затрагивают формат страницы отчета, а также для процедур обработки событий или макросов, которые должны выполняться только после проведения форматирования страницы (например макроса, который распечатывает итоговые суммы по странице), следует использовать событие **Печать (Print)** раздела отчета.

В некоторых случаях Microsoft Access возвращается к предыдущим разделам отчета для выполнения форматирования в несколько проходов. При возврате к каждой предыдущей секции отчета происходит событие **Возврат (Retreat)**, а событие **Форматирование** возникает несколько раз для каждого раздела. В процедурах обработки событий или в макросах, связанных с событием **Возврат**, допускается отмена всех изменений, которые были внесены при возникновении события **Форматирование** в данном разделе. Этот прием применяется в тех случаях, когда в процедурах обработки события или в макросах, связанных с событием **Форматирование**, выполняются действия, например, вычисление итоговых сумм по странице или изменение размеров раздела, которые необходимо выполнить для каждого раздела только один раз.

Событие «Форматирование» (Format) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtFormatMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtFormatMOX":1}

Для выполнения макроса, связанного с событием **Форматирование (Format)**, следует указать имя данного макроса в качестве значения свойства **Форматирование (OnFormat)**.

Дополнительные сведения

Для определения ситуации, в которой событие **Форматирование** происходит для некоторого разделанесколько раз, используется свойство **FormatCount**. Например, если раздел не умещается на одной странице и часть его переносится на следующую страницу отчета, то событие **Форматирование** возникает в данном разделе по одному разу для каждой страницы, и свойство **FormatCount** получает значение 2. Значение свойства **FormatCount** позволяет определить, сколько раз для каждого раздела выполняется макрос, связанный с событием **Форматирование**.

Вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Форматирование**, позволяет отменить форматирование раздела. При этом Microsoft Access не выполняет форматирование раздела для печати и переходит к печати следующего раздела. Макрокоманда **ОтменитьСобытие** позволяет пропустить раздел отчета при печати, не оставляя при этом пустого места на печатной странице.

Событие «Форматирование» (Format) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtFormatEPC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtFormatEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Форматирование (Format)**, следует выбрать в качестве значения свойства **Форматирование (OnFormat)** элемент [Процедура обработки события] и нажать кнопку **Построить** 

Синтаксис

Private Sub *имяРаздела*_**Format(**Cancel **As Integer,** FormatCount **As Integer)**

Процедура обработки события **Форматирование** использует следующие аргументы.

Аргумент	Описание
имяРаздела	Имя раздела отчета, к которому применяется процедура обработки события **Форматирование (Format)**.
Cancel	Определяет выполнение форматирования раздела. Заданное для аргумента «Cancel» значение **True** (–1) отменяет форматирование раздела.
FormatCount	**Целое** значение, показывающее, возникало ли событие **Форматирование** раздела несколько раз. Например, если раздел не умещается на одной странице, и часть его переносится на следующую страницу отчета, в Microsoft Access для аргумента «FormatCount» задается значение 2.

Дополнительные сведения

При отмене события форматирования Microsoft Access не форматирует раздел для печати и переходит к печати следующего раздела. Отмена форматирования позволяет пропустить раздел отчета при печати, не оставляя при этом пустого места на печатной странице.

Событие «Форматирование» (Format) - Пример процедуры обработки события

В данном примере выводится или скрывается поздравительное сообщение рядом с вычисляемым элементом управления, в котором содержатся данные об итогах месячных продаж продавцов. В разделах отчета, где итоговая сумма продаж превышает план по продажам, в надписи с именем «Поздравление» при печати выводится сообщение «Отличная работа. План перевыполнен!». В разделах отчета, где итоговая сумма продаж меньше намеченного плана, данная надпись делается скрытой.

Для проверки работы данного примера следует поместить процедуру обработки события в отчет с именем «Отчет о продажах», который содержит надпись «Поздравление», поле с именем «ИтоговаяСумма» (вычисляемый элемент управления, в котором выводится итоговая сумма продаж) и область данных с именем «Продажи».

```
Private Sub Продажи_Format(Cancel As Integer, _
      FormatCount As Integer)
   Const conSalesGoal = 1000
   If Me!ИтоговаяСумма > conSalesGoal Then
      Me!Поздравление.Visible = True
   Else
      Me!Поздравление.Visible = False
   End If
End Sub
```

События «Открытие» (Open), «Закрытие» (Close)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїS":"acevtOpenCloseC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtOpenCloseX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtOpenCloseMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtOpenCloseEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acevtOpenCloseA"}

- Событие **Открытие (Open)** возникает после открытия формы, но до вывода первой записи. Для отчетовэто событие возникает перед открытием отчета для печати или предварительного просмотра.
- Событие **Закрытие (Close)** возникает, когда форма или отчет закрываются и удаляются с экрана.

Дополнительные сведения

Для выполнения макроса или процедуры обработки событий, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Открытие (OnOpen)** или **Закрытие (OnClose)**.

Выполнение макроса или процедуры обработки события при возникновении события формы **Открытие** позволяет закрыть другое окно или перевести фокус на определенный элемент управления формы. Можно также запустить макрос или процедуру обработки события, которые позволяют запросить у пользователя необходимые сведения перед открытием или выводом на печать формы или отчета. Например, с помощью макроса или процедуры обработки события **Открытие** можно открыть специальное диалоговое окно, в котором пользователь вводит условия отбора записей, выводящиеся в форме, или диапазон дат для отчета.

Событие **Открытие** не возникает при активизации уже открытой формы, например, при переходе в эту форму из другого окна Microsoft Access или при переводе открытой формы в верхний слой с помощью макрокоманды **ОткрытьФорму (OpenForm)**. Однако в таких ситуациях возникает событие **Включение (Activate)**.

При открытии пользователем формы с базовым запросом Microsoft Access выполняет базовый запрос формы до запуска макроса или процедуры обработки события **Открытие**. Однако при открытии отчета с базовым запросом макрос или процедура обработки события **Открытие** запускается до выполнения базового запроса отчета. Это позволяет пользователю задать условия отбора до открытия отчета, например, в специальном диалоговом окне, которое открывается в ответ на событие **Открытие**.

Если в приложении загружено одновременно несколько форм, то для вывода и скрытия специальных панелей инструментов при перемещении фокуса между формами следует использовать события **Включение (Activate)** и **Отключение (Deactivate)** вместо события **Открытие**.

Событие **Открытие** возникает до события **Загрузка (Load)**, которое генерируется при открытии формы и выводе на экран записей формы.

При первом открытии формы возникает следующая цепочка событий:

Открытие ⇒ Загрузка ⇒ Изменение размера ⇒ Включение ⇒ Текущая запись

Событие **Закрытие (Close)** происходит после события **Выгрузка (Unload)**, которое возникает после закрытия формы, но до удаления формы с экрана.

При закрытии формы возникает следующая цепочка событий:

Выгрузка ⇒ Отключение ⇒ Закрытие

При возникновении события **Закрытие** разработчик может открыть другое окно или вывести запрос для пользователя с целью занесения имени пользователя в журнал для учета работы с формой или отчетом.

Разработчик, выбирающий, какое из двух событий, **Открытие** или **Загрузка**, использовать для запуска макроса или процедуры обработки события, должен учитывать, что событие **Открытие** можно отменить, а событие **Загрузка** - нет. Например, при динамическом построении источника записей для формы в процедуре обработки события формы **Открытие** возможна отмена открытия формы в случае, когда в источнике отсутствуют записи. Аналогично этому, допускается отмена события **Выгрузка**, но не допускается отмена события **Открытие**.

События «Открытие» (Open), «Закрытие» (Close) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtOpenCloseMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtOpenCloseMOX":1}

Для выполнения макроса, связанного с событием **Открытие (Open)** или **Закрытие (Close)**, следует указать имя данного макроса в качестве значения свойства **Открытие (OnOpen)** или **Закрытие (OnClose)**.

Дополнительные сведения

Макрокоманда **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Открытие**, позволяет отменить открытие формы или отчета. В макросе, связанном с событием **Закрытие**, вызов макрокоманды **ОтменитьСобытие** невозможен. Однако допускается отмена закрытия формы с помощью вызова макрокоманды **ОтменитьСобытие** в макросе, связанном с событием **Выгрузка (Unload)** формы.

При необходимости обратиться к элементам управления из макроса, связанного с событием **Открытие**, следует сначала передать фокус соответствующему элементу управления или записи. Например, для присвоения значения элементу управления в форме с помощью макрокоманды **ЗадатьЗначение (SetValue)** в макросе события **Открытие** необходимо сначала вызвать макрокоманду **КЭлементуУправления (GoToControl)** или **НаЗапись (GoToRecord)** для получения доступа к этому элементу управления или к содержащей его записи.

События «Открытие» (Open), «Закрытие» (Close) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtOpenCloseEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtOpenCloseEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Открытие (Open)** или **Закрытие (Close)**, следует выбрать в качестве значения свойства **Открытие (OnOpen)** или **Закрытие (OnClose)**, элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_Open(Cancel **As Integer)**
Private Sub Report_Open(Cancel **As Integer)**
Private Sub Form_Close( )
Private Sub Report_Close( )

Процедура обработки события **Открытие (Open)** использует следующий аргумент.

Аргумент	Описание
Cancel	Определяет открытие формы или отчета. Заданное для аргумента «Cancel» значение **True** (–1) отменяет открытие формы или отчета.

Дополнительные сведения

Отмена события **Закрытие (Close)** не допускается.

При необходимости обратиться к элементам управления из процедуры обработки события **Открытие (Open)**, следует сначала передать фокус форме, соответствующему элементу управления или записи с помощью метода **SetFocus** или с помощью методов **GoToRecord** или **GoToControl** объекта **DoCmd**.

Если в приложении в каждый момент времени загружена только одна форма, то можно включить инструкции вывода или скрытия панели инструментов в процедуру обработки события **Открытие (Open)** для формы. В следующем примере при открытии формы выводится панель инструментов «НоваяПанель»:

```
Sub Form_Open(Cancel As Integer)
   DoCmd.ShowToolbar "НоваяПанель", acToolbarYes
End Sub
```

События «Открытие» (Open), «Закрытие» (Close) - Пример процедуры обработки события

В данном примере демонстрируется отмена открытия формы при нажатии пользователем кнопки «Нет». На экран выводится окно сообщения с приглашением ввести данные о заказе. Если пользователь нажимает кнопку «Нет», то форма «Заказано» не открывается.

Для проверки работы данного примера следует поместить процедуру обработки события в форму.

```
Private Sub Form_Open(Cancel As Integer)
   Dim intReturn As Integer
   intReturn = MsgBox("Ввести сведения о заказе?", vbYesNo)
   Select Case intReturn
      Case vbYes
         DoCmd.OpenForm "Заказано" ' Открывает форму "Заказано".
      Case vbNo
         MsgBox "Не забудьте ввести сведения о заказе до 17.00."
         Cancel = True                    ' Отменяет событие "Открытие".
   End Select
End Sub
```

Событие «Печать» (Print)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtPrintC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtPrintX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtPrintMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtPrintEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acevtPrintA"}

Событие **Печать (Print)** возникает после форматирования данных разделаотчета для печати, но до вывода раздела на печать.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с данным событием, следует указать имя этого макроса или элемент [Процедура обработки события] в качестве значения свойства **Печать (OnPrint)**.

Для области данных отчета событие **Печать** возникает для каждой записи раздела отчета непосредственно перед тем, как Microsoft Access выводит содержимое записи на печать. Макросы или процедуры обработки события **Печать** имеют доступ к данным только в текущей записи.

Для заголовков групп отчета событие **Печать** возникает для каждой новой группы. Макросы или процедуры обработки события **Печать** имеют доступ к данным заголовка группы и к содержимому первой записи области данных. Для примечаний групп отчета событие **Печать** возникает для каждой новой группы. Макросы или процедуры обработки события **Печать** имеют доступ к данным в примечании группы и к содержимому последней записи области данных.

Событие **Печать** используется для запуска макроса или процедуры обработки событий только после того, как Microsoft Access подготовит данные для печати на текущей странице. Например, таким способом вычисляется итоговая сумма значений для текущей страницы, которая будет выведена в верхнем или нижнем колонтитуле.

Для изменения макета страницы, например для вывода или скрытия элементов управления, следует использовать событие **Форматирование (Format)**.

Событие **Печать** возникает только для разделов, выводящихся на печать. При необходимости обеспечить доступ к данным в разделах, которые не выводятся на печать(например, при расчетах сумм с накоплением с учетом не печатающихся разделов), вместо события **Печать** следует использовать событие **Форматирование**.

Событие **Страница (Page)** возникает после всех событий **Форматирование** в отчете и после события страницы **Печать**, но до начала физической печати страницы.

Событие «Печать» (Print) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtPrintMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtPrintMOX":1}

Для выполнения макроса, связанного с событием **Печать (Print)**, следует указать имя данного макроса в качестве значения свойства **Печать (OnPrint)**.

Дополнительные сведения

Для определения ситуации, в которой событие **Печать** происходит для некоторой записи несколько раз, используется свойство **PrintCount**. Например, если только часть записи помещается на одной странице, а остальное печатается на следующей странице, то событие **Печать** возникает дважды, и свойство **PrintCount** получает значение 2. Значение свойства **PrintCount** используется в макросах события **Печать**, которые должны быть выполнены для каждой записи только один раз. Примером может служить макрос для вычисления суммы с накоплением по всем записям, которая выводится в нижнем колонтитуле.

Использование макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе события **Печать** позволяет отменить печать раздела. При печати отчета место, где должен был располагаться отмененный раздел, Microsoft Access оставит на странице пустым. Макрокоманда **ОтменитьСобытие** позволяет пропустить при печати отчета запись и оставить вместо нее пустое место на странице.

Макрос, связанный с событием **Печать**, запускается только для тех разделов отчета, которые выводятся на печать.

Примечание. Нельзя использовать макрос, связанный с событием **Печать**, для определения свойств всех объектов.

Событие «Печать» (Print) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtPrintEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtPrintEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Печать (Print)**, следует выбрать в качестве значения свойства **Печать (OnPrint)**. элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub *имяРаздела***_Print(**Cancel **As Integer,** PrintCount **As Integer)**

Процедура обработки события **Печать** использует следующие аргументы.

Аргумент	Описание
имяРаздела	Имя разделаотчета, к которому применяется процедура обработки события **Печать (Print)**.
Cancel	Определяет, выводится ли раздел на печать. Заданное для аргумента «Cancel» значение **True** (–1) отменяет печать раздела.
PrintCount	**Целое** значение, определяющее число событий **Печать** для записи. Например, если запись не помещается на странице, и ее часть переносится на следующую страницу, то событие **Печать** возникает дважды, и аргумент «PrintCount» получает значение 2.

Дополнительные сведения

Для отмены вывода на печать раздела отчета следует присвоить аргументу «Cancel» значение **True** (–1) или вызвать метод **CancelEvent** объекта **DoCmd**. Microsoft Access оставляет пустое место на странице там, где должен был находиться раздел отчета. Использование метода **CancelEvent** или аргумента «Cancel» позволяет пропустить при печати отчета запись и оставить вместо нее пустое место на странице отчета.

Примечание. Нельзя использовать процедуру обработки события **Печать** для определения свойств всех объектов.

Событие «Печать» (Print) - Пример процедуры обработки события

В данном примере с помощью аргумента «PrintCount» определяется число событий **Печать (Print)** для раздела отчета. Если данное событие возникает не первый раз, то текущее событие раздела **Печать** отменяется.

Для проверки работы данного примера следует поместить процедуру обработки события в отчет, содержащий раздел с именем «СведенияОКлиентах».

```
Private Sub СведенияОКлиентах_Print(Cancel As Integer, _
      PrintCount As Integer)
   If PrintCount > 1 Then Cancel = True
End Sub
```

Событие «Текущая запись» (Current)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtCurrentC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtCurrentX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtCurrentMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtCurrentEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtCurrentA"}

Событие **Текущая запись (Current)** возникает, когда запись получает фокус и становится текущей, а также при обновлении экрана или при выполнении повторного запроса для формы.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с данным событием, следует указать имя этого макроса или элемент [Процедура обработки события] в качестве значения свойства **Текущая запись (OnCurrent)**.

Данное событие происходит при открытии формы и при каждом переходе фокуса от одной записи к другой. Microsoft Access выполняет макрос или процедуру обработки события **Текущая запись** до вывода первой или следующей записи.

Запуск макроса или выполнение процедуры обработки события формы **Текущая запись** позволяет вывести сообщение или выполнить синхронизацию записей в другой форме с текущей записью. Например, когда запись о клиенте становится текущей, можно вывести сведения о предыдущем заказе данного клиента. Если текущей становится запись о поставщике товаров, в форме «Поставщики» можно вывести список товаров, производимых данным поставщиком. Это событие используется также для вычислений, основанных на данных из текущей записи, или для изменения макета формы в зависимости от содержимого текущей записи.

Если в ответ на событие **Открытие (Open)** в макросе или в процедуре обработки события выполняется макрокоманда **КЭлементуУправления (GoToControl)** или **НаЗапись (GoToRecord)** или соответствующий метод объекта **DoCmd,** то возникает событие **Текущая запись**.

Событие **Текущая запись** возникает также при обновлении экрана формы или при выполнении повторного запроса по базовым таблицам или запросам формы. Например, оно возникает при выборе в меню **Записи** команды **Удалить фильтр** при вызове в макросе макрокоманды **Обновление** (Requery) или при вызове метода **Requery** в программе Visual Basic.

При первом открытии формы возникает следующая цепочка событий:

Открытие ⇒ Загрузка ⇒ Изменение размера ⇒ Включение ⇒ Текущая запись

Событие «Текущая запись» (Current) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtCurrentMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtCurrentMOX":1}

Для выполнения макроса, связанного с событием **Текущая запись (Current)**, следует указать имя этого макроса в качестве значения свойства **Текущая запись (OnCurrent)**.

Дополнительные сведения

В макросе, связанном с событием **Текущая запись**, нельзя использовать макрокоманду **ОтменитьСобытие (CancelEvent).** Однако допускается перевод фокуса на другую запись с помощью макрокоманды **НаЗапись (GoToRecord)**.

Событие «Текущая запись» (Current) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtCurrentEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtCurrentEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Текущая запись (Current)**, следует выбрать в качестве значения свойства **Текущая запись (OnCurrent)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_Current( )

Дополнительные сведения

Данное событие нельзя отменить с помощью метода **CancelEvent** объекта **DoCmd**. Однако с помощью метода **GoToRecord** объекта **DoCmd** можно перевести фокус на другую запись.

Событие «Текущая запись» (Current) - Примеры процедур обработки события

В данном примере в процедуре обработки события **Текущая запись (Current)** проверяется значение переключателя с именем «ПоставкиПрекращены». Если данный переключатель включен, то для фоновой области поля «НаименованиеТовара» задается красный цвет, что указывает на прекращение поставок данного товара.

Для проверки работы данного примера следует добавить процедуру обработки события в форму, содержащую переключатель с именем «ПоставкиПрекращены» и поле «НаименованиеТовара».

```
Private Sub Form_Current()
   If Me!ПоставкиПрекращены Then
      Me!НаименованиеТовара.BackColor = 255
   EndIf
End Sub
```

В следующем примере для свойства формы **Подпись (Caption)** задается значение, равное значению поля «НазваниеПоставщика». При перемещении фокуса от записи к записи в заголовке формы будет выводиться текущее название фирмы-поставщика.

Для проверки работы данного примера следует добавить процедуру обработки события в форму, содержащую поле с именем «НазваниеПоставщика».

```
Private Sub Form_Current()
   Me.Caption = Me!НазваниеПоставщика
End Sub
```

События «Включение» (Activate), «Отключение» (Deactivate)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtActivateDeactivateC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtActivateDeactivateX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtActivateDeactivateMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtActivateDeactivateEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtActivateDeactivateA"}

- Событие **Включение (Activate)** возникает, когда форма или отчет получает фокус и становится активным окном.
- Событие **Отключение (Deactivate)** возникает при переводе фокуса из окна формы или отчета в окно таблицы, запроса, формы, отчета, макроса, модуля или в окно базы данных.

Примечание. Событие **Отключение** не возникает при переводе фокуса из формы или отчета в диалоговое окно, в форму, для которой свойство **Всплывающее окно (PopUp)** имеет значение «Да», или в окно другого приложения.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Включение (OnActivate)** или **Отключение (OnDeactivate)**.

Формы и отчеты становятся активными при их открытии, выборе с помощью мыши или выборе элемента управления в окне формы или отчета, а также в результате выполнения метода **SetFocus** в программе Visual Basic (только для форм).

Событие **Включение** возникает только для видимых форм и отчетов.

Событие **Включение** возникает до события **Получение фокуса (GotFocus)**; событие **Отключение** возникает после события **Потеря фокуса (LostFocus)**.

При переключении между окнами двух открытых форм для формы, теряющей фокус, возникает событие **Отключение (Deactivate)**, а для формы, получающей фокус, возникает событие **Включение (Activate)**. Если эти формы не содержат видимых доступных элементов управления, событие **Потеря фокуса (LostFocus)** возникает в первой форме до события **Отключение (Deactivate)**, а событие **Получение фокуса (GotFocus)** возникает во второй форме после события **Включение (Activate)**.

При первом открытии формы возникает следующая цепочка событий:

Открытие ⇒ Загрузка ⇒ Изменение размера ⇒ Включение ⇒ Текущая запись

При закрытии формы возникает следующая цепочка событий:

Выгрузка ⇒ Отключение ⇒ Закрытие

События «Включение» (Activate), «Отключение» (Deactivate) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtActivateDeactivateMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtActivateDeactivateMOX":1}

Для выполнения макроса, связанного с событием **Включение (Activate)** или **Отключение (Deactivate)**, следует указать имя этого макроса в качестве значения свойства **Включение (OnActivate)** или **Отключение (OnDeactivate)**.

Дополнительные сведения

Макрокоманда **ПанельИнструментов** (ShowToolbar) используется в макросах, связанных с событием **Включение** или **Отключение**, для вывода на экран или скрытия специальных панелей инструментов. Событие **Включение** или **Отключение** следует использовать для вывода на экран или скрытия специальных панелей инструментов, если одновременно имеется несколько загруженных форм. Если каждый раз загруженной является только одна форма, допускается использование событий **Открытие (Open)** и **Закрытие (Close)**.

При скрытии специальной панели инструментов в форме с помощью макроса, связанного с событием **Отключение**, необходимо также обеспечить скрытие панели инструментов в ответ на событие **Выгрузка (Unload)** формы, поскольку событие **Отключение** не возникает при выгрузке формы. Не следует выводить или скрывать специальную панель инструментов в форме с помощью событий **Получение фокуса (GotFocus)** или **Потеря фокуса (LostFocus)**, так как эти события не возникают в формах, содержащих доступные элементы управления.

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросах, связанных с событием **Включение** или **Отключение**.

События «Включение» (Activate), «Отключение» (Deactivate) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtActivateDeactivateEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtActivateDeactivateEPX":1}

Для создания процедуры обработки событий, которая выполняется при возникновении события **Включение (Activate)** или **Отключение (Deactivate)**, следует выбрать в качестве значения свойства **Включение (OnActivate)** или **Отключение (OnDeactivate)**. элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_Activate( )
Private Sub Report_Activate( )
Private Sub Form_Deactivate( )
Private Sub Report_Deactivate( )

Дополнительные сведения

Процедуры обработки событий **Включение** и **Отключение** используются для вывода на экран или скрытия специальных панелей инструментов. При скрытии специальной панели инструментов в форме с помощью процедуры обработки события **Отключение** необходимо также обеспечить скрытие панели инструментов в ответ на событие **Выгрузка (Unload)** формы, поскольку событие **Отключение** не возникает при выгрузке формы. Не следует скрывать специальную панель инструментов в форме с помощью событий **Получение фокуса (GotFocus)** или **Потеря фокуса (LostFocus)**, так как эти события не возникают в формах, содержащих доступные элементы управления.

Отмена события **Включение** или **Отключение** не допускается.

События «Включение» (Activate), «Отключение» (Deactivate) - Пример процедур обработки событий

В данном примере демонстрируется вывод специальной панели инструментов с именем «НоваяПанель» при получении фокуса формой и скрытие этой панели инструментов при переходе фокуса в другое окно.

Для проверки работы данного примера следует добавить процедуру обработки события в форму, содержащую специальную панель инструментов с именем «НоваяПанель».

```
Private Sub Form_Activate()
    ' Выводит панель.
    DoCmd.ShowToolbar "НоваяПанель", acToolbarYes
End Sub

Private Sub Form_Deactivate()
    ' Скрывает панель.
    DoCmd.ShowToolbar "НоваяПанель", acToolbarNo
End Sub
```

Событие «Изменение» (Change)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtChangeC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtChangeX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS":"acevtChangeMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtChangeEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtChangeA"}

Событие **Изменение (Change)** возникает при изменении содержимого поля или текстового поля в поле со списком. Оно также возникает в элементе управления «Набор вкладок»при перемещении с одной страницы на другую. Данное событие также возникает, к примеру, при вводе символа непосредственно в поле или в поле со списком, а также при изменении значения свойства **Текст (Text)** элемента управления с помощью макроса или в программе Visual Basic.

Примечания

- Задание значения для элемента управления с помощью макроса или в процедуре Visual Basic не приводит к возникновению данного события в элементе управления. Необходимо непосредственно ввести значение в элемент управления или определить значение свойства **Текст** элемента управления.
- Данное событие определено только для элементов управления в формах и не определено для элементов управления в отчетах.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Изменение (OnChange)**.

Запуск макроса или выполнение процедуры обработки события при возникновении события **Изменение** позволяет согласовывать вывод данных в разных элементах управления. Возможен также вывод данных или формулы в одном элементе управления, а результатов вычислений в другом.

Событие **Изменение** не возникает при изменении значения вычисляемого элемента управления или при выборе элемента из списка в поле со списком.

Примечание. Событие **Изменение (Change)** может вызвать каскадные события. Подобная ситуация возникает, когда макрос или процедура обработки события, выполняемые в ответ на событие элемента управления **Изменение**, сами изменяют содержимое элемента управления - например, если они изменяют значение свойства, которое определяет значение элемента управления, такого как свойство **Текст** поля. Для того чтобы предотвратить возникновение каскадных событий, необходимо придерживаться следующих правил.

- По возможности, не связывать с событием **Изменение** элемента управления макрос или процедуру обработки события, которые изменяют содержимое этого элемента управления.
- Избегать создания двух или нескольких элементов управления, события **Изменение** которых влияют друг на друга - например, создание двух полей, каждое из которых обновляет другое.

Изменение содержимого поля или поля со списком с помощью клавиатуры вызывает события клавиатуры. в дополнение к событиям элемента управления, таким как **Изменение**. Например, если пользователь переходит на новую запись и вводит в поле записи символ ANSI, возникает следующая цепочка событий:

Клавиша вниз ⇒ Нажатие клавиши ⇒ До вставки ⇒ Изменение ⇒ Клавиша вверх

События **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)** для элементов управления «поле» или «поле со списком» возникают после того, как пользователь ввел новые или изменил существующие данные в этом элементе управления и перешел к другому элементу управления (или выбрал команду **Сохранить запись** в меню **Записи**), и

следовательно, после всех событий **Изменение** для данного элемента управления.

Для поля со списком, у которого свойство **Ограничиться списком (LimitToList)** имеет значение «Да», событие **Отсутствие в списке (NotInList)** возникает после ввода значения, отсутствующего в списке, при попытке перейти на другой элемент управления или сохранить запись. Это событие также возникает после всех событий **Изменение (Change)** поля со списком. В данном случае события **До обновления** и **После обновления** поля со списком не возникают, поскольку значение, отсутствующее в списке, не принимается Microsoft Access.

Событие «Изменение» (Change) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtChangeMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtChangeMOX":1}

Для выполнения макроса, связанного с событием **Изменение (Change)**, следует указать имя этого макроса в качестве значения свойства **Изменение (OnChange)**.

Дополнительные сведения

Макрос события **Изменение** используется для синхронизации данных в двух элементах управления; при изменении одного элемента управления выполняется действие, изменяющее второй элемент управления.

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросах, связанных с событием **Изменение**.

Событие «Изменение» (Change) - Процедуры обработки события

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtChangeEPC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtChangeEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Изменение (Change)**, следует выбрать в качестве значения свойства **Изменение (OnChange)**. элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub *имяЭлемента***_Change( )**

Процедура обработки события **Изменение** использует следующий аргумент.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Изменение (Change)**.

Дополнительные сведения

Отмена события **Изменение** не допускается.

Для предотвращения каскадных событийследует избегать создания процедур обработки события **Изменение** элемента управления, изменяющих содержимое этого элемента управления (например, с помощью задания значения свойства **Текст (Text)** для элемента управления). При необходимости создать подобную процедуру обработки событий следует описать специальную переменную и установить ее значение, которое будет предотвращать дополнительные изменения, пока текущее изменение не завершено.

Событие «Изменение» (Change) - Пример процедуры обработки события

В данном примере подсчитывается число символов, введенных в поле. При вводе более чем трех символов на экран выводится сообщение, и значение счетчика сбрасывается до нуля.

Для проверки работы данного примера следует добавить процедуру обработки события в форму с именем «Заказы», содержащую поле «ИндексПолучателя».

```
Private Sub ИндексПолучателя_Change()
   ' Переменная описывается как статическая для сохранения
   ' значения в промежутках между вызовами процедуры.
   Static intLetter As Integer

   ' Проверка числа введенных символов.
   ' При вводе более чем трех символов счетчик сбрасывается до нуля.
   If intLetter < 3 Then
      intLetter = intLetter + 1
   Else
      MsgBox "Допускается ввод только трех символов."
      intLetter = 0
      Forms!Заказы!ИндексПолучателя.Undo
   End If
End Sub
```

Событие «Ошибка» (Error)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtErrorC;vafctError;vaobjErr"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtErrorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtErrorMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS":"acevtErrorEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acevtErrorA"}

Событие **Ошибка (Error)** имеет место при возникновении ошибки выполнения Microsoft Access в формах или отчетах, имеющих фокус.В это число входят ошибки ядра базы данных Microsoft Jet, но не входят ошибки выполнения Visual Basic.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Ошибка (OnError)**.

Запуск макроса или выполнение процедуры обработки события при возникновении события **Ошибка** позволяет перехватить сообщение об ошибке Microsoft Access и вывести на экран специальное сообщение, содержащее информацию, специфическую для выполняющегося приложения.

Событие «Ошибка» (Error) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtErrorMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtErrorMOX":1}

Для выполнения макроса, связанного с событием **Ошибка (Error)**, следует указать имя этого макроса в качестве значения свойства **Ошибка (OnError)**.

Дополнительные сведения

Макрос, связанный с событием **Ошибка**, позволяет при каждом возникновении ошибки выполнения выводить сообщение общего характера. Однако макросы не возвращают коды ошибок и не позволяют определить, какая именно ошибка имела место; поэтому с данным событием обычно связывают процедуры обработки события.

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросах, связанных с событием **Ошибка**.

Событие «Ошибка» (Error) - Процедуры обработки события

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtErrorEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtErrorEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Ошибка (Error)**, следует выбрать в качестве значения свойства **Ошибка (OnError)**. элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_Error(DataErr **As Integer,** Response **As Integer)**
Private Sub Report_Error(DataErr **As Integer,** Response **As Integer)**

Процедура обработки события **Ошибка** использует следующие аргументы.

Аргумент	**Описание**
DataErr	Код ошибки, возвращаемый объектом **Err** при возникновении ошибки. «DataErr» используется в качестве аргумента функции **Error** для возвращения сообщения об ошибке, соответствующего коду ошибки.
Response	Определяет, будет ли выведено сообщение об ошибке. Значение аргумента «Response» задается с помощью одной из следующих встроенных констант.

Константа	**Описание**
acDataErrContinue	Ошибка игнорируется, и выполнение продолжается без вывода стандартного сообщения об ошибке Microsoft Access. В этом случае допускается вывод специального сообщения об ошибке вместо стандартного сообщения.
AcDataErrDisplay	(Значение по умолчанию). Выводится стандартное сообщение об ошибке Microsoft Access.

Дополнительные сведения

Отмена события **Ошибка** не допускается.

Событие «Ошибка» (Error) - Пример процедуры обработки события

В данном примере демонстрируется замена стандартного сообщения об ошибке на специальное сообщение. При обнаружении Microsoft Access повторяющегося значения в ключевом поле (код ошибки 3022) данная процедура обработки события выводит сообщение об ошибке, содержащее информацию, специфическую для выполняющегося приложения.

Для проверки работы данного примера следует добавить процедуру обработки события в форму с базовой таблицей, каждая запись которой содержит сведения о сотруднике, а также уникальный код сотрудника в качестве ключа.

```
Private Sub Form_Error(DataErr As Integer, Response As Integer)
   Const conDuplicateKey = 3022
   Dim strMsg As String

   If DataErr = conDuplicateKey Then
      Response = acDataErrContinue
      strMsg = "Каждая запись должна содержать уникальный " _
         & "код сотрудника. Проверьте введенные данные."
      MsgBox strMsg
   End If
End Sub
```

События «Получение фокуса» (GotFocus), «Потеря фокуса» (LostFocus)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtGotLostFocusC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtGotLostFocusX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtGotLostFocusMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS":"acevtGotLostFocusEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acevtGotLostFocusA"}

- Событие **Получение фокуса (GotFocus)** возникает, когда форма или элемент управления получает фокус.
- Событие **Потеря фокуса (LostFocus)** возникает при потере фокуса формой или элементом управления.

Примечание. События **Получение фокуса** и **Потеря фокуса** определены только для форм и элементов управления в формах, но не определены для элементов управления в отчетах.

Дополнительные сведения

Для выполнения макроса или процедуры обработки событий, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Получение фокуса (OnGotFocus)** или **Потеря фокуса (OnLostFocus)**.

Данные события происходят, когда фокус перемещается в результате действий пользователя, таких как нажатие клавиши TAB или выбор объекта с помощью мыши, а также в результате вызова метода **SetFocus** в программе Visual Basic или вызова в макросе макрокоманд **ВыделитьОбъект (SelectObject)**, **НаЗапись (GoToRecord)**, **КЭлементуУправления (GoToControl)** или **НаСтраницу (GoToPage)**.

Элемент управления может получить фокус только в том случае, если его свойства **Вывод на экран (Visible)** и **Доступ (Enabled)** имеют значение «Да». Форма может получить фокус только в случае, если в ней нет элементов управления, или если все видимые элементы управления формы являются недоступными. Если форма содержит доступные видимые элементы управления, то событие **Получение фокуса** для нее не возникает.

Запуск макроса или выполнение процедуры обработки события в ответ на событие **Получение фокуса** позволяет выполнить определенные действия при получении фокуса формой или элементом управления. Например, создание процедуры обработки события **Получение фокуса** для каждого элемента управления формы, позволяет предоставить пользователю справочные сведения о приложении, которые выводятся в поле в виде кратких инструкций или сообщений; кроме того, для повышения наглядности, при переходах фокуса от одного элемента управления к другому элементы управления становятся доступными или недоступными, видимыми или невидимыми.

Примечание. Чтобы задать порядок перемещений фокуса между элементами управления в форме при нажатии клавиши TAB, следует указать последовательность переходов или определить клавиши доступа для элементов управления.

Событие **Получение фокуса** отличается от события **Вход (Enter)** тем, что событие **Получение фокуса** возникает каждый раз, когда элемент управления получает фокус. Предположим, что пользователь выбирает с помощью мыши флажок в форме, затем с помощью другого нажатия кнопки мыши переключается на окно отчета, а затем опять устанавливает указатель на флажок формы и нажимает кнопку мыши для того, чтобы вывести окно формы на передний план. Событие **Получение фокуса** возникнет при каждом получении фокуса флажком, то есть два раза. В отличие от этого, событие **Вход** возникнет только при первом выборе флажка пользователем. Событие **Получение фокуса** возникает после события **Вход**.

Событие **Потеря фокуса (LostFocus)** отличается от события **Выход (Exit)** тем, что событие **Потеря фокуса** возникает каждый раз, когда элемент управления теряет фокус. Событие

Выход возникает только перед тем, как фокус с элемента управления перемещается на другой элемент управления в той же форме. Событие **Потеря фокуса** возникает после события **Выход**.

Если фокус переходит в другую форму на элемент управления, который не имел фокуса, когда форма была активной, то возникают события **Выход** и **Потеря фокуса** для последнего активного элемента управления в этой форме с последующими событиями **Вход** и **Получение фокуса** для элемента управления, получающего фокус.

При переводе с помощью мыши фокуса с элемента управления в главной форме на элемент управления в подчиненной форме возникают следующие события:

Выход (для элемента управления в главной форме)

⇓

Потеря фокуса (для элемента управления в главной форме)

⇓

Вход (для элемента управления в подчиненной форме)

⇓

Выход (для элемента управления в подчиненной форме, имевшего фокус)

⇓

Потеря фокуса (для элемента управления в подчиненной форме, имевшего фокус)

⇓

Вход (для элемента управления в подчиненной форме, получающего фокус)

⇓

Получение фокуса (для элемента управления в подчиненной форме, получающего фокус)

При переходе в подчиненной форме на элемент управления, который последним имел в ней фокус, для этого элемента возникают не два события **Вход** и **Получение фокуса**, а только одно событие **Вход**. Когда фокус переводится с элемента управления в подчиненной форме на элемент управления в главной форме, для элемента управления в подчиненной форме возникают не два события **Выход** и **Потеря фокуса**, а только одно событие **Выход**; при этом для элемента управления в главной форме возникают события **Вход** и **Получение фокуса**.

Примечание. Часто фокус переводится на другой элемент управления с помощью мыши или с помощью клавиатуры, например, нажатием клавиши TAB. При этом наряду с описанными в данном разделе событиями возникают события мыши или события клавиатуры.

При переключении между окнами двух открытых форм событие **Отключение (Deactivate)** возникает для первой формы, а событие **Включение (Activate)** для второй Если эти формы не содержат видимых доступных элементов управления, событие **Потеря фокуса (LostFocus)** возникает в первой форме до события **Отключение (Deactivate)**, а событие **Получение фокуса (GotFocus)** возникает во второй форме после события **Включение (Activate)**.

События «Получение фокуса» (GotFocus), «Потеря фокуса» (LostFocus) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtGotLostFocusEPC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtGotLostFocusEPX":1}

Для создания процедуры обработки событий, которая выполняется при возникновении события **Получение фокуса (GotFocus)** или **Потеря фокуса (LostFocus)**, следует выбрать в качестве значения свойства **Получение фокуса (OnGotFocus)** или **Потеря фокуса (OnLostFocus)**, элемент [Процедура обработки события] и нажать кнопку **Построить** 

Синтаксис

Private Sub Form_GotFocus( )
Private Sub *имяЭлемента*_**GotFocus( )**
Private Sub Form_LostFocus( )
Private Sub *имяЭлемента*_**LostFocus( )**

Процедуры обработки событий **Получение фокуса** и **Потеря фокуса** используют следующий аргумент.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Получение фокуса** или **Потеря фокуса**.

Дополнительные сведения

Процедура обработки события **Потеря фокуса** применяется для проверки условий на значение для введенных данных при потере фокуса элементом управления. Допускается также отмена или изменение условий, заданных пользователем в процедуре обработки события **Получение фокуса** объекта.

Кроме того, процедуры обработки событий **Получение фокуса** и **Потеря фокуса** используются для вывода на экран или скрытия и для включения или отключения других объектов.

Отмена события **Получение фокуса** или **Потеря фокуса** не допускается.

События «Получение фокуса» (GotFocus), «Потеря фокуса» (LostFocus) - Пример процедуры обработки события

В данном примере при переводе фокуса на переключатель выводится сообщение в надписи.

Для проверки работы данного примера следует поместить процедуру обработки события в форму, содержащую переключатель «Выбор» и надпись «Сигнал».

```
Private Sub Выбор_GotFocus()
   Me!Сигнал.Caption = "Фокус на переключателе 'Выбор'."
End Sub

Private Sub Выбор_LostFocus()
   Me!Сигнал.Caption = ""    ' Очистка надписи.
End Sub
```

События «Получение фокуса» (GotFocus), «Потеря фокуса» (LostFocus) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtGotLostFocusMOC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtGotLostFocusMOX":1}

Для выполнения макроса, связанного с событием **Получение фокуса (GotFocus)** или **Потеря фокуса (LostFocus)**, следует указать имя этого макроса в качестве значения свойства **Получение фокуса (OnGotFocus)** или **Потеря фокуса (OnLostFocus)**.

Дополнительные сведения

Макрос, связанный с событием **Потеря фокуса**, применяется для проверки условий на значение для введенных данных при потере фокуса элементом управления. Допускается также отмена или изменение условий, заданных пользователем в макросе события **Получение фокуса** объекта.

Кроме того, макросы, связанные с событиями **Получение фокуса** и **Потеря фокуса**, используются для вывода на экран или скрытия и для включения или отключения других объектов.

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросах, связанных с событиями **Получение фокуса** или **Потеря фокуса**.

События «Клавиша вниз» (KeyDown), «Клавиша вверх» (KeyUp)

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtKeyDownUpC;vastmSendKeys"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtKeyDownUpX":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acevtKeyDownUpMO":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acevtKeyDownUpEP":1} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅпїЅ":"acevtKeyDownUpA"}

- Событие **Клавиша вниз (KeyDown)** возникает, если пользователь отпускает клавишу в момент, когда фокус имеет форма или элемент управления. Данное событие возникает также при передаче нажатия клавиши в форму или элемент управления с помощью макрокоманды **КомандыКлавиатуры (SendKeys)** в макросе или с помощью инструкции **SendKeys** в программе Visual Basic.
- Событие **Клавиша вверх (KeyUp)** возникает, если пользователь отпускает клавишу в момент, когда фокус имеет форма или элемент управления. Данное событие возникает также при передаче нажатия клавиши в форму или элемент управления с помощью макрокоманды **КомандыКлавиатуры** в макросе или инструкции **SendKeys** в программе Visual Basic.

Примечание. События **Клавиша вниз** и **Клавиша вверх** определены только для форм и для элементов управления в формах. Они не определены для элементов управления в отчетах.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Клавиша вниз (OnKeyDown)** или **Клавиша вверх (OnKeyUp)**.

Для каждого из событий все нажатия клавиш передаются объекту, имеющему фокус. Форма может получить фокус только в том случае, когда в ней нет элементов управления, или когда все видимые элементы управления формы являются недоступными.

Форма также принимает все события клавиатуры, даже возникающие в элементах управления, если для свойства **Перехват нажатия клавиш (KeyPreview)** формы предварительно задано значение «Да». При этой настройке все события клавиатуры возникают сначала для формы, а затем для имеющего фокус элемента управления. Это позволяет организовать отклик формы на нажатие конкретных клавиш, вне зависимости от того, какой из элементов управления имеет фокус. Например, это позволяет определить стандартное действие, выполняющееся при нажатии сочетания клавиш CTRL+X в форме.

Если пользователь нажимает и удерживает клавишу, то попеременно возникают события **Клавиша вниз** и **Нажатие клавиши (KeyPress)**. (цепочка событий Клавиша вниз, Нажатие клавиши, Клавиша вниз, Нажатие клавиши и т.д.). Это продолжается до отпускания клавиши, после чего возникает событие **Клавиша вверх**.

События **Клавиша вниз** и **Клавиша вверх** возникают при нажатии большинства клавиш, однако, эти события обычно применяются для распознавания нажатий следующих специальных клавиш:

- клавиш расширенного набора, таких как функциональные клавиши;
- клавиш перемещения курсора, таких как HOME, END, PAGE UP, PAGE DOWN, СТРЕЛКА ВВЕРХ, СТРЕЛКА ВНИЗ, СТРЕЛКА ВПРАВО, СТРЕЛКА ВЛЕВО и TAB;
- сочетаний клавиш со стандартными управляющими клавишами SHIFT, CTRL или ALT;
- клавиш цифровой панели и цифровых клавиш обычной клавиатуры.

События **Клавиша вниз** и **Клавиша вверх** не возникают при нажатии следующих клавиш:

- ENTER, если форма содержит кнопку с заданным для свойства **По умолчанию (Default)**

значением «Да».

- ESC, если форма содержит кнопку с заданным для свойства **Отмена (Cancel)** значением «Да».

Совет. Для определения символа ANSI, соответствующего нажатой клавише, следует воспользоваться событием **Нажатие клавиши (KeyPress)**.

События **Клавиша вниз** и **Нажатие клавиши** возникают при нажатии клавиш или при передаче кодов клавиш, соответствующих символам ANSI. Событие **Клавиша вверх** возникает после любого события элемента управления, вызванного нажатием или передачей нажатия клавиши. Если в результате нажатия клавиши фокус переводится с одного элемента управления на другой, то событие **Клавиша вниз** возникает для первого элемента управления, а события **Нажатие клавиши** и **Клавиша вверх** для второго.

Дополнительные сведения содержатся в разделе Порядок событий клавиатуры и мыши.

Если в результате нажатия или передачи нажатия клавиши выводится модальное диалоговое окно, то возникают события **Клавиша вниз** и **Нажатие клавиши**, но событие **Клавиша вверх** не возникает.

События «Клавиша вниз» (KeyDown), «Клавиша вверх» (KeyUp) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtKeyDownUpMOC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtKeyDownUpMOX":1}

Для выполнения макроса, связанного с событием **Клавиша вниз (KeyDown)** или **Клавиша вверх (KeyUp)**, следует указать имя этого макроса в качестве значения свойства **Клавиша вниз (OnKeyDown)** или **Клавиша вверх (OnKeyUp)**.

Дополнительные сведения

Макросы, связанные с событиями **Клавиша вниз** или **Клавиша вверх**, позволяют организовать отклик на любое нажатие клавиш или передачу нажатия клавиш в имеющих фокус формах или элементах управления. Однако макросы не возвращают коды клавиш и не позволяют определить, какая из клавиш была нажата, поэтому обычно с данными событиями связывают процедуры обработки событий.

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросах, связанных с событиями **Клавиша вниз** или **Клавиша вверх**.

События «Клавиша вниз» (KeyDown), «Клавиша вверх» (KeyUp) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtKeyDownUpEPC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtKeyDownUpEPX":1}

Для создания процедуры обработки событий, которая выполняется при возникновении события **Клавиша вниз (KeyDown)** или **Клавиша вверх (KeyUp)**, следует выбрать в качестве значения свойства **Клавиша вниз (OnKeyDown)** или **Клавиша вверх (OnKeyUp)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_KeyDown(KeyCode **As Integer,** Shift **As Integer)**
Private Sub *имяЭлемента***_KeyDown(**KeyCode **As Integer,** Shift **As Integer)**
Private Sub Form_KeyUp(KeyCode **As Integer,** Shift **As Integer)**
Private Sub *имяЭлемента***_KeyUp(**KeyCode **As Integer,** Shift **As Integer)**

Процедуры обработки событий **Клавиша вниз** и **Клавиша вверх** используют следующие аргументы.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки событий **Клавиша вниз** или **Клавиша вверх**.
KeyCode	Код клавиши, например, **vbKeyF1** (для клавиши F1) или **vbKeyHome** (для клавиши HOME). Для указания кодов клавиш следует применять встроенные константы, выводящиеся в окне просмотра объектов. Для того чтобы не передавать нажатие клавиши объекту, следует присвоить аргументу «KeyCode» значение 0.
Shift	Определяет состояние управляющих клавиш SHIFT, CTRL и ALT во время возникновения события. Для проверки значения аргумента «Shift» допускается использование следующих встроенных констант в качестве поразрядных масок:

Константа	Описание
acShiftMask	Маска для клавиши SHIFT.
AcCtrlMask	Маска для клавиши CTRL.
AcAltMask	Маска для клавиши ALT.

Дополнительные сведения

Для проверки состояния управляющих клавиш следует сначала присвоить значение каждого из аргументов временной переменной целого типа, а затем сравнить значение аргумента «Shift» с соответствующей встроенной константой. Если результат применения оператора **And** к аргументу «Shift» и выбранной константе превышает нуль, это указывает, что была нажата соответствующая данной константе клавиша (SHIFT, CTRL или ALT), как демонстрируется в следующем примере:

```
ShiftDown = (Shift And acShiftMask) > 0
```

В процедуре обработки событий допускается проверка любых комбинаций условий, например:

```
If ShiftDown And CtrlDown Then
.  ' Выполняется, если одновременно нажаты клавиши SHIFT и CTRL.
.
.
```

```
End If
```

Процедуры обработки событий **Клавиша вниз** и **Клавиша вверх** позволяют различать введенные символы верхнего и нижнего регистра. Для этого следует проверить значения обоих аргументов, «KeyCode» и «Shift». Аргумент «KeyCode» определяет физически нажатую клавишу (при этом символы «A» и «a» соответствуют одной клавише), а аргумент «Shift» указывает состояние клавиши SHIFT, в результате чего символы «A» и «a» различаются.

Процедуры обработки событий **Клавиша вниз** и **Клавиша вверх** применяются при создании программ драйверов клавиатуры, в которых выполняются действия в ответ на нажатие и отпускание клавиш.

Пользователь имеет возможность организовать отклик формы на нажатие конкретных клавиш, вне зависимости от того, какой из элементов управления имеет фокус. Например, это позволяет определить стандартное действие, выполняющееся при нажатии сочетания клавиш CTRL+X в форме. Для того чтобы форма принимала все события клавиатуры, даже возникающие в элементах управления, до того как они будут переданы элементам управления, следует задать для свойства **Перехват нажатия клавиш (KeyPreview)** формы значение «Да». При этой настройке все события клавиатуры возникают сначала для формы, а затем для имеющего фокус элемента управления. Это позволяет организовать отклик формы на нажатие конкретных клавиш с помощью событий формы **Клавиша вниз**, **Нажатие клавиши (KeyPress)**, и **Клавиша вверх**. Для того чтобы элемент управления не получал нажатия клавиш, на которые уже осуществлен отклик, и в нем не возникали соответствующие события клавиатуры, следует задать значения 0 для аргументов «KeyCode» обоих событий, **Клавиша вниз** и **Клавиша вверх**, а также задать значение 0 для аргумента «KeyAscii» события **Нажатие клавиши** (если нажата клавиша, соответствующая символу ANSI). Если все три аргумента получают значение 0, то событие клавиатуры не принимается элементом управления.

Аргументы событий **Клавиша вниз**, **Нажатие клавиши** и **Клавиша вверх** вместе с аргументами событий **Кнопка вниз (MouseDown)**, **Кнопка вверх (MouseUp)** и **Перемещение указателя (MouseMove)** позволяют обеспечить работу приложения для любых пользователей, как работающих с мышью, так и использующих клавиатуру.

Отмена событий **Клавиша вниз** или **Клавиша вверх** не допускается.

События «Клавиша вниз» (KeyDown), «Клавиша вверх» (KeyUp) - Пример процедуры обработки события

В данном примере проверяется, какая клавиша нажата: SHIFT, CTRL или ALT.

Для проверки работы данного примера следует добавить процедуру обработки события в форму, содержащую поле «ДрайверКлавиатуры».

```
Private Sub ДрайверКлавиатуры_KeyDown(KeyCode As Integer, Shift As Integer)
   Dim intShiftDown As Integer, intAltDown As Integer
   Dim intCtrlDown As Integer

   ' Определяет нажатую клавишу с помощью поразрядной маски.
   intShiftDown = (Shift And acShiftMask) > 0
   intAltDown = (Shift And acAltMask) > 0
   intCtrlDown = (Shift And acCtrlMask) > 0
   ' Выводит сообщение о нажатой клавише.
   If intShiftDown Then MsgBox "Нажата клавиша SHIFT."
   If intAltDown Then MsgBox "Нажата клавиша ALT."
   If intCtrlDown Then MsgBox "Нажата клавиша CTRL."
End Sub
```

Событие «Нажатие клавиши» (KeyPress)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtKeyPressC;vafctAsc;vafctChr;vastmSendKeys"}
{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtKeyPressX":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS":"acevtKeyPressMO":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS
nїSnїSnїSnїSnїSnїS":"acevtKeyPressEP":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtKeyPressA"}

Событие **Нажатие клавиши (KeyPress)** возникает, если пользователь нажимает и отпускает клавишу или сочетание клавиш, соответствующих коду символа ANSI, в момент, когда фокус имеет форма или элемент управления. Данное событие возникает также при передаче нажатия клавиши ANSI в форму или элемент управления с помощью макрокоманды **КомандыКлавиатуры (SendKeys)** в макросе или с помощью инструкции **SendKeys** в программе Visual Basic.

Примечание. Событие **Нажатие клавиши** определено только для форм и элементов управления в формах и не определено для элементов управления в отчетах.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Нажатие клавиши (OnKeyPress)**.

Все нажатия клавиш передаются объекту, имеющему фокус. Форма может получить фокус только в том случае, когда в ней нет элементов управления, или когда все видимые элементы управления формы являются недоступными.

Форма также будет принимать все события клавиатуры, даже возникающие в элементах управления, если для свойства формы **Перехват нажатия клавиш (KeyPreview)** задано значение «Да». При этой настройке все события клавиатуры возникают сначала для формы, а затем для имеющего фокус элемента управления. Это позволяет организовать отклик формы на нажатие конкретных клавиш вне зависимости от того, какой элемент управления имеет фокус. Например, это позволяет определить стандартное действие, выполняющееся при нажатии сочетания клавиш CTRL+X в форме.

Если пользователь нажимает и удерживает клавишу, то попеременно возникают события **Клавиша вниз (KeyDown)** и **Нажатие клавиши (KeyPress)** (цепочка событий Клавиша вниз, Нажатие клавиши, Клавиша вниз, Нажатие клавиши и т.д.). Это продолжается до отпускания клавиши, после чего возникает событие **Клавиша вверх (KeyUp)**.

Событие **Нажатие клавиши** применяется для распознавания нажатия клавиш, соответствующих печатаемым символам, сочетаний клавиши CTRL со стандартными алфавитными символами или со специальными символами, а также клавиш ENTER и BACKSPACE. Для обработки нажатия клавиш, которое не распознается событием **Нажатие клавиши**, например, нажатия функциональных клавиш, клавиш перемещения курсора, а также любого сочетания данных клавиш со стандартными управляющими клавишами (ALT, SHIFT или CTRL) используются процедуры обработки событий **Клавиша вниз** и **Клавиша вверх**. В отличие от событий **Клавиша вниз** и **Клавиша вверх**, событие **Нажатие клавиши** не отражает реального состояния клавиш клавиатуры; вместо этого данное событие возвращает код символа ANSI, соответствующего нажатой клавише или сочетанию клавиш.

В событии **Нажатие клавиши** прописные и строчные буквы интерпретируются как два разных кода клавиш и, следовательно, как два разных символа.

Примечание. Клавиша BACKSPACE является частью набора символов ANSI, а клавиша DEL не является. При удалении символа в элементе управления с помощью клавиши BACKSPACE возникает событие **Нажатие клавиши**, при нажатии клавиши DEL это событие не возникает.

События **Клавиша вниз** и **Нажатие клавиши** возникают при нажатии клавиш или при передаче

кодов клавиш, соответствующих символам ANSI. Событие **Клавиша вверх** возникает после каждого события элемента управления, вызванного нажатием или передачей нажатия клавиши. Если в результате нажатия клавиши фокус переводится с одного элемента управления на другой, то событие **Клавиша вниз** возникает для первого элемента управления, а события **Нажатие клавиши** и **Клавиша вверх** для второго.

Например, если пользователь переходит на новую запись и вводит символ в первый элемент управления в этой записи, возникает следующая цепочка событий:

Текущая запись (для новой записи)

⇓

Вход (для первого элемента управления в новой записи)

⇓

Получение фокуса (для элемента управления)

⇓

Клавиша вниз (для элемента управления)

⇓

Нажатие клавиши (для элемента управления)

⇓

До вставки (для новой записи в форме)

⇓

Изменение (для элемента управления «поле» или «поле со списком»)

⇓

Клавиша вверх (для элемента управления)

Дополнительные сведения содержатся в разделе Порядок событий клавиатуры и мыши.

Событие «Нажатие клавиши» (KeyPress) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtKeyPressMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtKeyPressMOX":1}

Для выполнения макроса, связанного с событием **Нажатие клавиши (KeyPress)**, следует указать имя этого макроса в качестве значения свойства **Нажатие клавиши (OnKeyPress)**.

Дополнительные сведения

Макросы, связанные с событием **Нажатие клавиши**, позволяют организовать отклик на любое нажатие клавиш ANSI, в момент, когда фокус имеет форма или элемент управления. Однако макросы не возвращают коды клавиш ANSI и не позволяют определить, какая из клавиш была нажата, поэтому обычно с данным событием связывают процедуры обработки событий.

Допускается использование макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Нажатие клавиши**, чтобы отменить нажатие клавиши или сочетания клавиш, например, если требуется запретить ввод в элемент управления символов ANSI при определенных условиях.

Событие «Нажатие клавиши» (KeyPress) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtKeyPressEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtKeyPressEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Нажатие клавиши (KeyPress)**, следует выбрать в качестве значения свойства **Нажатие клавиши (OnKeyPress)**. элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_KeyPress(KeyAscii **As Integer)**
Private Sub *имяЭлемента***_KeyPress(**KeyAscii **As Integer)**

Процедура обработки события **Нажатие клавиши** использует следующие аргументы.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Нажатие клавиши**.
KeyAscii	Возвращает числовой код символа ANSI. Аргумент «KeyAscii» передается по ссылке; изменение значения данного аргумента приводит к передаче объекту другого символа. Значение 0, присвоенное аргументу «KeyAscii», отменяет нажатие клавиши, при этом нажатие клавиши в объект не передается.

Дополнительные сведения

Процедура обработки события **Нажатие клавиши** применяется для перехвата нажатий клавиш в поле или в поле со списком.Данное событие позволяет проверять, являются ли допустимыми вводящиеся символы, или выполнять форматирование непосредственно при вводе символов. Изменение значения аргумента «KeyAscii» приводит к изменению выводящегося в поле символа.

Значение аргумента «KeyAscii» преобразуется в символ с помощью следующего выражения:

Chr(KeyAscii**)**

Подобное преобразование позволяет работать с данным символом как со строковым значением. Обратное преобразование символа в код ANSI, который распознается элементом управления, проводится с помощью следующей синтаксической конструкции:

KeyAscii **= Asc(***символ***)**

Примечание. Сочетанию клавиш CTRL+@ соответствует код символа ANSI 0. Microsoft Access воспринимает аргумент «KeyAscii» со значением 0 как пустую строку (""), поэтому не следует применять в приложениях сочетание клавиш CTRL+@.

Пользователь имеет возможность организовать в открытой форме отклик на нажатие конкретных клавиш, вне зависимости от того, какой из элементов управления имеет фокус. Например, это позволяет определить стандартное действие, выполняющееся при нажатии сочетания клавиш CTRL+X в форме. Для того чтобы форма принимала все события клавиатуры, даже возникающие в элементах управления, до того как они будут переданы элементам управления, следует задать для свойства **Перехват нажатия клавиш (KeyPreview)** формы значение «Да». При этой настройке все события клавиатуры возникают сначала для формы, а затем для имеющего фокус элемента управления. Это позволяет организовать отклик формы на нажатие конкретных клавиш с помощью событий формы **Клавиша вниз (KeyDown)**, **Нажатие клавиши** и **Клавиша вверх (KeyUp)**. Для того чтобы элемент управления не получал нажатия клавиш, на которые уже осуществлен отклик, и в нем не возникали соответствующие

события клавиатуры, следует задать значения 0 для аргументов «KeyCode» обоих событий, **Клавиша вниз** и **Клавиша вверх**, а также задать значение 0 для аргумента «KeyAscii» события **Нажатие клавиши**. Если все три аргумента получают значение 0, то событие клавиатуры не принимается элементом управления.

Аргументы событий **Клавиша вниз**, **Нажатие клавиши** и **Клавиша вверх** вместе с аргументами событий **Кнопка вниз (MouseDown)**, **Кнопка вверх (MouseUp)** и **Перемещение указателя (MouseMove)** позволяют обеспечить работу приложения для любых пользователей, как работающих с мышью, так и использующих клавиатуру.

Не допускается отмена события **Нажатие клавиши** в процедуре обработки события, однако, это событие можно отменить в макросе.

Событие «Нажатие клавиши» (KeyPress) - Пример процедуры обработки события

В данном примере вводящийся в поле текст преобразуется к верхнему регистру, причем преобразование выполняется непосредственно при вводе каждого символа.

Для проверки работы данного примера следует добавить процедуру обработки события в форму, содержащую поле «ОбластьПолучателя».

```
Private Sub ОбластьПолучателя_KeyPress(KeyAscii As Integer)
   Dim strCharacter As String

   ' Преобразует код ANSI в строку символов.
   strCharacter = Chr(KeyAscii)
   ' Преобразует символ к верхнему регистру, а затем в код ANSI.
   KeyAscii = Asc(UCase(strCharacter))
End Sub
```

События «Загрузка» (Load), «Выгрузка»(Unload)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtLoadUnloadC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtLoadUnloadX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS":"acevtLoadUnloadMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtLoadUnloadEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtLoadUnloadA"}

- Событие **Загрузка (Load)** возникает при открытии формы и выводе на экран ее записей.
- Событие **Выгрузка (Unload)** возникает после закрытия формы, но до ее удаления с экрана. При повторной загрузке формы Microsoft Access вновь выводит форму и устанавливает начальные значения всех ее элементов управления.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Загрузка (OnLoad)** или **Выгрузка (OnUnload)**.

Событие **Загрузка** возникает при следующих действиях пользователя:

- запуск приложения;
- открытие формы с помощью кнопки **Открыть** в окне базы данных..
- выполнение в макросе макрокоманды **ОткрытьФорму (OpenForm)**.

Событие **Выгрузка** возникает при следующих действиях пользователя:

- нажатие кнопки **Закрыть** в окне формы или выбор команды **Закрыть** в меню **Файл** или в оконном меню формы;
- выполнение макрокоманды **Закрыть (Close)** в макросе;
- выход из приложения путем нажатия правой кнопки мыши на панели задач и последующего выбора команды **Закрыть**;
- выход из Windows при работающем приложении.

С помощью запуска макроса или процедуры обработки события при возникновении события формы **Выгрузка** можно указать стандартные значения для элементов управления или вывести результаты вычислений для данных, содержащихся в записях формы.

С помощью запуска макроса или процедуры обработки события при возникновении события формы **Выгрузка** имеется возможность запросить подтверждение необходимости выгрузки формы или указать действия, которые следует выполнить при выгрузке формы. Кроме того, существует возможность открыть другую форму или диалоговое окно с приглашением пользователю занести свое имя в журнал учета работы с формой.

При первом открытии формы возникает следующая цепочка событий:

Открытие ⇒ Загрузка ⇒ Изменение размера ⇒ Включение ⇒ Текущая запись

При решении вопроса, с каким событием, **Открытие** или **Загрузка**, следует связывать макрос или процедуру обработки события, следует принимать во внимание, что событие **Открытие** допускает отмену, а событие **Загрузка** не допускает. Например, при динамическом построении источника записей формы в процедуре обработки события **Открытие** формы пользователь имеет возможность отменить открытие формы, если источник записей пуст.

При закрытии формы возникает следующая цепочка событий:

Выгрузка ⇒ Отключение ⇒ Закрытие

Событие **Выгрузка** возникает до события **Закрытие (Close)**. Допускается отмена события **Выгрузка**; событие **Закрытие** отменить нельзя.

Примечание. При создании макросов или процедур обработки события для событий,

которые вызываются событием **Загрузка (Load)**, таких как **Включение (Activate)** и **Получение фокуса (GotFocus)**, необходимо следить за отсутствием конфликтов (например, в одном макросе или процедуре не следует выполнять действие, результат которого будет отменен в другом макросе или процедуре), а также не допускать возникновения каскадных событий.

События «Загрузка» (Load), «Выгрузка»(Unload) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtLoadUnloadMOC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtLoadUnloadMOX":1}

Для выполнения макроса, связанного с событием **Загрузка (Load)** или **Выгрузка (Unload)**, следует указать имя этого макроса в качестве значения свойства **Загрузка (OnLoad)** или **Выгрузка (OnUnload)**.

Дополнительные сведения

Макросы, связанные с событием **Загрузка**, используются для выполнения действий над элементами управления или записями в форме после открытия формы. Например, в таких макросах задают стандартные значения элементов управления.

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Загрузка**.

Допускается отмена выгрузки записей с помощью макрокоманды **ОтменитьСобытие** в макросах, связанных с событием **Выгрузка**. Эта макрокоманда вызывает также отмену закрытия формы.

Если в макросе, связанном с событием формы **Выгрузка**, выполняется макрокоманда **ОтменитьСобытие**, то закрытие формы становится невозможным. В этом случае необходимо либо исправить условие, которое вызвало выполнение макрокоманды **ОтменитьСобытие**, либо открыть макрос и удалить из него эту макрокоманду. Если форма является модальной, то после выполнения данной макрокоманды открытие макроса или работа в каком-либо другом окне приложения становится невозможной.

События «Загрузка» (Load), «Выгрузка»(Unload) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtLoadUnloadEPC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtLoadUnloadEPX":1}

Для создания процедуры обработки событий, которая выполняется при возникновении события **Загрузка (Load)** или **Выгрузка (Unload),** следует указать в качестве значения свойства **Загрузка (OnLoad)** или **Выгрузка (OnUnload)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_Load( )
Private Sub Form_Unload(Cancel **As Integer)**

Процедура обработки события **Выгрузка** использует следующий аргумент.

Аргумент	Описание
Cancel	Определяет, будет ли выгружена форма. Заданное для аргумента «Cancel» значение **True** (–1) приводит к отмене выгрузки формы.

Дополнительные сведения

В процедуру обработки события формы **Загрузка** формы можно поместить программы инициализации. Например, в таких программах задаются стандартные значения для элементов управления и присваиваются начальные значения переменным на уровне формы.

Событие **Загрузка** отменить нельзя.

Процедура обработки события **Выгрузка** может содержать любые инструкции проверки условий на значения в форме, выполняемые перед закрытием формы и сохранением в файле содержащихся в ней данных.

Если для аргумента «Cancel» задается значение **True** в модальной форме, то передача управления другим частям приложения становится невозможной.

События «Загрузка» (Load), «Выгрузка»(Unload) - Пример процедуры обработки события

В данном примере при загрузке формы в заголовке формы выводится текущая дата.

Для проверки работы данного примера следует добавить процедуру обработки события в форму:

```
Private Sub Form_Load()
   Me.Caption = Date
End Sub
```

В следующем примере выводится приглашение подтвердить закрытие формы.

Для проверки работы данного примера следует добавить процедуру обработки события в форму. В режиме формы при закрытии формы будет выведено диалоговое окно, в котором следует выбрать одну из кнопок «Да» или «Нет».

```
Private Sub Form_Unload(Cancel As Integer)
   If MsgBox("Закрыть форму?", vbYesNo) = vbYes Then
      Exit Sub
   Else
      Cancel = True
   End If
End Sub
```

События «Кнопка вниз» (MouseDown), «Кнопка вверх» (MouseUp)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtMouseDownUpC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtMouseDownUpX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtMouseDownUpMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS":"acevtMouseDownUpEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acevtMouseDownUpA"}

- Событие **Кнопка вниз (MouseDown)** возникает, когда пользователь нажимает на кнопку мыши.
- Событие **Кнопка вверх (MouseUp)** возникает, когда пользователь отпускает кнопку мыши.

Примечания

- События **Кнопка вниз** и **Кнопка вверх** определены только для форм, разделов форм и элементов управления в формах. Они не определены для элементов управления в отчетах.
- Данные события не определены для надписи, присоединенной к другому элементу управления, например, для подписи поля. Они определены только для «самостоятельных» надписей. Нажатие и отпускание кнопки мыши при указателе, установленном на присоединенной надписи, дает такой же эффект, как нажатие и отпускание кнопки мыши при указателе, установленном на элементе управления, к которому присоединена надпись. При этом происходят обычные события элемента управления и не возникают отдельные события для надписи.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Кнопка вверх (OnMouseUp)** или **Кнопка вниз (OnMouseDown)**.

События **Кнопка вниз** или **Кнопка вверх** позволяют указать конкретные действия, которые выполняются, когда пользователь нажимает или отпускает определенную кнопку мыши. В отличие от событий **Нажатие кнопки (Click)** и **Двойное нажатие кнопки (DblClick)** события **Кнопка вниз** и **Кнопка вверх** позволяют различить нажатия левой, правой и средней кнопки мыши. Допускается также создание программ для обработки специальных сочетаний из нажатия кнопки мыши и одной из управляющих клавиш SHIFT, CTRL или ALT.

Для того чтобы создать в форме событие **Кнопка вниз** или **Кнопка вверх**, следует нажать или отпустить кнопку мыши при указателе, установленном в пустой области или в области выделения записи формы. Для того чтобы создать событие **Кнопка вниз** или **Кнопка вверх** в разделе формы, следует нажать кнопку мыши при указателе, установленном в пустой области раздела.

Для событий **Кнопка вниз** или **Кнопка вверх** действуют следующие правила:

- Если кнопка мыши нажата в тот момент, когда указатель расположен на форме или на элементе управления, то данный объект принимает все последующие события мыши вплоть до завершающего события **Кнопка вверх** включительно.
- Если кнопки мыши нажимаются последовательно друг за другом, то объект, получивший первое событие мыши, принимает все последующие события мыши до того момента, пока все кнопки мыши не будут отпущены.

Для выполнения действий в ответ на перемещение указателя мыши следует использовать событие **Перемещение указателя (MouseMove)**.

События «Кнопка вниз» (MouseDown), «Кнопка вверх» (MouseUp) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtMouseDownUpMOC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtMouseDownUpMOX":1}

Для выполнения макроса, связанного с событием **Кнопка вниз (MouseDown)** или **Кнопка вверх (MouseUp)**, следует указать имя макроса в качестве значения свойства **Кнопка вверх (OnMouseUp)** или **Кнопка вниз (OnMouseDown)**.

Дополнительные сведения

Макросы, связанные с событием **Кнопка вниз** или **Кнопка вверх**, позволяют выполнить конкретные действия в ответ на нажатие или отпускание пользователем кнопки мыши в форме или на элементе управления. Однако макросы не возвращают код нажатой кнопки и не позволяют определить, какая именно кнопка мыши была нажата, поэтому обычно с данными событиями связывают процедуры обработки событий.

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Кнопка вниз** или **Кнопка вверх**, с одним исключением: допускается вызов макрокоманды **ОтменитьСобытие** в макросе, связанном с событием **Кнопка вниз** для отмены события, возникшего при нажатии правой кнопки мыши. Например, допускается отмена открытия контекстного меню и вывод специального контекстного меню.

События «Кнопка вниз» (MouseDown), «Кнопка вверх» (MouseUp) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtMouseDownUpEPC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtMouseDownUpEPX":1}

Для создания процедуры обработки событий, которая выполняется при возникновении события **Кнопка вниз (MouseDown)** или **Кнопка вверх (MouseUp)**, следует выбрать в качестве значения свойства **Кнопка вверх (OnMouseUp)** или **Кнопка вниз (OnMouseDown)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_MouseDown(Button **As Integer,** Shift **As Integer,** X **As Single,** Y **As Single)**
Private Sub *имяЭлемента***_MouseDown(**Button **As Integer,** Shift **As Integer,** X **As Single,** Y **As Single)**
Private Sub Form_MouseUp(Button **As Integer,** Shift **As Integer,** X **As Single,** Y **As Single)**
Private Sub *имяЭлемента***_MouseUp(**Button **As Integer,** Shift **As Integer,** X **As Single,** Y **As Single)**

Процедуры обработки событий **Кнопка вниз (MouseDown)** или **Кнопка вверх (MouseUp)** используют следующие аргументы.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Кнопка вниз (MouseDown)** или **Кнопка вверх (MouseUp)**.
Button	Определяет кнопку, которая была нажата (Mouse Down) или отпущена (Mouse Up) при возникновении соответствующего события. Для проверки значения аргумента «Button» допускается использование следующих встроенных констант в качестве поразрядных масок:

Константа	Описание
acLeftButton	Маска для левой кнопки мыши.
AcRightButton	Маска для правой кнопки мыши.
AcMiddleButton	Маска для средней кнопки мыши.

Shift	Определяет состояние управляющих клавиш SHIFT, CTRL и ALT в момент, когда нажимается или отпускается кнопка мыши, определяемая аргументом «Button». Для проверки значения аргумента «Shift» допускается использование одной из следующих встроенных констант в качестве поразрядных масок:

Константа	Описание
acShiftMask	Маска для клавиши SHIFT.
AcCtrlMask	Маска для клавиши CTRL.
AcAltMask	Маска для клавиши ALT.

X, Y	Определяют текущие x- и y-координаты указателя мыши. Значения аргументов «X» и «Y» всегда задаются в твипах.

Примечание. Если кнопка мыши нажимается в тот момент, когда указатель расположен на форме или на элементе управления, то данный объект принимает все последующие события мыши вплоть до завершающего события **Кнопка вверх** включительно. Это означает, что аргументы «X» и «Y» не всегда относятся к принимающему их объекту.

Дополнительные сведения

Для проверки аргумента следует сначала присвоить значение каждого из аргументов временной переменной типа **Integer,** а затем сравнить значение аргумента «Shift» или «Button» с соответствующей встроенной константой. Если результат применения оператора **And** к аргументу «Button» и выбранной константе превышает нуль, это указывает, что была нажата соответствующая данной константе кнопка мыши, как демонстрируется в следующем примере:

```
LeftDown = (Button And acLeftButton) > 0
```

В процедурах обработки событий допускается проверка любой комбинации условий, например:

```
If ShiftDown And CtrlDown Then
   .' Выполняется, если нажаты клавиши SHIFT и CTRL.
   .
   .
End If
```

Примечание. Аргумент «Button» событий **Кнопка вниз** и **Кнопка вверх** имеет существенное отличие от аргумента «Button» события **Перемещение указателя (MouseMove)**. Для событий **Кнопка вниз** и **Кнопка вверх** аргумент «Button» определяет одну и только одну кнопку в каждом из событий. Это означает, что при нажатии двух кнопок мыши возникают два события **Кнопка вниз** (и соответственно, два события **Кнопка вверх**), каждое из которых имеет собственное значение аргумента «Button». Для события **Перемещение указателя** аргумент «Button» определяет текущее состояние всех кнопок.

Аргументы событий **Клавиша вниз (KeyDown)**, **Нажатие клавиши (KeyPress)**. и **Клавиша вверх (KeyUp)** вместе с аргументами событий **Кнопка вниз**, **Кнопка вверх** и **Перемещение указателя** позволяют обеспечить работу приложения для любых пользователей, как работающих с мышью, так и использующих клавиатуру.

Не допускается отмена события **Кнопка вниз** или **Кнопка вверх**.

Использование встроенных констант в качестве поразрядных масок

Для проверки значений аргументов «Button» или «Shift» используются поразрядные маски.

Аргумент «Button» является двоичным полем, в котором младшие биты соответствуют левой (бит 0), правой (бит 1) и средней кнопке мыши (бит 2). Значения данных битов равняются, соответственно, 1, 2 и 4. В аргументе может быть установлен только один бит, показывающий, какая кнопка мыши генерирует событие.

Встроенные константы Microsoft Access, используемые в качестве поразрядных масок для аргумента «Button», имеют следующие значения.

Константа	Значение
acLeftButton	1
acRightButton	2
acMiddleButton	4

Аргумент «Shift» является двоичным полем, в котором младшие биты определяют состояние нажатия клавиш SHIFT (бит 0), CTRL (бит 1) и ALT (бит 2). Значения данных битов равняются, соответственно, 1, 2 и 4. В аргументе «Shift» могут быть установлены некоторые из битов, все биты или ни один из битов, что указывает на нажатие некоторых, всех или ни одной из управляющих клавиш. Например, при одновременном нажатии клавиш CTRL и ALT аргумент «Shift» получает значение 6.

Встроенные константы Microsoft Access, используемые в качестве поразрядных масок для аргумента «Shift», имеют следующие значения.

Константа	Значение
acShiftMask	1
acCtrlMask	2
acAltMask	4

Данные константы позволяют проверить любое сочетание нажатых клавиш и кнопок мыши без необходимости запоминать уникальную поразрядную маску для каждого сочетания клавиш. Бит устанавливается, если соответствующая клавиша или кнопка мыши была нажата.

События «Кнопка вниз» (MouseDown), «Кнопка вверх» (MouseUp) - Пример процедуры обработки события

В данном примере показано, как можно проверить, какая из кнопок мыши вызвала событие **Кнопка вниз (MouseDown)**.

Для проверки работы данного примера следует добавить процедуру обработки события в форму:

```
Private Sub Form_MouseDown(Button As Integer, Shift As Integer, _
      X As Single, Y As Single)
   If Button = acLeftButton Then
      MsgBox "Нажата левая кнопка мыши."
   End If
   If Button = acRightButton Then
      MsgBox "Нажата правая кнопка мыши."
   End If
   If Button = acMiddleButton Then
      MsgBox "Нажата средняя кнопка мыши."
   End If
End Sub
```

Событие «Перемещение указателя» (MouseMove)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïS":"acevtMouseMoveC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtMouseMoveX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtMouseMoveMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïS":"acevtMouseMoveEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acevtMouseMoveA"}

Событие **Перемещение указателя (MouseMove)** возникает при перемещении указателя мыши пользователем.

Примечания

- Событие **Перемещение указателя** определено только для форм, разделов форм и элементов управления в формах. Оно не определено для элементов управления в отчетах.
- Данное событие не определено для надписи, присоединенной к другому элементу управления, например, для подписи поля. Оно определено только для «самостоятельных» надписей. Перемещение указателя мыши по присоединенной надписи, дает такой же эффект, как и перемещение указателя в элементе управления, к которому присоединена надпись. При этом происходят обычные события элемента управления и не возникают отдельные события для надписи.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Перемещение указателя (OnMouseMove)**.

Событие **Перемещение указателя** генерируется непрерывно по мере того, как указатель мыши перемещается по объектам. При перемещении указателя мыши внутри границ объекта событие **Перемещение указателя** возникает для данного объекта, если только это событие не является частью серии событий мыши, относящихся к другому объекту.

Для того чтобы вызвать событие **Перемещение указателя** для формы, следует провести указатель по пустой области формы, по области выделения записи или по полосе прокрутки формы. Для того чтобы вызвать событие **Перемещение указателя** для раздела формы, следует провести указатель по пустой области раздела.

Примечания

- Перемещение самой формы может вызвать событие **Перемещение указателя**, даже если мышь неподвижна. Это событие генерируется и в том случае, когда форма перемещается под стационарным указателем мыши. Перемещение формы в макросе или в процедуре обработки события в ответ на событие **Перемещение указателя** может стать причиной возникновения каскадных событий (то есть событие **Перемещение указателя** будет непрерывно генерироваться снова и снова).
- Если два элемента управления расположены очень близко друг к другу, и указатель быстро проводится через узкий промежуток между ними, то событие **Перемещение указателя** для этого промежутка (например, событие раздела формы) может и не возникнуть. В подобных случаях необходимо предусмотреть отклик на перемещение указателя как в смежных элементах управления, так и в разделе формы.

Для запуска макросов или выполнения процедуры обработки событий в ответ на нажатие или отпускание кнопок мыши следует использовать события **Кнопка вниз (MouseDown)** и **Кнопка вверх (MouseUp)**.

Событие «Перемещение указателя» (MouseMove) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtMouseMoveMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtMouseMoveMOX":1}

Для выполнения макроса, связанного с событием **Перемещение указателя (MouseMove)**, следует указать имя этого макроса в качестве значения свойства **Перемещение указателя (OnMouseMove)**.

Дополнительные сведения

Макросы, связанные с событием **Перемещение указателя**, применяются для выполнения определенных действий при перемещении указателя пользователем в форме или в элементе управления. Однако макросы не возвращают код кнопки и не позволяют определить состояние кнопок мыши или координаты указателя на момент перемещения, поэтому обычно с данным событием связывают процедуры обработки событий.

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Перемещение указателя**.

Событие «Перемещение указателя» (MouseMove) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtMouseMoveEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtMouseMoveEPX":1}

Для создания процедуры обработки событий, которая выполняется при возникновении события **Перемещение указателя (MouseMove)**, следует выбрать в качестве значения свойства **Перемещение указателя (OnMouseMove)**. элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_MouseMove(Button **As Integer,** Shift **As Integer,** X **As Single,** Y **As Single)**
Private Sub *имяЭлемента***_MouseMove(**Button **As Integer,** Shift **As Integer,** X **As Single,** Y **As Single)**

Процедура обработки события **Перемещение указателя** использует следующие аргументы.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Перемещение указателя**.
Button	Определяет состояние кнопок мыши в момент возникновения события. Для проверки значения аргумента «Button» допускается использование следующих встроенных констант в качестве поразрядных масок:

Константа	Описание
acLeftButton	Маска для левой кнопки мыши.
AcRightButton	Маска для правой кнопки мыши.
AcMiddleButton	Маска для средней кнопки мыши.

Shift	Определяет состояние управляющих клавиш SHIFT, CTRL и ALT в момент, когда нажимается или отпускается кнопка мыши, определяемая аргументом «Button». Для проверки значения аргумента «Shift» допускается использование одной из следующих встроенных констант в качестве поразрядных масок:

Константа	Описание
acShiftMask	Маска для клавиши SHIFT.
AcCtrlMask	Маска для клавиши CTRL.
AcAltMask	Маска для клавиши ALT.

X, Y	Определяют текущие x- и y-координаты указателя мыши. Значения аргументов «X» и «Y» всегда задаются в твипах.

Дополнительные сведения

Для проверки аргумента следует сначала присвоить значение каждого из аргументов временной переменной типа **Integer,** а затем сравнить значение аргумента «Shift» или «Button» с соответствующей встроенной константой. Если результат применения оператора **And** к аргументу «Button» и выбранной константе превышает нуль, это означает, что нажата соответствующая данной константе кнопка мыши, как демонстрируется в следующем примере:

```
LeftDown = (Button And acLeftButton) > 0
```

В процедуре допускается проверка любой комбинации условий, например:

```
If ShiftDown And CtrlDown Then
```

```
    .' Выполняется, если нажаты клавиши SHIFT и CTRL.
    .
    .
End If
```

Примечание. Аргумент «Button» событий **Кнопка вниз (MouseDown)** и **Кнопка вверх (MouseUp)** имеет существенное отличие от аргумента «Button» события **Перемещение указателя**. Для события **Перемещение указателя** этот аргумент определяет текущее состояние всех кнопок; каждое событие **Перемещение указателя** может возникнуть при нескольких или всех нажатых кнопках или без нажатых кнопок. Например, если пользователь перемещает указатель, удерживая нажатыми левую и правую кнопки мыши, аргумент «Button» получает значение 3 (1 + 2). Для событий **Кнопка вниз** и **Кнопка вверх** аргумент «Button» определяет одну и только одну кнопку в каждом из событий.

Аргументы событий **Клавиша вниз (KeyDown)**, **Нажатие клавиши (KeyPress)**, и **Клавиша вверх (KeyUp)** вместе с аргументами событий **Кнопка вниз**, **Кнопка вверх** и **Перемещение указателя** позволяют обеспечить работу приложения для любых пользователей, как работающих с мышью, так и использующих клавиатуру. Например, часто возникает необходимость скоординировать фокус событий клавиатуры с фокусом событий мыши. Обычно оба фокуса находятся на одном объекте. Например, если пользователь переводит указатель в поле и нажимает кнопку мыши, фокусы событий клавиатуры и мыши перемещаются в это поле. Однако, перемещение указателя по элементу управления не переводит фокус события клавиатуры на этот элемент управления. Для перевода фокуса следует нажать кнопку мыши, воспользоваться клавишами перемещения курсора или перевести фокус на этот элемент управления в процедуре обработки события **Перемещение указателя**.

Поскольку событие **Перемещение указателя** возникает для каждой области формы, по которой перемещается указатель, то, если не принять специальные меры, существует вероятность возникновения каскадных событий **Перемещение указателя**. Рекомендуется принимать меры, чтобы в элементе управления или разделе, уже принявшем событие **Перемещение указателя**, пока указатель находится в данной области формы, это событие не возникало вновь. Для этого следует определить на уровне модуля переменную, значение которой указывает, произошло ли уже событие **Перемещение указателя** для данного раздела формы или элемента управления.

Отмена события **Перемещение указателя** не допускается.

Событие «Перемещение указателя» (MouseMove) - Пример процедуры обработки события

В данном примере определяется положение указателя мыши и осуществляется проверка, были ли нажаты левая кнопка мыши и/или клавиша SHIFT. По мере перемещения указателя его x- и y-координаты выводятся в надписи.

Для проверки работы данного примера следует поместить процедуру обработки события в форму, которая содержит надпись с именем «Координаты».

```
Private Sub Detail_MouseMove(Button As Integer, Shift As Integer, _
      X As Single, Y As Single)
   Dim intShiftDown As Integer, intLeftButton As Integer

   Me!Координаты.Caption = X & ", " & Y
   ' Создает маски для проверки клавиши SHIFT и левой кнопки.
   intShiftDown = Shift And acShiftMask
   intLeftButton = Button And acLeftButton
   ' Проверяет, нажаты ли клавиша SHIFT и левая кнопка.
   If intShiftDown And intLeftButton > 0 Then
      MsgBox "Нажаты клавиша SHIFT и левая кнопка мыши."
   End If
End Sub
```

Событие «Изменение размера» (Resize)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtResizeC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtResizeX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtResizeMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtResizeEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtResizeA"}

Событие **Изменение размера (Resize)** возникает при открытии формы и при любом изменении размера формы.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Изменение размера (OnResize)**.

Данное событие возникает при изменении размера формы в макросе или в процедуре обработки события, например, при изменении размера формы с помощью макрокоманды **СдвигРазмер (MoveSize)**.

Макросы или процедуры обработки события **Изменение размера** позволяют переместить или изменить размер элемента управления, когда изменяется размер содержащей этот элемент управления формы. Кроме того, событие **Изменение размера** позволяет выполнить пересчет вычисляемых элементов управления или сброс значений свойств, зависящих от размеров формы.

При первом открытии формы возникает следующая цепочка событий:

Открытие ⇒ Загрузка ⇒ Изменение размера ⇒ Включение ⇒ Текущая запись

Примечание. При использовании в макросе или в процедуре обработки события **Изменение размера** макрокоманд **СдвигРазмер (MoveSize)**, **Развернуть (Maximize)**, **Свернуть (Minimize)** и **Восстановить (Restore)** (или соответствующих методов объекта **DoCmd**) необходимо соблюдать осторожность. Эти макрокоманды могут запустить событие **Изменение размера** формы и таким образом вызвать каскадные события.

Событие «Изменение размера» (Resize) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtResizeMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtResizeMOX":1}

Для выполнения макроса, связанного с событием **Изменение размера (Resize)**, следует указать имя этого макроса в качестве значения свойства **Изменение размера (OnResize)**.

Дополнительные сведения

Для того чтобы при каждом изменении размера формы происходило обновление экрана, следует в макросе, связанном с событием **Изменение размера**, вызвать макрокоманду **ОбновитьОбъект (RepaintObject)**.

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросах, связанных с событием **Изменение размера**.

Событие «Изменение размера» (Resize) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtResizeEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtResizeEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Изменение размера (Resize)**, следует выбрать в качестве значения свойства **Изменение размера (OnResize)** элемент [Процедура обработки события] и нажать кнопку **Построить**  .

Синтаксис

Private Sub Form_Resize( )

Дополнительные сведения

Для того чтобы при каждом изменении размера формы происходило обновление экрана, следует в процедуре обработки события **Изменение размера** вызвать метод **Repaint**.

Отмена события **Изменение размера** не допускается.

Событие «Изменение размера» (Resize) - Пример процедуры обработки события

В данном примере демонстрируется применение процедуры обработки события **Изменение размера (Resize)** для обновления экрана при развертывании окна формы. При нажатии пользователем кнопки **Развернуть** форма развертывается и вызывается событие **Изменение размера**.

Для проверки работы данного примера следует поместить процедуры обработки события в форму «Контакты», которая содержит кнопку **Развернуть**.

```
Private Sub Развернуть_Click()
   DoCmd.Maximize
End Sub

Private Sub Form_Resize()
   Forms!Контакты.Repaint
End Sub
```

Событие «Возврат» (Retreat)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtRetreatC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtRetreatX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtRetreatMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtRetreatEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtRetreatA"}

Событие **Возврат (Retreat)** возникает, когда Microsoft Access возвращается к предыдущему разделу отчета во время форматирования отчета.

Примечание. Событие **Возврат** определено для всех разделов отчета за исключением верхних колонтитулов и нижних колонтитулов.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Возврат (OnRetreat)**.

При некоторых обстоятельствах Microsoft Access возвращается к предыдущему разделу отчета для того, чтобы определить расположение некоторых элементов управления и разделов отчета и возможности их размещения на выделенном пространстве. Примерами могут служить:

- Группы (за исключением верхних и нижних колонтитулов), для которых свойство **Не разрывать (KeepTogether)** имеет значение «Полную группу» или «Первую область данных».
- Подчиненные отчеты или подчиненные формы, для которых свойства **Расширение (CanGrow)** и/или **Сжатие (CanShrink)** имеют значение «Да».
- Разделы на последней странице отчета.

В подобных ситуациях во время проверки расположения разделов на печатной странице возникает событие **Форматирование (Format)** (однако событие **Печать (Print)** не возникает, поскольку разделы еще не выводятся на печать). Если какие-либо разделы не могут быть напечатаны на текущей странице, то происходит возврат к предыдущим разделам, которые действительно могут быть напечатаны, а не умещающиеся на странице разделы будут перенесены на следующую страницу. Событие **Возврат** возникает для каждого раздела, к которому было проведено возвращение. После печати текущей страницы для каждого из не напечатанных разделов снова возникает событие **Форматирование** при подготовке этих разделов к печати.

Например, каждый раз, когда Microsoft Access доходит при форматировании отчета до конца последней страницы, возникает событие **Возврат** для каждого из предыдущих разделов. Это продолжается, пока не будет достигнут первый раздел у верхнего края последней страницы. После этого снова возникают события **Форматирование** каждого из разделов на странице, за которыми следуют события **Печать**.

Процедура обработки события или макросы, вызываемые при возникновении события **Возврат**, позволяют отменить все изменения, которые были внесены при возникновении события **Форматирование** раздела. Это оказывается полезным, когда в макросе или в процедуре обработки события **Форматирование** выполняются определенные действия, такие как вычисление итоговых значений для страницы или изменение размеров раздела, которые следует выполнить только один раз для каждого раздела.

Событие **Возврат** применяется также для обеспечения нужного расположения элементов отчета. Например, с помощью этого события в отчете «Продажи по годам» в базе данных «Борей» осуществляется проверка, следует ли печатать верхний колонтитул (верхний колонтитул печатается на страницах, следующих за страницей, на которой напечатан заголовок группы, и не печатается на страницах, следующих за страницей, на которой напечатано

примечание группы).

Событие «Возврат» (Retreat) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtRetreatMOC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtRetreatMOX":1}

Для выполнения макроса, связанного с событием **Возврат (Retreat)**, следует указать имя этого макроса в качестве значения свойства **Возврат (OnRetreat)**.

Дополнительные сведения

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросах, связанных с событием **Возврат**.

Событие «Возврат» (Retreat) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtRetreatEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtRetreatEPX":1}

Для создания процедуры обработки событий, которая выполняется при возникновении события **Возврат (Retreat)**, следует выбрать в качестве значения свойства **Возврат (OnRetreat)** элемент [Процедура обработки события] и нажать кнопку **Построить** ...

Синтаксис

Private Sub *имяРаздела*_**Retreat( )**

Процедура обработки события **Возврат** использует следующий аргумент.

Аргумент	Описание
имяРаздела	Имя раздела отчета, к которому применяется процедура обработки события **Возврат**.

Дополнительные сведения

Отмена события **Возврат** не допускается.

Событие «Таймер» (Timer)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtTimerC;vafctTimer"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtTimerX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtTimerMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS":"acevtTimerEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïS":"acevtTimerA"}

Событие **Таймер (Timer)** возникает регулярно через интервал времени, заданный в значении свойства формы **Интервал таймера (TimerInterval)**.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя макроса или элемент [Процедура обработки события] в качестве значения свойства **Таймер (OnTimer)**.

Запуск макроса или выполнение процедуры обработки события в ответ на событие **Таймер** позволяет управлять действиями Microsoft Access, которые должны выполняться периодически по сигналу таймера. Например, можно проводить через заданные интервалы времени повторный запрос по базовым записям или обновление экрана.

Значение свойства **Интервал таймера (TimerInterval)** формы определяет интервал времени между событиями **Таймер** (в миллисекундах). Допустимые значения интервала составляют от 0 до 65 535 миллисекунд. Значение 0 свойства **Интервал таймера** отменяет возникновение событий **Таймер**.

Событие «Таймер» (Timer) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtTimerEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtTimerEPX":1}

Для создания процедуры обработки событий, которая выполняется при возникновении события **Таймер (Timer)**, следует выбрать в качестве значения свойства **Таймер (OnTimer)**. элемент [Процедура обработки события] и нажать кнопку **Построить** 

Синтаксис

Private Sub Form_Timer( )

Дополнительные сведения

Отмена события **Таймер** не допускается.

Событие «Таймер» (Timer) - Пример процедуры обработки события

В данном примере демонстрируются цифровые часы, с помощью которых в надписи в форме выводится текущее время по системным часам компьютера.

Для проверки работы данного примера следует добавить процедуру обработки события в форму, которая содержит надпись «Часы». Для обновления показаний часов каждую секунду следует задать значение 1000 миллисекунд для свойства **Интервал таймера (TimerInterval)** формы.

```
Private Sub Form_Timer()
   Часы.Caption = Time      ' Обновляет значение времени.
End Sub
```

Событие «Таймер» (Timer) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtTimerMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtTimerMOX":1}

Для выполнения макроса, связанного с событием **Таймер (Timer)**, следует указать имя этого макроса в качестве значения свойства **Таймер (OnTimer)**.

Дополнительные сведения

Не допускается вызов макрокоманды **ОтменитьСобытие (CancelEvent)** в макросах, связанных с событием **Таймер**. Однако, если задать для свойства формы **Интервал таймера (TimerInterval)** значение 0, события **Таймер** не возникают.

События «Удаление» (Delete), «До подтверждения Del» (BeforeDelConfirm), «После подтверждения Del» (AfterDelConfirm)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtDeleteBeforeAfterDelConfirmC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtDeleteBeforeAfterDelConfirmX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acevtDeleteBeforeAfterDelConfirmMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS":"acevtDeleteBeforeAfterDelConfirmEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïSnïS":"acevtDeleteBeforeAfterDelConfirmA"}

- Событие **Удаление (Delete)** возникает при выполнении пользователем определенного действия, например нажатия клавиши DEL для удаления записи, но до фактического удаления записи.
- Событие **До подтверждения Del (BeforeDelConfirm)** возникает после удаления пользователем одной или нескольких записей в буфер, но до вывода диалогового окна Microsoft Access с приглашением подтвердить удаление.
- Событие **После подтверждения Del (AfterDelConfirm)** возникает после подтверждения пользователем и фактического удаления записей или при отмене удаления.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Удаление (OnDelete)**, **До подтверждения Del (BeforeDelConfirm)** или **После подтверждения Del (AfterDelConfirm)**.

После удаления записи эта запись помещается во временный буфер. Событие **До подтверждения Del** возникает после события **Удаление** (или, если удаляется несколько записей, после того, как событие **Удаление** произошло для всех удаляемых записей), но до вывода диалогового окна **Подтверждение удаления**. Отмена события **До подтверждения Del** приводит к восстановлению записи или записей из буфера, и диалоговое окно с приглашением подтвердить удаление не выводится.

Событие **После подтверждения Del** возникает после фактического удаления записи или записей, или после отмены всех удалений. Если событие **До подтверждения Del** не отменяется, то событие **После подтверждения Del** возникает после вывода диалогового окна **Подтверждение удаления**. Событие **После подтверждения Del** возникает даже в том случае, когда событие **До подтверждения Del** было отменено. Сведения о состоянии операции удаления возвращаются в процедуре обработки события **После подтверждения Del**. Например, в макросе (или в процедуре обработки) для события **После подтверждения Del** возможен пересчет итоговых значений, которые затрагиваются операцией удаления записей.

При отмене события **Удаление** события **До подтверждения Del** и **После подтверждения Del** не возникают и диалоговое окно с приглашением подтвердить удаление не выводится.

Примечание. Если в диалоговом окне **Параметры** (открывающемся с помощью меню **Сервис**) на вкладке **Правка/поиск** в группе **Подтверждение** снят флажок **Изменения записей**, то события **До подтверждения Del** и **После подтверждения Del** не возникают, и диалоговое окно **Подтверждение удаления** не выводится.

В макросе или в процедуре обработки события, выполняемых в ответ на событие **Удаление (Delete)**, пользователь имеет возможность предотвратить или разрешить удаление записи только при выполнении определенных условий. Можно также использовать событие **Удаление** для вывода диалогового окна с приглашением подтвердить удаление записи.

Для того чтобы удалить запись, следует выбрать команду **Удалить запись** в меню **Правка**. При этом удаляется текущая запись (указанная с помощью области выделения записи). Для удаления записи можно также выбрать в меню **Правка** команду **Выделить запись** и нажать

клавишу DEL. При выборе команды **Удалить запись**, области выделения для текущей записи или команды **Выделить запись** возникают события **Выход (Exit)** и **Потеря фокуса (LostFocus)** для элемента управления, имеющего фокус. Если какие-либо данные в записи были изменены, то перед событиями **Выход** и **Потеря фокуса** возникают события **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)**. Если в области выделения записи была выделена запись, не являющаяся текущей, то для этой записи возникает также событие **Текущая запись (Current)**.

После удаления записи фокус перемещается на следующую за удаленной запись, и для нее возникает событие **Текущая запись (Current)**, за которым следуют события **Вход (Enter)** и **Получение фокуса (GotFocus)** для первого элемента управления в этой записи.

Событие **До подтверждения Del** возникает непосредственно перед тем, как Microsoft Access выводит диалоговое окно **Подтверждение удаления**, в котором появляется приглашение подтвердить удаление. После ответа пользователя, подтверждающего или отменяющего удаление, возникает событие **После подтверждения Del**.

При одновременном удалении нескольких записей событие **Удаление** возникает после удаления каждой записи. Это позволяет получить доступ к данным в каждой записи перед ее фактическим удалением и выборочно подтверждать или отменять удаление каждой из записей в макросе или процедуре обработки события **Удаление**. При удалении нескольких записей событие **Текущая запись (Current)** для записи, следующей за последней удаленной записью, и события **Вход (Enter)** и **Получение фокуса (GotFocus)** для первого элемента управления в этой записи не возникают до завершения удаления всех выделенных записей. Другими словами, до удаления всех выделенных записей для каждой из них возникает событие **Удаление** и не возникают другие события. События **До подтверждения Del** и **После подтверждения Del** также не возникают до удаления всех выделенных записей.

События «Удаление» (Delete), «До подтверждения Del» (BeforeDelConfirm), «После подтверждения Del» (AfterDelConfirm) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtDeleteBeforeAfterDelConfirmMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtDeleteBeforeAfterDelConfirmMOX":1}

Чтобы запустить макрос при возникновении события **Удаление (Delete)**, **До подтверждения Del (BeforeDelConfirm)** или **После подтверждения Del (AfterDelConfirm)**, следует указать имя данного макроса в качестве значения свойства **Удаление (OnDelete)**, **До подтверждения Del (BeforeDelConfirm)** или **После подтверждения Del (AfterDelConfirm)**.

Дополнительные сведения

Макросы, связанные с событием **Удаление**, используются для вывода диалогового окна **Подтверждение удаления** с приглашением подтвердить удаление записи перед ее фактическим удалением.

Макросы, связанные с событиями **До подтверждения Del** или **После подтверждения Del**, используются для выполнения действий в ответ на удаление записи пользователем. Однако макросы, связанные с событиями **До подтверждения Del** и **После подтверждения Del**, не позволяют задавать значения аргументов, определяющих вывод диалогового окна с приглашением подтвердить удаление, и возвращать аргумент, определяющий состояние операции удаления, поэтому с данными событиями обычно связывают процедуры обработки событий.

Макрокоманда **ОтменитьСобытие (CancelEvent)** в макросе, связанном с событием **Удаление**, позволяет отменить удаление.

Макрокоманда **Отменить событие** в макросе, связанном с событием **До подтверждения Del**, позволяет отменить удаление для всех удаляемых записей. При этом диалоговое окно с приглашением подтвердить удаление не выводится. Однако даже при отмене события **До подтверждения Del** возникает событие **После подтверждения Del**.

Не допускается использование макрокоманды **Отменить событие** в макросе, связанном с событием **После подтверждения Del**.

События «Удаление» (Delete), «До подтверждения Del» (BeforeDelConfirm), «После подтверждения Del» (AfterDelConfirm) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtDeleteBeforeAfterDelConfirmEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtDeleteBeforeAfterDelConfirmEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении событий **Удаление (Delete)**, **До подтверждения Del (BeforeDelConfirm)** или **После подтверждения Del (AfterDelConfirm)**, следует выбрать в качестве значения свойства **Удаление (OnDelete)**, **До подтверждения Del (BeforeDelConfirm)** или **После подтверждения Del (AfterDelConfirm)** элемент [Процедура обработки событий] и нажать кнопку **Построить** ...

Синтаксис

Private Sub Form_Delete(Cancel **As Integer)**
Private Sub Form_BeforeDelConfirm(Cancel **As Integer,** Response **As Integer)**
Private Sub Form_AfterDelConfirm(Status **As Integer)**

Процедура обработки события **Удаление** использует следующий аргумент:

Аргумент	Описание
Cancel	Определяет возникновение события **Удаление (Delete)**. Заданное для аргумента «Cancel» значение **True** (–1) отменяет событие **Удаление**.

Процедура обработки события **До подтверждения Del** использует следующие аргументы:

Аргумент	Описание
Cancel	Определяет возникновение события **До подтверждения Del (BeforeDelConfirm)**. Заданное для аргумента «Cancel» значение **True** отменяет событие **До подтверждения Del** и вывод диалогового окна **Подтверждение удаления**. Если событие отменяется, восстанавливаются исходные записи, однако при этом все равно возникает событие **После подтверждения Del (AfterDelConfirm)**. Если аргумент «Cancel» получает значение **True**, аргумент «Response» игнорируется. Если аргумент «Cancel» имеет значение **False** (0) (значение по умолчанию), для определения отклика на событие удаления в Microsoft Access используется значение аргумента «Response».
Response	Определяет, выводит ли Microsoft Access диалоговое окно **Подтверждение удаления** с приглашением подтвердить удаление записи. В аргументе «Response» используются следующие встроенные константы:

Константа	Описание
acDataErrContinue	Удаление записей без вывода диалогового окна **Подтверждение удаления**. Задание аргументу «Cancel» значения **False** и аргументу «Response» значения **acDataErrContinue** позволяет Microsoft Access удалять записи без приглашения подтвердить удаление.

AcDataErrDisplay	Используется по умолчанию. Выводит диалоговое окно **Подтверждение удаления**.

Процедура обработки события **После подтверждения Del** использует следующий аргумент:

Аргумент	**Описание**
Status	Определяет состояние операции удаления. В аргументе «Status» используются следующие встроенные константы:

Константа	**Описание**
acDeleteOK	Показывает, что удаление проведено успешно.
AcDeleteCancel	Показывает, что удаление отменено в программе Visual Basic.
AcDeleteUserCancel	Показывает, что удаление отменено пользователем.

Дополнительные сведения

Отмена события **После подтверждения Del (AfterDelConfirm)** не допускается.

Данные события позволяют управлять операцией удаления. Если пользователь удаляет, например, запись о сотруднике, то возникает событие **Удаление**. Разработчик имеет возможность включить в процедуру обработки события **Удаление** программу, сохраняющую содержимое полей «КодСотрудника» и «Фамилия» во временном наборе переменных. Когда возникает событие **До подтверждения Del**, для отключения вывода диалогового окна **Подтверждение удаления** следует определить аргумент «Response» с помощью константы **acDataErrContinue**. Это позволит вывести специальное диалоговое окно. При возникновении события **После подтверждения Del** и задании для аргумента «Status» константы **acDeleteOK** можно очистить временные переменные или сохранить информацию о сотруднике в другой таблице.

События «Удаление» (Delete), «До подтверждения Del» (BeforeDelConfirm), «После подтверждения Del» (AfterDelConfirm) - Пример процедур обработки событий

В данном примере демонстрируется, как запретить пользователям удаление записей из таблицы.

Для проверки данного примера включите следующую процедуру обработки события в форму, созданную на основе таблицы. Переведите форму в режим таблицы и попытайтесь удалить запись.

```
Private Sub Form_Delete(Cancel As Integer)
   Cancel = True
   MsgBox "Запись удалить нельзя."
End Sub
```

В следующем примере демонстрируется использование процедуры обработки события **До подтверждения Del (BeforeDelConfirm)** для отмены вывода стандартного диалогового окна **Подтверждение удаления** и вывода специального диалогового окна при удалении записи. Кроме того, демонстрируется использование процедуры обработки события **После подтверждения Del (AfterDelConfirm)** для вывода сообщений о нормальном выполнении операции удаления или о ее отмене в программе Visual Basic или пользователем.

Для проверки данного примера включите следующую процедуру обработки события в форму, созданную на основе таблицы или запроса.

```
Private Sub Form_BeforeDelConfirm(Cancel As Integer, _
      Response As Integer)
   ' Подавляет вывод стандартного диалогового окна.
   Response = acDataErrContinue
   ' Выводит специальное диалоговое окно.
   If MsgBox("Удалить запись?", vbOKCancel) = vbCancel Then
      Cancel = True
   End If
End Sub

Private Sub Form_AfterDelConfirm(Status As Integer)
   Select Case Status
      Case acDeleteOK
         MsgBox "Удаление выполнено нормально."
      Case acDeleteCancel
         MsgBox "Удаление отменено в программе."
      Case acDeleteUserCancel
         MsgBox "Удаление отменено пользователем."
   End Select
End Sub
```

События «До вставки» (BeforeInsert), «После вставки» (AfterInsert)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtBeforeAfterInsertC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtBeforeAfterInsertX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtBeforeAfterInsertMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїS":"acevtBeforeAfterInsertEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїS":"acevtBeforeAfterInsertA"}

- Событие **До вставки (BeforeInsert)** возникает при вводе пользователем первого символа в новую запись, но до фактического создания записи.
- Событие **После вставки (AfterInsert)** возникает после добавления новой записи.

Примечание. Задание значения элемента управления с помощью макроса или программы Visual Basic не приводит к возникновению этих событий.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этими событиями, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **До вставки (BeforeInsert)** или **После вставки (AfterInsert)**.

Процедура обработки события или макрос, связанные с событием **После вставки**, используются для выполнения повторного запроса для набора записей при каждом добавлении новой записи.

События **До вставки (BeforeInsert)** и **После вставки (AfterInsert)** подобны событиям **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)**. Эти события возникают в следующем порядке:

До вставки ⇒ До обновления ⇒ После обновления ⇒ После вставки.

В следующей таблице приведены описания момента возникновения каждого из этих событий.

Событие	Момент возникновения
До вставки	Пользователь вводит первый символ в новую запись.
До обновления	Пользователь обновляет запись.
После обновления	Запись обновлена.
После вставки	Обновленная запись является новой записью.

Если первый символ новой записи вводится в поле или в поле со списком, событие **До вставки** возникает до события **Изменение (Change)**.

События «До вставки» (BeforeInsert), «После вставки» (AfterInsert) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtBeforeAfterInsertMOC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtBeforeAfterInsertMOX":1}

Чтобы запустить макрос при возникновении событий **До вставки (BeforeInsert)** и **После вставки (AfterInsert)**, следует указать имя данного макроса в качестве значения свойства **До вставки (BeforeInsert)** или **После вставки (AfterInsert)**.

Дополнительные сведения

Макросы, связанные с данными событиями, применяются для вывода сообщений или других полезных сведений. Например, макрос, связанный с событием **До вставки**, позволяет выводить на экран сведения, необходимые пользователю при вводе новой записи.

Макрокоманда **Отменить событие (CancelEvent)** в макросе, связанном с событием **До вставки**, позволяет отменить создание новой записи. При вызове данной макрокоманды фокус возвращается на новую пустую запись (введенный пользователем символ будет удален).

Не допускается использование макрокоманды **ОтменитьСобытие** в макросе, связанном с событием **После вставки**.

События «До вставки» (BeforeInsert), «После вставки» (AfterInsert) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtBeforeAfterInsertEPC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtBeforeAfterInsertEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении событий **До вставки (BeforeInsert)** и **После вставки (AfterInsert)**, следует выбрать в качестве значения свойства **До вставки (BeforeInsert)** или **После вставки (AfterInsert)** элемент [Процедура обработки события] и нажать кнопку **Построить** 

Синтаксис

Private Sub Form_BeforeInsert(Cancel **As Integer)**
Private Sub Form_AfterInsert( )

Процедура обработки события **До вставки** использует следующий аргумент.

Аргумент	Описание
Cancel	Определяет возникновение события **До вставки (BeforeInsert)**. Задание для аргумента «Cancel» значения **True** (–1) отменяет событие **До вставки**.

Дополнительные сведения

Отмена события **После вставки** не допускается.

События «До вставки» (BeforeInsert), «После вставки» (AfterInsert) - Пример процедур обработки событий

В данном примере демонстрируется использование процедуры обработки события **До вставки (BeforeInsert)** для проверки, действительно ли пользователь хочет вставить новую запись. Процедура обработки события **После вставки (AfterInsert)** используется для выполнения повторного запроса к источнику записей для формы «Сотрудники» после добавления записи.

Для проверки данного примера включите следующую процедуру обработки события в форму «Сотрудники», созданную на основе таблицы или запроса. Переведите форму в режим таблицы и попытайтесь вставить запись.

```
Private Sub Form_BeforeInsert(Cancel As Integer)
   If MsgBox("Вставить новую запись?", vbOKCancel) = vbCancel Then
      Cancel = True
   End If
End Sub

Private Sub Form_AfterInsert()
   Forms!Сотрудники.Requery
End Sub
```

Событие «При обновлении» (Updated)

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtUpdatedC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtUpdatedX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïS":"acevtUpdatedMO":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïSnïSnïS nïSnïSnïSnïSnïSnïS":"acevtUpdatedEP":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acevtUpdatedA"}

Событие **При обновлении (Updated)** возникает при изменении данных в объекте OLE.

Примечание. Событие **При обновлении** определено только для элементов управления в формах. Оно не определено для элементов управления в отчетах.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **При обновлении (OnUpdated)**.

Это событие используется для проверки, были ли данные в объекте OLE изменены со времени последнего сохранения.

Событие **При обновлении** возникает при модификации данных в объекте OLE. Это обновление может быть результатом изменения объекта в приложении, в котором был создан объект, или изменения одной из связанных копий данного объекта. Как следствие, данное событие не синхронизуется с другими событиями элемента управления Microsoft Access.

Примечания

- Событие **При обновлении (Updated)** и события **До обновления (BeforeUpdate)** и **После обновления (AfterUpdate)** для присоединенной и свободной рамки объекта не связаны друг с другом. Событие **При обновлении** возникает при изменении данных в объекте OLE, а события **До обновления** и **После обновления** возникают при обновлении данных. Хотя эти события и не являются жестко связанными друг с другом, при изменении данных в объекте OLE обычно возникают все три события. Событие **При обновлении** как правило (но не обязательно) возникает до событий **До обновления** и **После обновления**.
- Элемент управления-календарь, имеющийся в Microsoft Access 97, более не поддерживает событие **При обновлении**. Если база данных, содержащая данный элемент, преобразуется из предыдущей версии Microsoft Access к Microsoft Access 97, все программы, связанные с событием **При обновлении** для элемента управления-календаря, должны использоваться с событием **После обновления**.

Событие «При обновлении» (Updated) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtUpdatedMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acevtUpdatedMOX":1}

Чтобы запустить макрос при возникновении события **При обновлении (Updated)**, следует указать имя данного макроса в качестве значения свойства **При обновлении (OnUpdated)**.

Дополнительные сведения

При обновлении данных в объекте OLE для вывода сообщения можно использовать макросы, связанные с событием **При обновлении**. Однако макросы не позволяют возвращать код, определяющий способ обновления объекта OLE, поэтому с данным событием как правило используются процедуры обработки событий.

В макросе, связанном с событием **При обновлении**, не допускается использование макрокоманды **ОтменитьСобытие (CancelEvent)**.

Событие «При обновлении» (Updated) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtUpdatedEPC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїS":"acevtUpdatedEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **При обновлении (Updated)**, следует выбрать в качестве значения свойства **При обновлении (OnUpdated)** элемент [Процедура обработки события] и нажать кнопку **Построить** 

Синтаксис

Private Sub *имяЭлемента***_Updated(**Code **As Integer)**

Процедура обработки события **При обновлении** использует следующие аргументы.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **При обновлении (Updated)**.
Code	Определяет способ обновления объекта OLE. В аргументе «Code» используются следующие встроенные константы:

Константа	Описание
acOLEChanged	Указывает, что данные в объекте были изменены.
AcOLESaved	Указывает, что данные в объекте были сохранены в приложении, в котором объект был создан. Это обычно происходит, когда пользователь выбирает в приложении команду **Обновить** или другую команду, используемую для тех же целей.
AcOLEClosed	Указывает, что файл, содержащий данные объекта, был закрыт в приложении, в котором объект был создан.
AcOLERenamed	Указывает, что файл, содержащий данные объекта, был переименован в приложении, в котором объект был создан.

Дополнительные сведения

Чтобы определить, изменялись ли данные в объекте OLE после его последнего сохранения, используйте в процедуре обработки события **При обновлении** общую переменную, указывающую, что объект необходимо сохранить. После сохранения объекта OLE задайте переменной исходное значение.

Отмена события **При обновлении** не допускается

Событие «Отсутствие в списке» (NotInList)

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS. nїSnїSnїSnїSnїS":"acevtNotInListC"} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acevtNotInListX":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїS":"acevtNotInListMO":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїSnїSnїS nїSnїSnїSnїSnїSnїS":"acevtNotInListEP":1} {ewc HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acevtNotInListA"}

Если пользователь вводит в тестовое поле поля со списком значение, которое не предусмотрено для данного поля со списком, возникает событие **Отсутствие в списке (NotInList)**.

Примечание. Событие **Отсутствие в списке (NotInList)** определено только для элементов управления в формах. Оно не определено для элементов управления в отчетах.

Дополнительные сведения

Для выполнения макроса или процедуры обработки события, связанных с этим событием, следует указать имя данного макроса или элемент [Процедура обработки события] в качестве значения свойства **Отсутствие в списке (OnNotInList)**.

Данное событие позволяет пользователю добавить новое значение в список поля со списком.

Для возникновения события **Отсутствие в списке** свойство **Ограничиться списком (LimitToList)** должно иметь значение «Да».

Событие **Отсутствие в списке** не вызывает события **Ошибка (Error)**.

Событие **Отсутствие в списке** возникает для поля со списком, в котором свойство **Ограничиться списком** имеет значение «Да», в момент, когда после ввода отсутствующего в списке значения делается попытка перейти на другой элемент управления или сохранить запись. Это событие возникает после всех событий **Изменение (Change)** для данного поля со списком.

Если для свойства **Автоподстановка (AutoExpand)** задано значение «Да», то при вводе пользователем символов в поле со списком, Microsoft Access отбирает в списке совпадающие значения. Если символы, вводимые пользователем, совпадают с первыми символами некоторого значения, имеющегося в списке (например, пользователь вводит значение «Иван», а в списке имеется «Иванов»), то при переходе к другому элементу управления или при сохранении записи событие **Отсутствие в списке (NotInList)** не возникает. Символы, которые Microsoft Access добавляет к символам, введенным пользователем, (в данном примере «ов») выбираются в поле со списком. Если в таких случаях требуется разрешить возникновение события **Отсутствие в списке** (например, когда пользователь добавляет в список поля со списком новое значение «Иван»), то после последнего символа нового значения следует ввести символ SPACE, BACKSPACE или DEL.

Если для свойства **Ограничиться списком (LimitToList)** задано значение «Да», и список поля со списком раскрыт, то при вводе пользователем символов в поле со списком, Microsoft Access отбирает в списке совпадающие значения, даже если свойство **Автоподстановка (AutoExpand)** имеет значение «Нет». При нажатии пользователем клавиши ENTER или при переходе к другому элементу управления или к другой записи совпадающее значение выводится в поле со списком. В этом случае событие **Отсутствие в списке** не возникает. Чтобы разрешить возникновение события **Отсутствие в списке**, пользователь не должен раскрывать список поля со списком.

Событие «Отсутствие в списке» (NotInList) - Макросы

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acevtNotInListMOC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acevtNotInListMOX":1}

Чтобы запустить макрос при возникновении события **Отсутствие в списке (NotInList)**, следует указать имя данного макроса в качестве значения свойства **Отсутствие в списке (OnNotInList)**.

Дополнительные сведения

Макросы, связанные с событием **Отсутствие в списке**, применяются для добавления нового значения в список поля со списком. При возникновении события **Отсутствие в списке** следует открыть специальное диалоговое окно и ввести новое значение в один из его элементов управления. Этот элемент управления должен быть присоединен к полю таблицы или запроса, которое является источником записей для поля со списком. После этого следует сохранить запись в специальном диалоговом окне и выполнить повторный запрос к полю со списком.

В макросе, связанном с событием **Отсутствие в списке**, не допускается использование макрокоманды **ОтменитьСобытие (CancelEvent)**.

Событие «Отсутствие в списке» (NotInList) - Процедуры обработки событий

{ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅ. пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtNotInListEPC"} {ewc HLP95EN.DLL,DYNALINK,"пїЅпїЅпїЅпїЅпїЅпїЅ":"acevtNotInListEPX":1}

Для создания процедуры обработки события, которая выполняется при возникновении события **Отсутствие в списке (NotInList)**, следует выбрать в качестве значения свойства **Отсутствие в списке (OnNotInList)** элемент [Процедура обработки события] и нажать кнопку **Построить** 

Синтаксис

Private Sub *имяЭлемента***_NotInList(**NewData **As String,** Response **As Integer)**

Процедура обработки события **Отсутствие в списке** использует следующие аргументы.

Аргумент	Описание
имяЭлемента	Имя элемента управления, для которого вызывается процедура обработки события **Отсутствие в списке (NotInList)**.
NewData	Строка, в которой Microsoft Access передает в процедуру обработки событий текст, введенный пользователем в текстовое поле поля со списком.
Response	Определяет способ обработки события **Отсутствие в списке (NotInList)**. В аргументе «Response» используются следующие встроенные константы:

Константа	Описание
acDataErrDisplay	Используется по умолчанию. Задает вывод стандартного сообщения о событии. Данная константа используется, когда требуется запретить пользователю добавление нового значения в список поля со списком.
acDataErrContinue	Стандартное сообщение не выводится. Данная константа используется при выводе специального сообщения для пользователя. Например, в процедуре обработки события может быть открыто специальное диалоговое окно с вопросом, следует ли сохранить новое значение. При нажатии кнопки «Да» процедура обработки события добавит новое значение в список и задаст в аргументе «Response» константу **acDataErrAdded**. При нажатии кнопки «Нет» процедура обработки событияи задаст в аргументе «Response» константу **acDataErrContinue**.
acDataErrAdded	Сообщение не выводится, но пользователь получает возможность добавить новое значение в список поля со списком в процедуре

обработки события **Отсутствие в списке (NotInList)**. После добавления значения Microsoft Access обновляет список, выполняя повторный запрос к полю со списком. После этого строковое значение снова сравнивается со списком и значение аргумента «NewData» сохраняется в поле, к которому присоединено поле со списком. Если строковое значение в списке не обнаружено, Microsoft Access выводит сообщение об ошибке.

Дополнительные сведения

Допускается создание процедуры обработки события **Отсутствие в списке (NotInList)**, которая предоставляет пользователю возможность добавить новый элемент в список поля со списком. Например, в такой процедуре может быть добавлена запись в таблицу или список, служащие источником данных для поля со списком.

Для добавления нового элемента в список процедура обработки события должна добавить значение аргумента «NewData» в источник данных поля со списком. Конкретный способ зависит от типа источника данных поля со списком, который определяется значениями свойств **Тип источника строк (RowSourceType)** и **Источник строк (RowSource)**. В примере для данного раздела демонстрируется добавление нового значения в список значений для поля со списком.

Для того чтобы позволить пользователю изменить значение, введенное в поле со списком, (например в специальном диалоговом окне), необходимо сделать значением поля со списком новое значение, введенное в специальном диалоговом окне. Это приведет к сохранению нового значения в поле, к которому присоединено поле со списком. Константа **acDataErrContinue** в аргументе «Response» обеспечит добавление нового значения в список поля со списком.

Событие «Отсутствие в списке» (NotInList) - Пример процедуры обработки событий

В следующем примере событие **Отсутствие в списке (NotInList)** используется для добавления элемента в поле со списком.

Для проверки данного примера создайте в форме поле со списком с именем «Цвета». Задайте для свойства поля со списком **Ограничиться списком** значение «Да». Для заполнения поля со списком выберите для свойства **Тип источника строк (RowSourceType)** значение «Список значений» и введите в ячейку свойства **Источник строк (RowSource)** список значений, разделенных точкой с запятой. Для данного свойства можно ввести, например, следующие значения: Красный; Зеленый; Синий.

Добавьте следующую процедуру обработки события в форму. Переключитесь в режим формы и введите новое значение в поле со списком.

```
Private Sub Colors_NotInList(NewData As String, _
      Response As Integer)
   Dim ctl As Control

' Возвращает объект Control, указывающий на поле со списком.
   Set ctl = Me!Colors
' Приглашение подтвердить ввод нового значения.
   If MsgBox("Значение отсутствует в списке. Добавить?", vbOKCancel) = vbOK
Then
' Значение аргумента "Response" определяет добавление в список.
      Response = acDataErrAdded
' Добавляет значение аргумента "NewData" в источник строк.
      ctl.RowSource = ctl.RowSource & ";" & NewData
   Else
' При нажатии кнопки "Отмена" подавляет вывод
' сообщения об ошибке и отменяет изменения.
      Response = acDataErrContinue
      ctl.Undo
   End If
End Sub
```

Примечание. В вышеприведенном примере новый элемент добавляется в свободное поле со списком. При добавлении элемента в присоединенное поле со списком необходимо добавить значение в поле источника данных. В большинстве случаев невозможно ограничиться добавлением одного поля в новую запись - для соответствия структуре данных в таблице и выполнения условий на данные может понадобиться несколько полей. Например, в новой записи необходимо заполнить значения всех ключевых полей. При динамическом добавлении элементов в присоединенное поле со списком необходимо вывести пользователю приглашение заполнить все обязательные поля, сохранить новую запись, а затем выполнить повторный запрос к полю со списком для вывода нового значения.

Пример синхронизации форм с помощью макроса

Следующий макрос открывает форму «Список товаров» в нижнем правом углу формы «Поставщики», которая содержит товары текущего поставщика. Следующий пример демонстрирует использование команд **ВыводНаЭкран (Echo)**, **Сообщение (MsgBox)**, **ОстановитьМакрос (StopMacro)**, **ОткрытьФорму (OpenForm)** и **СдвигРазмер (MoveSize)**. Здесь также показано применение условных выражений в макрокомандах **Сообщение (MsgBox)** и **ОстановитьМакрос (StopMacro)**.

Для просмотра аргументов каждой макрокоманды нажмите кнопку ....

	Условие	Команда	Пояснение
			Связан с кнопкой «Просмотр товаров» в форме «Поставщики».
...		ВыводНаЭкран	Запрещает изменение экрана при выполнении макроса (вывод на экран выключен).
...	IsNull([КодПоставщика])	Сообщение	Если текущий поставщик отсутствует в форме «Поставщики,» то выводится сообщение...
...	...	ОстановитьМакрос	... и выполнение макроса останавливается.
...		ОткрытьФорму	Открывает форму «Список товаров» и выводит товары текущего поставщика.
...		СдвигРазмер	Располагает форму «Список товаров» в нижнем правом углу формы «Поставщики».

Аргументы макрокоманды «ВыводНаЭкран» (Echo)

Аргумент	Значение
Включить вывод	Нет
Текст строки состояния	

Аргументы макрокоманды «Сообщение» (MsgBox)

Аргумент	Значение
Сообщение	Выберите поставщика, чьи товары нужно отобразить, и вновь нажмите кнопку «Просмотр товаров».
Сигнал	Да
Тип	Отсутствует
Заголовок	

Аргументы макрокоманды «ОстановитьМакрос» (StopMacro)

У этой команды нет аргументов.

Аргументы макрокоманды «ОткрытьФорму» (OpenForm)

Аргумент	Значение
Имя формы	Список товаров
Режим	Таблица
Имя фильтра	
Условие отбора	[КодПоставщика] = Forms![Поставщики]! [КодПоставщика]
Режим данных	Только чтение
Режим окна	Обычное

Аргументы макрокоманды «СдвигРазмер» (MoveSize)

Аргумент	Значение
От левого края	2.6см
От верхнего края	2.7см
Ширина	4.1см
Высота	1.25см

Пример применения фильтра с помощью макроса

Следующий макрос содержит набор макрокоманд для фильтрации записей из формы «Поставщики». Пример демонстрирует использование макрокоманд **ПрименитьФильтр (ApplyFilter)** и **ПоказатьВсеЗаписи (ShowAllRecords)**. Демонстрируется также применение условий для определения выбранного выключателя в группе параметров формы. Каждая строка макроса связана с выключателем, который предназначен для выбора набора записей, начинающихся на букву А,Б,В и т. д. или всех записей.

Для просмотра аргументов каждой макрокоманды нажмите кнопку  .

	Условие	Команда	Пояснение
			Связана с кнопками «А»-«Я » и кнопкой «Все» в форме «Поставщики».
	[Фильтр по имени организации] =1	ПрименитьФильтр	Фильтр по именам компаний, которые начинаются с букв А, Б, В, Г, или Д.
	[Фильтр по имени организации] =2	ПрименитьФильтр	Фильтр по именам компаний, начинающимся на букву Б.
	[Фильтр по имени организации] =3	ПрименитьФильтр	Фильтр по именам компаний, начинающимся на букву В или З.

... Строки команды для букв с Г по Ю имеют такой же формат как и строки для букв с А по В ...

	[Фильтр по имени организации] =26	ПрименитьФильтр	Фильтр по именам компаний, начинающимся на букву Я, Ж, Ш или Е.
	[Фильтр по имени организации] =27	ПоказатьВсеЗаписи	Показывает все записи.

Аргументы команды «ПрименитьФильтр» (ApplyFilter) (1)

Аргумент	Значение
Имя фильтра	
Условие отбора	[Название] Like "[ААБВГД]*"

Аргументы команды «ПрименитьФильтр» (ApplyFilter) (2)

Аргумент	Значение
Имя фильтра	
Условие отбора	[Название] Like "В*"

Аргументы команды «ПрименитьФильтр» (ApplyFilter) (3)

Аргумент	Значение
Имя фильтра	
Условие отбора	[Название] Like "[С3]*"

Аргументы команды «ПрименитьФильтр» (ApplyFilter) (4)

Аргумент	Значение
Имя фильтра	
Условие отбора	[Название] Like "[ЯЖШЕ]*"

Аргументы команды «ПоказатьВсеЗаписи» (ShowAllRecords)

У этой команды нет аргументов.

Пример проверки данных на допустимость с помощью макроса

Следующий макрос для проверки на допустимость проверяет почтовые индексы, введенные в форму «Поставщики». Здесь демонстрируется использование макрокоманд **Сообщение (MsgBox)**, **ОтменитьСобытие (CancelEvent)** и **КЭлементуУправления (GoToControl)**. Условное выражение осуществляет проверку страны и почтового индекса в записи формы. Если почтовый индекс не соответствует формату данной страны, то выводится сообщение и сохранение текущей записи отменяется. Затем управление передается элементу «ПочтовыйИндекс» для исправления ошибочных данных.

Для просмотра аргументов каждой макрокоманды нажмите кнопку .

	Условие	Команда	Пояснение
			Связан со свойством **До обновления (BeforeUpdate)** формы «Поставщики».
	[Страна] In ("Франция", "Италия", "Испания") And Len([ПочтовыйИндекс]) <> 5	Сообщение	Если длина почтового индекса не равна 5, то выводится сообщение и...
	...	ОтменитьСобытие	... событие отменяется.
	[Страна] In ("Австралия", "Сингапур") And Len([ПочтовыйИндекс]) <> 4	Сообщение	Если длина почтового индекса не равна 4, то выводится сообщение и...
	...	ОтменитьСобытие	... событие отменяется.
	([Страна] = "Канада") And ([Индекс] Not Like"[a-z][0-9] [a-z] [0-9][a-z][0-9]")	Сообщение	Если почтовый индекс не соответствует формату для Канады, то выводится сообщение и...
	...	ОтменитьСобытие	... событие отменяется.
		КЭлементуУправления	Перемещает фокус на элемент управления «ПочтовыйИндекс».

Аргументы макрокоманды «Сообщение» (MsgBox) (1)

Аргумент	Значение
Сообщение	Индекс должен состоять из 5 символов.
Сигнал	Да
Тип	Информационное
Заголовок	Ошибка в почтовом индексе

Аргументы макрокоманды «ОтменитьСобытие» (CancelEvent) (1)

У этой команды нет аргументов.

Аргументы макрокоманды «Сообщение» (MsgBox) (2)

Аргумент	Значение
Сообщение	Индекс должен состоять из 4 символов.
Сигнал	Да
Тип	Информационное
Заголовок	Ошибка в почтовом индексе

Аргументы макрокоманды «ОтменитьСобытие» (CancelEvent) (2)

У этой команды нет аргументов.

Аргументы макрокоманды «Сообщение» (MsgBox) (3)

Аргумент	Значение
Сообщение	Индекс имеет недопустимое значение. Пример канадского индекса: H1J 1C3
Сигнал	Да
Тип	Информационное
Заголовок	Ошибка в почтовом индексе

Аргументы макрокоманды «ОтменитьСобытие» (3)

У этой команды нет аргументов.

Аргументы макрокоманды «КЭлементуУправления» (GoToControl)

Аргумент	Значение
Имя элемента	Индекс

Пример установки значения элемента управления с помощью макрокоманды

Следующий макрос открывает форму «Ввод товара» в нижней части формы «Поставщики». Здесь демонстрируется использование команд **ВыводНаЭкран (Echo)**, **ОткрытьФорму (OpenForm)**, **ЗадатьЗначение (SetValue)** и **КЭлементуУправления (GoToControl)**. Команда **ЗадатьЗначение (SetValue)** устанавливает значение элемента управления «Поставщик» в форме «Ввод товара», равное значению этого элемента в форме «Поставщики». Затем команда **КЭлементуУправления (GoToControl)** передает фокус полю «Марка», с которого начинается ввод нового товара.

Для просмотра аргументов каждой макрокоманды нажмите кнопку  .

	Команда	Пояснение
		Связан с кнопкой «Ввод товара» в форме «Поставщики».
	ВыводНаЭкран	Останавливает обновление экрана на время работы макроса (вывод на экран отключен).
	ОткрытьФорму	Открывает форму «Ввод товара».
	ЗадатьЗначение	Присваивает элементу «Поставщик» значение для текущего поставщика из формы «Поставщики».
	КЭлементуУправления	Переходит к элементу управления «Категория».

Аргументы макрокоманды «ВыводНаЭкран» (Echo)

Аргумент	Значение
Включить вывод	Нет
Текст строки состояния	

Аргументы макрокоманды «ОткрытьФорму» (OpenForm)

Аргумент	Значение
Имя формы	Товары
Режим	Форма
Имя фильтра	
Условие отбора	
Режим данных	Добавление
Режим окна	Обычное

Аргументы макрокоманды «ЗадатьЗначение» (SetValue)

Аргумент	Значение
Элемент	Forms![Товары]![КодПоставщика]
Выражение	КодПоставщика

Аргументы макрокоманды «КЭлементуУправления» (GoToControl)

Аргумент	Значение
Имя элемента	КодТипа

Свойство ReplicationConflictFunction

{ewc HLP95EN.DLL,DYNALINK,"nїSnїS.
nїSnїSnїSnїSnїS":"acproReplicationConflictFunctionC;damthCreateProperty;daobjProperty"} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїS":"acproReplicationConflictFunctionX":1} {ewc
HLP95EN.DLL,DYNALINK,"nїSnїSnїSnїSnїSnїSnїSnїSnїS":"acproReplicationConflictFunctionA;daobjDocument"}

Свойство **ReplicationConflictFunction** позволяет использовать специальную программу для разрешения конфликтов, возникающих при синхронизации реплик. После задания значения данного свойства Microsoft Access при требовании разрешить конфликт вызывает определенную пользователем функцию разрешения конфликтов вместо встроенного средства разрешения конфликтов.

Значения

В качестве значения свойства **ReplicationConflictFunction** задается строка, представляющая имя процедуры типа **Function,**, которую требуется вызвать. Значение обязательно должно быть именем процедуры типа **Function**; а не именем процедуры типа **Sub.** Если значение данного свойства не задано, Microsoft Access вызывает средство разрешения конфликтов.

При первом определении значения данного свойства в программе Visual Basic необходимо создать свойство и добавить его в семейство **Properties** определяемого пользователем объекта **Document.** В процедуру следует включить блок обработки ошибок, проверяющий, существует ли данное свойство в семействе **Properties**. Более подробное описание и пример см. в раздел справки для метода **CreateProperty.**

При создании и добавлении свойства **ReplicationConflictFunction** необходимо задать значение **True** (–1) для необязательного аргумента *DDL* метода **CreateProperty**. Этот аргумент гарантирует, что пользователь, не имеющий разрешения **dbSecWriteDef**, не сможет изменить или удалить новый объект **Property**. Если значение **True** не задано для этого аргумента, то специальная программа разрешения конфликтов не будет запущена.

Для того чтобы снова вернуться к встроенному средству разрешения конфликтов, следует удалить данное свойство из семейства **Properties**.

Допускается также добавление или удаление данного свойства на вкладке **Прочие** в окне диалога **Свойства** *имяБазыДанных*, которое открывается командой **Свойства базы данных** из меню **Файл**..

Дополнительные сведения

Если одна и та же запись реплицированной базы данных изменена в одной или нескольких репликах, то при синхронизации реплик в наборе возникает конфликт. Для разрешения этого конфликта вызывается встроенное средство разрешения конфликтов Microsoft Access. Для каждого конфликта мастер предоставляет пользователю выбор той из измененных записей, которая содержит правильные данные.

Для разрешения конфликтов в приложении могут применяться специальные процедуры, которые заменяют встроенные программы при установке значения свойства **ReplicationConflictFunction**. Следует создать отдельную функцию, которая будет служить точкой входа для созданной пользователем программы разрешения конфликтов. Затем задать в качестве значения свойства **ReplicationConflictFunction** имя этой функции.

При создании собственной процедуры разрешения конфликтов репликации пользователь должен предусмотреть обработку конфликтов как при репликации макета, так и при репликации данных. Для того чтобы ознакомиться с примером такой процедуры, откройте базу данных мастера Wzcnf80.mda в программной папке Microsoft Access и просмотрите исходный код.

Пример использования свойства ReplicationConflictFunction

Следующая функция устанавливает свойство Microsoft Access объекту доступа к данным (DAO). Эта функция может применяться для установки свойства **ReplicationConflictFunction**. Если свойство еще не присутствует в семействе **Properties** определяемого пользователем объекта **Document**, то Microsoft Access создаст и добавит его в семейство. Для установки этого свойства следует задать значение **True** (–1) для необязательного аргумента `blnDDL`.

```
Function SetAccessProperty(obj As Object, strName As String, _
      intType As Integer, varSetting As Variant, _
      Optional blnDDL As Boolean) As Boolean
   Dim prp As Property
   Const intPropNotFound As Integer = 3270

   On Error GoTo Error_SetAccessProperty
   ' Ссылается исключительно на семейство Properties.
   obj.Properties(strName) = varSetting
   obj.Properties.Refresh
   SetAccessProperty = True

Exit_SetAccessProperty:
   Exit Function

Error_SetAccessProperty:
   If Err = intPropNotFound Then
      ' Проверяется передан ли необязательный аргумент.
      If Not IsMissing(blnDDL) Then
         ' Создает свойство, помечает тип, устанавливает начальное
значение, задает DDL.
         Set prp = obj.CreateProperty(strName, intType, varSetting, blnDDL)
      Else
         ' Создает свойство, помечает тип, устанавливает начальное
значение.
         Set prp = obj.CreateProperty(strName, intType, varSetting)
      End If
      ' Добавляет объект Property в семейство Properties.
      obj.Properties.Append prp
      obj.Properties.Refresh
      SetAccessProperty = True
      Resume Exit_SetAccessProperty
   Else
      MsgBox Err & ": " & vbCrLf & Err.Description
      SetAccessProperty = False
      Resume Exit_SetAccessProperty
   End If
End Function
```

Следующая процедура вызывает функцию SetAccessProperty для установки свойства **ReplicationConflictFunction**:

```
Sub SetConflictResolver()
   Dim dbs As Database, ctr As Container, doc As Document
   Dim blnReturn As Boolean

   ' Возвращает ссылку на текущую базу данных.
   Set dbs = CurrentDb
   ' Возвращает ссылку на контейнер Databases.
```

```
    Set ctr = dbs.Containers!Databases
    ' Возвращает ссылку на объект SummaryInfo Document.
    Set doc = ctr.Documents!userdefined
    blnReturn = SetAccessProperty(doc, _
       "ReplicationConflictFunction", dbText, "CustomResolver", True)
    ' Вычисляет возвращаемое значение.
    If blnReturn = True Then
       Debug.Print "Свойство успешно установлено."
    Else
       Debug.Print "Не удалось установить свойство."
    End If
End Sub
```

Метод AccessError

{ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acmthAccessErrorC;daobjError;vaobjErr"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acmthAccessErrorX":1} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïSnïSnïSnïS":"acmthAccessErrorA"}

Метод **AccessError** возвращает описание ошибки Microsoft Access или объекта доступа к данным.

Синтаксис

[*приложение*]**.AccessError(***кодОшибки***)**

Метод **AccessError** использует следующие аргументы.

Аргумент	Описание
приложение	Необязательный аргумент. Представляет объект **Application**.
кодОшибки	Код ошибки, для которой возвращается описание.

Дополнительные сведения

Метод **AccessError** позволяет получить описание конкретной ошибки Microsoft Access или объекта доступа к данным (DAO) без фактического возникновения ошибки.

Метод **Raise** используется для вызова ошибки Visual Basic. После этого в свойстве **Description** объекта **Err** становится доступной строка описания ошибки.

Метод **Raise** не предназначен для вызова ошибки Microsoft Access или ошибки объекта доступа к данным (DAO). Кроме того, метод **AccessError** используют для возвращения описания ошибки без ее вызова.

Метод **AccessError** может использоваться для возвращения описания ошибки из события формы «Ошибка» (Error).

При возникновении ошибки Microsoft Access строка описания возвращается методом **AccessError** или с помощью свойства **Описание (Description)** объекта Visual Basic **Err**.

Пример использования метода AccessError

Следующая функция возвращает строку описания для ошибки с любым допустимым кодом:

```
Function ErrorString(lngError As Long) As String
   Const conAppError = "Ошибка, определяемая приложением или объектом."

   On Error Resume Next
   Err.Raise lngError
   If Err.Description = conAppError Then
      ErrorString = AccessError(lngError)
   ElseIf Err.Description = "" Then
      MsgBox "Для данного кода ошибки не предусмотрено сообщение."
   Else
      ErrorString = Err.Description
   End If
End Function
```

Функция CodeDb

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctCodeDBC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctCodeDbX":1}

Функцию **CodeDb** используют в модуле для определения имени объекта **Database,** описывающего открытую базу данных, в которой выполняется текущая программа. Функция **CodeDb** обеспечивает доступ к объектам доступа к данным, являющимся частью библиотечной базы данных.

Например, функция **CodeDb** может быть использована в модуле библиотечной базы данных для создания объекта **Database**, описывающего библиотечную базу данных. После этого становится возможным открытие и изменение набора записей, выбираемого из таблицы в библиотечной базе данных.

Синтаксис

Set *базаДанных* **= CodeDb**

Функция **CodeDb** использует следующий аргумент.

Аргумент	Описание
базаДанных	Объектная переменная типа **Database.**

Дополнительные сведения

Функция **CodeDb** возвращает объект **Database**, у которого значением свойства **Name** является полное (включая путь) имя файла базы данных, из которой была вызвана данная функция. Функция **CodeDb** используется для работы с объектами доступа к данным в библиотечной базе данных.

При вызове данной функции в библиотечной базе данных текущей остается база данных, из которой была вызвана функция, даже во время выполнения программы модуля библиотечной базы данных. Для ссылки на объекты доступа к данным в библиотечной базе данных необходимо знать имя объекта **Database**, представляющего библиотечную базу данных.

Предположим, например, что в библиотечной базе данных имеется таблица, содержащая список сообщений об ошибках. Для того чтобы выполнить из программы обработку данных в этой таблице следует с помощью функции **CodeDb** определить имя объекта **Database**, содержащего ссылку на библиотечную базу данных, в которой содержится таблица.

Если функция **CodeDb** вызывается в текущей базе данных, то она возвращает имя текущей базы данных аналогично функции **CurrentDb.**

Функция CodeDb, пример

В данном примере функция **CodeDb** возвращает объект **Database**, определяющий ссылку на библиотечную базу данных. В библиотечной базе данных содержатся таблица «Errors» и текущая выполняемая программа. После того как эти сведения считываются с помощью функции **CodeDb**, функция **GetErrorString** открывает табличный набор записей, базовой таблицей которого является таблица «Errors». Затем по значению типа **Integer**, переданному функции, из поля «ErrorData» извлекается сообщение об ошибке.

```
Function GetErrorString(ByVal intError As Integer) As String
   Dim dbs As Database, rst As RecordSet

   ' Переменная представляет базу данных, в которой выполняется программа.
   Set dbs = CodeDb
   ' Создает табличный объект Recordset.
   Set rst = dbs.OpenRecordSet("Errors", dbOpenTable)
   ' Определяет индекс по ключевому полю ("ErrorID").
   rst.Index = "PrimaryKey"
   ' Поиск номера ошибки, переданного функции GetErrorString.
   rst.Seek "=", intError
   ' Возвращает сообщение об ошибке.
   GetErrorString = rst.Fields!ErrorData.Value
   rst.Close
End Function
```

Функции CreateControl, CreateReportControl

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctCreateControlC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïSnïS":"acfctCreateControlX":1}

- Функция **CreateControl** создает элемент управления в указанной открытой форме.
- Функция **CreateReportControl** создает элемент управления в указанном открытом отчете.

Предположим, что требуется разработать специальную программу мастера, создающего форму определенного типа. В этом случае функция **CreateControl** обеспечивает добавление мастером элементов управления в создаваемую форму.

Синтаксис

CreateControl(*имяФормы***,** *типЭлемента***[,** *раздел***[,** *главный***[,** *имяПоля***[,** *слева***[,** *сверху***[,** *ширина***[,** *высота***]]]]]]])**

CreateReportControl(*имяОтчета***,** *типЭлемента***[,** *раздел***[,** *главный***[,** *имяПоля***[,** *слева***[,** *сверху***[,** *ширина***[,** *высота***]]]]]]])**

Функции **CreateControl** и **CreateReportControl** используют следующие аргументы.

Аргумент	Описание	
имяФормы, *имяОтчета*	Строковое выражение, определяющее имя открытой формы или отчета, в которых создается элемент управления.	
типЭлемента	Одна из следующих встроенных констант, определяющих тип создаваемого элемента управления. Пользователь имеет возможность просматривать эти константы в окне просмотра объектов и вставлять их в собственные программы. Нажмите кнопку **Просмотр объектов** ... на панели инструментов **Visual Basic**. Выберите **Access** в поле со списком **Проект/Библиотека** и **Constants** в списке **Классы**.	
	Константа	**Элемент управления**
	acLabel	Надпись
	acRectangle	Прямоугольник
	acLine	Линия
	acImage	Рисунок
	acCommandButton	Кнопка
	acOptionButton	Переключатель
	acCheckBox	Флажок
	acOptionGroup	Группа параметров
	acBoundObjectFrame	Присоединенная рамка объекта
	acTextBox	Поле
	acListBox	Список
	acComboBox	Поле со списком
	acSubform	Подчиненная форма
	acObjectFrame	Свободная рамка объекта
	acPage	Страница
	acPageBreak	Конец страницы
	acCustomControl	элемент ActiveX
	acToggleButton	Выключатель
	acTabCtl	набор вкладок
раздел	Одна из следующих из встроенных констант, определяющих	

раздел, в котором создается элемент управления.

Константа	Раздел
acDetail	(По умолчанию) Область данных
acHeader	Заголовок формы или отчета
acFooter	Примечание формы или отчета
acPageHeader	Верхний колонтитул
acPageFooter	Нижний колонтитул
acGroupLevel1Header	Заголовок группы уровня 1 (только в отчетах)
acGroupLevel1Footer	Примечание группы уровня 1 (только в отчетах)
acGroupLevel2Header	Заголовок группы уровня 2 (только в отчетах)
acGroupLevel2Footer	Примечание группы уровня 2 (только в отчетах)

Если в отчете имеются другие уровни группировки, то соответствующие пары областей заголовка и примечаний групп нумеруются последовательно, начиная с 9.

главный	Строковое выражение, определяющее имя главного элемента управления для присоединенного элемента управления. Для элементов управления, не являющихся присоединенными, данный аргумент должен быть пропущен, либо его значением может быть пустая строка.
имяПоля	Имя поля, с которыми связывается данный элемент управления. Если создается элемент управления неприсоединенный к полю, значением данного аргумента должна быть пустая строка.
слева, сверху	Числовые выражения, задающие координаты верхнего левого угла элемента управления в единицах твип.
ширина, высота	Числовые выражения, задающие ширину и высоту элемента управления в единицах твип.

Дополнительные сведения

Функции **CreateControl** и **CreateReportControl** используются в специальных программах мастеров для создания элементов управления в форме или отчете. Обе эти функции возвращают объект **Control.**

Функции **CreateControl** и **CreateReportControl** применимы только в режиме конструктора формы и в режиме конструктора отчета, соответственно.

Аргумент *главный* позволяет определить связь между главным и подчиненным элементами управления. Например, если поле имеет присоединенную подпись, то поле является главным (родительским) элементом управления, а подпись подчиненным (дочерним). При создании подписи значением аргумента *главный* должна быть строка, указывающая имя главного элемента управления. При создании поля значением аргумента *главный* должна быть пустая строка.

Аргумент *главный* необходимо задавать также для флажков, переключателей или выключателей, помещаемых в группу. Группа является главным элементом управления для содержащихся в ней флажков, переключателей и выключателей. Только подписи, флажки, переключатели и выключатели могут иметь главный элемент управления. Все эти элементы управления могут быть созданы и как самостоятельные, т.е. не имеющие главного элемента управления.

Значение аргумента *имяПоля* задается в соответствие с типом создаваемого элемента управления, а также с учетом того, является ли этот элемент присоединенным к полю в таблице. Присоединенным элементом может быть поле, список, поле со списком, группа и присоединенная рамка объекта. Для переключателя, выключателя и флажка присоединение допускается, если они не входят в группу.

Если указать в аргументе *имяПоля* имя поля таблицы, создается присоединенный элемент управления. Все свойства присоединенного элемента управления автоматически получают значения соответствующих свойств поля таблицы. Например, элемент управления автоматически получает то же значение свойства **Условие на значение (ValidationRule).**

Примечание. Мастер, создающий элементы управления в новых или существующих формах или отчетах, должен предварительно открыть документ в режиме конструктора.

Удалить элемент управления из формы или отчета позволяют инструкции **DeleteControl** и **DeleteReportControl.**

Функции CreateControl, CreateReportControl, пример

В данном примере сначала создается форма, имеющая базовую таблицу «Заказы». Затем функция **CreateControl** создает поле и присоединенную к нему подпись в форме.

```
Sub NewControls()
   Dim frm As Form
   Dim ctlLabel As Control, ctlText As Control
   Dim intDataX As Integer, intDataY As Integer
   Dim intLabelX As Integer, intLabelY As Integer

   ' Создает новую форму с базовой таблицей «Заказы».
   Set frm = CreateForm
   frm.RecordSource = "Заказы"
   ' Задает координаты элементов управления.
   intLabelX = 100
   intLabelY = 100
   intDataX = 1000
   intDataY = 100
   ' Создает в области данных неприсоединенное поле стандартных размеров.
   Set ctlText = CreateControl(frm.Name, acTextBox, , "", "", _
      intDataX, intDataY)
   ' Создает надпись, присоединенную к полю.
   Set ctlLabel = CreateControl(frm.Name, acLabel, , ctlText.Name, _
      "NewLabel", intLabelX, intLabelY)
   ' Восстанавливает окно формы.
   DoCmd.Restore
End Sub
```

Функции CreateForm, CreateReport

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctCreateFormC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctCreateFormX":1}

- Функция **CreateForm** создает форму и возвращает объект **Form.**
- Функция **CreateReport** создает отчет и возвращает объект **Report.**

Предположим, например, что требуется разработать специальную программу мастера, создающего отчет о продажах. В этом случае функция **CreateReport** обеспечивает создание мастером нового отчета, базирующегося на указанном шаблоне отчета.

Синтаксис

CreateForm([*базаДанных*[**,** *шаблонФормы*]]**)**
CreateReport([*базаДанных*[**,** *шаблонОтчета*]]**)**

Функции **CreateForm** и **CreateReport** используют следующие аргументы.

Аргумент	Описание
базаДанных	Строковое выражение, задающее имя базы данных, содержащей шаблон, используемый для создания формы или отчета. Если данный аргумент опущен, используется текущая база данных (значение, возвращаемое функцией **CurrentDb**). Если задана база данных отличная от текущей, то она должна быть открыта как библиотечная база данных.
шаблонФормы, *шаблонОтчета*	Строковое выражение, задающее имя формы или отчета, используемых как шаблон при создании новой формы или отчета. Если данный аргумент опущен, используется шаблон, указанный на вкладке **Формы/отчеты** в диалоговом окне **Параметры** (доступном по команде **Параметры** из меню **Сервис**).

Дополнительные сведения

Функции **CreateForm** и **CreateReport** используются в специальных программах мастеров для создания новых форм или отчетов.

При вызове функций **CreateForm** и **CreateReport** открывается новое свернутое в значок окно формы в режиме конструктора или отчета в режиме конструктора, соответственно.

В аргументах *шаблонФормы* и *шаблонОтчета* допускается указание как имен форм или отчетов, специально созданных для использования в качестве шаблонов, так и любых других форм или отчетов из базы данных, указанной в аргументе *базаДанных*. Если в аргументе *базаДанных* задана не текущая база, то она должна быть открыта как библиотечная база данных. Для получения дополнительных сведений по библиотечным базам данных см. главу 12 «Использование библиотечных баз данных и библиотек динамической компоновки» в руководстве *Разработка приложений для Microsoft Access 97.*

Примечание. Если шаблон формы или отчета создан в другой базе данных, то вместо загрузки библиотечной базы данных этот шаблон может быть импортирован в текущую базу. Убедитесь, что имя шаблона выводится в поле **Шаблон формы** или **Шаблон отчета** на вкладке **Формы/отчеты** в диалоговом окне **Параметры**. В дальнейшем формы и отчеты, создаваемые с помощью функций **CreateForm** или **CreateReport** будут базироваться на этом шаблоне.

Если в аргументе *шаблонФормы* или *шаблонОтчета* указано неверное имя, то Visual Basic

использует шаблон формы или отчета, указанный в поле **Шаблон формы** или **Шаблон отчета** на вкладке **Формы/отчеты** в диалоговом окне **Параметры**.

При создании формы или отчета с помощью функций **CreateForm** и **CreateReport** для свойства формы или отчета **Наличие модуля (HasModule)** устанавливается значение **False** (0). Если требуется, чтобы новая форма или отчет имели модуль класса, то для этого свойства следует установить значение **True** (–1).

Функции **CreateForm** и **CreateReport** создают формы и отчеты, свернутые в значки.

Функции CreateForm, CreateReport, примеры

В данном примере в текущей базе данных создается отчет с использованием шаблона, указанного в поле **Шаблон отчета** на вкладке **Формы/отчеты** в диалоговом окне **Параметры**.

```
Sub NormalReport()
   Dim rpt As Report

   Set rpt = CreateReport     ' Создаем отчет со свернутым окном.
   DoCmd.Restore              ' Восстанавливаем окно.
End Sub
```

В следующем примере в базе данных «Борей» создается новая форма на основе формы «Клиенты». В свойстве формы **Источник записей (RecordSource)** указывается таблица «Клиенты» из базы данных «Борей». Данную программу следует запускать в демонстрационной базе данных «Борей».

```
Sub NewForm()
   Dim frm As Form

   ' Создает форму на основе формы "Клиенты".
   Set frm = CreateForm( , "Клиенты")
   DoCmd.Restore
   ' Указывает в свойстве "Источник записей" таблицу "Клиенты".
   frm.RecordSource = "Клиенты"
End Sub
```

Функция CreateGroupLevel

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctCreateGroupLevelC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctCreateGroupLevelX":1}

Функция **CreateGroupLevel** задает поле или выражение, по которому выполняется группировка или сортировка данных в отчете.

Предположим, что требуется разработать специальную программу мастера, предоставляющего пользователю при разработке отчета возможность выбрать поля, по которым производится группировка данных в отчете. В этом случае следует вызвать в программе мастера функцию **CreateGroupLevel** для создания групп в соответствие с выбором пользователя.

Синтаксис

CreateGroupLevel(*отчет***,** *выражение***,** *заголовок***,** *примечание***)**

Функция **CreateGroupLevel** использует следующие аргументы.

Аргумент	**Описание**
отчет	Строковое выражение, задающее имя отчета, в котором определяется новый уровень группировки.
выражение	Строковое выражение, определяющее поле или выражение, по которому выполняется группировка или сортировка данных.
заголовок, *примечание*	Значение типа **Integer,** указывающее, будет ли группа, определяемая полем или выражением, иметь заголовок и/или примечание группы. Если в аргументе *заголовок* и/или *примечание* задано значение **True** (-1), то группа, определяемая полем или выражением, будет иметь присоединенную область заголовка и/или примечаний. При значении **False** (0) аргумента *заголовок* и/или *примечание* группа не имеет соответствующих присоединенных областей. Допускается одновременное создание заголовка и примечаний. При этом оба аргумента должны принимать значение **True**.

Дополнительные сведения

Функцию **CreateGroupLevel** используют при разработке программы мастера, создающего отчет с областями групп или итоговыми полями. Эта функция задает группировку или сортировку данных по указанному полю или выражению и создание областей заголовка и примечаний для данной группы.

Функция **CreateGroupLevel** доступна только в режиме конструктора отчета.

В Microsoft Access создаваемые в отчете группы регистрируются в массиве свойства **Уровень группировки (GroupLevel).** При вызове функции **CreateGroupLevel** в массив добавляется новый уровень группировки, определяемый аргументом *выражение*. Функция **CreateGroupLevel** возвращает значение, являющееся указателем на позицию нового уровня группировки в массиве. Первому полю или выражению, определяющему группировку, присваивается уровень 0, второму - уровень 1 и т.д. В отчетах поддерживается до десяти уровней группировки (0-9).

Если указать в аргументе *заголовок* и/или *примечание* значения **True**, то свойства **Заголовок группы (GroupHeader)** и/или **Примечание группы (GroupFooter)** получают значение «Да», и в отчете для группы данного уровня создаются области заголовка и/или примечаний.

После создания областей заголовка и/или примечаний пользователь имеет возможность определить другие свойства для данного уровня группировки: **Группировка (GroupOn),**

Интервал (GroupInterval) и **Не разрывать (KeepTogether).** Значения этих свойств могут быть заданы в программе Visual Basic или интерактивно в окне **Сортировка и группировка**, которое выводится при нажатии кнопки **Сортировка и группировка** ... на панели инструментов **Конструктор отчетов.**

Примечание. Мастер, создающий новый уровень группировки в новом или существующем отчете, должен открыть этот отчет в режиме конструктора.

Функция CreateGroupLevel, пример

В данном примере задается группировка по полю «ДатаРазмещения» в отчете «ОтчетЗаказы». Отчет должен быть открыт в режиме конструктора. Поскольку аргументы *заголовок* и *примечание* получают значение **True** (–1), отчет создается с заголовком и примечанием. Далее в программе определяются размеры этих областей.

```
Sub CreateGL()
   Dim varGroupLevel As Variant

   ' Задает группировку по полю "ДатаРазмещения".
   varGroupLevel = CreateGroupLevel("ОтчетЗаказы", "ДатаРазмещения", _
      True, True)
   ' Задает высоту областей заголовка и примечаний.
   Reports!ОтчетЗаказы.Section(acGroupLevel1Header).Height = 400
   Reports!ОтчетЗаказы.Section(acGroupLevel1Footer).Height = 400
End Sub
```

Инструкции DeleteControl, DeleteReportControl

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acstmDeleteControlC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acstmDeleteControlX":1}

- Инструкция **DeleteControl** позволяет удалить указанный элемент управления из формы.
- Инструкция **DeleteReportControl** позволяет удалить указанный элемент управления из отчета.

Предположим, что имеется процедура, которую требуется запускать при первом подключении каждого пользователя к базе данных. В этом случае удобно создать в форме кнопку и указать для нее в свойстве **Нажатие кнопки (OnClick)** имя нужной процедуры. После того как пользователь подключился к базе данных и запустил процедуру, ненужную кнопку можно динамически удалить из формы с помощью инструкции **DeleteControl**.

Синтаксис

DeleteControl *имяФормы*, *имяЭлемента*
DeleteReportControl *имяОтчета*, *имяЭлемента*

Инструкции **DeleteControl** и **DeleteReportControl** используют следующие аргументы.

Аргумент	Описание
имяФормы, *имяОтчета*	Строковое выражение, определяющее имя формы или отчета, из которых удаляется элемент управления.
имяЭлемента	Строковое выражение, определяющее имя удаляемого элемента управления.

Дополнительные сведения

Инструкции **DeleteControl** и **DeleteReportControl** доступны только в режиме конструктора формы и в режиме конструктора отчета, соответственно.

Примечание. Мастер, удаляющий элемент управления из формы или отчета, должен предварительно открыть форму или отчет в режиме конструктора.

Инструкции DeleteControl, DeleteReportControl, пример

В данном примере создается форма, содержащая кнопку, и выводится сообщение, предлагающее пользователя удалить эту кнопку. При нажатии кнопки **Да** в диалоговом окне из формы удаляется кнопка.

```
Sub DeleteCommandButton()
   Dim frm As Form, ctlNew As Control
   Dim strMsg As String, intResponse As Integer, intDialog As Integer

   ' Создает новую форму и получает указатель на нее.
   Set frm = CreateForm
   ' Создает новую кнопку.
   Set ctlNew = CreateControl(frm.Name, acCommandButton)
   ' Восстанавливает окно формы.
   DoCmd.Restore
   ' Задает текст подписи.
   ctlNew.Caption = "Новая кнопка"
   ' Задает размер элемента управления.
   ctlNew.SizeToFit
   ' Предлагает удалить кнопку.
   strMsg = "Удаляем кнопку " & ctlNew.Name &". Продолжить?"
   ' Задает кнопки, которые выводятся в диалоговом окне.
   intDialog = vbYesNo + vbCritical + vbDefaultButton2
   intResponse = MsgBox(strMsg, intDialog)
   If intResponse = vbYes Then
      ' Удаляет элемент управления.
      DeleteControl frm.Name, ctlNew.Name
   End If
End Sub
```

Функция SysCmd

{ewc HLP95EN.DLL,DYNALINK,"nïSnïS. nïSnïSnïSnïSnïS":"acfctSysCmdC"} {ewc HLP95EN.DLL,DYNALINK,"nïSnïSnïSnïSnïS":"acfctSysCmdX":1}

Функция **SysCmd** используется для выполнения одного из следующих действий:

- вывод в строке состояния индикатора выполнения или указанного текста;
- возвращение сведений о Microsoft Access и файлах приложения;
- возвращение сведений о состоянии объекта базы данных, показывающих, открыт ли данный объект, является ли объект новым, или был ли объект изменен, но не сохранен.

Например, в специальной программе мастера функция **SysCmd** позволяет вывести индикатор выполнения, демонстрирующий успешное выполнение программы при создании мастером формы.

Синтаксис

ReturnValue **= SysCmd(***действие***[, *текст*][, *значение*])**
ObjectState **= SysCmd(***действие***[, *типОбъекта*][, *имяОбъекта*])**

Функция **SysCmd** использует следующие аргументы.

Аргумент	Описание	
действие	Одна из перечисленных ниже встроенных констант, определяющих тип выполняемого действия.	
	Следующий набор констант определен для индикатора выполнения. При успешном выполнении данных действий функция **SysCmd** возвращает значение **Null**. В противном случае Microsoft Access генерирует ошибку выполнения.	
	acSysCmdInitMeter	Инициализирует индикатор выполнения. При использовании данной константы действия пользователь должен задать значения аргументов *текст* и *значение*.
	acSysCmdUpdateMeter	Обновляет индикатор выполнения с помощью указанного значения. При использовании данной константы действия необходимо задать значение аргументов *текст* и *значение*.
	acSysCmdRemoveMeter	Удаляет индикатор выполнения.
	acSysCmdSetStatus	Выводит *текст* в строке состояния.
	acSysCmdClearStatus	Сбрасывает текст в строке состояния.
	Следующий набор констант возвращает сведения о Microsoft Access.	
	acSysCmdRuntime	Возвращает значение **True** (–1), если запущена версия

		Microsoft Access, предназначенная только для выполнения.
	acSysCmdAccessVer	Возвращает номер версии Microsoft Access.
	acSysCmdIniFile	Возвращает имя файла .ini, используемого Microsoft Access.
	acSysCmdAccessDir	Возвращает имя каталога, в котором хранится файл Msaccess.exe.
	acSysCmdProfile	Возвращает значение параметра **/profile**, указанного при запуске Microsoft Access с командной строки.
	acSysCmdGetWorkgroupFile	Возвращает путь к файлу рабочей группы (System.mdw).
	Следующая константа возвращает сведения о состоянии объекта базы данных.	
	acSysCmdGetObjectState	Возвращает сведения о состоянии объекта базы данных. При использовании данной константы действия необходимо задать значение аргументов *типОбъекта* и *имяОбъекта*.

текст	Строковое выражение, определяющее текст, выводящийся в строке состояния с выравниванием по левому краю. Данный аргумент требуется указать, когда в аргументе *действие* указаны константы **acSysCmdInitMeter**, **acSysCmdUpdateMeter** или **acSysCmdSetStatus**; неприменим при других значениях аргумента *действие*.
значение	Числовое выражение, определяющее состояние индикатора выполнения. Данный аргумент требуется указать, когда в аргументе *действие* указана константа **acSysCmdInitMeter**; неприменим при других значениях аргумента *действие*.
типОбъекта	Одна из следующих встроенных констант: **acTable** **acQuery** **acForm** **acReport** **acMacro** **acModule** Данный аргумент требуется указать, когда в аргументе *действие* указана константа **acSysCmdGetObjectState**; неприменим при других значениях аргумента *действие*.
имяОбъекта	Строковое выражение, являющееся допустимым именем

объекта базы данных для типа, указанного в аргументе *типОбъекта*. Данный аргумент требуется указать, когда в аргументе *действие* указана константа **acSysCmdGetObjectState**; неприменим при других значениях аргумента *действие*.

Дополнительные сведения

Вызов функции **SysCmd** с константами индикатора выполнения позволяет вывести в строке состояния индикатор выполнения для операции известной длительности или требующей известного числа шагов и обновлять состояние индикатора в соответствие с ходом выполняемой операции.

Первым шагом при выводе индикатора выполнения в строке состояния является вызов функции **SysCmd** с константой **acSysCmdInitMeter** в аргументе *действие* и заданными значениями аргументов *текст* и *значение*. При вызове с константой **acSysCmdInitMeter** аргумент *значение* определяет максимальное значение индикатора, т.е. 100 процентов.

Для того чтобы обновить состояние индикатора в соответствие с ходом выполняемой операции, следует вызвать **SysCmd** с константой **acSysCmdUpdateMeter** в аргументе *действие* и заданным значениям аргумента *значение*. При вызове с константой **acSysCmdUpdateMeter** аргумент *значение* определяет текущее значение индикатора. Например, если было указано максимальное значение 200, то при обновлении индикатора значением 100 индикатор будет выведен заполненным наполовину.

Для изменения текста, выводящегося в строке состояния, следует вызвать **SysCmd** с константой **acSysCmdSetStatus** в аргументе *действие* и новым текстом сообщения в аргументе *текст*. Например, при выполнении операции сортировки удобно вывести в строке состояния сообщение «Сортировка...», а после завершения сортировки удалить это сообщение. Максимальная длина строки в аргументе *текст* составляет примерно 80 символов. Поскольку текст в строке состояния выводится с помощью пропорционального шрифта, реальная длина сообщения определяется конкретным содержимым строки *текст*.

Увеличение ширины текста сообщения приводит к уменьшению ширины индикатора выполнения. Если сообщение, размер которого превышает ширину строки состояния, задается при константе **acSysCmdInitMeter** в аргументе *действие* , то функция **SysCmd** игнорирует сообщение и не выводит никакого текста в строке состояния. Если же такой текст задается при константе **acSysCmdSetStatus** в аргументе *действие*, то **SysCmd** выводит сообщение, но обрезает текст по размеру строки состояния.

Не допускается указание в аргументе *текст* пустой строки (""). Для того чтобы удалить существующее сообщение, выведенное в строке состояния, следует задать в аргументе *текст* единственный пробел. Следующие инструкции иллюстрируют удаление текста из строки состояния:

```
varReturn = SysCmd(acSysCmdInitMeter, " ", 100)
varReturn = SysCmd(acSysCmdSetStatus, " ")
```

Если в строке состояния выведен индикатор выполнения, то указание текста сообщения путем вызова функции **SysCmd** с константой **acSysCmdSetStatus** в аргументе *действие* приводит к автоматическому удалению индикатора.

Вызов функции **SysCmd** с другими константами действия позволяет получить системную информацию о Microsoft Access, в том числе номер исполняемой версии Microsoft Access, является ли данная версия только исполняемой, адрес каталога исполняемого файла Microsoft Access, а также имя файла инициализации Microsoft Access.

Примечание. Как общие, так и специализированные настройки Microsoft Access сохраняются в реестре Windows, поэтому при разработке приложения использование файла .ini является излишним. Константа **acSysCmdIniFile** определена для совместимости с предыдущими версиями Microsoft Access.

Чтобы получить сведения о состоянии конкретного объекта базы данных, следует вызвать функцию **SysCmd** с константой **acSysCmdGetObjectState** в аргументе *действие* и заданными аргументами *типОбъекта* и *имяОбъекта*. Различаются четыре состояния объекта: не открыт или не существует, открытый, новый, а также измененный, но не сохраненный.

Например, при разработке программы мастера, вставляющего новое поле в таблицу, требуется определить, была ли структура таблицы изменена, но еще не сохранена, чтобы сохранить ее перед новым изменением. Для этого следует проверить значение, возвращаемое функцией **SysCmd**.

Функция **SysCmd** с константой **acSysCmdGetObjectState** в аргументе *действие* может возвращать любую комбинацию следующих констант.

Константа	Состояние объекта базы данных
acObjStateOpen	Открыт
acObjStateNew	Новый
acObjStateDirty	Изменен, но не сохранен

Примечание. Если объект, указанный в аргументе *имяОбъекта* не открыт или не существует, функция **SysCmd** возвращает нуль.

Функция SysCmd, примеры

В данном примере создается объект **Recordset** типа статического набора записей, считывается каждая запись и выводится индикатор выполнения, показывающий текущую позицию в наборе записей.

```
Function ReadRecords(strTableName As String) As Integer
   Const conBadArgs = -1
   Dim dbs As Database, rst As Recordset
   Dim lngCount As Long, strMsg As String
   Dim varReturn As Variant, lngX As Long

   ReadRecords = 0
   If strTableName <> "" Then
      DoCmd.Hourglass True
      Set dbs = CurrentDb
      On Error Resume Next
      Set rst = dbs.OpenRecordSet(strTableName)
      ' Определяет число записей.
      rst.MoveLast
      rst.MoveFirst
      If Err Then
         ReadRecords = conBadArgs
      End If
      lngCount = rst.RecordCount
      On Error GoTo 0
      If lngCount Then
         strMsg = "Чтение " & UCase$(strTableName) & "..."
         varReturn = SysCmd(acSysCmdInitMeter, strMsg, lngCount)
         ' Выводит сообщение в строке состояния.
         For lngX = 1 To lngCount
            varReturn = SysCmd(acSysCmdUpdateMeter, lngX)
         ' Обновляет индикатор.
            .                              ' Обработка записи.
            .
            .
            rst.MoveNext                   ' Переход на следующую запись.
         Next lngX
         varReturn = SysCmd(acSysCmdClearStatus)
         GoSub CloseObjects
         ReadRecords = lngCount            ' Возвращает число записей.
         Exit Function
      End If
   End If
   ' Таблица не найдена или не содержит записей.
   strMsg = "Таблица '" & strTableName & "'не найдена или не содержит
записей.'"
   MsgBox strMsg, vbInformation, "ReadRecords"
   GoSub CloseObjects
   Exit Function
CloseObjects:
   On Error Resume Next
   rst.Close
   dbs.Close
   On Error GoTo 0
   DoCmd.Hourglass False
   Return
```

```
End Function
```

В следующем примере демонстрируется использование функции **SysCmd** с константой **acSysCmdGetObjectState** для определения состояния активного объекта базы данных.

```
Dim intObjType As Integer, strObjName As String, intObjState As Integer

intObjType = Application.CurrentObjectType
strObjName = Application.CurrentObjectName
intObjState = SysCmd(acSysCmdGetObjectState, intObjType, strObjName)
If intObjState <> 0 Then
    .
    .
    .
End If
```