Примеры для Microsof

Microsoft Excel '97 обладает новой средой для разработки макросов Visual Basic Integrated Development Environment (IDE). Если раннее вам приходилось заниматься разработкой макросов, то вы обратите внимание, что в новой среде разработчика модули макросов отображаются по-новому и выглядят ни как листы книги.

Эта книга содержит листы с объяснением процедур, которые используются для автоматизации типичных для Microsoft Excel задач. С каждого листа можно перейти в среду разработки макросов IDE для просмотра текста программ. Процедуры следует скопировать в модули и изменить, если это необходимо.

Для просмотра примеров выберите соответствующее название справа от этой надписи.

Функции листа Functions

Работа с массивами

Повтор действий

Размещение подписей на диаграмме

t® Excel

Доступ к данным

Программирование объектов в Office

Програмирование событий

явных или подразумеваемых, в том числе
тексты процедур Visual Basic передаются «как
енеры службы поддержки Microsoft («Microsoft
стоянии ни проводить работы по их изменению
ю таких изменений. Если ваш опыт создания
аммного обеспечения или поставкой готовых

Примеры работы с функциями листа

Данный лист содержит примеры формул, которые используются при решении типичных задач. Ячейки, содержащие формулы, окрашены голубым цветом. Чтобы увидеть формулу, установите курсор над ячейкой, появится примечание к данной ячейке. Для перехода из режима просмотра формулы в режим просмотра значения и обратно используйте клавиши CTRL+`. Если необходимо получить дополнительные сведения о функции листа, выберите ячейку, содержащую функцию, а затем нажмите кнопку "Изменить формулу" (=) на стандартной панели инструментов.

Отключение вывода значения ошибки

Обычно, если в формуле листа используется недопустимое значение, то выводится значение ошибки (#ДЕЛ/0!, #Н/Д, #ЗНАЧ!,#ССЫЛКА! или #ЧИСЛО!). Например, если в формуле деления в знаменателе стоит ссылка на ячейку, содержащую ноль, то будет выведено значение ошибки #ДЕЛ/0!.

Операнд A	Операнд B	Исходная формула	Новая формула
25	0	#DIV/0!	

Обычно для обработки ошибок в формулах используется следующая конструкция:

=ЕСЛИ(ЕОШИБКА(<формула>),"",<формула>)

где <формула> — формула, для которой необходимо отключить вывод значения ошибки. Если при вычислении формулы возникает ошибка, то формула возвращает "" (нулевое значение), в другом случае возвращается результат вычисления формулы.

Другой способ отмены вывода значения ошибки основан на новом средстве условного форматирования Excel '97. Оно позволяет отформатировать отображение ячейки по-другому в зависимости от содержимого этой ячейки. Для значения ошибки проделайте следующие шаги:

1. Выберите ячейку для форматирования (в нашем случае G19).
2. В меню **Формат** выберите команду **Условное форматирование**.
3. В первом поле со списком выберите значение "формула".
4. В следующем поле введите: =ЕОШИБКА(G19).
5. Нажмите кнопку "Формат" и установите нужный формат, в нашем случае используйте для шрифта синий цвет такой же, как и цвет заливки.
6. Нажмите кнопку "ОК".

В результате значение ошибки не будет отображаться в ячейке, так как цвет шрифта для значения ошибки совпадает с цветом фона данной ячейки.

Операнд A	Операнд B	Исходный формат	Новый формат
25	0	#DIV/0!	

Индексирование, поиск вхождений и подстановка знач

Одной из типичных задач обработки списков является поиск значения на пересечении столбца и строки в прямоугольной области из ячеек листа. Мастер подстановок помогает написать формулы для выполнения этой задачи.

Чтобы воспользоваться этим средством, выберите ячейку внутри нужного диапазона данных, затем в меню **Сервис** выберите команду **Мастер**, а затем — **Поиск...**, далее следуйте указаниям мастера.

Чтобы получить более подробные сведения, используйте поиск по ключевым словам "мастер подстановок" на вкладке **Предметный указатель** справочной системы.

Формулы условного суммирования

Другой типичной для Excel задачей является вычисление суммы значений при определенных условиях. Мастер суммирования помогает написать формулы, которые выполняют эту задачу.

Чтобы воспользоваться этим средством, выберите ячейку внутри списка, который следует просуммировать, затем в меню **Сервис** выберите команду **Мастер**, а затем — **Частичная сумма**, далее следуйте указаниям мастера.

Чтобы получить более подробные сведения, используйте поиск по ключевым словам "мастер суммирования" на вкладке **Предметный указатель** справочной системы.

Работа с массивами

КРАТКОЕ ОПИСАНИЕ

Самым простым способом переноса содержимого массива на лист является использования цикла, например For...Next. Этот цикл предоставляет возможность одновременного индексирования внутри массива и переноса элементов массива по их адресам назначения. Тот же результат может быть получен и без применения цикла, используя в VBA свойство FormulaArray для объекта типа Range.

В VBA одномерный массив имеет вид строки. Следовательно, если диапазон ячеек листа, в который заносится содержимое массива, является так же строкой, то для передачи массива достаточно одного оператора FormulaArray. Например:

```
Sub ArrayDump1()            'Для одномерного массива
 Dim x(1 To 10) As Double    'Описание массива из 10 элементов
 For j = 1 To 10              'Вычисление значений массива
   x(j) = j * j
 Next j
 'Перенос содержимого массива в строку листа
 Range(Cells(2, 1), Cells(2, 10)).FormulaArray = x
End Sub
```

Пример и
Array

Приведенный выше пример хорошо работает только тогда, когда назначенные ячейки листа являются одной строкой. Когда же содержимое массива нужно перенести в ячейки, расположенные в столбце листа, этот пример работает неправильно.

В случае многостроковых данных, представленных двумерным массивом, необходимо изменить ориентацию массива. Чтобы сделать это, необходимо описать массив как двумерный, состоящий из нескольких строк и одного столбца. Например:

Пример и
Array

```
Sub arraydump2()     'Для двумерного массива в виде столбца
  'Описание массива расположенного в одном столбце
  Dim x(1 To 10, 1 To 1) As Double
  'Вычисление значений массива
  For j = 1 To 10
     x(j, 1) = j * j
  Next j
  'Перенос содержимого массива в столбец листа
  Range(Cells(1, 2), Cells(10, 2)).FormulaArray = x
End Sub
```

Двумерный массив, приведенный в этом примере, позволяет представить данные массива в виде столбца, следовательно, массив можно перенести на лист без применения цикла.

программы
Dump1

программы
Dump2

Повтор действий

Часто приходится выполнять одну операцию над группой элементов. Этими элементами могут быть ячейки, листы в книге или книги в приложении.

Чтобы автоматизировать повторяющиеся действия, следует записать макрос. Хотя при записи макроса невозможно записать циклы, используйте эту процедуру для создания исходного текста модуля VBE. Затем измените полученную таким образом программу, применяя различные циклы в зависимости от выполняемой задачи.

В приведенных ниже примерах в ячейках столбца A содержатся числа и в зависимости от числа, находящегося в ячейке, изменяется цвет соответствующей ячеки столбца B.

Сначала следует записать по шагам процесс изменения цвета заливки для ячейки. Чтобы записать макрос, выберите команду **Макрос** в меню **Сервис**, а затем — команду **Начать запись**.

Пока идет запись макроса выберите команду **Ячейки** в меню **Формат**, а затем на вкладке **Вид** выберите цвет заливки ячейки. В этом примере используется желтый цвет (.ColorIndex=6). Прекратите запись макроса, нажав кнопку **Остановить запись** на панели инструментов **Остановка записи**.
Текущая выбранная ячейка изменит цвет и будет записан следующий макрос:

```
Sub Recorded_Macro()
'
' Макрос записан 30.06.97
'
    With Selection.Interior
        .ColorIndex = 6
        .Pattern = xlSolid
    End With
End Sub
```

Пример программы Recorded_Macro

Примечание: если в процессе записи макроса были выбраны ячейки, макрос может содержать строку похожую на следующую запись `Range("A3").Select`. Удалите эту строку так как она указывает на выбранную ячейку при каждом запуске макроса. Если необходимо, чтобы эта ячейка была выбрана первой, сохраните эту строку в макросе.

Теперь можно незначительно изменить программу и добавить в нее одну из циклических структур.

Цикл For Each...Next

Если известен диапазон ячеек, для которого необходимо применить записанный макрос, следует использовать цикл **For Each…Next**.

В этом примере цвет в ячейке столбца B изменится, если ячейка столбца A содержит число больше 20. Чтобы сделать это, следует добавить в начало макроса оператор **If**, а в конец — оператор **With**. Цвет ячейки изменится только при выполнении условия в операторе **If**.

Наконец, так как нужно изменить соответствующую ячейку одного столбца, находящегося справа от столбца A (столбец B), следует изменить свойство **Selection** в записанной программе у метода **Offset** для объекта циклическая ячейка (cell_in_loop).
В результате программа должна выглядеть следующим образом:

```
Sub For_Each_Next_Sample()
'
' Макрос записан 30.06.97
'
For Each cell_in_loop In Range("A1:A5")
  If cell_in_loop.Value > 20 Then
    With cell_in_loop.Offset(0, 1).Interior
        .ColorIndex = 6
        .Pattern = xlSolid
    End With
  End If
Next
End Sub
```

Пример программы For Each..Next

Цикл For…Next

Если известно количество прогона программы, следует воспользоваться циклом **For..Next**. Используйте наш пример, если необходимо проверить только 10 ячеек вниз, начиная с выбранной ячейки. После изменения программа примет следующий вид:

```
Sub For_Next_Sample()

For Counter = 0 To 9
  If Selection.Offset(Counter, 0).Value > 20 Then
    With Selection.Offset(Counter, 1).Interior
        .ColorIndex = 6
        .Pattern = xlSolid
    End With
  End If
Next
End Sub
```

Пример программы For..Next

Здесь объект **Selection** использован для того, чтобы программа просматривала 10 ячеек в столбце, начиная с активной ячейки. Переменная **Counter** возрастает при каждом прохождении через цикл и может использоваться внутри циклической структуры. Здесь она использована как аргумент **Offset**, для определения смещения текущей ячейки от начальной ячейки. Так, если программа была запущена при активной ячейке A1, то при первом вхождении в цикл переменная **Counter** равна 0, это означает, что смещение от ячейки A1 так же равно 0. Это и записано в следующем выражении Selection.Offset(Counter, 0).Value.

Цикл Do…

Если необходимо остановить цикл при определенном условии, следует использовать цикл **Do…**. Эта циклическая структура позволяет проверить свойства или условия до вхождения в цикл. В приведенном ниже примере, разрешается вход в цикл до тех пор, пока номер строки ячеек `Selection.Offset(Counter, 0).Row` не превысит 100. Это может быть полезно, если необходимо остановить выполнение цикла ниже сотой строки.

```
Sub Do_Loop_Sample()

Counter = 0
Do Until Selection.Offset(Counter, 0).Row > 100
  If Selection.Offset(Counter, 0).Value > 20 Then
    With Selection.Offset(Counter, 1).Interior
        .ColorIndex = 6
        .Pattern = xlSolid
    End With
  End If
  Counter = Counter + 1
Loop
```

Пример программы Do…Loop

Примечание: существует более трех видов цикла **Do…**. Они обеспечивают большую гибкость в применении этого цикла.

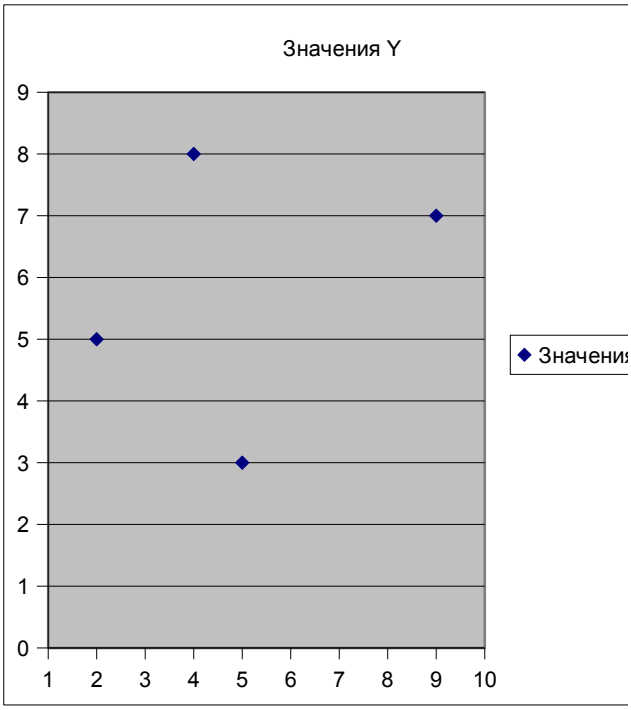
Чтобы получить более подробные сведения об этих и других циклических структурах, в справочной системе по VBE следует осуществить поиск по ключевому слову "цикл".

Размещение подписей на декартовой диаграмме

Подписи	Значения X	Значения Y
Данные1	2	5
Данные2	9	7
Данные3	5	3
Данные4	4	8
Данные5	1	4

Разместить подписи на диаграмме

Вернуть начальный вид



Microsoft Excel не размещает подписи на декартовой диаграмме автоматически. Чтобы это сделать, следует использовать макрос.

Этот макрос демонстрирует автоматическое размещение подписей на декартовой диаграмме. Предполагается, что данные для диаграммы и связанные с ними подписи располагаются на одном листе и имеют туже форму представления, что и данные в окрашенных ячейках, расположенных выше.

Чтобы расположить подписи на диаграмме, нажмите кнопку "Разместить подписи на диаграмме".

Чтобы убрать подписи с диаграммы, нажмите кнопку "Вернуть начальный вид".

Для просмотра текста программы нажмите кнопку "Текст программы".

Текст программы

R Y

Доступ к данным

КРАТКОЕ ОПИСАНИЕ
Объекты доступа к данным (DAO) предоставляют возможность непосредственной работы с базой данных. Чтобы использовать DAO, следует создать ссылку в VBE на объектную библиотеку Microsoft DAO 3.5 Object Library (выберите в меню **Сервис** команду **Ссылки**, а затем в появившемся диалоговом окне выберите соответствующее название библиотеки). Небходимо так же установить соответствующие драйверы ISAM и ODBC. Более подробные сведения по установке ODBC драйверов находятся в разделе "Установка ODBC драйвера" справочной системы. Сведния об установке DAO или ISAM находятся в разделе "Установка и удаление отдельных компонентов Microsoft Office или Microsoft Excel" справочной системы. Пути, используемые в примерах, указывают на каталоги, в которые устанавливаются компоненты Microsoft Office, при установке по умолчанию. Если Microsoft Office был установлен в другое место, следует изменить пути в VBE до запуска примеров программ.
ПРИМЕР RDO

В этом примере данные извлекаются из базы данных (с помощью RDO) и размещаются в таблице запроса на новом листе Excel. Этот метод следует использовать при подключении к базе данных SQL. RDO не использует объекты доступа к данным, которые уменьшают объем свободной памяти.

```
Sub RDOExample()
Dim ws As Workspace
Dim rs As Recordset
Dim qt As QueryTable
Dim c As Connection
Dim ConnectStr As String
Dim NewSheet As Object
    'Если Microsoft Office не был установлен в каталог, используемый при установке
    'по умолчанию, следует изменить соответствующий каталог до запуска программы
    ConnectStr = "odbc;Driver={Microsoft dBase Driver (*.dbf)};DBQ=c:\program files\
microsoft office\office;"
    'Создание рабочей области, которая позволяет обойти DAO
    Set ws = CreateWorkspace("w1", "admin", "", dbUseODBC)
    'Открытие подключения к папке, которая содержит файлы DBF
    Set c = ws.OpenConnection("", 1, 0, ConnectStr)
    'Извлечь все записи таблицы и установить передачу
    'данных только записями (ускорение передачи данных)


    Set rs = c.OpenRecordset("select * from orders", dbOpenForwardOnly)
    'Создание нового листа в активной книге
    Set NewSheet = Worksheets.Add
    'Расположить новую таблицу запроса, содержащую набор записей, на новом листе
    Set qt = NewSheet.QueryTables.Add(rs, Range("a1"))
    'Синхронное обновление данных в таблице запроса
    qt.Refresh False
    'Закрытие набора записей
    rs.Close
    'Закрытие подключения
    c.Close
End Sub
```

ВОССТАНОВЛЕНИЕ НАБОРА ЗАПИСЕЙ DAO

В этом примере используется база данных "Борей.mdb" из Microsoft Access для создания таблицы запроса на новом листе, затем восстанавливается набор записей в другой таблице базы данных.

```
Sub ResettingDAORecordset()
Dim db As Database
Dim rs As Recordset
Dim qt As QueryTable
Dim NewSheet As Object
Dim nwind As String
    'Используемый по умолчанию при установке, путь к базе данных Борей.mdb
    nwind = "C:\Program files\microsoft office\office\samples\Борей.mdb"
    'Открытие базы данных Борей.mdb
    Set db = DBEngine.Workspaces(0).OpenDatabase(nwind)
    'Открытие набора записей, содержащего все записи из таблицы "Клиенты"
    Set rs = db.OpenRecordset("Клиенты")
    'Создание нового листа в активной книге
    Set NewSheet = Worksheets.Add
    'Разместить новую таблицу запроса, содержащую набор записей, на новом листе
    Set qt = NewSheet.QueryTables.Add(rs, Range("a1"))




    'Синхронное обновление данных в таблице запроса
    qt.Refresh False
    'Закрытие набора записей
    rs.Close
    'Открытие набора записей, содержащего все записи из таблицы "Заказы"
    Set rs = db.OpenRecordset("Заказы")
    'Изменение набора записей для сущуествующей таблицы запроса
    Set qt.Recordset = rs
    'Синхронное обновление таблицы запроса для просмотра данных на экране
    qt.Refresh False
    'Закрытие набора записей
    rs.Close
    'Закрытие базы данных
    db.Close
End Sub
```

ПЕРЕДАЧА ДАННЫХ ИЗ MICROSOFT ACCESS

В этом примере используется DAO для создания запроса по двум таблицам из базы данных "Борей.mdb". Если запрос уже существует, то программа обработки ошибок удалит его.

```
Sub RetrieveAccessData()
Dim Nsql As String, Ncriteria As String, Njoin As String
Dim nwind As String
Dim h As Integer
Dim db As Database, qry As Object
Dim rec As Recordset, NewSheet As Object
    'Используемый по умолчанию при установке, путь к базе данных Борей.mdb
    nwind = "C:\Program files\microsoft office\office\samples\Борей.mdb"
    'При обнаружении ошибки перейти к метке errorhandler:
    On Error GoTo errorhandler
    'Открытие базы данных
    Set db = DBEngine.Workspaces(0).OpenDatabase(nwind)
    'Предложения SQL для запроса (скопированы из окна MS Query's SQL)
    Nsql = "SELECT DISTINCTROW Типы.Категория, Товары.Марка, Товары.ЕдиницаИзмерения,
Товары.Цена "



    Njoin = "FROM Типы INNER JOIN Товары ON Типы.КодТипа = Товары.КодТипа "
    Ncriteria = "WHERE ((([Товары].ПоставкиПрекращены)=No) AND (([Товары].НаСкладе)>20));"
    'Создание запроса с именем TempQuery
    Set qry = db.CreateQueryDef("TempQuery")
    'Запоминание предложения SQL в переменной TempQuery
    qry.sql = Nsql & Njoin & Ncriteria
    'Открытие набора записей, созданного запросом TempQuery
    Set rec = qry.OpenRecordset()
    'Создание новой страницы в книге, содержащей эту программу
    Set NewSheet = ThisWorkbook.Sheets.Add(after:=Worksheets("Доступ к данным"),
Type:=xlWorksheet)
    'Поместить имена полей в первую строку нового листа
    For h = 0 To rec.Fields.Count - 1
        NewSheet.[a1].Offset(0, h).Value = rec.Fields(h).Name
    Next h
```

```
    'Копирование динамического набора в Excel
    NewSheet.[a2].CopyFromRecordset rec
    'Удаление временного запроса
    db.QueryDefs.Delete "TempQuery"
    'Закрытие базы данных
    db.Close
'Выход из подпрограммы, если ошибки не обнаружены
Exit Sub
'Удаление подпрограммой обработки ошибок существующего запроса
errorhandler:
    If DBEngine.Errors(0).Number = 3012 Then
        db.QueryDefs.Delete "TempQuery"
        'Продолжить выполнение программы со строки, в которой обнаружена ошибка
        Resume
    Else
        MsgBox Error(Err)
    End If
End Sub
```

ИЗВЛЕЧЕНИЕ ДАННЫХ С ПОМОЩЬЮ ДРАЙВЕРОВ ISAM

В этом примере данные извлекаются из файла базы данных и размещаются на новом листе. Необходимо, чтобы соответствующие ISAM драйверы были установлены.

```
Sub RetrieveISAMdata()
Dim Db As database, rec As Recordset
Dim dPath As String
Dim h As Integer, NewSheet As Object
    'Путь к каталогу с файлами dbf, используемый при стандартной установке
    dPath = "C:\Program files\microsoft office\office"
    'Открытие базы данных
    Set Db = DBEngine.Workspaces(0).OpenDatabase(dPath, 0, 0, "dBase III")


    'Открытие набора записей со всеми записями из orders.dbf
    Set rec = Db.OpenRecordset("SELECT * FROM orders")
    'Вставка нового листа в книгу, содержащую эту программу
    Set NewSheet = ThisWorkbook.worksheets.Add(after:=Worksheets("Доступ к данным"))
    'Цикл по полям и размещение имен полей на листе
    For h = 0 To rec.Fields.Count - 1
        NewSheet.[a1].Offset(0, h).Value = rec.Fields(h).Name
    Next h
    'Копирование динамического набора в Excel
    NewSheet.[a2].CopyFromRecordset rec
    'Закрытие  базы данных
    Db.Close
End Sub
```

ПРОСМОТР ТАБЛИЦ В БАЗЕ ДАННЫХ

В этом примере выводятся имена таблиц из примера базы данных "Борей.mdb" Microsoft Access.
Примечание: для запуска этой программы необходимо установить Microsoft Access 97.

```
Sub ListTables()
Dim db As Database, TableCount As Long, i As Long
Dim dPath As String
    'Стандартное расположение файла Борей.mdb
    dPath = "C:\Program files\microsoft office\office\samples\Борей.mdb"
    'Открытие базы данных Борей.mdb
    Set db = DBEngine.Workspaces(0).OpenDatabase(dPath)
    'Установить значение переменной равное числу таблиц
    TableCount = db.TableDefs.Count
    'Цикл по всем таблицам
    For i = 0 To TableCount - 1
        'Вывод имен таблиц
        MsgBox db.TableDefs(i).Name
    Next
    'Закрытие базы данных
    db.Close
End Sub
```

ВЫВОД ПОЛЕЙ ИЗ БАЗЫ ДАННЫХ

В этом примере выводятся имена полей из таблицы "Customer". Если файл customer.dbf не установлен, прочтите раздел "Установка и удаление отдельных компонентов Microsoft Office или Microsoft Excel" из справочной системы.

```
Sub List_Fields()
Dim db As Database, rec As Recordset
Dim fieldcount As Long, i As Long, dPath As String
    'Стандартный каталог для примеров файлов dbf
    dPath = "C:\Program files\microsoft office\office"
    'Открытие базы данных
    Set db = OpenDatabase(dPath, 0, 0, "dBase III")
    'Открытие всех записей из customer.dbf
    Set rec = db.OpenRecordset("SELECT * FROM customer")
    'Подсчет числа полей
    fieldcount = rec.Fields.Count
    'Цикл по всем полям
    For i = 0 To fieldcount - 1
        'Вывод имен полей
        MsgBox rec.Fields(i).Name
    Next
    'Закрытие базы данных
    db.Close
End Sub
```

ПОДКЛЮЧЕНИЕ К БАЗЕ ДАННЫХ SQL

В этом примере выводятся имена полей из базы данных SQL. Необходимо иметь доступ к базе данных SQL , а так же — правильно созданный DSN. Чтобы получить более подробные сведения о DSN, выберите на панели управления Windows значок 32bit ODBC.

```
Sub ODBC_Connection()
Dim db As Database, i As Long
    'Показать диалоговое окно DSN, которое позволяет пользователю выбрать dsn,
    'а также получить более подробные сведения
    Set db = DBEngine.Workspaces(0).OpenDatabase("", , , "ODBC;")
    'Подсчет числа таблиц
    TableCount = db.TableDefs.Count
    'Цикл для вывода имен всех таблиц
    For i = 0 To TableCount - 1
        MsgBox db.TableDefs(i).Name
    Next
    'Закрытие базы данных
    db.Close
End Sub
```

ОБНАРУЖЕНИЕ ОШИБОК DAO

В следующем примере показывается, как найти ошибки DAO. Показанное на экране сообщение с описанием ошибки, часто является полезной иформацией для пользователя. Знание номера ошибки позволяет программисту найти и исправить характерные ошибки.

```
Sub Trap_DAO_Error()
Dim d As database, r As Recordset
    'При обнаружении ошибки перейти к метке errorhandler:
    On Error GoTo errorhandler
    'Не показывать встроенные сообщения об ошибках
    Application.DisplayAlerts = False
    'Попытка открыть несуществующую базу данных
    Set d = DBEngine.Workspaces(0).OpenDatabase("c:\xl95\db4.mdb")
    Exit Sub 'Выход из подпрограммы, если ошибки не обнаружены
errorhandler:
    MsgBox DBEngine.Errors(0).Description 'Текст сообщения об ошибке
    MsgBox DBEngine.Errors(0).Number 'Номер ошибки
    MsgBox DBEngine.Errors(0).Source 'Место ошибки
    MsgBox DBEngine.Errors(0).HelpContext
End Sub
```

СОЗДАНИЕ ТАБЛИЦЫ

В этом примере показано, как создать новую таблицу внутри существующей базы данных. Здесь используется база данных "Борей", устанавливаемая вместе с Microsoft Access 97.

```
Sub Create_Table()
Dim t As Object, f As Object, d As Database
Dim dPath As String
    'При обнаружении ошибки перейти к метке errorhandler:
    On Error GoTo errorhandler
    'Путь к базе данных Борей.mdb, используемый при установке по умолчанию
    dPath = "C:\Program files\microsoft office\office\samples\Борей.mdb"
    'Открытие базы данных Борей.mdb
    Set d = DBEngine.Workspaces(0).OpenDatabase(dPath)
    'Создание описания новой таблицы
    Set t = d.CreateTableDef("Новая_таблица")



    'Добавление полей в таблицу "Новая_таблица"
    Set f = t.CreateField("Поле1", dbDate)
        t.Fields.Append f 'Добавление поля к существующим полям
    Set f = t.CreateField("Поле2", dbText)
        t.Fields.Append f
    Set f = t.CreateField("Поле3", dbLong)
        t.Fields.Append f
    ' Сохранение описания таблицы добавлением ее набор TableDefs
    d.TableDefs.Append t
    'Закрытие базы данных
    d.Close
Exit Sub
errorhandler:
    MsgBox DBEngine.Errors(0)
End Sub
```

Пример программы RDOExample

Пример программы восстановления набора записей DAO ResettingDAORecordset

Пример программы передачи данных RetrieveAccessData

Пример программы извлечения данных с помощью драйверов ISAM (RetrieveISAMdata)

Пример программы вывода имен таблиц ListTables

Пример программы имен полей из таблицы List_Fields

Пример программы подключения к базе данных SQL ODBC_Connection

Пример программы обнаружения ошибок DAO Trap_DAO_Error

Пример программы создания таблицы Create_Table

Программирование объектов в Office

КРАТКОЕ ОПИСАНИЕ

Программирование объектов — особенность модели компонентов объектов (COM), технологии, в которой программы предоставляют доступ к своим объектам средствам разработки, макроязыкам и другим приложениям, которые поддерживают программирование. Например, приложение листа предоставляет доступ к листу, диаграмме, ячейке или к диапазону ячеек, каждые из которых принадлежат к разным типам объектов. Текстовый процессор предоставляет для использования такие объекты как приложение, документ, абзац, предложение, закладка или выделенный объект. В следующих примерах демонстрируется автоматизация задач, возникающих между Microsoft Excel и другими программами Microsoft.

Чтобы получить более подробные сведения о программировании объектов, обратитесь к справочной системе VBA.

MICROSOFT ACCESS

В этом примере выводится каталог, в котором находится Microsoft Access.

Пример программы MS_Access

```
Sub MS_Access()
Dim AccDir As String
Dim acc As Object
    'Связь OLE с Access
    Set acc = CreateObject("access.application")
    'Поиск пути для файла msaccess.exe
    AccDir = acc.SysCmd(Action:=acSysCmdAccessDir)
    'Вывод на экран найденного пути
    MsgBox "MSAccess.exe находится в каталоге " & AccDir
    'Освобождение памяти для переменных
    Set acc = Nothing
End Sub
```

MICROSOFT WORD

В этом примере копируется диаграмма с листа "Подписи на диаграмме" в новый документ Microsoft Word.

Пример программы MS_Word

```
Sub MS_Word()
Dim wd As Object
    'Создание сеанса работы с Microsoft Word
    Set wd = CreateObject("word.application")
    'Копирование диаграммы с листа "Подписи на диаграмме"
    Worksheets("Подписи на диаграмме").ChartObjects(1).Chart.ChartArea.Copy
    'Отображение документа на экране
    wd.Visible = True
    'Активизировать MS Word
    AppActivate wd.Name
    With wd
      'Открытие нового документа в Microsoft Word
      .Documents.Add
      'Вставка абзаца
      .Selection.TypeParagraph
      'Вставка диаграммы
      .Selection.PasteSpecial link:=True, DisplayAsIcon:=False, Placement:=wdInLine
    End With
    Set wd = Nothing
End Sub
```

MICROSOFT POWERPOINT

В этом примере копируется диаграмма с листа "Подписи на диаграмме" в новую презентацию Microsoft PowerP

Пример программы MS_PowerPoint

```
Sub MS_PowerPoint()
Dim ppt As Object, pres As Object
    'Создание сеанса работы с Microsoft PowerPoint
    Set ppt = CreateObject("powerpoint.application")
    'Копирование диаграммы с листа "Подписи на диграмме"
    Worksheets("Подписи на диаграмме").ChartObjects(1).Copy
    'Создание новой презентации в Microsoft PowerPoint
    Set pres = ppt.Presentations.Add
    'Добавление нового слайда
    pres.Slides.Add 1, ppLayoutBlank
    'Показать на экране PowerPoint
    ppt.Visible = True
    'Активизировать PowerPoint
    AppActivate ppt.Name
    'Вставка диаграммы
    ppt.ActiveWindow.View.Paste
    Set ppt = Nothing
End Sub
```

MICROSOFT OUTLOOK

В этом примере данные переносятся в новую задачу Outlook. Чтобы увидеть новую задачу, запустите Outlook и выберите "Задачи" на панели Outlook.
Примечание: появление новой задачи на экране может занять несколько минут.

Пример программы MS_Outlook

```
Sub MS_Outlook()
Dim ol As Object, myItem As Object
    'Создание сеанса работы с Microsoft Outlook
    Set ol = CreateObject("outlook.application")
    'Создание задачи
    Set myItem = ol.CreateItem(olTaskItem)
    'Добавление данных в новую задачу
    With myItem
        .Subject = "Новая задача VBA"
        .Body = "Задача создана с помощью программирования объектов в Microsoft Excel"
        .NoAging = True
        .Close (olSave)
    End With
    'Удаление объекта из памяти
    Set ol = Nothing
End Sub
```

MICROSOFT BINDER

В этом примере создается новый файл Microsoft Binder, добавляются в него разделы, производятся действия с разделом Microsoft Excel, а затем файл сохраняется.

Пример программы MS_Binder

```
Sub MS_Binder()
Dim MyBinderNew As Object, Section As Object
Dim MyVar As String, MyFile As String
    'Создание сеанса работы с Microsoft Office Binder
    Set MyBinder = CreateObject("office.binder")
    'Отобразить на экране подшивку
    MyBinder.Visible = True
    'Добавление нового раздела Word в подшивку
    MyBinder.Sections.Add Type:="word.document"
    'Добавление существующего файла Excel перед первым разделом
    MyFile = Application.Path & "\examples\solver\solvsamp.xls"
    Set NewSection = MyBinder.Sections.Add(FileName:=MyFile, Before:=1)
```

```
    With NewSection
        'Изменение имени раздела
        .Name = "Поиск решения: примеры"
        'Значение из ячейки A2 на третьем листе книги
        MyVar = .Object.Worksheets(3).Range("A2").Value
    End With
    'Сохранение сеанса работы как файла mybinder.obd, перезаписывание
    'существующего файла
    MyBinder.SaveAs Application.Path & "\mybinder.obd", bindOverwriteExisting
    'Не отображать подшивку
    MyBinder.Visible = False
    'Отключение объекта, освобождение ресурсов
    Set MyBinder = Nothing
    'Показать значение из ячейки A2 с третьего листа раздела Excel
    MsgBox "Данные берутся из ячейки A2 в разделе 'Поиск решения: примеры'." &
        Chr(10) & Chr(10) & MyVar
End Sub
```

Point.

Програмирование событий

Часто возникает необходимость в средстве автоматического запуска макроса при возникновении определенного события. В Microsoft Excel 97 такая возможность предусмотрена. События теперь всегда связаны с объектом, например листом или книгой. В этом примере обсуждается только несколько из возможных событий и используется окно сообщений для демонстрации работы программы. Чтобы получить более подробные сведения об этих и других событиях, используйте поиск по ключевому слову "события" или по названию объекта (например "события листа") на вкладке "Поиск" справочной системы по VBE.

Событие BeforeDoubleClick

Чаще всего используется событие листа с именем **BeforeDoubleClick**. В самом простом случае программа, использующая это событие выглядит следующим образом:

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Excel.Range, Cancel As Boo
    MsgBox "Был произведен двойной щелчок по ячейке " & Target.Address
    Cancel = True
End Sub
```

Это событие происходит, при выборе ячейки листа с двойным нажатием кнопки мыши. Параметр **Target** передается в макрос, таким образом сообщается, что по ячейке был произведен двойной щелчок. Аргумент **Cancel** имеет значение False по умолчанию, но его можно изменить на True внутри программы. Устанавливая значение аргумента **Cancel** в True, отменяется выполняемая по умолчанию операция для события. В этом случае, такой операцией является переход в режим редактирования выбранной ячейки. На текущем листе перехода в этот режим не происходит, так как параметру **Cancel** присвоено значение True. Если возникнет необходисмость восстановить операцию, выполняемую по умолчанию, удалите строку `Cancel=True`.

Событие Change

Другим, часто используемым событием, является событие листа — **Change**. Оно возникает при вводе нового значения в любую ячейку листа.

```
Private Sub Worksheet_Change(ByVal Target As Excel.Range)
    MsgBox "Лист изменен в ячейке " & Target.Address
End Sub
```

Примечание: это событие не возникает при пересчете листа. Существует событие с именем **Calculate**, которое возникает при пересчете листа.

Событие BeforeClose

Полезным оказывается событие **BeforeClose**., которое относится ко всей книге. Это событие может быть использовано для выполнения вспомогательных операций перед сохранением или закрытием файла.

```
Sub Workbook_BeforeClose(Cancel As Boolean)
    a = MsgBox("Не сохранять файл перед закрытием?", vbYesNo)
        If a = vbNo Then Cancel = True
End Sub
```

В этом примере выводится сообщение, если пользователь не сохранил книгу перед ее закрытием.

Использование событий с объектом приложения

Перед использованием событий с объектом приложения, необходимо создать новый модуль класса и описать объект типа приложения с событием. Например, предположим что новый модуль класса создан и называется EventClassModule. Новый модуль класса содержит следующую программу.

```
Public WithEvents App As Application
```

Далее новый объект, описанный с событием, появится в поле со списком объектов в модуле класса. Теперь для нового объекта можно записать процедуры события. (При выборе нового объекта в поле со списком объектов, допустимые события для этого объекта будут перечислены в поле со списком процедур.)

Перед запуском процедур необходимо соединить описанный объект в модуле класса с объектом приложения. Это можно сделать с помощью следующей программы из любого модуля.

```
Dim X As New EventClassModule

Sub InitializeApp()
Set X.App = Application
End Sub
```

Запустите процедуру InitializeApp. Объект App в модуле класса указывает на объект приложения Microsoft Excel. Процедуры события в модуле класса будут выполняться при возникновении этого события.

Примечание: чтобы получить более подробные сведения обо всех событиях, обратитесь к справочной с

lean)

истеме.