Contents

A chap who lived in Rawalpindi
Who did not speak Greek, French or Hindi
    Who wanted to buy
    What he called Delphi
While actually it should have been Delphi.

Disclaimer

Screen Resolution

When designing forms, it is sometimes helpful to write the code so that the screen and all of its objects are displayed at the same size no matter what the screen resolution is. Here is some code to show how that is done:

```
implementation
const
  ScreenWidth: LongInt = 800; {I designed my form in 800x600 mode.}
  ScreenHeight: LongInt = 600;

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
var
  i, OldFormWidth: integer;
begin
  scaled := true;
  if (screen.width <> ScreenWidth) then begin
    OldFormWidth := width;
    height := longint(height) * longint(screen.height) DIV ScreenHeight;
    width := longint(width) * longint(screen.width) DIV ScreenWidth;
    scaleBy(screen.width, ScreenWidth);
    font.size := (Width DIV OldFormWidth) * font.size;
  end;
end;
```

Then, you will want to have something that checks to see that the font sizes are OK. Before you change the font's size, you would need to ensure the object actually has a font property by checking the RTTI. This can be done as follows:

```
USES TypInfo;  {Add this to your USES statement.}

var
  i: integer;
begin
  for i := componentCount - 1 downto 0 do
    with components[i] do
    begin
      if GetPropInfo(ClassInfo, 'font') <> nil  then
        font.size := (NewFormWidth DIV OldFormWidth) * font.size;
    end;
end;
```

{This is the long way to do the same thing.}

```
var
  i: integer;
  p: PPropInfo;
begin
  for i := componentCount - 1 downto 0 do
    with components[i] do
    begin
```

```
      p := GetPropInfo(ClassInfo, 'font');
      if assigned(p) then
        font.size := (NewFormWidth DIV OldFormWidth) * font.size;
    end;
end;
```

Note: not all objects have a FONT property. This should be enough to get you started. The font size changes are based on changing the font.size of the form. To notice small font size changes, try using a true type font.

Note: The following are issue to bear in mind when scaling Delphi applications (forms) on different screen resolutions:

* Decide early on in the form design stage whether you're going to allow the form to be scaled or not. The advantage of not scaling is that nothing changes at runtime. The disadvantage of not scaling is that nothing changes at runtime (your form may be far too small or too large to read on some systems if it is not scaled).

* If you're NOT going to scale the form, set Scaled to False.

* Otherwise, set the Form's Scaled property to True.

* Set AutoScroll to False. AutoScroll = True means 'don't change the form's frame size at runtime' which doesn't look good when the form's contents do change size.

* Set the form's font to a scaleable TrueType font, like Arial. MS San Serif is an ok alternate, but remember that it is still a bitmapped font. Only Arial will give you a font within a pixel of the desired height. NOTE: If the font used in an application is not installed on the target computer, then Windows will select an alternative font within the same font family to use instead. This font may not match the same size of the original font any may cause problems.

* Set the form's Position property to something other than poDesigned. poDesigned leaves the form where you left it at design time, which for me always winds up way off to the left on my 1280x1024 screen - and completely off the 640x480 screen.

* Don't crowd controls on the form - leave at least 4 pixels between controls, so that a one pixel change in border locations (due to scaling) won't show up as ugly overlapping controls.

* For single line labels that are alLeft or alRight aligned, set AutoSize to True. Otherwise, set AutoSize to False.

* Make sure there is enough blank space in a label component to allow for font width changes - a blank space that is 25% of the length of the current string display length is

a little too much, but safe. (You'll need at least 30% expansion space for string labels if you plan to translate your app into other languages) If AutoSize is False, make sure you actually set the label width appropriately. If AutoSize is True, make sure there is enough room for the label to grow on its own.

* In multi-line, word-wrapped labels, leave at least one line of blank space at the bottom. You'll need this to catch the overflow when the text wraps differently when the font width changes with scaling. Don't assume that because you're using large fonts, you don't have to allow for text overflow - somebody else's large fonts may be larger than yours!

* Be careful about opening a project in the IDE at different resolutions. The form's PixelsPerInch property will be modified as soon as the form is opened, and will be saved to the DFM if you save the project. It's best to test the app by running it standalone, and edit the form at only one resolution. Editing at varying resolutions and font sizes invites component drift and sizing problems.

* Speaking of component drift, don't rescale a form multiple times, at design time or a runtime. Each rescaling introduces roundoff errors which accumulate very quickly since coordinates are strictly integral. As fractional amounts are truncated off control's origins and sizes with each successive rescaling, the controls will appear to creep northwest and get smaller. If you want to allow your users to rescale the form any number of times, start with a freshly loaded/created form before each scaling, so that scaling errors do not accumulate.

* Don't change the PixelsPerInch property of the form, period.

* In general, it is not necessary to design forms at any particular resolution, but it is crucial that you review their appearance at 640x480 with small fonts and large, and at a high-resolution with small fonts and large before releasing your app. This should be part of your regular system compatibility testing checklist.

* Pay close attention to any components that are essentially single-line TMemos - things like TDBLookupCombo. The Windows multi-line edit control always shows only whole lines of text - if the control is too short for its font, a TMemo will show nothing at all (a TEdit will show clipped text). For such components, it's better to make them a few pixels too large than to be one pixel too small and show not text at all.

* Keep in mind that all scaling is proportional to the difference in the font height between runtime and design time, NOT the pixel resolution or screen size. Remember also that the origins of your controls will be changed when the form is scaled - you can't very well make components bigger without also moving them over a bit.

Form Design Help

Displaying a form as being the same size even when the Screen Resolution varies.
How can I restore a window to its last state when I run it again?
What is the order of event handlers when a form is created and shown?
Iconizing an application and keeping it that way.
Hiding the caption bar
How do I set the WindowState to wsMinimized when I minimize a form?
How do I move one image across a background image?
How do I use a form several times?
How do I use a case statement to determine which object calls the procedure?
How do I do screen updates all at once (without a ripple effect).
How do I initialte a DDE link to the program manager to create a new group?
I want to know how I can make a variable that is a "pointer" to Canvas.Font.
How do I use one of the cursor files in the c:\delphi\images\cursors?
How can I tell if the right mouse button was pressed?
How do I trap for right mouse clicks on my VBX and have a popup menu display?
How do I create a floating palette window?
How do I print a form?
How do I close the help file when I close the application?
How do I display a JPEG or PCX file?
How do I put a repeating bitmap on the background of an MDI main form?
Where is the best place to open a splash screen on program start up?
GetMinMax
How do I do something on the form's OnActivate method?
How do I paint the background of my form with a bitmap?
How can I have an animated icon (when the form is minimized)?
How do I fake TTabbedNotebook with multiple forms?
How do I make it so that only the form I select comes to the top? (i.e. *without* the main form)
How can I trap for my own hotkeys?
How do I paint with a cross-hatched brush?
How do I put the current time on the title bar of my form?
How do I place the mouse anywhere on the form that I want?
How do I size a form to fit a bitmap?

When doing date math on calculated fields, it is important to ensure that all values being used are properly matched as to type. The double method (not in the docs) casts the value to a useable type.

In the following method, d1, and d2 (part of table1) can be of either date or dateTime type. d3 is an integer field.

```
procedure TForm1.Table1CalcFields(DataSet: TDataset);
var
  t1, t2: tDateTime;
begin
  table1d1.asDateTime := Date + 2; {or table1d1.value := date + 2;}
  table1d2.asDateTime := Date - 2;
  t1 := table1d1.asDateTime;
  t2 := table1d2.asDateTime;
  table1d3.asInteger := trunc(double(t1) - double(t2));
end;
```

Database Help

Topics:
general
paradox
dBASE

other:
IDAPI calls
ODBC
SQL

isDigit

```
function isDigit(ch: char): boolean;
begin
   if ch in ['0'..'9'] then
      isDigit := true
   else
      isDigit := false;
end;
```

Stuff

This function will remove one part of a string and replace it with another.

```
function stuff( ToString: string;
                FirstByte, NumOfBytes: integer;
                FromString: string ): string;
begin
  delete(toString, FirstByte, NumOfBytes);
  insert(FromString, ToString, FirstByte);
  stuff := ToString;
end;
```

StrStr

Find one string in another. StrStr is the "C" function name. The dBASE equivalent is at().

```
function StrStr(LookHere, FindThis: string): integer;
var
   p: PChar;
   s1, s2: array[0..255] of Char;
begin
   StrPCopy(s1, LookHere);
   StrPCopy(s2, FindThis);
   p := StrPos(s1, s2);
   StrStr := (p - s1) + 1;
end;
```

IsUpper

```
function isUpper(ch: char): boolean;
begin
   if ch in ['A'..'Z'] then
      isUpper := true
   else
      isUpper := false;
end;
```

IsLower

```
function isLower(ch: char): boolean;
begin
  if ch in ['a'..'z'] then
    isLower := true
  else
    isLower := false;
end;
```

ToUpper

```
function toUpper(ch: char): char;
begin
  toUpper := chr(ord(ch) and $DF);
end;
```

ToLower

```
function toLower(ch: char): char;
begin
  toLower := chr(ord(ch) or $20);
end;
```

upper

```
function Upper(s: string): string;
var
  i: integer;
begin
  for i := 1 to length(s) do
    if isLower(s[i]) then s[i] := toUpper(s[i]);
  Upper := s;
end;
```

lower

```
function Lower(s: string): string;
var
  i: integer;
begin
  for i := 1 to length(s) do
    if isUpper(s[i]) then s[i] := toLower(s[i]);
  Lower := s;
end;
```

proper

```
function Proper(s: string): string;
var
  i: integer;
  CapitalizeNetLetter: boolean;
begin
  s := Lower(s);
  CapitalizeNetLetter := true;
  i := 1;
  repeat
    if CapitalizeNetLetter then if isLower(s[i]) then
      s[i] := toUpper(s[i]);
    if s[i] = ' ' then CapitalizeNetLetter := true
    else CapitalizeNetLetter := false;
    i := i + 1;
  until i > length(s);
  Proper := s;
end;
```

String Help

IntToHexStr
HexStrToInt
IntToBinaryStr
isDigit
isUpper
isLower
toUpper
toLower
upper
lower
proper
rTrim()
lTrim()
AllTrim()

How do I format Numbers (adding commas to a longint)?
Is there a way to use a Pascal string as a null terminated string?
How do I manipulate a TStringList in a DLL?
How can I parse a PChar?
How do I do a BreakApart()?
How do I do a search and replace in a string?
How can I get the command line parameters?
How can I determine the length in pixels of a string after a specific font has been applied to it?
How do I determine if two strings sound alike?
How do I store dates beyond the year 2000?
How do I find one string inside another *with wildcards*?

iconized apps

How do I keep the form in icon form when I run it? In the private section of the form object's declaration, put:

```
PROCEDURE WMQueryOpen(VAR Msg : TWMQueryOpen); message WM_QUERYOPEN;
```

In the implementation section, put this method:

```
PROCEDURE TForm1.WMQueryOpen(VAR Msg : TWMQueryOpen);
begin
  Msg.Result := 0;
end;
```

That's it! The form will always remain iconic. OBTW, of course you must set WindowState to wsMinimized in the form's properties initially.

"C" help

Type conversions
Static variables in Pascal
What is the Object Pascal equivalent of C's "union" reserved word?
How do I do pointer arithmetic in Delphi?
How do I translate this 'C' delaration to ObjectPascal?
How can I emulate the "C" function: StrTok()?

C to Pascal

```
 C type               Pascal type
-------------------------------
 unsigned char       byte
 char                char
 char name[arSize]   array [0..arSize - 1] of Char
 int                 integer
 unsigned int        word
 long                longint
 unsigned long       longint
 float               single
 double              double
 char far *          PChar
 char *              PChar if large memory model, Word in small and medium
 SomeType far *      Var aName: SomeType
                     or PSomeType (type PSomeType = ^SomeType) if you
                     need the option to pass Nil as a value
 struct              record
 union               record with variants
 enum                enumerated type with compiler switch $Z+ set
                     (default is $Z-!)
```

Q: If the argument of a C function is of type float, how should it be declared in Pascal: real, single or double?

A: Guaranteed "real" is NOT the answer. The Real data type is a 6-byte floating point number that's completely specific to Borland Pascal. Single and Double correspond precisely to the IEEE standard 4-byte and 8-byte floating point types. There's also an Extended type which is the 10-byte floating point format used internally by the numeric co-processor. A "C" float is a Pascal SINGLE! The double type is called double in C, too.

```
type Plong = ^longint;

function foo
  ( arg1 : Plong; arg2: Pchar; arg3, arg4, arg5, arg6 : double ) : longint;
```

This was translated from the following C declaration:

```
long foo( long FAR * arg1,
          char FAR * arg2
          float arg3,
          float arg4,
          float arg5,
          float arg6 );
```

In general, when an argument in a C function contains "FAR *" and it's NOT a pointer-to-char, instead of making the corresponding Pascal argument a pointer, you simply make it a VAR parameter. So your function header would begin:

```
function foo(VAR arg1 : Long; ...
```

Q (follow up): Since Foo *returns* a pointer, should arg1, the longint pointer, be declared as a variable parameter in the declaration of Foo?

A: If this weird foo function returns a pointer to its first arguement, you can do it like this:

```
FUNCTION Foo(VAR bar : LongInt; ...) : PLongInt;
BEGIN
  ...
  Foo := @bar;
END;
```

... or you can do it like this:

```
FUNCTION Foo(bar : PLongInt; ...) : PLongInt;
BEGIN
  ...
  Foo := Bar;
END;
```

In the latter case, you'll have to dereference Bar every time you use it in the function. The former is probably easier on YOU. If the implementation of the function is in a DLL written by others and it does a lot more than return the pointer, the second option may be the only one open to you.

Static vars in Pascal

While there are no static data members but there are static methods. The workaround for the former is to use a "typed constant" which is the Pascal equivalent of a C static and is declared thus:

```
const
  MyStaticInt : integer = 12345 ;
```

It cannot be declared as part of a class but it can be local to a proc/fn/method and will retain it's value between calls in true static fashion.

Hex to Decimal

Q: I want to write a simple application that converts numbers from hex to decimal. My form has two edit boxes, one for the decimal number and another for the hex number. I add some code to each edit box's OnChange handler and as I type in one, the other updates in real time. The problem I am wondering about, is this. The OnChange for one, changes the edit box text in the other, firing the OnChange in that one, which in turn updates the other and fires its OnChange and so on ad infinitum.

A: One solution would be to do nothing in the OnChange event handler unless ActiveControl is equal to the control that is calling the event handler.

```
procedure TForm1.Edit2Change(Sender: TObject);
begin
 if ActiveControl = Edit2 then
    Edit1.Text := IntToHex(StrToInt(Edit2.Text),0);
end;

procedure TForm1.Edit1Change(Sender: TObject);
begin
 if ActiveControl = Edit1 then
    Edit2.Text := IntToStr(StrToInt('$'+Edit1.Text));
end;
```

Options: Instead of OnChange. Use OnKeyDown. This way 1 Keydown equates to 1 loop as you intend.

Caption bar Hiding

Q: I'm looking for a way to hide the Caption (or Title) bar of my application. I want to have a Sizable Window with no Caption Bar. Is this possible?

A: try this method connected to onCreate event:

```
 procedure TForm1.FormCreate(Sender: TObject);
 begin
     SetWindowLong(Handle,GWL_STYLE, GetWindowLong(Handle,GWL_STYLE) AND NOT
WS_CAPTION);
     ClientHeight:=Height;
 end;
```

it's not very elegant but...

Porting Apps Help

Keeping streams and collections

Streams and collections

Q: I have 2 heavily used BP7 libraries to migrate to Delphi. Both use a lot of collections & Streams. I'm probably not alone in this situation. Can we mix Objects & classes in the same application? If not, I'm ready to dump Streams, but can collections be saved???

Is there any way of saving that code without a major rewrite?

(I tried to USES OBJECT but I'm getting all sorts of compiler complaints...)

A: Yes, it's actually pretty straightforward - I have a lot of code that does that. I have Objects as the first item in my "uses" statement. You DO need to tell Delphi where to find the OWL run-time library source (\delphi\source\rtl70, I think). Then, you just do either

```
anObject := new(PMyObject, init);
```

for old-style or

```
anObject := TMyObject.create;
```

for new-style objects. Make sure you don't mix up which is which though, or you'll get compiler "invalid variable reference" messages (which sometimes confuse me if I've accidentally treated an old-style object like a VCL one, or vice versa).

In fact I prefer the old streaming in many ways. As far as I can see, you need to descend an object from TComponent in order to get automatic streaming in VCL.

Topics

Command line

How do I get everything on the command line?
How can I get the command line parameters?

Other

Using Crystal Reports, how do I let the user select a printer at run-time?

Multi-tasking

This function should be called occasionally whenever your app does something that seizes the CPU, like a long disk copy or a large loop. This will allow for pseudo multi-tasking (Windows style) to take place.

The easiest way is `application.ProcessMessages`. If you want to do it by hand, here it is:

Note: A delay function won't do quite the same thing.

```
function Yield: Boolean;
var
  msg: TMsg;
begin
  while PeekMessage(msg, 0, 0, 0, PM_REMOVE) do
  begin
    if msg.message = WM_QUIT then
    begin
      PostQuitMessage(msg.wParam);
      Yield := TRUE;
      EXIT
    end
    else
    begin
      TranslateMessage(msg);
      DispatchMessage(msg)
    end
  end;
  Yield := FALSE
end;
```

Formatting Numbers

This function will add commas to a longint.

```
function FormatNumber(l: longint): string;
var
  len, count: integer;
  s: string;
begin
  str(l, s);
  len := length(s);
  for count := ((len - 1) div 3) downto 1 do
    begin
      insert(',', s, len - (count * 3) + 1);
      len := len + 1;
    end;
  FormatNumber := s;
end;
```

And if you are using Delphi, there is, of course, the easy way:

```
function FormatNumber(l: longint): string;
begin
  FormatNumber := FormatFloat('#,##0', StrToFloat(IntToStr(l)));
end;
```

Hi/Lo order byte

How do you extract the high or low order byte from a word? How do you insert it? There are built in methods hi() and lo() for extracting, but for those that want to know how to do it on their own, here it is in its most efficient form (if I do say so myself <G>). The functions for inserting bytes are not in Delphi. Note: Assembler functions return the contents of the AX register.

```
function GetHiByte(w: word): byte; assembler;
asm
   mov ax, w
   shr ax, 8
end;

function GetLoByte(w: word): byte; assembler;
asm
   mov ax, w
end;

function SetHiByte(b: byte; w: word): word; assembler;
asm
   xor ax, ax
   mov ax, w
   mov ah, b
end;

function SetLoByte(b: byte; w: word): word; assembler;
asm
   xor ax, ax
   mov ax, w
   mov al, b
end;
```

Another way of doing it: How about REAL FAST, without using assembler (i.e. let the compiler do the work for you)???

```
 Type
   TWord2Byte = record
                  Lo,Hi: Byte;
                end;

 var W: Word;
     B: Byte;
 begin
   W := $1234;
   B := TWord2Byte(W).Hi;
   writeln(TWord2Byte(W).Hi);
   { going back }
   TWord2Byte(W).Lo := $67;
   TWord2Byte(W).Hi := $98; { no shl needed! }
 end.
```

Dynamic objects

This will place a TImage object on the form and fill it with a picture.

```
uses ExtCtrls;
procedure TForm1.Button1Click(Sender: TObject);
var
  ti: tImage;
begin
  ti := tImage.create(self);
with ti do
begin
  parent := self;
  autosize := true;
  Picture.LoadFromFile('c:\windows\MyBmp.bmp');
  show;
end;
end;
```

To make it always fit the window, do it this way:

```
with ti do
begin
  align := alClient;
  stretch := true;
  autosize := true;
end;
```

Navigator button check

Is there any way to determine if a particular button on a TDBNavigator control is enabled? (Buttons is a protected property.)

<Warning: slimy hack alert!>

```
type TDBNavCracker = class(TDBnavigator);

if TDBNavCracker(DBNavigator1).Buttons[nbEdit].Enabled then {};
```

Dynamic assigning at runtime

Q: Is there an easy way to assign speedbutton properties via iteration at runtime? (The Speedbutton properties in my application are very dynamic.) That is I don't want to have to do the following for every speedbutton property that changes:

```
Toolbar.Speedbutton1.Glyph := GetGlyph(1);
Toolbar.Speedbutton2.Glyph := GetGlyph(2);
```

But rather something that like

```
For I := 1 to NumSpeedButtons do
  Toolbar.Speedbutton[I].Glyph := GetGlyph(I);
```

A: This code fragment might put you on the right track. It iterates through all the components on the form. I suppose you could use the Tag property to control which Glyph is which in your GetGlyph function.

```
for I := 0 to ComponentCount-1 do
if Components[I] is TSpeedbutton then
  TSpeedButton(Components[I]).Glyph := GetGlyph();
```

Another option is to build your own array just for SpeedButtons (much like TForm builds it for all components).

Q: Is there a way I can generically respond to all Speedbutton clicks?

```
{ Not this. }
procedure TToolbar.SpeedButton0Click(Sender: TObject);
begin
  SpeedButtonAction(0);
end;

{ More like this. }
procedure TToolbar.AnySpeedButtonClick(Sender:TObject, ButtonNo:byte);
begin
  SpeedButtonAction(ButtonNo);
end;
```

A: Set all the OnClick events to point to the following method, and assign the appropiate value to the Tag properties of the SpeedButtons.

```
procedure TForm1.AnySpeedButtonClick(Sender:TObject)
begin
  SpeedButtonAction((Sender as TSpeedButton).Tag);
end;
```

Note: If your speed buttons do indeed have names SpeedButton1, SpeedButton2, etc., you can use the form's FindComponent method to pretend they're an array:

```
FOR N := 1 TO NumButtons DO
  WITH FindComponent('SpeedButton'+Str(N)) AS TSpeedButton DO { whatever } ;
```

Linking with OBJ files

Q: I have an OBJ file that has several assembler routines compiled into it. I wouldn't want to have to rewrite them. Is there a way to use them in a Delphi app?

A: You don't indicate if these return values or not. In Pascal this matters.

If they don't, include the following near the front of your code:

```
Procedure TurnOn;  External;
Procedure TurnOff; External;
```

If they return values, use:

```
Function TurnOn: Integer; External;
Function TurnOff: Integer; External;
```

Replace Integer with whatever datatype they return.

In either case follow that with:

```
{$L filename.obj}
```

to link in the obj file.

1. Remove "device=w31s.386" from the 386enhanced section of system.ini.

2. There are problems with w32sys.dll, win32s16.dll, win32s.ini, etc. files in \windows\system that you can remove. Also, windows\system\win32s should be removed. (It is a general 32 bit windows problem.)

3. If there is an "Error... Could not find object" message before the GPF, recopy the IDAPI.CFG from the cd-rom. It could be corrupt.

IDAPI specs

TITLE : Here are the CURRENT maximum limits for some common IDAPI objects. These may change for next release.

48 // Max clients in system
32 // Max sesssions per client
32 // Max open databases per session
32 // Max loaded drivers
64 // Max sessions in system
4000 // Max cursors per session
100 // Max passwords per session
16 // Max entries in error stack
127 // Max locks of a given type on a given table

BLOB handles per cursor:
Paradox: max (16, two times the number of BLOB fields in the table)
dBASE: Two times the number of BLOB field in the table

Simple windows program

This is a simple program to show how a windows program is written from scratch in BP (with no OWL). There is also an example of this in \delphi\demos\generic.

```
{Lloyd Linklater;  2-6-95}
{c:\bp\examples\win\generic.pas has much of this in it.}
program WinClk;
{$R sumthing}  {Resource file is sumthing.res}
uses WinTypes, WinProcs;
const
  AppName = 'WinClk';

{***************************}
{The export is used to force the far call model and to generate
special entry code so that it can be called by Windows.}

function About(Dialog: HWnd; Message, WParam: Word;
               LParam: Longint): Bool; export;
begin
  About := True;
  case Message of
    wm_InitDialog: Exit;
    wm_Command: if (WParam = id_Ok) or (WParam = id_Cancel) then
    begin
      EndDialog(Dialog, 1);
      Exit;
    end;
  end;
  About := False;
end;  {About}

{***************************}

function WindowProc(Window: HWnd; Message, WParam: Word;
                    LParam: Longint): Longint; export;
var
  AboutProc: TFarProc;
begin
  WindowProc := 0;
  case Message of
{The WM_COMMAND message is sent to a window when the user selects
an item from a menu, when a control sends a notification
message to its parent window, or when an accelerator keystroke
is translated. }
    wm_Command: case WParam of
      301 : {Help | About selected from the menu.}
        begin
          AboutProc := MakeProcInstance(@About, HInstance);
          DialogBox(HInstance, 'AboutBox', Window, AboutProc);
          FreeProcInstance(AboutProc);
          Exit;
        end;
      101 : {EXIT selected from the menu.}
        begin
          PostQuitMessage(0); {Puts a wm_Quit message on the queue.}
```

```
          halt;
        end;
      end;
    wm_Destroy:
      begin
        PostQuitMessage(0);
        Exit;
      end;
  end;  {case Message}
  WindowProc := DefWindowProc(Window, Message, WParam, LParam);
end;  {WindowProc}

{***************************}
{This MUST be called WinMain.}

procedure WinMain;
var
  Window: HWnd;
  Message: TMsg;
const
  WindowClass: TWndClass = (
    style:         0;
    lpfnWndProc:   @WindowProc; {Function pointer to the message handling
code.}
    cbClsExtra:    0;
    cbWndExtra:    0;
    hInstance:     0;
    hIcon:         0;
    hCursor:       0;
    hbrBackground: 0;
    lpszMenuName:  'MyFirst';
    lpszClassName: AppName);
begin
  if HPrevInst = 0 then {If there is not another copy running then...}
  begin {...a window class must be declared since it is not declared
already.}
    WindowClass.hInstance := HInstance;
    WindowClass.hIcon := LoadIcon(hInstance, 'ICON_1');
    WindowClass.hCursor := LoadCursor(0, idc_Arrow);
    WindowClass.hbrBackground := GetStockObject(white_Brush);
    if not RegisterClass(WindowClass) then Halt(255);
  end;
  Window := CreateWindow(AppName,               {Class name}
                         'Windoze Clock',      {Window name}
                         ws_OverlappedWindow,  {style}
                         cw_UseDefault,        {X}
                         cw_UseDefault,        {Y}
                         cw_UseDefault,        {Width}
                         cw_UseDefault,        {Height}
                         0,                    {WndParent}
                         0,                    {Menu}
                         HInstance,            {Instance}
                         nil);                 {structure creation parameter}

  {CmdShow is used only when ShowWindow is used to display the
  app's main window.  Otherwise it uses one of the sw_ constants.}
  ShowWindow(Window, CmdShow);
```

```
  UpdateWindow(Window);

  {Messages are not sent directly to the app by windows, so we must use
  the OBLIGATORY message loop to keep getting messages and using them
  until we get the 'go away now' message.  If the message is WM_QUIT, then
  GetMessage() returns a 0.  Message is of type TMsg.
    TMsg = record
      hwnd: HWnd;
      message: Word;
      wParam: Word;
      lParam: LongInt;
      time: Longint;
      pt: TPoint;
    end;}
  while GetMessage(Message, 0, 0, 0) do
  begin
    {This translates virtual-key messages into character messages.}
    TranslateMessage(Message);
    DispatchMessage(Message); {The translated message is now 'mailed' out.}
  end;
  Halt(Message.wParam);
end;  {WinMain}

{**************************}

begin
  WinMain; {By this time it is simple <G>}
end.
```

Enter as Tab

Q: How do I make it so that when the user hits <enter>, it goes to the next object as though he had hit the tab key?

A: You need to trap the keystroke and set up your own response to it. Try this:

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  keyPreview := true; {To turn the event "ON".}
end;

procedure TMainForm.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then
  begin
    Key := #0;
    PostMessage(Handle, WM_NEXTDLGCTL, 0, 0);{next control}
    {PostMessage(Handle, WM_NextDLGCTL, 1, 0);} {previous control}
  end;
end;
```

Here is some info from compu-serve (FWIW)

<<I am trying to override the default behavior of the left & right arrow keys in a dbgrid. I am using the OnKeyDown event of DBgrid. Overriding the right arrow as follows works fine, but trying to override the left arrow with Shift-Tab doesn't:>>

I haven't tried to mess around with the left and right arrow keys, so the following may not work. Note that I have this implemented at the form level, with the key preview (I think that's what its called) property set to true.

Basic answer however, is that you have to us the key up procedure when you access shift, control, and alt. Look in help under on key up. It will explain much better than I can. I use the following to emulate, at least in part, the same type keystrokes as in pdoxwin. I haven't implemented the page down and page up yet. Hope this helps

```
procedure TFreightRateForm.FormKeyUp(Sender: TObject; var Key: Word; Shift:
TShiftState);
Begin
  if (Key = VK_F9) and not((shift=[ssalt]) or (shift=[ssshift])
            or (shift=[ssctrl])) then
     case FreightTbl.State of
       dsEdit : FreightTbl.Post;
       dsInsert : FreightTbl.Post ;
    else
       FreightTbl.edit;
    end ;
    if ((Shift = [ssAlt]) and (Key = VK_BACK)) then
      FreightTbl.Cancel;
    if (Key = VK_Insert) then
```

```
      if FreightTbl.State = dsEdit then
        FreightTbl.Insert;
end;
```

Filling an outline from a table

Here is how to fill an outline component from a table. It has been written so that the name, address, etc appears as a branch from the root company name.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  t: TTable;
  indx, FieldCounter: integer;
begin
t := TTable.create(self);
with t do
begin
  DatabaseName := 'pw'; {personal alias}
  TableName := 'customer.db';
  open;
  first; {probably redundant}
  while not eof do
  begin
    indx := outline1.add(0, fields[1].AsString);
    for FieldCounter := 2 to 7 do
      outline1.addChild(indx, fields[FieldCounter].AsString);
    next;
  end; {end of the while statement}
  close;
end;  { End of the with statement. }
end; {end of the procedure}
```

filling a listbox from a table

Here is how to do it when the field is not a string: (In this case, it is a date field.)

```
var
  QDate: TTable;
begin
  listbox1.Clear;
  QDate := TTable.create(application);
  with QDate do
  begin
    databasename := 'DBDEMOS';
    TableName := 'orders.db';
    open;
    First;
    Repeat
      listbox1.Items.Add(FieldByName('SaleDate').AsString);
      Next;
    until EOF;
    Close;
  end;
end;
```

Here's one way of populating a list box with your fields ID and Name:

```
  with MyQuery do
  begin
{ this is to prevent flicker in any controls using this dataset }
    DisableControls;

    try
      First;
      while not EOF do
      begin

       { Here's the actual statement you asked for... }
        with MyListBox.Items do
          Objects[ Add( FieldByName('Name').AsString ) ] :=
            TObject( PChar( FieldByName('ID').AsInteger ) );

        Next;
      end;
    finally
      EnableControls;
    end;
  end;
```

And here's how you can extract the ID from the currently selected item in the listbox:

```
  with MyListBox do
    MyInteger := Longint( Items.Objects[ ItemIndex ] );
```

This error indicates that SHARE.EXE (or VSHARE.386) wasn't loaded.

bitmap pasting

Q: When I paste a glyph/BMP with white in it into a button, the white disappears. The glyph/BMP appears just fine if I paste into into a button in Paradox. What gives?

A: Delphi selects whatever color is in the lower left corner of your bitmap as the transparent color for the button as it rests on the button. If the problem is that you _want_ the white to appear in the BitBtns in Delphi, then edit the bitmaps so that the lower left corner bit is a color that appears nowhere else in the bitmap. (That yucky olive color works -- who uses that? <g>)

Navigator control use

Q: I have a form that uses several TDBGrids. It has only one navigator control. How do I write it so that I can use the navigator control so that it works with whatever grid is active?

A: Use this line in the Enter event of each grid:

```
TDBNavigator1.dataSource := (sender as TDBGrid).dataSource;
```

TListBox with tabs

Q: Do you have to do anything special to get tabs to be expanded in a listbox? I am just getting little line characters instead of tabs.

A: Yes. The default behavior of a listbox will not show tabs, but you can 'roll your own' by inheriting TListbox and going from there. You need to OR lbs_UseTabStops into the window style in the create params.

sound

Q: How do I make sound the way that it worked in BP7?

A: Here is some code, that makes the sound of Turbo Pascal:

Note: This code is untried and contains elements that must be adjusted for Delphi.

```
{This should be re-written using TTimer.}
Function Waiting(ms: LongInt): BOOLEAN;
VAR
  TickCount: LongInt;
Begin
  Waiting:= false;
  TickCount:= GetTickCount;
  While GetTickCount - TickCount < ms do yield; {see multi-tasking}
  Waiting:= true;
End;

{Works.}
Procedure NoSound;
Begin
  port[$61]:= port[$61] and $FC;
End;

{Works, but just plays one tone.}
Function Sound(freq, dauer: Word): BOOLEAN;
VAR
  b: byte;
Begin
  if freq > 18 then begin
    freq:= word(1193181 div longint(freq));
    b:= port[$61];
    if (b and 3)=0 then
    begin
      port[$61]:= b or 3;
      port[$43]:= $b6;
    end;
    port[$42]:= byte(freq);
    port[$42]:= byte(freq shr 8);
  end;
  Waiting(dauer);
  NoSound;
End;
```

Memory Model

Q: Which memory model does Delphi use?

A: Delphi uses a mixed memory model. The defaults are:

- Methods are far
- Procedures in an interface section are far
- Procedures only used in an implementation section are near
- Heap data and all pointers in general (including class instances) are far
- Global variables are near (DS based)
- Procedure parameters and local variables are near (SS based)
- Procedures declared FAR or EXPORT are far
- Virtual memory tables are far for the new class model and near for the old

This scheme has been used by Borland Pascal for a very long time. I find it flexible and efficient.

Since all public procedures, methods and pointers are 32bit already, Delphi32 won't have to change any of that. It's likely that Delphi32 will switch to 32bit addressing for the data and stack segments too, but that shouldn't affect any of your code either. What will affect it is the change of Integer from 16 to 32 bit.

Linking tables with queries

Q: How do I link tables with queries?

A: If the master table is also a query you can use this code for it:

```
select * from customer
```

The detail table has a couple of things to do. First, you need a couple lines of code.

```
select * from orders
 where custNo = :custNo
```

The `:custNo` part will refer back to whatever the dataSource points to. It should be pointing to the dataSource that points to the master table. In this way, the values are passed in as a parameter.

message handling

Q: If a component doesn't relay the message, do I have to write my own version that passes the messages that I need, or is there way to tap (from a TForm or else where) into the message loop, and grap what I need?

A: To respond to, say, the wm_paint message, you would add --

```
procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
```

to your component. The you could have

```
procedure TWhateverComponent.WMPaint(var Message: TWMPaint);
begin
   {have your way with the component}
end;
```

This will work for any windows message. Most components respond to the more popular messages already, and you can override their event handlers.

Dialing a phone

Dialer is a small non visual component which allows you to dial phone numbers from your Delphi applications. I am not a great expert in communications but it works fine for my modem. You can modify it as much as you wish.

Dialer has four published properties, which will appear in you Object Inspector.

ComPort - Set a communication port of your modem (dpCom1..dpCom4);
Confirm - true if you wish dialer to ask you if you are sure to dial the number;
Method - Dialing method - Pulse or Tone
NumberToDial - string, which contains Phone Number you wish to dial e.g. '911' :)

You can set these properties from Object Inspector or during the run-time.

There is one public procedure: `Execute`

After you add an icon representing dialer, you can use TButton component to run it. e.g.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Dialer1.Execute;
end;
```

You can create the Dialer component "On Fly", without adding its icon to your form:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  TempDialer : TDialer;
begin
  TempDialer:=TDialer.Create(Self);
  with TempDialer do
  begin
    ComPort:=dpCom4;
    Confirm:=true;
    Method:=dmTone;
    NumberToDial:='1(222)333-4444';
    Execute;
    Free;
  end;
end;
```

In this case don't forget to add to your uses statement Dialer unit.

To install this control in you VCL place it in your C:\DELPHI\LIB directory and from IDE Options Menu select Install Components. In the Install Components dialog box click Add Button, then in Add Module box type C:\DELPHI\LIB\DIALER.PAS, click OK, then in the Install Components Dialog box click OK again and wait a while. Dialer icon will appear in the Samples section of your Components Palette.

```
unit Dialer;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs;

type

  TComPort = (dpCOM1,dpCOM2,dpCOM3,dpCOM4);
  TMethod  = (dmTone,dmPulse);

  TDialer = class(TComponent)
  private
    { Private declarations }
    FComPort : TComPort;
    FNumberToDial : string;
    FConfirm : boolean;
    FMethod : TMethod;
  protected
    { Protected declarations }
  public
    { Public declarations }
    procedure Execute;
  published
    property ComPort : TComPort read FComPort
                write FComPort;
    property Confirm : boolean read FConfirm
                write FConfirm;
    property Method  : TMethod read FMethod
                write FMethod;
    property NumberToDial : string read FNumberToDial
                write FNumberToDial;
    { Published declarations }
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Samples', [TDialer]);
end;

procedure TDialer.Execute;
var
  s : string;
  CId : Integer;
  Status : Integer;
  Buf : array[1..32] of Char;
begin
  if FConfirm then
  begin
```

```
   if MessageDlg('About to dial the number '+FNumberToDial+'. Are you sure?',
      mtConfirmation, [mbYes,mbNo], 0)=mrNo then Exit;
  end;
  {Create a string to send to modem}
  s:=Concat('ATDT',FNumberToDial,^M^J);
  if FMethod=dmPulse then s[4]:='P';
  {Open Com Port}
  StrPCopy(@Buf,'COM ');
  Buf[4]:=Chr(49+Ord(FComPort));
  CId:=OpenComm(@Buf,512,512);
  if CId<0 then
  begin
    MessageDlg('Unable to open '+StrPas(@Buf),mtError,
               [mbOk], 0);
    Exit;
  end;
  {Send phone number to modem}
  StrPCopy(@Buf,s);
  Status:=WriteComm(CId,@Buf,StrLen(@Buf));
  if Status>=0 then
  begin
    MessageDlg('Pick up the phone',mtInformation,
               [mbOk], 0);
    WriteComm(CId,'ATH'^M^J,5);
  end
  else
    MessageDlg('Unable to dial number',mtError,
               [mbOk], 0);
  {Close communication port}
  CloseComm(CId);
end;

end.
```

Q: How can I make the active TEdit one color, and every other TEdit a default color?

A: Assign a procedure to the `screen.OnActiveControlChange` event.

```
unit Killer2;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure DoActiveControl(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  OldControl: TComponent;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  screen.OnActiveControlChange := DoActiveControl;
end;

procedure TForm1.DoActiveControl(Sender: TObject);
begin
{This goes first in case the active control is not a TEdit.}
  if assigned(OldControl) then
  begin
    (OldControl as TEdit).color := clWhite;
    (OldControl as TEdit).font.color := clBlack;
  end;
  if activeControl is TEdit then
  begin
    (activeControl as TEdit).color := clNavy;
    (activeControl as TEdit).font.color := clYellow;
    OldControl := activeControl as TEdit;
```

```
  end;
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  screen.OnActiveControlChange := nil; {prevents a GPFault}
end;

end.
```

binary files

Q: How do I read and write binary files?

A: If the data can be stored in its binary form, it's as simple in Delphi as it was in BP7:

```
type
  TUserNotes = record
    TimeStored : TDateTime;
    Comment : string[20];
    TaxCost : real;
    NetCost : real;
    Altered : Boolean;
  end;

TUserNotesFile = file of TUserNotes;
    var
       UserNotes : TUserNotes;
       F: TUserNotesFile;
begin
       System.Assign(F,'MYDATA.DAT');
       Rewrite(F);
       { fill in the fields of "UserNotes" }
       Write(F,UserNotes);
       Close(F);
end;
```

Q: How do I get a file's date and time stamp?

A:

```
function GetFileDate(TheFileName: string): string;
var
  FHandle: integer;
begin
  FHandle := FileOpen(TheFileName, 0);
  result := DateTimeToStr(FileDateToDateTime(FileGetDate(FHandle)));
  FileClose(FHandle);
end;
```

minimizing

When I minimize a form by clicking the minimize button, the WindowState does NOT get set to wsMinimized! So I had to work around the problem by setting WindowState like so:

```
procedure TForm1.SysMen(var msg:TMessage);
begin
 if msg.wParam = SC_MINIMIZE then
   WindowState := wsMinimized; {...or whatever code you want.}
 else
   Inherited;
end;
```

Reason: When you select the apps minimize, you are minimizing the TApplication, not the TMainForm. TMainForm will be hidden, and TApplications Icon will be shown.

constructors

Q: Can I give a constructor any name?

A: Constructors in C++ do have names. It just so happens that the constructor names are all the same, and equal to the class name. In Object Pascal, constructors do have names, and they can be different from the class name. Furthermore, each constructor must have different name. Typically, constructors in Delphi are called Create. To invoke a constructor, you just call it. For example,

```
var
  LB : TListBox;
begin
  LB := TListBox.Create( ... );
```

Q: How do I invoke a named constructor?

Q: What are the arguments of a constructor?

A: Anything you want them to be. Of course, if your creating a descendant class and using virtual constructors, then you don't have much choice in the parameters, they must be the same as the ancestor.

Q: Why does the example of constructors and destructors for the class TShape also define TObject.Free?

A: Yes, this is confusing. This is only showing that when an object calls its Free method, the Destroy destructor eventually gets called. Could use a comment in the example code <g>.

Q: Are destructors named as well? If so, how do I invoke them?

A: Yes, destructors are named. In Delphi, they are typically called Destroy. As stated above, whenever you need to destroy an object, it is safer to call the Free method. The rule is that if you create the object in code, then you are responsible for destroying (i.e. Freeing) it. If you create a component using the form designer, then the Form handles destroying the components.

Also relevant information:
How do I close a file that was opened in a DLL (Delphi made) and called from VB?

VB to DLL

Q: How Do I pass the following struct from VB to a Delphi created DLL **by reference** would be appreciated (what should the parameters on the Delphi side look like?). I chose this structure to sort of represent most scenarios (if I can successfully pass this one, anything should be OK)

```
Type MyType
   Value1 As Double
   Value2 As Integer
   Value3 As String * 20
   Value4 As String
End Type
```

A: You will have some trouble with Value4. Passing VB strings to/from DLLs takes special handling. The easiest way to do it is to pass a pre-declared VB string to the DLL and have the DLL return the length and the resulting string back.

The others are easy.

```
  Value1 As Double      ->  Value1 : Longint;
  Value2 AS Integer     ->  Value2 : integer;
  Value3 As String * 20 ->  Value3 : array[0..19] of char;
```

User defined structures in Pascal are called records.

```
  type
    myStructure = record
      Value1 : longint;
      Value2 : integer;
      Value3 : array [0..19] of char;
    end;
```

I think you could define Value4 as a PChar in Pascal, but when you got it back in VB, you would have to search it for the ASCII 0 end byte and change the length of your string to the number of characters before the zero.

If I remeber correctly, from my evaluation 'beta' copy with source code, there were some functions in the VCL that handle VB strings. Can't say if they are in the shipping version because I havn't recieved the VCL source yet. A quick search of the online docs don't reveal them.

Here's some general type conversions from VB to Pascal (in table form; I hope this formats correctly on CIS.):

```
 VB Declare As         VB Call With                Translates to Pascal Type
 -------------         ------------                -------------------------
 By Val S As String    Any String or Variant       PChar
 I As Integer          Any Integer                 ^Integer (pointer to
integer)
 L As Long             Any Long                    ^Longint (pointer to
longint)
 S As Rect             Any Variable of same type   ^TRect   (pointer to TRect)
 By Val As Integer     Any Integer                 BOOL (word boolean)
  "   "   "                                        Word
  "   "   "                                        Integer
  "   "   "                                        hWnd
  "   "   "                                        hDC
  "   "   "                                        ...and so on; all word
types.
 By Val As Long        Any Long                    Longint
 I As Integer          the first element of I(0)   ^array of integer
 As Any                Any Variable (By Val when
                       String)                     Pointer (PChar when string)
 As Any                By Val 0&                   nil
```

Q: This would probably work, but I'd still really like to know how to return a string (even a null-terminated one) from a DLL function to VB.

A: The problem with returning PChars from DLLs is that the DLL has to be responsible for cleaning up the memory, but it doesn't know when the caller finishes. Most people use the following style, which is NOT bullet proof, but works in most instances:

```
Var ReturnBuffer :Array [0..255] of Char;  { Must be outside function! }

function Dir_Get(ACaption, AInitDir, DirText: PChar ) : PChar;
Begin
  ...
  Result := StrPCopy(ReturnBuffer, 'Result Text');
End;
```

The only time you have a problem is when someone calls the function while someone else is still using the ReturnBuffer. (Just about guaranteed not to happen, except maybe under Windows 95, but could under 3.1 if VB doesn't copy the string).

rTrim()

```
{Supresses trailing blanks}
function RTrim(s: string): string;
begin
  while s[length(s)] = ' ' do dec(s[0]);
  result := s;
end;
```

overriding events

I have a form on which I want to place about 40 or 50 images. However, I want to create these images at run-time and I want to place them on the form at run-time. So for each of these images, I want to put up a messagebox (or something of that sort) whenever the user clicks on one of the images. I guess the crux of the matter is that I want to do all of this at run-time and not a design-time. So, I am confused on how to override the OnClick event of Timage for my new image type, TMyImage.

```
TMyImage = class(TImage)
   procedure OnClick(Sender: TObject); override; { When I put override here,
it gives me an error }
   private
   public
end;

procedure TMyImage.OnClick(Sender: TObject); override; {Once again, an
error }
```

And, of course, it doesn't work even if I replace override with virtual or even if I don't put anything there.

bitmap motion

The basic way to move an image across a background image is this. First, you create a monochrome bitmap the same size as the moving image, with the black silhouette of the e.g. airplane against a white background - call this the mask. You also must create a same-sized bitmap to hold the portion of the background that you'll be overwriting - call it the storage. Then the process of making one movement of the image goes like this:

1) copy storage onto background at its old location
2) copy a rectangle from background to storage at new location
3) copy the image to the background using SRCINVERT
4) copy the mask to the background using SRCAND
5) copy the image to the background again using SRCINVERT

Nasty? Well, perhaps, but that's the way it's done. Copying an image with SRCINVERT XORs the pixels with the background. XOR the same image twice and you return to the original background. But because we ANDed the mask between the two XORs, the black part of the mask contains whatever was in the image with no trace of the background.

Here's the catch. These operations can all be handled using CopyRect, but there's an impossibly annoying flicker of the entire background image. I'm not a big graphics dude; I couldn't figure out how to eliminate the flicker. So it seems you'd have to go down to the metal and use the Windows API function BitBlt. Maybe someone with mreo graphics experience can prove me wrong here.

GetEnvStr

Q: Borland decided that accessing environment variables from Windows Programs is a Bad Thing. Why do they force you to use the "obsolete" WinDos unit?

A: Actually, using WinDos in a Delphi app is a *bad* thing. Many of the functions that are in WinDos have been moved to SysUtils. However, the GetEnvVar function didn't make it. Furthermore, the TDateTime type got moved to the system unit -AND- its definition changed. Therefore, if you use WinDos in a Delphi application, you will not be able to use the new Date and Time functions of Delphi. Therefore, to fix this problem I wrote the following unit which contains a GetEnvStr function.

```
{============= DLPHIDOS.PAS =================}

unit DlphiDos;

interface

  function GetEnvStr( VarName : string ) : string;

implementation

  uses
    WinProcs, SysUtils;

  {============================================================}
  {= GetEnvStr - Get Dos Environment Variable Setting          =}
  {=                                                           =}
  {= This function is a modified version of the GetEnvVar      =}
  {= function that appears in the WinDos unit that comes       =}
  {= with Delphi.  This function's interface uses Pascal       =}
  {= strings instead of null-terminated strings.               =}
  {============================================================}

  function GetEnvStr( VarName : string ) : string;
  var
    Len     : Word;
    EnvStz  : PChar;
    NameStz : array[ 0..180 ] of Char;
  begin
    StrPCopy( NameStz, VarName );     { Covert VarName to PChar }
    Len := StrLen( NameStz );
    EnvStz := GetDosEnvironment;      { EnvStz holds entire env }

    while EnvStz^ <> #0 do
    begin                  { Pick off Variable Name and Compare }
      if ( StrLIComp( EnvStz, NameStz, Len ) = 0 ) and
         ( EnvStz[ Len ] = '=' ) then
      begin          { Convert to Pascal string before returing }
        Result := StrPas( EnvStz + Len + 1 );
        Exit;
      end;
      Inc( EnvStz, StrLen( EnvStz ) + 1 );   { Jump to Next Var }
    end;
```

```
    Result := '';
  end;
end.
```

DLL sample

Without units

First the DLL "framework" that you wanted, save as DLLFRAME.DPR:

```
{--------------------DLLFRAME.DPR-------------------------}
library Dllframe;

uses WinTypes;

function  GetString : string ; export ;

begin
  GetString := 'Hello from the DLL!' ;
end ;

exports
  GetString ;

begin
end.
{---------------------------------------------------------}
```

Now here's the calling program, save it as DLLCALL.DPR:

```
{--------------------DLLCALL.DPR--------------------------}
program Dllcall;

uses
  Dialogs;

{$R *.RES}

function GetString : string ; far ; external 'DLLFRAME' ;

begin
  MessageDlg( GetString, mtInformation, [ mbOK ], 0 ) ;
end.
```

With units

Here's the calling program, save it as DLLCALL.DPR:

```
{--------------------DLLCALL.DPR--------------------------}
program Dllcall;

uses
  Dialogs;

{$R *.RES}

function GetString : string ; far ; external 'DLLFRAME' ;
```

```
begin
  MessageDlg( GetString, mtInformation, [ mbOK ], 0 ) ;
end.
{-------------------------------------------------------------}
```

The DLL "framework" that you wanted, save as DLLFRAME.DPR:

```
{---------------------DLLFRAME.DPR-------------------------}
library Dllframe;

uses DLLUnit;

exports
  GetString;

begin
end.
{-------------------------------------------------------------}
```

The unit we will save as dllunit.pas:

```
{---------------------dllunit.pas-------------------------}
unit DLLUnit;
interface

uses WinTypes;

function GetString: string; export;

implementation

function GetString: string;
begin
  GetString := 'Hello from the DLL!' ;
end ;

begin
end.
```

tStringGrid use

Q: How do I populate a TStringGrid with strings from a file...and save the strings after user editing back to a file?

A:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  F: System.Text;
  S: String;
  I: Integer;
begin
AssignFile(F, 'C:\AUTOEXEC.BAT');
Reset(F);
I := 1;
while not Eof(F) do
  begin
  ReadLn(F, S);
  StringGrid1.Cells[1, I] := S;
  Inc(I);
  end;
CloseFile(F);
AssignFile(F, 'C:\CONFIG.SYS');
Reset(F);
I := 1;
while not Eof(F) do
  begin
  ReadLn(F, S);
  StringGrid1.Cells[2, I] := S;
  Inc(I);
  end;
CloseFile(F);
end;
```

I get an error when I do this:

I use the help file example for the ADD method and I get an error message "comma expected".
I can't run Reportsmith more than 2 or 3 times before it stops running.
I can't open my DBF file because the MDX is missing.
I can only display my ODBC data when it is read only. (PW works OK.)
There is a leak in my GUI resources that comes from TCustomDBGrid.
I get a bitmap pasting error when I do a PasteFromClipboard to a TDBImage.
I get a Capability Not Supported error when passing params in SQL.
I get an "Error creating cursor handle" message.

GPF and error messages

GPF in COMPOBJ.DLL while loading.
I'm getting a GPF when my Delphi app calls a function in a DLL that returns a single.
ERROR in Complib.DCL while trying to initialize BDE
Error while compiling: Can't write EXE file. Disk full(?).
Runs in design mode, but not from program manager.
Runtime error 219
error 94 "." expected
Error $2C09
Error 105

Error #	Error Message
1	Invalid function number
2	File not found
3	Path not found
4	Too many open files
5	File access denied
6	Invalid file handle
12	Invalid file access code
15	Invalid drive number
16	Cannot remove current directory
17	Cannot rename across drives
100	Disk read error
101	Disk write error
102	File not assigned
103	File not open
104	File not open for input
105	File not open for output
106	Invalid numeric format
200	Division by zero
201	Range check error
202	Stack overflow error
203	Heap overflow error

204 Invalid pointer operation
205 Floating point overflow
206 Floating point underflow
207 Invalid floating point operation
210 Object not initialized
211 Call to abstract method
212 Stream registration error
213 Collection index out of range
214 Collection overflow error
215 Arithmetic overflow error
216 General protection fault

Error # Error Message

TDBGrid field focus

Q: How do I set focus on a specific field on a TDBGrid?

A:

```
DBGrid1.SelectedField := Table1Field1;
DBGrid1.SetFocus;
```

Password automation

Q: I have a paradox table that uses a password. How do I make it so that the form that uses the table comes up without prompting the user for the password?

A: The table component's ACTIVE property must be set to FALSE. (If it is active before you have added the password, you will be prompted.) Then, put this code on the form's create event:

```
session.AddPassword('My secret password');
table1.active := true;
```

Once you close the table, you can remove the password with `RemovePassword('My secret password')`, or you can remove all current passwords with `RemoveAllPasswords`. (Note: This is for Paradox tables only.)

Q: I have a form that is a sort of template. I want to be able to create and show the same form several times (with different data in the fields). How do I use the same form several times?

A: You need to make modeless window by calling create and show for each form instance, like this:

```
with TMyForm.create(self) do show;
```

To demonstrate how to use and control these new forms, here is an example that changes the caption and name of each form that is created. You have access to it through the form's component array. This example uses an about box (named "box") as the other form. Also, there is a variable called "TheFormCount" that keeps track of how many times the form is instantiated.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with TBox.create(self) do begin
    Name := 'AboutBox_' + intToStr(TheFormCount);
    caption := 'About Box # ' + intToStr(TheFormCount);
    Show;
  end;
  inc(TheFormCount);
end;
```

These forms can be found and used by their name by means of the FindComponent method used something like this:

```
with Form1.FindComponent('AboutBox_' + IntToStr(Something)) as TForm do
    DoSomethingHere;
```

Screen blanking

Q; How would I write a basic screen saver (like, blank the screen) in Delphi?

A: An easy implementation would be something like this:

```
var
  f: tForm;

begin
  f := tForm.create(self);
  f.WindowState := wsMaximized;
  f.color := black;
  f.borderStyle := bsNone;
  f.show;
end;
```

There is more to do, of course. You must have a way back from there. Perhaps an actual form that has a mouse click event programmed to f.close it. Also, you want to hide the mouse. Etc, etc. But, this answers the question.

memo field value insertion

Q: This question concerns how to transfer the text in a TMemo component on a form to a TMemofield in a Paradox table. In the application in question, there is a TMemo field on the form and, when a Paradox record is read from the table, shows the contents of the memo field on the form by doing a TMemofield.lines.assign(TMemo). This works fine to show the memo field on the form. Then, the user changes the text and we want to move the changed text back to the TMemofield. You can't just do a TMemofield := TMemo, like you can in Objectpal, and I've tried every combination I can think of and it either generates a syntax error or runtime error. I don't want to use a db-aware component because of the way the application is designed.

A: This is not necessarily so very easy. I made an example for you.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  t: TTable;
begin
t := TTable.create(self);
with t do
begin
  DatabaseName := 'MyAlias'; {personal alias}
  TableName := 'MyTbl.db';
  open;
  edit;
  insert;
  fieldByName('TheField').assign(memo1.lines); {This is it!}
  post; {required!!!}
  close;
end;  { End of the with statement. }
end;
```

Q: How do I handle TEdit text with windows messages only?

A:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  p: pChar;
  i: integer;
begin
  i := edit1.perform(wm_GetTextLength, 0, 0) + 1;
  p := AllocMem(i);
  edit1.perform(wm_GetText, i, longint(p));
  edit2.perform(wm_SetText, 0, longint(p));
  FreeMem(p, i);
end;
```

VCL and components Help

Topics
Specific components (by name)
Dynamic components
general component information
custom components
using components

Controlling components

How do you control a scroll bar manually?
What is the easiest way to change the control menu of a form based application ?
How do I hide the caption bar?

Property Editors

Gray Paper - Delphi Property Editors for Beginners
How do I make a component that uses the built in editor for a TStrings property?

created on the fly

Q. "How can VCL components be created on the fly at run-time?"

A. * The following code will create a modal password form at runtime. The TPasswordForm type is assumed to be created already in a seperate unit.

```
with TPasswordForm.Create(Application) do
begin  ( i.e TForm1, TPasswordForm etc. }
  ShowModal;
  Free;
end;
```

* The following are the general steps to add a component to a form at run-time:

1. Create an instance variable of the component type that you wish to create {i.e. TButton }. Note: instance variables are used to point to an actual instance of an object. They are not objects themselves.
2. Use the component's Create constructor method to create an instance of the component and assign the instance reference to the instance variable created in step 1.
3. Assign a parent to the component's Parent property (i.e. Form1, Panel1, etc)
   property.
4. Set any other properties that are necessary (i.e. Width, Height).
5. Finally, to make the component appear on the form by setting the component's Visible property to True.
6. When done with the component make sure the component's Free method is called.

The following demonstrates how to add a TButton component to the current form at run-time:

```
var
  TempButton : TButton;  { This is only a pointer to a TButton }
begin
  TempButton := TButton.Create(Self); { Self refers to the form }
  TempButton.parent := Self;          { Must assign the Parent }
  TempButton.Caption := 'Run-time';   { Assign properties now }
  TempButton.Visible := True;         { Show to button }
end;
```

Note: The TempButton's Free method must be called before or during the time that the form gets closed.

* Creating components using RTTI (Run time Type Information)

With Delphi's run time type information (RTTI), an object can easily create another instance of itself by calling the NewInstance method or using it's ClassType method which returns it's class and then using that to call the Create constructor:

```
NewObject := ExistingObject.ClassType.Create(...) ;
```

The DynaInst project in the \Delphi\Demos\DynaInst directory demonstrates how to create controls at runtime using RTTI.

popup menu

Q. How can the component that was right clicked be determined while in an event handler of a popup MenuItem?

A. Use the PopupComponent property of the PopupMenu component to determine what control was right clicked.

```
procedure TForm1.PopupItem1Click(Sender: TObject);
begin
  Label1.Caption := PopupMenu1.PopupComponent.ClassName;
end;
```

The form's ActiveControl property can also be used, however, the active control may not necessarily be the control that caused the popup menu to appear.

control specs

Q. "What limits are known of the standard Delphi controls?"

A. There can be 16368 tabs in a Tab Control.
There can be 5440 Items in a ComboBox.
There can be 5440 Items in a ListBox.
There can be 570 pages in a NoteBook control.
There can be 16368 palette pages in the Delphi Component Palette.

Note: Going beyond these limits may cause serious errors.

font size in pixels

Q. How can I determine the Length in pixels of a string after a specific font has been applied to it?

A. The two methods, TextHeigh and TextWidth, can be used to determine both the text height and width of a string in pixels. These methods can only be accessed through components that have a Canvas property such as TForm. The TPanel component does not have access to its Canvas property by default because it is protected.

If a component doesn't have a Canvas property then The following function will return the text width based on the font passed.

```
  function GetTextWidth(CanvasOWner: TForm; Text : String; TextFont :
TFont): Integer;
  var
     OldFont : TFont;
  begin
    OldFont := TFont.Create;
    try
      OldFont.Assign( CanvasOWner.Font );
      CanvasOWner.Font.Assign( TextFont );
      Result := CanvasOWner.Canvas.TextWidth(Text);
      CanvasOWner.Font.Assign( OldFont );
    finally
      OldFont.Free;
    end;
  end;
```

message handling (early)

Q. How can I get messages before my application's window procedure is called?

A. the following project source demonstrates how to get Window messages before the application's window procedure is called. It is rare, if ever, that this needs to be done. In most cases assigning a procedure to the Application.OnMessage will accomplish the same thing.

```
program Project1;

uses
Forms, messages, wintypes, winprocs,
Unit1 in 'UNIT1.PAS' {Form1};

{$R *.RES}

var
  OldWndProc: TFarProc;

function NewWndProc(hWndAppl: HWnd; Msg, wParam: Word;
                    lParam: Longint): Longint; export;
begin
  NewWndProc := 0; { Default WndProc return value }

  { * * * Handle messages here; The message number is in Msg * * * }

  NewWndProc := CallWindowProc(OldWndProc, hWndAppl, Msg,
                wParam, lParam);
end;

begin
  Application.CreateForm(TForm1, Form1);
  OldWndProc := TFarProc(GetWindowLong(Application.Handle,
                GWL_WNDPROC));
  SetWindowLong(Application.Handle, GWL_WNDPROC,
                longint(@NewWndProc));
  Application.Run;
end.
```

listbox with h_scrollbar

Q. How can I get a horizontal scrollbar on a list box?

A. There isn't a property to do this but a message can be sent to a listbox component. For example, the message could be sebt in the form's OnActivate:

```
procedure TForm1.FormActivate(Sender: TObject);
begin
 SendMessage(Listbox1.Handle, LB_SetHorizontalExtent,
             1000, Longint(0));
end;
```

TMemo tabstops

Q. How can the tab stops be set in a TMemo control?

A. To change the tab stops for a multiline edit control (i.e. a TMemo) send the EM_SetTabStops message to the component. The Tabs array indicates where the tab stops will be located. Since the WParam parameter to SendMessage is 1, then all tab stops will be set to the value passed in the Tabs array. Remember to set the WantTabs property of TMemo to True to enable the tabs.

```
procedure TForm1.FormCreate( Sender : TObject );
const
   Tabs : array[ 1..1 ] of Integer = ( 10 );
begin
  SendMessage( Memo1.Handle, EM_SetTabStops, 1, Longint( @Tabs ) );
end;
```

splash screens

Q. Where is the best place to open a splash screen on program start up?

A. The best place to open a splash screen is in the project source file after the first FormCreate and before the Run This is accomplished by creating a form on the fly and then displaying it before the app is actual opened.

```
 program Project1;

uses Forms,  Unit1 in 'UNIT1.PAS' {Form1}, Splash;

{$R *.RES}
var
  SplashScreen : TSplashScreen;  {in the Splash unit}
begin
try
SplashScreen := TSplashScreen.Create(Application);
  SplashScreen.Show;
  SplashScreen.update; {To paint the splash screen}
  Application.CreateForm(TForm1, Form1);
  {
   do other CreatForms or any other processing
   before the app is to be opened
  }
 SplashScreen.Close;
finally  {Make sure the splash screen gets released}
  SplashScreen.Free;
end;
Application.Run;

end.
```

If the name of an included .RES file is the same as the name of a .DPR file Delphi wll overwrite it with it's own .RES file.

scrollbar control

Q. How can you do scrolling functions in a TForm component using keyboard commands? For example, scrolling up and down when a PgUp or PgDown is pressed. Is there some simple way to do this or does it have to be programmed by capturing the keystrokes and manually responding to them?

A. Form scrolling is accomplished by modifying the VertScrollbar or HorzScrollbar Postion properties of the form. The following code demonstrates how to do this:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
const
  PageDelta = 10;
begin
  With VertScrollbar do
    if Key = VK_NEXT then  Position := Position+PageDelta
    else if Key = VK_PRIOR then Position := Position-PageDelta;
end;
```

To fill multiple lines of a TString object use the SetText property. The null terminated string is a concatination of each line of text delimeted by a carriage return character #13 in between For example the following statment:

```
Listbox1.Items.SetText('aaaaa'#13'bbbbb'#13'ccccc')
```

will display the following in a listbox window:

```
aaaaa
bbbbb
ccccc
```

Control array emulation

Q. "Is it possible to create something akin to the control array in VB? For example, I want an group of buttons with a common event handler whereby the event handler picks up an integer value for the particular button. In VB this would be done via the control array index."

A. One way do to this is to set the Tag field for each button to a different number and then create a common OnClick event handler that looks at the Sender's (as a TButton) Tag field. Assign the same OnClick event handler to all the buttons in the group. The OnClick event handler would look something like this:

```
procedure TForm1.Button1Click(Sender: TObject); var cap: string;
begin
  case TButton(sender).Tag of
    1: ShowMessage('1st Button Pressed');
    2: ShowMessage('2nd Button Pressed');
    3: ShowMessage('3rd Button Pressed');
  end;
end;
```

Q: I spend alot of time checking datatypes so that accessing the same property of different objects does GPF.

The code tends to look like this:

```
fldname:='';
fld:=Components[I];
if (Components[I] is TDBedit) then
   fldname:=TDBEdit(fld).DataField
else if (Components[I] is TDBLookupList) then
   fldname:=TDBlookupList(fld).DataField
else if (Components[I] is TDBLookupCombo)then
   fldname:=TDBlookupCombo(fld).DataField
else if (Components[I] is TDBListBox) then
   fldname:=TDBListBox(fld).DataField
else if (Components[I] is TDBComboBox) then
   fldname:=TDBComboBOx(fld).DataField
else if (Components[I] is TDBCheckBox) then
   fldname:=TDBCheckBox(fld).DataField
else if (Components[I] is TDBRadioGroup) then
   fldname:=TDBRadioGroup(fld).DataField
else if (Components[I] is TDBMemo) then
   fldname:=TDBMemo(fld).DataField;
```

Is there a way to use the classtype property in a case statement?
Would that be better?

A: No and yes. I suggest that you declare constants that represent each class.

```
const
  val_TEdit = 1;
  val_TButton = 2;
{etc, etc}
```

Then, set the tag of each object to equal the object's constant. Then the case statement is easy.

```
case (sender as tComponent).tag of
{ . . . }
end;
```

Q: How can I verify if a string is a valid date?

```
function ValidDate(const s: String): Boolean;
begin
  Result := True;
  try
    StrToDate(S);
  except
    ON EConvertError DO Result := False;
  end;
end;
```

case use with objects

Q: How do I use a case statement to determine which object calls the procedure?

A: Use the object's TAG property.

```
case (sender as tButton).tag of
  0:  blah;
  1:  blah_blah;
end;
```

This is an invalid typecast error. Look for an AS statement that is wrong.

Q: How do I put the current time on the title bar of my form?

A: Note: The placement of the time varies according to whether it is Win95 or below, as well as the form's size. If the form is too narrow, this may write the time over the top of the control buttons (maximize, minimize, etc).

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls;

type
  TForm1 = class(TForm)
    Timer1: TTimer;
    procedure Timer1Timer(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  dc: hDC;

implementation

{$R *.DFM}

procedure TForm1.Timer1Timer(Sender: TObject);
var
  TheTime: array[0..80] of char;
begin
  StrPCopy(TheTime, TimeToStr(time));
  TextOut(dc, width DIV 2, 5, TheTime, StrLen(TheTime));
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  dc := GetWindowDC(handle);
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  ReleaseDC(handle, dc); {This *must* be manually released.}
end;

end.
```

Q: How do I use my own bitmap on the toolbar?

A: First, make a .DCR file (component resource) that has a bitmap (24 x 24) using the image editor. Name the bitmap the same as the class name of your component. ALL CAPS!!!

The DCR filename must match the PAS filename.
The bitmap's name must match the components classname (e.g. TFOO).

exponent function

Q: I am migrating from VB. Where is the exponent function?

A: There isn't one, but it is simple to make one using Ln(). Ln() gives us the natural logarithm of a number. Using that, we can get the exponentiation easily. E.g. If we want X^Y (or X to the Y power) we would write the code like this:

```
ExpXY := Exp(Ln(X) * Y);
```

To use a generic function, declare the formal parameters and function result as Extended, and the conversions from the actual parameters and back to your result variable will be done automatically. (Ln is the natural logarithm of a number).

```
{ Reproduced from post by Dr. Bob }
  function SwartPower(X: LongInt; N: Word): LongInt;
  var R: LongInt;
  begin
    if N = 0 then SwartPower := 1
    else
    begin
      if N = 1 then SwartPower := X
      else
      begin
        if Odd(N) then
        begin
          R := X;
          Dec(N)
        end
        else R := 1;

        while N > 1 do { inner loop O(log N) }
        begin
          if Odd(N) then R := R * X;
          X := X * X;
          N := N div 2
        end;

        if R > 1 then
          SwartPower := R * X
        else { save last multiplication with 1 }
          SwartPower := X
      end
    end
  end {SwartPower};
```

control menu changing

Q: What is the easiest way to change the control menu of a form based application ?

Example, I was to add an option called "Always on top" that can be checked and unchecked at the user's will. This of course, should be there in addition to the "Move", "Restore", "Maximize" etc. other control menu options.

A: type

```
  TForm1 = class(TForm);
...
  private
    procedure WMSysCommand(VAR Message: TWMSysCommand);
      message WM_SYSCOMMAND;
...

procedure TForm1.WMSysCommand(var Message: TWMSysCommand);
begin
  Inherited;
  IF Mesage.CmdType AND $FFF0 = $F200 THEN
    MessageBeep(0); {replace with code to DO something}
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  AppendMenu(GetSystemMenu(Handle, False), MF_STRING,
    $F200, '&Wahoo!');
end;
```

This adds a new item named "Wahoo!" to the system menu, with a command ID of $F200. I picked $F200 at random as a number near the regular SC_xxxx constants, but greater than any of them. Note that the ID for a system command must be evenly divisibly by 16, as Windows uses the lowest 4 bits of the ID. That's also why you AND the value with $FFF0 before comparing it to the specific ID.

Put this on the MouseDown method (not the MouseClick):

```
if button = mbRight then doSomething;
```

text tables (ASCII)

Q: How do I make an ASCII text table from a paradox table?

A:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  t1, t2: tTable;  {t1 = PW table; t2 = ASCII version}
begin
  t1 := tTable.create(self);
  with t1 do begin
    DataBaseName := 'DBDEMOS';
    tableName := 'customer.db';
    open;
  end;
  t2 := tTable.create(self);
  with t2 do begin
    DataBaseName := 'DBDEMOS';
    tableName := 'myfile.txt';
    TableType := ttASCII;
    createTable;
    open;
    edit;
    BatchMove(t1, batCopy);
    close;
  end;
  t1.close;
end;
```

Use LockWindowUpdate(form1.handle) to stop it. LockWindowUpdate(0) will reset it as there can be only one window locked at a time.

Group creation with DDE

Q: How do I initialte a DDE link to the program manager to create a new group?

A: Here it is from c:\delphi\demos\dde\ddedemo.dpr:

```
var
  Name: string;
  Macro: string;
  Cmd: array[0..255] of Char;
begin
  if GroupName.Text = '' then {an edit field on the form}
    MessageDlg('Group name can not be blank.', mtError, [mbOK], 0)
  else
  begin
    Name := GroupName.Text;
    Macro := Format('[CreateGroup(%s)]', [Name]) + #13#10;
    StrPCopy (Cmd, Macro);
    DDEClient.OpenLink;
    if not DDEClient.ExecuteMacro(Cmd, False) then
      MessageDlg('Unable to create group.', mtInformation, [mbOK], 0);
    DDEClient.CloseLink;
    GroupName.SelectAll;
  end;
end;
```

formatting a data field

Q: How do I set a format for a data field?

A:
1. Select the table object, and double click.

2. Select the field that you want to format.

3. Use the DisplayFormat and the EditFormat properties to do what you want.
DisplayFormat works for when the field does not have focus.
EditFormat works for when the field has focus.

Use the commands as you would for the first parameter of the FormatFloat function, but without the quotes.

uses sequence error

Q: Does anyone know about a syntax problem with the ordering of the USES statement?

```
program CrtApp;

  Program CrtApp;

  (*uses                  {This order works!}
  U_X87_0a in 'G:\HS_LIB\UTILITY\U_X87_0a.PAS',
  WinCrt; *)

  uses WinCrt,        {This order does not work!}
  U_X87_0a in 'G:\HS_LIB\UTILITY\U_X87_0A.PAS';

  begin   Writeln('Delphi');   end.
```

A: Known problem. Grab the updated DLIB utility from the libraries and add SYSUTILS and TOOLHELP to DELPHI.DSL (or extract and remove WINCRT).

pointers and assignments

Q: if I write "Font := Canvas.Font" Font is not a pointer to Canvas.Font, Font takes on the properties of Canvas.Font. I want to know how I can make a variable that is a "pointer" to Canvas.Font.

In pointers it would have been:

```
procedure X( T : TType );
var
  vP : ^TType
begin
  vP := @T;
  {both of the following make changes to T}
  T.Size := 10;
  vP^.Size := 12;
end;
```

what I want is:

```
procedure X( canvas : TCanvas );
var
   F : TFont
begin
  F := canvas.Font;  {<-- ?? assigns copy instead of reference}
  {I need both of the following to affect canvas' Font property}
  F.Size := 10;
  canvas.Size := 12;
end;
```

Sorry if it isn't clear ... does anyone remember what a pointer is now that Delphi is here?

A: The problem you are seeing is that in your context the variable Font is not a variable of a class, or an instance of a class. It is a property, and the assignment operator (:=) is being translated by the compiler into an Assign type operation by the declaration of that property.

For example this code:

```
  var
    MyFontPtr : TFont;
  begin
    ...
    MyFontPtr := SomeCanvas.Font;
```

does result in an address being copied. Whereas this code:

```
  begin
    ...
    with Form1.Canvas do
        Font := SomeCanvas.Font;
```

results in the following code:

```
Form1.Canvas.SetFont(SomeCanvas.Font);
```

which 'copies' the data in SomeCanvas.Font to Form1.Canvas.Font.

See also pointers and dynamic memory

dynamic array sizing

Q: Is there any way to dynamically redimension an array?

A: When you dynamically allocate space for an array, you *can* dynamically allocate the size as needed. e.g.

```
 type
  TIntegerArray = array[0..32767] of integer;  {Types don't allocate memory }
  PIntegerArray = ^TIntegerArray;
 var
  I1,I2 : PIntegerArray;
 begin
  GetMem(I1,500*SizeOf(Integer));  { I1 now points to a 500 element array }
  GetMem(I2,1000*SizeOf(Integer)); { I2 now points to a 1000 element array }
...
```

Ok, now *THIS* kind of variable-sized array has been available in Pascal for many a long year. That is, the kind where you decide the size at run-time but don't change it. Here is an example that works with records: E.g.

```
TYPE
  VarArray = Array[0..65520 DIV SizeOf(MyRecord)] OF MyRecord;
  ptrVarArray = ^VarArray;
VAR
  MyArray : ptrVarArray;
  GetMem(MyArray, NumNeeded*SizeOf(MyRecord));
```

See? You define an array TYPE as large as possible, given the almost-64k limit on the size of a single variable. You define a pointer to that type. And you allocate just enough memory to hold the actual number needed.

FWIW, You can definitely use a TList too, but you'll need to define a simple OBJECT that holds your variant record, because TLists only hold TObjects and their descendants.

****

Here is a later post on C-Serve:

ReAllocMem comes closest. You allocate your array on the heap with some incantations like

```
 Type
   TIntArray = Array [0..High(Word) div Sizeof(Integer) -1] of Integer;
   { declares the maximum size possible for an array of Integers }
   PIntArray = ^TIntArray;

 Procedure AllocArray( Var pArr: PIntArray; items: Word;
```

```
                      Var maxIndex: Word);
  Begin
    If items > 0 Then Begin
      GetMem( pArr, items * Sizeof( Integer ));
      maxIndex := Pred( items );
    End
    Else
      pArr := Nil;
  End;

Procedure ReDimArray( Var pArr: PIntArray; newItems: Word;
                      Var maxIndex: Word );
  Begin
    If pArr = Nil Then
      AllocArray( pArr, newItems, maxIndex )
    Else Begin
      ReAllocMem( pArr, Succ(maxIndex)*Sizeof(Integer),
                  newItems*Sizeof(Integer));
      maxIndex := Pred( newItems );
    End;
  End;

Procedure DisposeArray( Var pArr: PIntArray; maxIndex: Word );
  Begin
    FreeMem( pArr, Succ(maxIndex)*SizeOf(Integer));
  End;

Var
 pMyArray: PIntArray;
 maxIndex, i: Word;

Begin
  try
    AllocArray( pMyArray, 100, maxIndex );
    For i:= 0 To maxIndex Do
      pMyArray^[i] := i;
    ...
    RedimArray( pMyArray, 200, maxIndex );
    For i:= 0 To maxIndex div 2 Do
      pMyArray^[Succ(maxIndex div 2)+i] := Sqr(pMyArray^[i]);
    ....
  finally
    DisposeArray( pMyArray, maxIndex );
  end;
..
```

"YIKES" i hear you say, "do i have to do this kind of gyrations for each array type i might need???". Well, you could, but it is not very difficult to write a set of generic procedures that will work for every base type you might use for an array. We assume that the array type is always declared with a lower bound of 0 and also use Cardinal instead of Word so the procedures will automagically expand to handle > 64K arrays under Delphi32.

```
Procedure AllocArray( Var pArr: Pointer; items, itemsize: Cardinal;
                      Var maxIndex: Cardinal);
  Begin
```

```
    If items > 0 Then Begin
      GetMem( pArr, items * itemsize);
      maxIndex := Pred( items );
    End
    Else Begin
      pArr := Nil;
      maxIndex := 0;  { WARNING! This is still an invalid index here! }
    End;
  End;

Procedure ReDimArray( Var pArr: Pointer; newItems, itemsize: Cardinal;
                      Var maxIndex: Cardinal );
  Begin
    If pArr = Nil Then
      AllocArray( pArr, newItems, itemsize, maxIndex )
    Else Begin
      ReAllocMem( pArr, Succ(maxIndex)*itemsize,
                  newItems*itemsize);
      maxIndex := Pred( newItems );
    End;
  End;

Procedure DisposeArray( Var pArr: Pointer; itemsize, maxIndex: Cardinal );
  Begin
    FreeMem( pArr, Succ(maxIndex)*itemsize);
  End;
```

To use these procedures to make a dynamic array of Double, for example, you would proceed as follows:

```
type
  {we can directly declare a pointer to an array, no need to declare
   the array first}
  PDoubleArray = ^Array [0..High(Cardinal) div Sizeof(Double) -1] of
                  Double;

Var
  pDbl: PDoubleArray;
  maxIndex, i: Cardinal;
  deg2arc: Double;

Begin
  deg2arc := Pi/180.0;
  try
    AllocArray( pDbl, 360, Sizeof( Double ), maxIndex );
    For i:= 0 To maxIndex Do
      pDbl^[i] := Sin( Float(i) * deg2arc );
    ReDimArray( pDlb, 720, Sizeof( Double ), maxIndex );
    For i:= 360 To maxIndex Do
      pDbl^[i] := Cos( Float(i-360) * deg2arc );
  finally
    DisposeArray( pDbl, Sizeof(Double), maxIndex );
  end;
```

And now the final icing: all this was typed off forehead; no idea if it will even compile

<eg>! And a safety net feature is still missing: AllocArray and RedimArray should raise an exception if you try to allocate an array > 64Kbyte under Delphi16. I left this out since i'm not really familiar with these beasties (exceptions) yet. Delphi may do it anyway if range checking is enabled.

Here is Peter Below's approach:

time for my canned reply re. dynamic arrays:

You allocate your array on the heap with some incantations like

```
Type
  TIntArray = Array [0..High(Word) div Sizeof(Integer) -1] of Integer;
  { declares the maximum size possible for an array of Integers }
  PIntArray = ^TIntArray;

Procedure AllocArray( Var pArr: PIntArray; items: Word;
                      Var maxIndex: Word);
  Begin
    If items > 0 Then Begin
      GetMem( pArr, items * Sizeof( Integer ));
      maxIndex := Pred( items );
    End
    Else
      pArr := Nil;
  End;

Procedure ReDimArray( Var pArr: PIntArray; newItems: Word;
                      Var maxIndex: Word );
  Begin
    If pArr = Nil Then
      AllocArray( pArr, newItems, maxIndex )
    Else Begin
      ReAllocMem( pArr, Succ(maxIndex)*Sizeof(Integer),
                  newItems*Sizeof(Integer));
      maxIndex := Pred( newItems );
    End;
  End;

Procedure DisposeArray( Var pArr: PIntArray; maxIndex: Word );
  Begin
    FreeMem( pArr, Succ(maxIndex)*SizeOf(Integer));
  End;

Var
 pMyArray: PIntArray;
 maxIndex, i: Word;

Begin
  try
    AllocArray( pMyArray, 100, maxIndex );
    For i:= 0 To maxIndex Do
      pMyArray^[i] := i;
    ...
```

```
    RedimArray( pMyArray, 200, maxIndex );
    For i:= 0 To maxIndex div 2 Do
      pMyArray^[Succ(maxIndex div 2)+i] := Sqr(pMyArray^[i]);
    ....
  finally
  end;
```

"YIKES" I hear you say, "do I have to do this kind of gyrations for each array type i might need???". Well, you could, but it is not very difficult to write a set of generic procedures that will work for every base type you might use for an array. We assume that the array type is always declared with a lower bound of 0 and also use Cardinal instead of Word so the procedures will automagically expand to handle > 64K arrays under Delphi32.

```
Procedure AllocArray( Var pArr: Pointer; items, itemsize): Card
```

: I want to be able to right click on my VBX and have a popup menu display. How do I trap for that?

A: Here it is:

```
procedure TForm1.FormMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if button = mbRight then
    WITH Sender AS TControl DO
      WITH ClientToScreen(Point(X,Y)) DO
      BEGIN
        PopupMenu1.PopupComponent := TComponent(Sender);
        PopupMenu1.Popup(X,Y);
      END;
end;
```

Note: The form's popupMenu property must be empty, or it will popup from everywhere. If you want the form to be the only place showing the popup, place this method on the form's OnMouseDown event. If you want the VBX to be the only place, then place it on the VBX's OnMouseDown event, etc.

callbacks

Q: I have a unit that does a callback function (from a DLL). In this unit, I have declared a global variable. When I do the callback, it is unable to 'see' the global variable. Both functions are in the same unit. Why is this and what do I do?

A: Turn off smart callbacks and add this:

{$K-}

Because of how things are being passed to the CPU (register manipulation) the scope has changed through a callback.

cursor loading

Q: How do I use one of the cursor files in the c:\delphi\images\cursors?

A: Load the cursor into a RES file. (The image editor will let you do that.) Then do this:

```
{$R c:\programs\delphi\MyFile.res}

const PutTheCursorHere_Dude = 1; {arbitrary number}

procedure stuff;
begin
  screen.cursors[PutTheCursorHere_Dude] := LoadCursor(hInstance,
pChar('cursor_1'));
  screen.cursor := PutTheCursorHere_Dude;
end;
```

ADD (syntax error)

Q: I use the help file example for the ADD method and I get an error message "comma expected". Why?

A: The ADD method requires an undocumented fourth parameter. (OOPS!) It says whether the field is required. (true = required)

Here is the working code for that example:

```
var
  t: tTable;
begin
t := tTable.create(self);
with t do
  begin
  Active := False;
  DatabaseName := 'lwl';
  TableName := 'hoser';
  TableType := ttParadox;
  with FieldDefs do
    begin
    Clear;
    Add('Field1', ftInteger, 0, false);
    Add('Field2', ftInteger, 0, false);
    end;
  with IndexDefs do
    begin
    Clear;
    Add('Field1Index', 'Field1', [ixPrimary, ixUnique]);
    end;
  CreateTable;
  end;
end;
```

packing a dBASE table

Q: How do I pack a dBASE table?

A: To pack a dBASE table that has been opened with a TTable use the BDE function DBIPackTable. There are two basic steps to do this:

1. Add the following units to our uses clause: DBITYPES, DBIPROCS and DBIERRS.

2) Then call the BDE function as follows:

```
DBIPackTable(Table1.DbHandle, Table1.Handle, 'TABLENAME.DBF', szDBASE, TRUE);
```

Note: The table must be opened in exclusive mode.

Sybase: Updated tables

In order to create an updatable Sybase SQL query using the TQuery component you need to specify the name of the table's owner if it isn't yourself.
e.g. If you wish to `set RequestLive = TRUE` (and Active) your SQL statement should read something like this:

```
Select * from 'myname.tablename'    {notice the quotes}
```

Using a SQL statement like this:
```
Select * from tablename
```
returns errors because no owner was specified. Additionally you MUST include quotes when specifying the owner name or the database engine will complain. Other then that, it works like a charm!

General SQL:

How can I reference a field name with a space in a query?
How do I pass a variable to a query?
How do I display a record that has the table's MAX value?
How can I use the aggregate functions (avg, sum, count, max, min) with a table?
I get an "Error creating cursor handle" message.
How do I get the result from a TStoredProc?

Sybase:

Updated tables
Error Message: 'connection is in use by another statement'.

Interbase:

How can I make it run faster?

Optimizing code

Original code:

```
if ActiveControl is TDBEdit then
  (ActiveControl as TDBEdit).CutToClipboard
else if ActiveControl is TDBMemo then
  (ActiveControl as TDBMemo).CutToClipboard;
```

New and improved code:

```
if ActiveControl is TCustomMemo then
  TCustomMemo(ActiveControl).CutToClipboard;
```

This works the same as what you had with less overhead. TCustomMemo is the ancestor to both TMemo and TDBMemo, so the typecast works fine. Also, using "is" and "as" in the same statement is redundant and expensive.

Q: I have a table grid, edit field, and a pushbutton. When I do a FindNearest from the pushbutton

```
Table1.FindNearest([edit1.text]);
```

it only shows the actual data if I write in the whole value to be found. If I enter only partial values, the scroll bar shows that a record was found, but it doesn't display. How do I get it to display?

A: The answer is that it DOES display. Because of the way that the value is evaluated, you should pad the string with zeros to the size of the (in this case) CustNo. (CustNo is a number field. If you enter a partial value, it is seen as a lower value number.)

Make sure that you say `report.InitialValues.clear` before you call the report.

Q: How do I create complex indices on dBASE tables in Delphi?

A: Compound indices must be expressions, as follows:

```
MyTable.AddIndex('MyTagName', 'FIELD1NAME+FIELD2NAME', [ixExpression]);
```

This is required on compound indices. You can even use other expressions, such as `UPPER(FIELD1+FIELD2)`. Note that `[ixExpression]` is required, but NOT documented in any help file I can find, and `[ixCaseInsensitive]` plain doesn't work for dBase files. Note further that such expression indices rule out using SetRange, and perhaps other methods. However, I've heard that using the "long winded" version of SetRange will work.

e.g.

```
with table1 do
begin
  SetRangeStart;
  FieldsByName('FIELD1').AsString := 'myString';
  FieldsByName('FIELD2').AsString := '  1';
  SetRangeEnd;
  FieldsByName('FIELD1').AsString := 'myString';
  FieldsByName('FIELD2').AsString := '999';
  ApplyRange;
end;
```

Q: What is the Object Pascal equivalent of C's "union" reserved word?

A: It is a variant record in Delphi. Variant records allow for different types to occupy the same space (but not at the same time, of course). The space allocated will be as large as the largest possible variation, but there need be no needless duplication of memory allocated. This is how a variant record is declared:

```
purchase = record
  amount: real;
  {more stuff}
  case MethodOfPayment of
    check:     (check number: integer;
                checkAmt: real;
                LicenseNumber: string[20]);
    CreditCard:(card: CardType;
                ExpMonth: 1..12;
                ExpYear: YearType);
  end;
end;
```

Unions in Delphi/Pascal are not a reserved word as much as an operator. Here is how a union (Pascal style) is done:

```
var
  a, b, c: set of char;

begin
  a := ['a'..'z'];
  b := ['a', 'c', 'e', 'g'];
  c := ['a'..'d', 'z'];

  if a - b = ['b', 'd', 'f', 'h'..'z'] then
    blah; {difference}

  if b + c = ['a'..'d', 'e', 'g', 'z'] then
    blah; {union}

  if b * c = ['a', 'c'] then
    blah; {intersection}
end;
```

```
SubString('Lloyd is the greatest!!!' from 1 to 5)
```

TMemo

How can the tab stops be set in a TMemo control?
How can I tell the length in bytes of a memo field?
How can I get to the bottom of a memo field with code?
How do I fill a memo field from a table?
How do I fill a TMemo from a PChar?
How do I know which line number I am currently on in a TMemo?

Error code 105 means "file not open for output". You should add the WinCRT unit to your uses list. (This is mentioned in MANUALS.TXT in your \DELPHI directory).

Q: How do I execute a program and have my code wait until it is finished?

A:

```
uses Wintypes,WinProcs,Toolhelp,Classes,Forms;

Function WinExecAndWait(Path : string; Visibility : word) : word;
var
  InstanceID : THandle;
  PathLen : integer;
begin
  { inplace conversion of a String to a PChar }
  PathLen := Length(Path);
  Move(Path[1],Path[0],PathLen);
  Path[PathLen] := #00;
  { Try to run the application }
  InstanceID := WinExec(@Path,Visibility);
  if InstanceID < 32 then { a value less than 32 indicates an Exec error }
     WinExecAndWait := InstanceID
  else begin
    Repeat
      Application.ProcessMessages;
    until Application.Terminated or (GetModuleUsage(InstanceID) = 0);
    WinExecAndWait := 32;
  end;
end;
```

MDX and DBF files

Q: I have a DBF table that I can't open because it is expecting an MDX (production index) file. I lost (deleted) that file. What do I do now?

A: Byte 28 in the DBF file header indicates whether an MDX file is needed. (0 = no MDX needed; 1 = MDX needed.) Here is some code that makes the change for you. It uses a radio group where the first item is NO MDX (element 0), and the second item is USES MDX (element 1). This app also uses the TOpenDialog component.

```
procedure TForm1.Button1Click(Sender: TObject);
var
  f: file of byte;
  b: byte;
begin
  if OpenDialog1.execute then
  begin
    assignFile(f, OpenDialog1.FileName);
    reset(f);
    seek(f, 28);
    b := RadioGroup1.ItemIndex; {0 = no MDX; 1 = use MDX}
    write(f, b);
    closefile(f);
  end;
  OpenDialog1.FileName := '*.dbf'; {reset the filter}
end;
```

Q: How do I do pointer arithmetic in Delphi?

A: If you are doing dynamic memory allocation, it is done like this:

```
uses WinCRT;

procedure TForm1.Button1Click(Sender: TObject);
var
  MyArray: array[0..30] of char;
  b: ^char;
  i: integer;
begin
  StrCopy(MyArray, 'Lloyd is the greatest!!!');
  b := @MyArray;
  for i := StrLen(MyArray) downto 0 do
  begin
    write(b^);
    inc(b);
  end;
end;
```

The amount by which the pointer is incremented is the correct size. (i.e. It is the size of the object pointed to.) The following code proves that.

```
var
  P1, P2 : ^LongInt;
  L : LongInt;
begin
  P1 := @L;
  P2 := @L;
  Inc(P2);
  L := Ofs(P2^) - Ofs(P1^); {L = 4; sizeof(longInt)}
end;
```

Q: I try to use ODBC, but when I set the table to active, it won't let me. I can display the data if it is in read only mode. I can edit this information correctly if I use paradox for windows 5.0. What gives?

A: You have separate installs of PW and Delphi that point to different IDAPI directories. Have them point to the same place.

Q: How can I tell if share is loaded from Delphi?

A:

```
function IsShareLoaded: Boolean;
var
  f: file of word;
  data: word;
  IsShareInstalled: Boolean;
begin
  assign(f, 'im_here.not');
  rewrite(f);
  write(f, data);
  asm
    mov IsShareInstalled, true
    mov bx, TFileRec(f).handle
    xor cx, cx
    xor dx, dx
    mov si, 0
    mov di, 2
    mov al, 0
    mov ah, $5C
    int $21
    jc  @@NoError
    dec IsShareInstalled
    @@NoError:
  end; {asm section}

  result := IsShareInstalled;
  close(f);
  erase(f);
end;
```

```
function IntToBinaryStr(TheVal: LongInt): string;
var
  counter: LongInt;
begin
{This part is here because we remove leading zeros.   That
means that a zero value would return an empty string.}
  if TheVal = 0 then begin
    result := '0';
    exit;
  end;

  result := '';
  counter := $80000000;

  {Suppress leading zeros}
  while   ((counter and TheVal) = 0) do begin
    counter := counter shr 1;
    if (counter = 0) then break; {We found our first "1".}
  end;

  while counter > 0 do begin
    if (counter and TheVal) = 0 then result := result + '0'
    else   result := result + '1';
    counter := counter shr 1;
  end;
end;
```

The equivalent is a query:
SELECT MAX(Amount) FROM ORDERS

Q: How do I create a floating palette window?

A: The tricky thing is to make sure the palette window always appears "in front" of the main window, but not necessarily "on top" of all other windows. This effect can be achieved by overriding the CreateParams method for your palette form. For example,

```
procedure TForm2.CreateParams( var Params: TCreateParams );
begin
  inherited CreateParams( Params );
  with Params do
  begin
    Style := Style or ws_Overlapped;
    WndParent := Form1.Handle;
  end;
end;
```

Q: Why aren't Delphi EXE files as small as BP files? (It always seems to be 150K+.)

A: The difference that you're seeing is the difference in OWL vs. VCL. If BP7 *could* compile a Delphi app (it can't because of language extensions), the executable would be the same size (give or take a hundred bytes) as the Delphi compiled executable. So to answer your question, Delphi already does compile and executable as small as BP7 does.

```
Program ExitWin;
uses WinProcs;
begin
  ExitWindowsExec('', '');
end.
```

will create a 2816 byte executable with the Delphi compiler. Then, if you run W8LOSS.EXE, it will shrink it down to 2463 bytes!!!

Note: This program will exit and restart windows.

****

Here is one that came about as a result of a small competition on compuserve. Here is the essence of the thread:

Hi James,

> The program also iterates through the active tasks looking at module names. If it can go smaller, I'd love to see how but I won't be upset if it stays at 17k. :)

Well, I'm willing to go for half of your size (I love a good challenge before breakfast)...

> This exe makes sure no Office apps are running by looking for specific module names in the task list. If one is found, we throw up that darn message box and quit.

Groetjes,
Dr. "I want to climb the mountain because it's there!" Bob

****

You're on! :)

The great way to learn is to see someone do something just a little bit better. So here you go...

MSOFFICE
VB
WINWORD

EXCEL
MSACCESS
PP4

These are the modules names. Basicall, if one of these is running you got to throw up a message box and then get out. It none are running, you need to ShellExecute another EXE, passing it any command line arguments this program receives.

I'm looking forward to your results.

'I have been to the top of the mountain, and it was good." - Beevis and Butthead <g>

James Foxall

****

Hi James,

You had an executable of 17Kb, right? Well, I've done it in less than 2.5 Kb (in fact, I ended up with 2133 bytes - including on-line help <g>). You'll have to supply the setup routine as command-line arguments, but other than that it'll work just fine.

(It'll probably be somewhat smaller if you hardcode the 'install sequence' but I didn't know what they were, so...)

```
{$A+,B-,D-,F-,G+,I-,K+,L-,N-,P-,Q-,R-,S-,T-,V-,W-,X+,Y-}
{$M 32768,0}
program NoOffice;
{
      File: NOOFFICE.PAS
    Author: Bob Swart
  Compiler: Borland Pascal 7.01
   Purpose: To quit when any MS Office module is loaded,
            otherwise run the command-line arguments...
     Usage: NOOFFICE Setup /X
   Example: NOOFFICE NOTEPAD C:\CONFIG.SYS

 Code Size: 1154 Bytes
 Data Size:  338 Bytes
 .EXE Size: 2560 Bytes (2133 after running W8LOSS.EXE)
}
uses WinTypes, WinProcs;
Const
  Run: String[127] = #0;
Const
  Office : Array[0..5] of PChar =
    ('MSOFFICE','VB','WINWORD','EXCEL','MSACCESS','PP4');
var i: Integer;
begin
  if ParamCount = 0 then
    MessageBox(0,'Usage: NOOFFICE <setup>', Office[0],
```

```
                  MB_ICONINFORMATION OR MB_OK)
  else
  begin
    for i:=0 to 5 do
    begin
      if GetModuleHandle(Office[i]) <> 0 then { found? }
      begin
        MessageBox(0,'MS-Office is still running!',Office[i],
                   MB_ICONHAND OR MB_OK);
        Halt
      end
    end;
    for i:=ParamCount downto 1 do Run := ParamStr(i) + #32 + Run;
    WinExec(@Run[1],SW_SHOW)
  end
end.
```

Q: How do I access ACCESS tables?

A: The ODBC driver provided with Access 2.0 is designed to work only within the Microsoft Office environment. To work with ODBC/Access in Delphi, you need the Microsoft ODBC Desktop Driver kit, part# 273-054-030 available from Microsoft Direct for $10.25US (post on WINEXT for where to get it in your country if you are not in the US). It is also available on the Jan. MSDN, Level 2 (Development Platform) CD4 \ODBC\X86 as part of the ODBC 2.1 SDK. Be aware that your redistribution rights for the Desktop Drivers are pretty restricted by Microsoft. For info on (and objections to) the restrictions post on the WINEXT forum.

You also need the following ODBC files.

Minimum:

ODBC.DLL	03.10.1994, Version 2.00.1510
ODBCINST.DLL	03.10.1994, Version 2.00.1510
ODBCINST.HLP	11.08.1993
ODBCADM.EXE	11.08.1993, Version 1.02.3129

Better:

ODBC.DLL	12.07.1994, Version 2.10.2401
ODBCINST.DLL	12.07.1994, Version 2.10.2401
ODBCINST.HLP	12.07.1994
ODBCADM.EXE	12.07.1994, Version 2.10.2309

The following steps will get you started in Delphi

1. Using the ODBC Administrator, set-up a datasource for your database. Be sure to specify a path to your mdb file. For the purposes of this explanation we'll say that the datasource name is MYDSN.

2. Load the BDE Configuration utility.

3. Select New Driver.

4. Give the driver a name (call it ODBC_MYDSN).

5. In the driver combo box select, "Microsoft Access Driver (*.mdb)

6. In the name combo box select MYDSN

7. Go to the Alias page.

8. Select New Alias.

9. Enter MYDSN for name.

10. For Alias Type, select ODBC_MYDSN.

11. In Delphi, drop a DataSource, Table, and DBGrid on your form.

12. Set DBGrid1.DataSource to DataSource1.

13. Set DataSource1.DataSet to Table1.

14. Set Table1.DatabaseName to MYDSN.

15. In the TableName property in Table1, click the downarrow, you will see the "Login" dialog. Press OK, after a short pause you will see a dropdown list with all your table names. Select one.

16. Set the Active property in Table1 to True, the data from your table will be displayed in the grid.

IF ARCHITECTS HAD TO WORK LIKE PROGRAMMERS...

Dear Mr. Architect:

Please design and build me a house. I am not quite sure of what I need, so you should use your discretion.

My house should have between two and forty-five bedrooms. Just make sure the plans are such that the bedrooms can be easily added or deleted. When you bring the blueprints to me, I will make the final decision of what I want. Also, bring me the cost breakdowns for each configuration so that I can arbitrarily pick one at a later time.

Keep in mind that the house I ultimately choose must cost less than the one I am currently living in. Make sure, however, that you correct all the deficiencies that exist in my current house (the floor of my kitchen
vibrates when I walk across it, and the walls don't have nearly enough insulation in them).

As you design, also keep in mind that I want to keep yearly maintenance costs as low as possible. This should mean the incorporation of extra-cost features like aluminum, vinyl, or composite siding. (If you choose not to specify aluminum, be prepared to explain your decision in detail.)

Please take care that modern design practices and the latest materials are used in construction of the house, as I want it to be a showplace for the most up-to-date ideas and methods. Be alerted, however, that kitchen should be designed to accommodate (among other things) my 1952 Gibson refrigerator.

To assure that you are building the correct house for our entire family, you will need to contact each of my children, and also our in-laws. My mother-in-law will have very strong feelings about how the house should be designed, since she visits us at least once a year. Make sure that you weigh all of thses options carefully and come to the right decision. I, however, retain the right to overrule any decisions that you make.

Please don't bother me with small details right now. Your job is to develop the overall plans for the house and get the big picture. At this time, for example, it is not appropriate to be choosing the color of the carpeting. However, keep in mind that my wife likes blue.

Also, do not worry at this time about acquiring the resources to build the house itself. Your first priority is to develop detailed plans and specifications. Once I approve these plans, however, I would expect the
house to be under roof within 48 hours.

While you are designing this house specifically for me, keep in mind that sooner or later I will have to sell it to someone else. It therefore should have appeal to a wide variety of

potential buyers. Please make sure before you finalize the plans that there is a consensus of the potential homebuyers
in my area that they like the features this house has.

I advise you to run up and look at the house my neighbor build last year, as we like it a great deal. It has many things that we feel we also need in our new home, particularily the 75-foot swimming pool. With careful engineering, I believe that you can design this into our new house without impacting the construction cost.

Please prepare a complete set of blueprints. It is not necessary at this time to do the real design, since they will be used only for construction bids. Be advised, however, that you will be held accountable for any increase of construction costs as a result of later design changes.

You must be thrilled to be working on as an interesting project as this! To be able to use the latest techniques and materials and to be given such freedom in your designs is something that can't happen very often. Contact me as soon as possible with your ideas and completed plans.

PS: My wife has just told me that she disagrees with many of the instructions I've given you in this letter. As architect, it is your responsibility to resolve these differences. I have tried in the past and have been unable to accomplish this. If you can't handle this responsibility, I will have to find another architect.

PPS: Perhaps what I need is not a house at all, but a travel trailer. Please advise me as soon as possible if this is the case.

Q: How do I print a form?

A: Prints all visible TLabel, TEdit, TMemo, TDBText, TDBEdit and TDBMemo components on the form with proper place, size and font. Set the Form Scrollbar.Range to 768 Horz and 1008 Vert for a 8 X 10.5 page at 96formPPI.

```
USES Printers;

procedure TForm1.SpeedButton1Click(Sender: TObject);
var
  C : array[0..255] of char;
  CLen, ScaleX, ScaleY, I : integer;
  Format : Word;  DC : HDC;
  MComp : TMemo;  R: TRect;
begin
  Printer.BeginDoc;
  DC := Printer.Canvas.Handle;
  ScaleX := GetDeviceCaps(DC, LOGPIXELSX) div PixelsPerInch;
  ScaleY := GetDeviceCaps(DC, LOGPIXELSY) div PixelsPerInch;
  for I := 0 to ComponentCount-1 do
    if (Components[I] is TCustomLabel) or (Components[I] is TCustomEdit) then
    begin
      MComp := TMemo(Components[I]);
      if (MComp.visible) then
      begin
        Printer.Canvas.Font := MComp.Font;
        DC := Printer.Canvas.Handle; {so DrawText knows about font}
        R := MComp.BoundsRect;
        R.Top := (R.Top + VertScrollBar.Position) * ScaleY;
        R.Left := (R.Left + HorzScrollBar.Position) * ScaleX;
        R.Bottom := (R.Bottom + VertScrollBar.Position) * ScaleY;
        R.Right := (R.Right + HorzScrollBar.Position) * ScaleY;
        if (not(Components[I] is TCustomLabel)) and (MComp.BorderStyle =
bsSingle)
          then Printer.Canvas.Rectangle(R.Left,R.Top,R.Right,R.Bottom);
        Format := DT_LEFT;
        if (Components[I] is TEdit) or (Components[I] is TCustomMaskEdit)
then
          Format := Format or DT_SINGLELINE or DT_VCENTER
        else
        begin
          if MComp.WordWrap then Format := DT_WORDBREAK;
          if MComp.Alignment = taCenter then Format := Format or DT_CENTER;
          if MComp.Alignment = taRightJustify then Format := Format or
DT_RIGHT;
          R.Bottom := R.Bottom + Printer.Canvas.Font.Height + 1;
        end;
        CLen := MComp.GetTextBuf(C,255);
        R.Left := R.Left + ScaleX + ScaleX;
        DrawText(DC, C, CLen, R, Format);
      end;
    end;
  Printer.EndDoc;
end;
```

Q: How do I do outline drag and drop?

A:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Grids, Outline;

type
  TForm1 = class(TForm)
    Outline1: TOutline;
    Outline2: TOutline;
    procedure OutlineDragDrop(Sender, Source: TObject; X, Y: Integer);
    procedure OutlineMouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure OutlineDragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.OutlineDragDrop(Sender, Source: TObject; X, Y: Integer);
begin
  with Sender as TOutline do
    begin
      AddChild(GetItem(x,y),
          TOutline(Source).Items[TOutline(Source).SelectedItem].Text);
    end;

end;

procedure TForm1.OutlineMouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if Button = mbLeft then
    with Sender as TOutline do
      begin
        if GetItem(x,y) >= 0 then
          BeginDrag(False);
      end;
end;

procedure TForm1.OutlineDragOver(Sender, Source: TObject; X, Y: Integer;
```

```
  State: TDragState; var Accept: Boolean);
begin
  if (Source is TOutline) and (TOutline(Source).GetItem(x,y) <>
TOutline(Source).SelectedItem) then
    Accept := True
  else
    Accept := False;
end;

end.
```

TStringGrid

How do I populate a TStringGrid with strings from a file...and save the strings after user editing back to a file?
How do I highlight selected fields on a TStringGrid?
How do I make characters in a string in a TStringGrid different colors?
Here is a custom TStringGrid that will allow for inserting a whole row at a time.
How do I right justify a column of numeric data in a TStringGrid?

MkDir error (a compuserve thread)

Q:
Is it my imagination or is there something wrong with MkDir() function ?!?!

I am using it as the online help instructs :

```
var
   cNewDir: string;

begin
   cNewDir := txtName.Text;
   MkDir(cNewDir);
   etc.
end;
```

but the compiler seems to fail with an error 94 "." expected and the cursor flashing after the word MkDir and before the (???

Can it really be expecting a "." ???

A:
The error would indicate that you've redefined the identifier "MkDir" somewhere in your app, and the compiler, due to scoping rules, is using your definition of "MkDir" rather than the SYSTEM units definition.

My guess... you've named your unit MkDir.

Reply:
All I can say is *Damn your good*. I can't believe it was that. I can't believe you guessed it from the little information I gave you. I have now kicked myself several times for being so stupid.

Thanks for your help!

Q: What is a Callback function, and how do I create one?

A: A call back function is a function which you write, but is called by some other program or module, such as windows. To create a callback function, you must first declare a function type, the funciton itself, and implement the function. In the interface section:

{ In main program interface }

```
type
  TCallBackFunction = function(s: string): integer;
  CallMe(s: string): integer;
```

And in the Implementation section:

```
{ In main program implementation }

procedure TestCallBack(CallBackFunction: TCallBackFunction); far; external
'Other';
{ Note that 'other' is a Dll containing the procedure TestCallBack }

function CallMe(s: PChar): integer;
begin
  { what ever you need to do }
  CallMe := 1; { What ever you need to return }
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  TestCallBack(CallMe);
end;
```

Note that in 'Other' you would also declare a function type, and use it like this:

```
{ in library Other interface }
type
  TMainFunction = function(s: string): integer;
  TestCallBack(MainFunc: TMainFunction);
{ in library Other implementation }
TestCallBack(MainFunc: TMainFunction);
var
  result: integer;
begin
  result:=MainFunc('test');
end;
```

The Top-14 ways things would be different if Microsoft built cars.
(drumroll, please...)

1. A particular model year of car wouldn't be available until AFTER that year, instead of before.

2. Every time they repainted the lines on the road, you'd have to buy a new car.

3. Occasionally your car would just die for no reason, you'd have to restart it. For some strange reason, you would just accept this.

4. You could only have one person at a time in your car, unless you bought a car '95 or a car NT, but then you'd have to buy more seats.

5. You would be constantly pressured to upgrade your car. Wait a sec, it's that way NOW!

6. Sun Motorsystems would make a car that was solar powered, twice as reliable, 5 times as fast, but only ran on 5% of the roads.

7. The oil, alternator, gas, engine warning lights would be replaced with a single "General Car Fault" warning light.

8. People would get excited about the "new" features in Microsoft cars, forgetting completely that they had been available in other brands for years.

9. We would still be waiting on the "6000 sux 58'" model to come out.

10. We'd all have to switch to Microsoft Gas (tm).

11. Lee Iacocca would be hired on as Bill G.'s chauffeur.

12. The US government would be GETTING subsidies from an automaker, instead of giving them.

13. New seats will force everyone to have the same size ass.

14. Ford, General Motors and Chrysler would all be complaining because Microsoft was putting a radio in all its models.

Q. How do I change the color of a grid cell in a TDBGrid?

A. Enter the following code in the TDBGrid's OnDrawDataCell event:

```
Procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
  Field: TField; State: TGridDrawState);
begin
   If gdFocused in State then
      with (Sender as TDBGrid).Canvas do
      begin
         Brush.Color := clRed;
         FillRect(Rect);
         TextOut(Rect.Left, Rect.Top, Field.AsString);
      end;
end;
```

Set the Default drawing to true. With this, it only has to draw the highlighted cell. If you set DefaultDrawing to false, you must draw all the cells yourself with the canvas properties.

Q. How do I show the contents of a memo field in a DBGrid?

A. Use the following code for the OnDrawDataCell event of the DBGrid. Note: before running create a TMemoField object for the memo field by double clicking on the TTable component and adding the memo field.

```
    procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
      Field: TField; State: TGridDrawState);
    var
      P  : array [0..50] of char;   {array size is number of characters
needed}
      BS : tBlobStream;             {from the memo field}
      S  : String;
    begin
      If Field is TMemoField then begin
      with (Sender as TDBGrid).Canvas do
        begin
          {Table1Notes is the TMemoField}
          BS := tBlobStream.Create(Table1Notes, bmRead);
          FillChar(P,SizeOf(P),#0);               {terminate the null string}
          BS.Read(P, 50);           {read 50 chars from memo into blobStream}
          BS.Free;
          S := StrPas(P);
          while Pos(#13, S) > 0 do             {remove carriage returns and}
            S[Pos(#13, S)] := ' ';             {line feeds}
          While Pos(#10, S) > 0 do
            S[Pos(#10, S)] := ' ';
          FillRect(Rect);                          {clear the cell}
          TextOut(Rect.Left, Rect.Top, S);         {fill cell with memo data}
        end;
      end;
    end;
```

Q. How do I do a locate on a non-indexed field?

A. The following function can be added to your to your unit and called as follows:

```
  Locate(Table1, Table1LName, 'Beman');
```

Table1 is your table component, Table1LName is TField you've add with the fields editor (double click on the table component) and 'Beman' is the name you want to find.

```
(* Locate will find SValue in a non-indexed table *)
Function Locate( const oTable: TTable; const oField: TField;
const sValue: String): Boolean;
var
  bmPos  : TBookMark;
  bFound : Boolean;
begin
  Locate := False;
  bFound := False;
  If not oTable.Active then Exit;
  If oTable.FieldDefs.IndexOf( oField.FieldName ) < 0 then Exit;
  bmPos := oTable.GetBookMark;
  With oTable do
  begin
    DisableControls;
    First;
    While not EOF do
      if oField.AsString = sValue then
      begin
        Locate := True;
        bFound := True;
        Break;
      end
      else Next;
  end ;
  If (Not bFound) then oTable.GotoBookMark( bmPos);
  oTable.FreeBookMark( bmPos );
  oTable.EnableControls;
end;
```

Q. How do I override the default message handler for my Application?

A. You create a dynamic method that is indexed to the message constant that you want to override. If you want to override the CM_DIALOGKEY message you would declare the following procedure in the public section of you class declaration:

```
Procedure CMDialogKey(var Message: TCMDialogKey);message CM_DIALOGKEY;
```

It is common practice to declare the procedure name the same as the message name minus the underscore. Your message handler would look something like this:

```
Procedure TForm1.CMDialogKey(var Message: TCMDialogKey);
begin
  if CharCode = VK_TAB then begin
    {Process the Alt+Tab key here}
    result := 1;
    exit;
  end;
  inherited;
end;
```

Setting result to 1 stops further processing. The inherited statement passes control to the parent handler. Be sure to execute inherited for all the cases you don't want to handle. Do not execute it for the ones you do handle.

Q. How can VCL components be created on the fly at run-time?

A. * The following code will create a modal password form at runtime. The TPasswordForm type is a TForm descendent class defined either in the current unit or in a separate unit referenced by the current unit's uses clause.

```
with TPasswordForm.Create(Application) do
begin                                ( i.e TForm1, TPasswordForm etc. }
   ShowModal;                        { Display form as a modal window }
   Free;                             { Free the form when it is closed }
end;
```

* The following are the general steps to add a component to a form at run-time:

1. Declare an instance variable of the component type that you wish to create {i.e. TButton }. Note: instance variables are used to point to an actual instance of an object. They are not objects themselves.
2. Use the component's Create constructor to create an instance of the component and assign the instance to the instance variable declared in step 1. All components' Create constructors take a parameter - the component's owner. Except in special circumstances, you should always pass the form as the owner parameter of the component's Create constructor.
3. Assign a parent to the component's Parent property (i.e. Form1, Panel1, etc). The Parent determines where the component will be displayed, and how the component's Top and Left coordinates are interpreted. To place a component in a groupbox, set the component's Parent property to the groupbox. For a component to be visible, it must have a parent to display itself within.
4. Set any other properties that are necessary (i.e. Width, Height).
5. Finally, make the component appear on the form by setting the component's Visible property to True.
6. If you created the component with an owner, you don't need to do anything to free the component - it will be freed when the owner is destroyed. If you did not give the component an owner when you created it, you are responsible for making sure the component is freed when it is no longer needed.

The following demonstrates how to add a TButton component to the current form at run-time:

```
var
  TempButton : TButton;  { This is only a pointer to a TButton }
begin
  TempButton := TButton.Create(Self); { Self refers to the form }
  TempButton.Parent := Self;          { Must assign the Parent }
  TempButton.Caption := 'Run-time';   { Assign properties now }
  TempButton.Visible := True;         { Show to button }
end;
```

Since the button was created with an owner, it will be freed automatically when its owner, the form, is freed.

Q. How can I get a horizontal scrollbar on a list box?

A. Send a LB_SetHorizontalExtent message to the listbox's window handle. For example, the message could be sent in the form's OnCreate:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 SendMessage(Listbox1.Handle, LB_SetHorizontalExtent,
             1000, Longint(0));
end;
```

TCompatibleStream

--- Mike Scott --- Mobius Ltd

----------COMPSTRM.PAS----------

```
unit CompStrm;

interface

uses Classes ;

type
  TCompatibleStream       = class ;

  { TStreamObject }

  TStreamObject = class( TComponent )
    constructor Load( S : TCompatibleStream ) ; virtual ; abstract ;
    procedure Store( S : TCompatibleStream ) ; virtual ; abstract ;
    function  GetObjectType : word ; virtual ; abstract ;
  end ;

  TStreamObjectClass = class of TStreamObject ;

  { TCompatibleStream }

  TCompatibleStream = class( TFileStream )
    function  ReadString : string ;
    procedure WriteString( var S : string ) ;
    function  StrRead : PChar ;
    procedure StrWrite( P : PChar ) ;
    function  Get : TStreamObject ; virtual ;
    procedure Put( AnObject : TStreamObject ) ; virtual ;
  end ;

{ Register Type : use this to register your CompatibleStream objects
                  with the same ID they had in OWL }

procedure RegisterType( AClass : TStreamObjectClass ;
                        AnID   : word ) ;

implementation

uses SysUtils, Controls ;

var Registry : TList ;  { holds object ID and class information }

{ TClassInfo }

type
  TClassInfo = class( TObject )
    ClassType : TStreamObjectClass ;
    ClassID   : word ;
    constructor Create( AClassType : TStreamObjectClass ;
```

```
                        AClassID   : word ) ; virtual ;
  end ;

constructor TClassInfo.Create( AClassType : TStreamObjectClass ;
                               AClassID   : word ) ;

var AnObject : TStreamObject ;

begin
  if not Assigned( AClassType ) then
    Raise EInvalidOperation.Create( 'Nil Class passed to TClassInfo.Create' )
;
  if not AClassType.InheritsFrom( TStreamObject ) then
    Raise EInvalidOperation.Create( 'Class ' + AClassType.ClassName +
                                    ' is not a descendent of TStreamObject' )
;
  ClassType := AClassType ;
  ClassID := AClassID ;
end ;


{ registry search functions }

function  FindClassInfo( AClass : TClass ) : TClassInfo ;

var i : integer ;

begin
  for i := Registry.Count - 1 downto 0 do begin
    Result := TClassInfo( Registry.Items[ i ] ) ;
    if Result.ClassType = AClass then exit ;
  end ;
  Raise EInvalidOperation.Create( 'Class ' + AClass.ClassName +
                          ' has not been registered for streaming' ) ;
end ;


function  FindClassInfoByID( AClassID : word ) : TClassInfo ;

var i : integer ;
    AName : string[ 31 ] ;

begin
  for i := Registry.Count - 1 downto 0 do begin
    Result := TClassInfo( Registry.Items[ i ] ) ;
    AName := TClassInfo( Registry.Items[ i ] ).ClassType.ClassName ;
    if Result.ClassID = AClassID then exit ;
  end ;
  Raise EInvalidOperation.Create( 'Class ID ' + IntToStr( AClassID ) +
                                  ' does not correspond to a registered
class' ) ; end ;


procedure RegisterType( AClass : TStreamObjectClass ;
                        AnID   : word ) ;

var i : integer ;
```

```
begin
  { see if it's been registered already }
  for i := Registry.Count - 1 downto 0 do
    with TClassInfo( Registry[ i ] ) do if ClassType = AClass then
  begin
    if ClassID <> AnID then
      Raise EInvalidOperation.Create( 'Class ' + AClass.ClassName +
                                      ' already registered with ID ' +
                                      IntToStr( ClassID ) ) ;
    exit ;
  end ;
  Registry.Add( TClassInfo.Create( AClass, AnID ) ) ;
end ;


{ TCompatibleStream }

function  TCompatibleStream.ReadString : string ;

begin
  ReadBuffer( Result[ 0 ], 1 ) ;
  if byte( Result[ 0 ] ) > 0 then ReadBuffer( Result[ 1 ], byte( Result[ 0 ]
) ) ; end ;


procedure TCompatibleStream.WriteString( var S : string ) ;

begin
  WriteBuffer( S[ 0 ], 1 ) ;
  if Length( S ) > 0 then WriteBuffer( S[ 1 ], Length( S ) ) ;
end ;


function TCompatibleStream.StrRead : PChar ;

var L : Word ;
    P : PChar ;

begin
  ReadBuffer( L, SizeOf( Word ) ) ;
  if L = 0 then StrRead := nil else
  begin
    P := StrAlloc( L + 1 ) ;
    ReadBuffer( P[ 0 ], L ) ;
    P[ L ] := #0 ;
    StrRead := P ;
  end ;
end ;


procedure TCompatibleStream.StrWrite( P : PChar ) ;

var L : Word ;

begin
  if P = nil then L := 0 else L := StrLen( P ) ;
```

```
  WriteBuffer( L, SizeOf( Word ) ) ;
  if L > 0 then WriteBuffer( P[ 0 ], L ) ;
end;


function  TCompatibleStream.Get : TStreamObject ;

var AClassID : word ;

begin
  { read the object ID, find it in the registry and load the object }
  ReadBuffer( AClassID, sizeof( AClassID ) ) ;
  Result := FindClassInfoByID( AClassID ).ClassType.Load( Self ) ;
end ;


procedure TCompatibleStream.Put( AnObject : TStreamObject ) ;

var AClassInfo : TClassInfo ;
    ANotedPosition : longint ;
    DoTruncate     : boolean ;

begin
  { get the object in the registry }
  AClassInfo := FindClassInfo( AnObject.ClassType ) ;

  { note the position in case of a problem }
  ANotedPosition := Position ;
  try
    { write out the class id and call the store method }
    WriteBuffer( AClassInfo.ClassID, sizeof( AClassInfo.ClassID ) ) ;
    AnObject.Store( Self ) ;
  except
    { return to previous position and if we're at the EOF then truncate }
    DoTruncate := Position = Size ;
    Position := ANotedPosition ;
    if DoTruncate then Write( ANotedPosition, 0 ) ;
    Raise ;
  end ;
end ;


{ exit proc stuff to clean up the registry }

procedure DoneCompStrm ; far ;

var i : integer ;

begin
  { free the registry }
  for i := Registry.Count - 1 downto 0 do TObject( Registry.Items[ i ] ).Free
;
  Registry.Free ;
end ;


begin
```

```
  Registry := TList.Create ;
  AddExitProc( DoneCompStrm ) ;
end.
```

expression index search

Below is a generic (with some limits) search function that works on expression tags of dBase files.

It is alpha code, barely tested, so use at your own risk <s>. Do let me know if you have a problem. I may not be able to help you as I can be rather busy, but at least, I'll know about it, and in time, may be able to solve the bugs.

The main limit is that it currently only supports character and date fields in the expression. If your expression contains other fields, the function will not work. I may expand it if the need arises for me, or you can try to expand it if you need the extra capability.

A second limit is that each field can only be used once in the expression.

The next enhancement of this will optionally permit SoftSeeks (i.e. seek on the closest value instead of on a specific value).

The next enhancement should also allocate memory for the right size of the Key, instead of putting an arbitrary value of 250 bytes.

Here is an example of how to call the generic SearchExpr function:

```
{current index key expression is 'DTOS(CRT_DATE)+CRT_CODE'}

Table1.SetKey;
Table1.FieldByName('CRT_DATE').AsString := '2/2/94'; {assign search values}
Table1.FieldByName('CRT_CODE').AsString := 'T002';   {assign search values}

if SeekExpr(Table1.Handle, [Table1.FieldByName('CRT_DATE'),
                            Table1.FieldByName('CRT_CODE')] ) then
  ShowMessage('Found!')
else
  ShowMessage('Not found!');

{note that the record pointer is now on the record it found, if any}
```

Here is the code for the generic search function:

```
function TDBFTest1.SeekExpr(hTable: HDBIcur;
                            aFlds: array of TField): boolean;

  const
    KeyBufSize = 250;        {maximum size of key}

  var
    pRecBuf:   pByte;        {contains a record in memory}
    pKeyBuf:   pByte;        {contains actual key}
    TblProps:  CURProps;     {table properties }
    rslt:      DBIResult;    {result of BDE call}
```

```
  i:         integer;      {counter}

  tmpPSZ:    PCHAR;
  tmpDate:   LongInt;
begin
  {get size of record buffer}
  DbiGetCursorProps(hTable, TblProps);

  {allocate memory for record buffer}
  GetMem(pRecBuf, TblProps.iRecBufSize * sizeof(BYTE));
  GetMem(pKeyBuf, KeyBufSize); {allocate mem for key}

  {initialize record buffer}
  DbiInitRecord(hTable, pRecBuf);

  {set search values in record buffer}
  for i := 0 to High(aFlds) do
  begin
    {TStringField}
    if (aFlds[i] is TStringField) then
      begin
        GetMem(tmpPSZ, Length(aFlds[i].AsString)+1);
        StrPCopy(tmpPSZ, aFlds[i].AsString);
        DbiPutField(hTable,              {TTable}
                    aFlds[i].FieldNo,    {TField}
                    pRecBuf,             {pointer to record buffer}
                    (tmpPSZ));           {pointer to value}
        FreeMem(tmpPSZ, Length(aFlds[i].AsString)+1);
      end
    {TDateField}
    else if (aFlds[i] is TDateField) then
      begin
        tmpDate := Trunc(aFlds[i].AsDateTime);
        DbiPutField(hTable,              {TTable}
                    aFlds[i].FieldNo,    {TField}
                    pRecBuf,             {pointer to record buffer}
                    (@tmpDate));         {pointer to value}
      end
  end;

  {create a key for the above values}
  DbiExtractKey(hTable, pRecBuf, pKeyBuf);

  {search using this key (stored in pKeyBuf)}
  table1.updatecursorpos;
  rslt := DbiGetRecordForKey(hTable, True, 0, 0, pKeyBuf, nil);
  table1.refresh; {update data aware components}

  if rslt <> 0 then
    Result := False
  else
    result := True;

  ShowRecord; {display new record number -- see below}

  {free memory allocated for buffer}
  FreeMem(pRecBuf, TblProps.iRecBufSize * sizeof(BYTE));
```

```
  FreeMem(pKeyBuf, KeyBufSize);
end;

procedure TDBFTest1.ShowRecord;
{Show a dBase record number}
{NOTE: it assumes there is a label component named RecordNo.}
var
  myrecprop: RECprops;
  recno: integer;
begin
  table1.updatecursorpos;
  recno := 0;
  dbiGetRecord(table1.handle, dbiNOLOCK, nil, @myrecprop);
  recno := myrecprop.iPhyRecNum;
  RecordNo.Caption := inttostr(recno);
end;
```

Q: I want to resize my TDrawGrid columns so that the grid fits exactly on the form even when the form is resized. This is what I am trying, but it is always off by a bit.

```
var
  i, WidthOfCols: integer;
begin
  WidthOfCols := 0;
  for i := 1 to DrawGrid1.ColCount - 1 do
    WidthOfCols := DrawGrid1.ColWidths[i];
   DrawGrid1.ColWidths[0] := form1.width - WidthOfCols;
end;
```

What do I do?

A: You have it close, but no cigar.

```
var
  i, WidthOfCols: integer;
begin
  WidthOfCols := 0;
  for i := 1 to DrawGrid1.ColCount - 1 do
    WidthOfCols := DrawGrid1.ColWidths[i];
  DrawGrid1.ColWidths[0] :=
    form1.ClientWidth - WidthOfCols - (DrawGrid1.ColCount + 1);
end;
```

What this is doing is different from what you were doing on a few points.

1. The form's client area must be used instead of the form's width. This is because we don't want the frame size figured into the value.
2. We need to subtract the lines of the grid itself. The width of a column excludes the lines that define it. We use the ColCount + 1 because there are lines on both the left and right sides of it.

Note: run this procedure on the form.create and the form.resize to keep everything looking good.

Q: How o I use Access under Delphi?

A: Guidlines for working with Access under Delphi/BDE/ODBC:

The ODBC driver provided with Access 2.0 (ODBCJT16.DLL with a file size of <65k) is designed to work only within the Microsoft Office environment. To work with ODBC/Access in Delphi, you need the Microsoft ODBC Desktop Driver (ODBCJT16.DLL with a file size of approx 260k) kit, part# 273-054-030 available from Microsoft Direct for $10.25US (post on WINEXT for where to get it in your country if you are not in the US). It is also available on the Jan. MSDN, Level 2 (Development Platform) CD4 \ODBC\X86 as part of the ODBC 2.1 SDK. Be aware that your redistribution rights for the Desktop Drivers are pretty restricted by Microsoft. For info on (and objections to) the restrictions post on the WINEXT forum.

You also need the following ODBC files.

Minimum:

ODBC.DLL	03.10.1994, Version 2.00.1510
ODBCINST.DLL	03.10.1994, Version 2.00.1510
ODBCINST.HLP	11.08.1993
ODBCADM.EXE	11.08.1993, Version 1.02.3129

Better:

ODBC.DLL	12.07.1994, Version 2.10.2401
ODBCINST.DLL	12.07.1994, Version 2.10.2401
ODBCINST.HLP	12.07.1994
ODBCADM.EXE	12.07.1994, Version 2.10.2309

The following steps will get you started in Delphi

1. Using the ODBC Administrator, set-up a datasource for your database. Be sure to specify a path to your mdb file. For the purposes of this explanation we'll say that the datasource name is MYDSN.

2. Load the BDE Configuration utility.

3. Select New Driver.

4. Give the driver a name (call it ODBC_MYDSN).

5. In the driver combo box select, "Microsoft Access Driver (*.mdb)

6. In the name combo box select MYDSN

7. Go to the Alias page.

8. Select New Alias.

9. Enter MYDSN for name.

10. For Alias Type, select ODBC_MYDSN.

11. In Delphi, drop a DataSource, Table, and DBGrid on your form.

12. Set DBGrid1.DataSource to DataSource1.

13. Set DataSource1.DataSet to Table1.

14. Set Table1.DatabaseName to MYDSN.

15. In the TableName property in Table1, click the downarrow, you will see the "Login" dialog. Press OK, after a short pause you will see a dropdown list with all your table names. Select one.

16. Set the Active property in Table1 to True, the data from your table will be displayed in the grid.

It would incur a major compiler overhead. In order to efficiently generate code, the compiler needs to have available a scalar type which encompasses the entire range of possible values for all scalar types. If you wanted to accommodate both signed and unsigned 32-bit values, that range would be -2147483548..4294967295, and that range can't be contained in 32 bits.

Q: I'm having a little trouble trying to write a routine that will iterate through all the directories and sub-directories of a given hard disk. What I need to to is calculate the total file size of every file on the hard disk that matches a certain filespec. How do I do this?

A: Here is the long version for Delphi:

```
{$I-}
{$M 65000,0,1024}
program KillDir;
uses crt,dos;

type String12 = string[12];
var TotalSize : longint;
    ThisSize : longint;

procedure UpString(var S:string);
var i:word;
begin
  for i := 1 to length(S) do
    S[i] := upcase(S[i]);
end;

function AbortIt:boolean;
var ch : char;
begin
  AbortIt := false;
  if not(KeyPressed) then Exit;
  ch := readkey;
  if ch = #0 then ch := chr(ord(readkey) or $80);
  if (ch = ^C) or (ch = ^X) or (ch = ^Q) or (ch = #$1B) then
    AbortIt := true;
end;

function GetSize(var F:file):string12;
var RawSize : longint;
    Size:string;
begin
   Reset(F,1);
   RawSize := FileSize(F);
   Close(F);
   ThisSize := RawSize;
   if IOresult <> 0 then {nop};
   if RawSize < 10000 then
   begin
     str(RawSize,Size);
   end
   else if RawSize < (1024*999) then
   begin
     str(RawSize div 1024,Size);
     Size[length(Size)+1] := 'K';
     inc(Size[0]);
   end
   else
   begin
```

```
     str(RawSize div (1000*1024),Size);
     Size[length(Size)+1] := 'M';
     Inc(Size[0]);
   end;
   while length(Size) < 4 do
   begin
     Size := ' '+Size;
   end;
   GetSize := Size;
end;

function SizeIt(Which:string):byte;
var DirInfo:SearchRec;
    f : file;
    Attr,Result:word;
    Size,Who:string12;
    Current:string;
begin
  SizeIt := 1;
  if IOresult <> 0 then {nop};
  GetDir(0,Current);
  chdir(Which);
  if IOresult <> 0 then Exit;
  Who := '*.*';
  findfirst(Who, $3F, DirInfo);
  while DosError = 0 do
  begin
    if AbortIt then
    begin
      SizeIt := 2;
      Exit;
    end;
    if (DirInfo.Name <> '.') and (DirInfo.Name <> '..') then
    begin
      Assign(F,DirInfo.Name);
      GetFAttr(F,Attr);
      if (Attr and Directory) <> 0 then
      begin
        Result := SizeIt(DirInfo.Name);
        SizeIt := Result;
        if Result <> 0 then Exit;
      end
      else
      begin
        SetFAttr(F,0);
        Size := GetSize(F);
        Who := DirInfo.Name+'            ';
        writeln(Current+'\'+Which,'  ',Who,' Size:',Size);
        TotalSize := TotalSize+ThisSize;
      end;
    end;
    FindNext(DirInfo);
  end;
  if IOresult <> 0 then {nop};
  ChDir(Current);
  SizeIt := 0;
end;
```

```
var Where:string;
    Current:string;
    Yorn:string;
    Result:word;

begin
  TotalSize := 0;
  writeln;
  Writeln('Directory Sizer V1.01  Written by Michael Day as of 05 Sept 94');
  if ParamCount < 0 then
  begin
    writeln('Format is: FSIZE DIRNAME');
    writeln('This program will find the size of all files and all
directories');
    writeln('in and below the directory DIRNAME.');
    halt(1);
  end;

  Where := ParamStr(1);
  UpString(Where);
  if pos(Where,':') <> 0 then
  begin
    writeln('Sorry, you cannot size directories on another drive with this
program.');
    writeln('Please move to that drive first.');
    halt(2);
  end;

  if IOresult <> 0 then {nop};
  GetDir(0,Current);
  chdir(Where);
  if IOresult <> 0 then
    Result := 1
  else
    Result := 0;
  chdir(Current);


  if Result = 0 then
  begin

    writeln('This will find the size of ALL files and ALL directories in and
below:');
    writeln(Current+'\'+Where);
    Result := SizeIt(Where);
    chdir(Current);

    write('Total size of the directory');
    write(Current+'\'+Where);
    writeln(TotalSize);

    if Result = 2 then
    begin
      writeln('Directory size operation terminated by the user.');
      Halt(3);
```

```
    end;
  end;

  if Result = 1 then
  begin
    writeln('Error finding directory: ',Where);
    writeln('The directory probably does not exist.');
    halt(4);
  end;

end.
```

Here is the short version written for BP7 in DOS:

This program simply lists the names of the files, but you could change the output line do accumulate sizes just as easily.

```
program search;
uses dos;
var i:integer;
    d:dirstr; n:namestr; x:extstr;

  procedure helpmsg;
  begin
    writeln('SEARCH filespec [filespec]...');
  end;

  procedure dosearch(const d:string);
  var sr:searchrec;

    procedure dofilesearch;
    begin
      findfirst(d+n+x,archive+readonly,sr);
      while doserror=0 do begin
        writeln(d+sr.name);        { THE output }
        findnext(sr);
      end;
    end;

    procedure dodirsearch;
    begin
      findfirst(d+'*.*',directory,sr);
      while doserror=0 do begin
        if (sr.attr and directory = directory)
        and (sr.name[1]<>'.') then              { ignores "." and ".." }
          dosearch(d+sr.name+'\');
        findnext(sr);
      end;
    end;

  begin {dosearch}
    dofilesearch;
    dodirsearch;
  end;
```

```
begin
  if paramcount<1 then helpmsg
  else begin
    for i:=1 to paramcount do begin
      fsplit(paramstr(i),d,n,x);
      dosearch(d);
    end;
  end;
end.
```

Q: How can I draw text right on the glyph of a speedbutton?

A: You can do it in the form's OnCreate event handler. Something like this:

```
WITH BitBtn1.Glyph, Canvas DO
BEGIN
  R := BitBtn1.ClientRect;
  InflateRect(R, -4, -4);
  Width := R.Right - R.Left + 1;
  Height := R.Bottom  - R.Top + 1;
  SetTextAlign(Handle, TA_CENTER);
  Font := Self.Font;
  TextOut(Width DIV 2, 8, 'Foo');
  TextOut(Width DIV 2, 24, 'Bar');
END;
```

```
Memo1.Parent:= TabbedNoteBook1.Pages.Objects[TabbedNoteBook1.PageIndex] as
TWinControl;
```

TDBGrid

How do I set focus on a specific field on a TDBGrid?
How do I change the color of a grid cell in a TDBGrid?
How do I show the contents of a memo field in a TDBGrid?
How do I use a navigator control with multiple tdbGrids?
How do I resize my TDrawGrid columns so that the grid fits exactly on the form?
How do you tell which record and which field of a TDBGrid is current?
How do I change the color of a grid cell in a TDBGrid?
How do I surface the MouseDown Event?
How do I to one character type-ahead in a TDBGrid?
How do I calculate the width of a TDBGrid so that it will display all the fields?
How do I put components into a TDBGrid? (i.e. checkbox, combobox, etc.)
How can I tell which row and column is currently selected?
How do I setup the column list in code for a dynamically created TTable?

file management routines

```
unit FmxUtils;

interface

uses SysUtils, WinTypes, WinProcs, Classes, Consts;

type
  EInvalidDest = class(EStreamError);
  EFCantMove = class(EStreamError);

procedure CopyFile(const FileName, DestName: TFileName);
procedure MoveFile(const FileName, DestName: TFileName);
function GetFileSize(const FileName: string): LongInt;
function FileDateTime(const FileName: string): TDateTime;
function HasAttr(const FileName: string; Attr: Word): Boolean;
function ExecuteFile(const FileName, Params, DefaultDir: string;
  ShowCmd: Integer): THandle;

implementation

uses Forms, ShellAPI;

const
  SInvalidDest = 'Destination %s does not exist';
  SFCantMove = 'Cannot move file %s';

procedure CopyFile(const FileName, DestName: TFileName);
var
  CopyBuffer: Pointer; { buffer for copying }
  TimeStamp, BytesCopied: Longint;
  Source, Dest: Integer; { handles }
  Destination: TFileName; { holder for expanded destination name }
const
  ChunkSize: Longint = 8192; { copy in 8K chunks }
begin
  Destination := ExpandFileName(DestName); { expand the destination path }
  if HasAttr(Destination, faDirectory) then { if destination is a
directory... }
    Destination := Destination + '\' + ExtractFileName(FileName); { ...clone
file name }
  TimeStamp := FileAge(FileName); { get source's time stamp }
  GetMem(CopyBuffer, ChunkSize); { allocate the buffer }
  try
    Source := FileOpen(FileName, fmShareDenyWrite); { open source file }
    if Source < 0 then raise EFOpenError.Create(FmtLoadStr(SFOpenError,
[FileName]));
    try
      Dest := FileCreate(Destination); { create output file; overwrite
existing }
      if Dest < 0 then raise EFCreateError.Create(FmtLoadStr(SFCreateError,
[Destination]));
      try
        repeat
          BytesCopied := FileRead(Source, CopyBuffer^, ChunkSize); { read
```

```
chunk }
          if BytesCopied > 0 then { if we read anything... }
            FileWrite(Dest, CopyBuffer^, BytesCopied); { ...write chunk }
        until BytesCopied < ChunkSize; { until we run out of chunks }
      finally
        FileClose(Dest); { close the destination file }
{        SetFileTimeStamp(Destination, TimeStamp);} { clone source's time
stamp }{!!!}
      end;
    finally
      FileClose(Source); { close the source file }
    end;
  finally
    FreeMem(CopyBuffer, ChunkSize); { free the buffer }
  end;
end;


{ MoveFile procedure }
{
  Moves the file passed in FileName to the directory specified in DestDir.
  Tries to just rename the file.  If that fails, try to copy the file and
  delete the original.

  Raises an exception if the source file is read-only, and therefore cannot
  be deleted/moved.
}

procedure MoveFile(const FileName, DestName: TFileName);
var
  Destination: TFileName;
begin
  Destination := ExpandFileName(DestName); { expand the destination path }
  if not RenameFile(FileName, Destination) then { try just renaming }
  begin
    if HasAttr(FileName, faReadOnly) then  { if it's read-only... }
      raise EFCantMove.Create(Format(SFCantMove, [FileName])); { we wouldn't
be able to delete it }
      CopyFile(FileName, Destination); { copy it over to destination...}
      DeleteFile(FileName); { ...and delete the original }
  end;
end;

{ GetFileSize function }
{
  Returns the size of the named file without opening the file.  If the file
  doesn't exist, returns -1.
}

function GetFileSize(const FileName: string): LongInt;
var
  SearchRec: TSearchRec;
begin
  if FindFirst(ExpandFileName(FileName), faAnyFile, SearchRec) = 0 then
    Result := SearchRec.Size
  else Result := -1;
end;
```

```
function FileDateTime(const FileName: string): System.TDateTime;
begin
  Result := FileDateToDateTime(FileAge(FileName));
end;

function HasAttr(const FileName: string; Attr: Word): Boolean;
begin
  Result := (FileGetAttr(FileName) and Attr) = Attr;
end;

function ExecuteFile(const FileName, Params, DefaultDir: string;
  ShowCmd: Integer): THandle;
var
  zFileName, zParams, zDir: array[0..79] of Char;
begin
  Result := ShellExecute(Application.MainForm.Handle, nil,
    StrPCopy(zFileName, FileName), StrPCopy(zParams, Params),
    StrPCopy(zDir, DefaultDir), ShowCmd);
end;

end.
```

&&

Q: How do I write programs using function pointers?

A:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

procedure SayIt(s: string);
var
  Form1: TForm1;
  x: procedure(s: string);

implementation
{$R *.DFM}
procedure SayIt(s: string);
begin
  showMessage(s);
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  x := SayIt;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  x(Edit1.text);
end;

end.
```

```
application.HelpCommand(help_quit, 0);
```

Q: Is there a way to use a Pascal string as a null terminated string?

A: Yes.

```
s[ord(s[0]) + 1] := #0;   { Make it null terminated. }
WinExec(@s[1], sw_ShowNormal);   { @s[1] is the last step. }
```

Q: How can I check to see if there is a disk in the "A" drive?

A:

```
var
  ErrorMode: word;
begin
  ErrorMode := SetErrorMode(SEM_FailCriticalErrors);
  try
    if DiskSize(1) = -1 then
      ShowMessage('Drive not ready');
  finally
    SetErrorMode(ErrorMode);
  end;
end;
```

They must be maintained indexes.

Q: How do I manipulate a TStringList in a DLL?

A:

```
{ Here is the DLL code. }
library Str_DLL;

uses classes;

procedure AddToStrList(s: string; sList: TStringList); export;
begin
  sList.add(s);
end;

exports
  AddToStrList index 1;

begin
end.

{************************************}

{ Here is how to use it.  This is from a form that has
two pushbuttons:  one to add a string to a string list,
and one to display the current string list in a list box.
The string to be added comes from a TEdit.  Default names
are used. }
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    Edit1: TEdit;
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  MyStringList: TStringList;

implementation

{$R *.DFM}
```

```
procedure AddToStrList(s: string; sList: TStringList);
  far; external 'str_dll';

procedure TForm1.Button1Click(Sender: TObject);
begin
  AddToStrList(edit1.text, MyStringList);
end;

procedure TForm1.Button2Click(Sender: TObject);
var
  i: integer;
begin
  listbox1.items.clear;
  listbox1.items.asign(MyStringList);
end;

begin
  MyStringList := TStringList.create;
end.
```

```
{remove leading blanks (spaces)}
function LTrim(s: string): string;
var i: integer;
begin
  i := 1;
  while s[i] = ' ' do inc(i);
  result := copy(s, i, length(s) - i + 1);
end;
```

```
{remove leading blanks (spaces)}
function LTrim(s: string): string;
var i: integer;
begin
  i := 1;
  while s[i] = ' ' do inc(i);
  result := copy(s, i, length(s) - i + 1);
end;

{Supresses trailing blanks}
function RTrim(s: string): string;
begin
  while s[length(s)] = ' ' do dec(s[0]);
  result := s;
end;

function AllTrim(s: string): string;
begin
  result := rTrim(lTrim(s));
end;
```

Q: How do I make a new component that has a string editor just like the one that comes with the regular components?

A:

```
unit MyEds;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TPropertyEditor = class(TButton)
  private
    { Private declarations }
    FStringStuff: TStrings;
  protected
    { Protected declarations }
  public
    { Public declarations }
    constructor create(Aowner: TComponent); override;
    destructor destroy; override;
    procedure SetStrings(s: TStrings);
  published
    { Published declarations }
    property StringStuff: TStrings read FStringStuff write SetStrings;
  end;

procedure Register;

implementation

constructor TPropertyEditor.create(AOwner: TComponent);
begin
  inherited create(AOwner);
  FStringStuff := TStringList.create;
end;

destructor TPropertyEditor.destroy;
begin
  FStringStuff.free; {Plugs the memory leak.}
  inherited destroy;
end;

procedure TPropertyEditor.SetStrings(s: TStrings);
begin
  if FStringStuff <> s then {This conditional is optional.}
    FStringStuff.Assign(s);
end;

procedure Register;
begin
  RegisterComponents('Lloyd', [TPropertyEditor]);
end;
```

```
end.
```

Q: ERROR in Complib.DCL while trying to initialize BDE

A: There are four(4) reasons this can occur when they try to run Delphi:

1) The IDAPI Section in the WIN.INI file is bad or missing. It should read something like:

```
[IDAPI]
CONFIGFIL01=C:\IDAPI\IDAPI.CFG
DLLPATH=C:\IDAPI
```

2) Share NOT loaded with correct parameters OR not loaded at all. It should be:

```
SHARE /F:4096 /L:40
```

Note: even if they have WFW and VSHARE it may be a good idea to load share anyway.

3) COMPLIB.DCL could be corrupted. Copy a new copy from the CD .. \RUNIMAGE\DELPHI\BIN\COMPLIB.DCL

4) The last resort could be and has been the case in a previous call or two... The Windows install is bad and a re-install of Windows is recomended. If you suspect this try to compare common .EXEs and .DLLs (e.g. -- VER.DLL, USER.EXE and KERNEL.EXE).

I noticed a leak in my GUI resources and traced it back to TCustomDBGrid. A field called FTitleFont (type TFont) is created but never destroyed. I added the following lines to the TCustomDBGrid.Destroy (destructor) method, and the leak seems to be plugged.

```
destructor TCustomDBGrid.Destroy;
begin
  FDataLink.Free;
  FDataLink := nil;
  FIndicators.Free;

  {Here are the two new lines.}
  FTitleFont.Free;
  FTitleFont:=Nil;

  inherited Destroy;
  ReleaseBitmap;
end;
```

huge numbers

By: Abe Timmerman; Alkmaar, The Netherlands
Send improvements to: A.Timmerman@beta.hsholland.nl

This unit uses an array of bytes to represent a LARGE number. The number is binairy-stored in the array, with the Least Significant Byte (LSB) first and the Most Significant Byte (MSB) last, like all Intel-integer types.

Arithmetic is not 10-based or 2-based, but 256-based, so that each byte represents one (1) digit.

The HugeInttype numbers are Signed Numbers.

When Compiled with the R+ directive, ADD and MUL wil generate an "Arithmetic Overflow Error" (RunError(215)) when needed. Otherwise use the "HugeIntCarry" variable.

Use the "HugeIntDiv0" variable to check on division by zero.

Use {$DEFINE HugeInt_xx } or "Conditional defines" from the "Compiler options" for sizing, where xx is 64, 32 or 16, otherwhise HugeIntSize will be set to 8 bytes.

```
unit HugeInts;
interface

const
{$IFDEF HugeInt_64 }
  HugeIntSize = 64;
{$ELSE}{$IFDEF HugeInt_32 }
  HugeIntSize = 32;
{$ELSE}{$IFDEF HugeInt_16 }
  HugeIntSize = 16;
{$ELSE}
  HugeIntSize = 8;
{$ENDIF}{$ENDIF}{$ENDIF}
  HugeIntMSB  = HugeIntSize-1;

type
  HugeInt = array[0..HugeIntMSB] of Byte;

const
  HugeIntCarry: Boolean = False;
  HugeIntDiv0:  Boolean = False;


procedure HugeInt_Min(var a: HugeInt);                { a := -a }
procedure HugeInt_Inc(var a: HugeInt);                { a := a + 1 }
procedure HugeInt_Dec(var a: HugeInt);                { a := a - 1 }
```

```
procedure HugeInt_Add(a, b: HugeInt; var R: HugeInt);  { R := a + b }
procedure HugeInt_Sub(a, b: HugeInt; var R: HugeInt);  { R := a - b }
procedure HugeInt_Mul(a, b: HugeInt; var R: HugeInt);  { R := a * b }
procedure HugeInt_Div(a, b: HugeInt; var R: HugeInt);  { R := a div b }
procedure HugeInt_Mod(a, b: HugeInt; var R: HugeInt);  { R := a mod b }

function HugeInt_IsNeg(a: HugeInt): Boolean;
function HugeInt_Zero(a: HugeInt): Boolean;
function HugeInt_Odd(a: HugeInt): Boolean;
function HugeInt_Comp(a, b: HugeInt): Integer;          {-1:a<b; 0; 1:a>b
}
procedure HugeInt_Copy(Src: HugeInt; var Dest: HugeInt);{ Dest := Src }

procedure String2HugeInt(AString: string; var a: HugeInt);
procedure Integer2HugeInt(AInteger: Integer; var a: HugeInt);
procedure HugeInt2String(a: HugeInt; var S: string);

                              implementation

procedure HugeInt_Copy(Src: HugeInt; var Dest: HugeInt);
{ Dest := Src }
begin
  Move(Src, Dest, SizeOf(HugeInt));
end;{ HugeInt_Copy }

function HugeInt_IsNeg(a: HugeInt): Boolean;
begin
  HugeInt_IsNeg := a[HugeIntMSB] and $80 > 0;
end;{ HugeInt_IsNeg }

function HugeInt_Zero(a: HugeInt): Boolean;
var i: Integer;
begin
  HugeInt_Zero := False;
  for i := 0 to HugeIntMSB do
    if a[i] <> 0 then Exit;
  HugeInt_Zero := True;
end;{ HugeInt_Zero }

function HugeInt_Odd(a: HugeInt): Boolean;
begin
  HugeInt_Odd := a[0] and 1 > 0;
end;{ HugeInt_Odd }

function HugeInt_HCD(a: HugeInt): Integer;
var i: Integer;
begin
  i := HugeIntMSB;
  while (i > 0) and (a[i] = 0) do Dec(i);
  HugeInt_HCD := i;
end;{ HugeInt_HCD }

procedure HugeInt_SHL(var a: HugeInt; Digits: Integer);
{ Shift "a" "Digits", digits (bytes) to the left,
  "Digits" bytes will 'fall off' on the MSB side
  Fill the LSB side with 0's }
var t: Integer;
```

```
b: HugeInt;
begin
  if Digits > HugeIntMSB then
    FillChar(a, SizeOf(HugeInt), 0)
  else if Digits > 0 then
    begin
      Move(a[0], a[Digits], HugeIntSize-Digits);
      FillChar(a[0], Digits, 0);
    end;{ else if }
end;{ HugeInt_SHL }

procedure HugeInt_SHR(var a: HugeInt; Digits: Integer);
var t: Integer;
begin
  if Digits > HugeIntMSB then
    FillChar(a, SizeOf(HugeInt), 0)
  else if Digits > 0 then
    begin
      Move(a[Digits], a[0], HugeIntSize-Digits);
      FillChar(a[HugeIntSize-Digits], Digits, 0);
    end;{ else if }
end;{ HugeInt_SHR }

procedure HugeInt_Inc(var a: HugeInt);
{ a := a + 1 }
var
  i: Integer;
  h: Word;
begin
  i := 0; h := 1;
  repeat
    h := h + a[i];
    a[i] := Lo(h);
    h := Hi(h);
    Inc(i);
  until (i > HugeIntMSB) or (h = 0);
  HugeIntCarry := h > 0;
  {$IFOPT R+ }
    if HugeIntCarry then RunError(215);
  {$ENDIF}
end;{ HugeInt_Inc }

procedure HugeInt_Dec(var a: HugeInt);
{ a := a - 1 }
var Minus_1: HugeInt;
begin
  { this is the easy way out }
  FillChar(Minus_1, SizeOf(HugeInt), $FF); { -1 }
  HugeInt_Add(a, Minus_1, a);
end;{ HugeInt_Dec }

procedure HugeInt_Min(var a: HugeInt);
{ a := -a }
var i: Integer;
begin
  for i := 0 to HugeIntMSB do
    a[i] := not a[i];
```

```
  HugeInt_Inc(a);
end;{ HugeInt_Min }

function HugeInt_Comp(a, b: HugeInt): Integer;
{ a = b: ==0; a > b: ==1; a < b: ==-1 }
var
  A_IsNeg, B_IsNeg: Boolean;
  i:                Integer;
begin
  A_IsNeg := HugeInt_IsNeg(a);
  B_IsNeg := HugeInt_IsNeg(b);
  if A_IsNeg xor B_IsNeg then
    if A_IsNeg then HugeInt_Comp := -1
    else HugeInt_Comp := 1
  else
    begin
      if A_IsNeg then HugeInt_Min(a);
      if B_IsNeg then HugeInt_Min(b);
      i := HugeIntMSB;
      while (i > 0) and (a[i] = b[i]) do Dec(i);
      if A_IsNeg then { both negative! }
        if a[i] > b[i] then HugeInt_Comp := -1
        else if a[i] < b[i] then HugeInt_Comp := 1
        else HugeInt_Comp := 0
      else { both positive }
        if a[i] > b[i] then HugeInt_Comp := 1
        else if a[i] < b[i] then HugeInt_Comp := -1
        else HugeInt_Comp := 0;
    end;{ else }
end;{ HugeInt_Comp }

procedure HugeInt_Add(a, b: HugeInt; var R: HugeInt);
{ R := a + b }
var
  i: Integer;
  h: Word;
begin
  h := 0;
  for i := 0 to HugeIntMSB do
    begin
      h := h + a[i] + b[i];
      R[i] := Lo(h);
      h := Hi(h);
    end;{ for }
  HugeIntCarry := h > 0;
  {$IFOPT R+ }
    if HugeIntCarry then RunError(215);
  {$ENDIF}
end;{ HugeInt_Add }

procedure HugeInt_Sub(a, b: HugeInt; var R: HugeInt);
{ R := a - b }
var
  i: Integer;
  h: Word;
begin
  HugeInt_Min(b);
```

```
  HugeInt_Add(a, b, R);
end;{ HugeInt_Sub }

procedure HugeInt_Mul(a, b: HugeInt; var R: HugeInt);
{ R := a * b }
var
  i, j, k:          Integer;
  A_end, B_end:     Integer;
  A_IsNeg, B_IsNeg: Boolean;
  h:                Word;
begin
  A_IsNeg := HugeInt_IsNeg(a);
  B_IsNeg := HugeInt_IsNeg(b);
  if A_IsNeg then HugeInt_Min(a);
  if B_IsNeg then HugeInt_Min(b);
  A_End := HugeInt_HCD(a);
  B_End := HugeInt_HCD(b);
  FillChar(R, SizeOf(R), 0);
  HugeIntCarry := False;
  for i := 0 to A_end do
    begin
      h := 0;
      for j:= 0 to B_end do
        if (i + j) < HugeIntSize then
          begin
            h := h + R[i+j] + a[i] * b[j];
            R[i+j] := Lo(h);
            h := Hi(h);
          end;{ if }
      k := i + B_End + 1;
      while (k < HugeIntSize) and (h > 0) do
        begin
          h := h + R[k];
          R[k] := Lo(h);
          h := Hi(h);
          Inc(k);
        end;{ while }
      HugeIntCarry := h > 0;
     {$IFOPT R+}
        if HugeIntCarry then RunError(215);
     {$ENDIF}
    end;{ for }
  { if all's well... }
  if A_IsNeg xor B_IsNeg then HugeInt_Min(R);
end;{ HugeInt_Mul }

procedure HugeInt_DivMod(var a: HugeInt; b: HugeInt; var R: HugeInt);
{ R := a div b  a := a mod b }
var
  MaxShifts, s, q:  Integer;
  d, e:             HugeInt;
  A_IsNeg, B_IsNeg: Boolean;

begin
  if HugeInt_Zero(b) then
    begin
      HugeIntDiv0 := True;
```

```
      Exit;
    end{ if }
  else HugeIntDiv0 := False;
  A_IsNeg := HugeInt_IsNeg(a);
  B_IsNeg := HugeInt_IsNeg(b);
  if A_IsNeg then HugeInt_Min(a);
  if B_IsNeg then HugeInt_Min(b);
  if HugeInt_Comp(a, b) < 0 then
    { a<b; no need to divide }
    FillChar(R, SizeOf(R), 0)
  else
    begin
      FillChar(R, SizeOf(R), 0);
      repeat
        Move(b, d, SizeOf(HugeInt));
        { first work out the number of shifts }
        MaxShifts := HugeInt_HCD(a) - HugeInt_HCD(b);
        s := 0;
        while (s <= MaxShifts) and (HugeInt_Comp(a, d) >= 0) do
          begin
            Inc(s);
            HugeInt_SHL(d, 1);
          end;{ while }
        Dec(s);
        { Make a new copy of b }
        Move(b, d, SizeOf(HugeInt));
        { Shift d as needed }
        HugeInt_ShL(d, S);
        { Use e = -d for addition, faster then  subtracting d }
        Move(d, e, SizeOf(HugeInt));
        HugeInt_Min(e);
        Q := 0;
        { while a >= d do a := a+-d and keep trek of # in Q}
        while HugeInt_Comp(a, d) >= 0 do
          begin
            HugeInt_Add(a, e, a);
            Inc(Q);
          end;{ while }
        { OOps!, one too many subtractions; correct }
        if HugeInt_IsNeg(a) then
          begin
            HugeInt_Add(a, d, a);
            Dec(Q);
          end;{ if }
        HugeInt_SHL(R, 1);
        R[0] := Q;
      until HugeInt_Comp(a, b) < 0;
      if A_IsNeg xor B_IsNeg then HugeInt_Min(R);
    end;{ else }
end;{ HugeInt_Div }

procedure HugeInt_DivMod100(var a: HugeInt; var R: Integer);
{ This works on positive numbers only
  256-Based division: R := a mod 100; a:= a div 100; }
var
  Q: HugeInt;
  S: Integer;
```

```
begin
  R := 0; FillChar(Q, SizeOf(Q), 0);
  S := HugeInt_HCD(a);
  repeat
    r := 256*R + a[S];
    HugeInt_SHL(Q, 1);
    Q[0] := R div 100;
    R := R mod 100;
    Dec(S);
  until S < 0;
  Move(Q, a, SizeOf(Q));
end;{  HugeInt_DivMod100 }

procedure HugeInt_Div(a, b: HugeInt; var R: HugeInt);
begin
  HugeInt_DivMod(a, b, R);
end;{ HugeInt_Div }

procedure HugeInt_Mod(a, b: HugeInt; var R: HugeInt);
begin
  HugeInt_DivMod(a, b, R);
  Move(a, R, SizeOf(HugeInt));
end;{ HugeInt_Mod }

procedure HugeInt2String(a: HugeInt; var S: string);
  function Str100(i: Integer): string;
  begin
    Str100 := Chr(i div 10 + Ord('0')) + Chr(i mod 10 + Ord('0'));
  end;{ Str100 }
var
  R:      Integer;
  Is_Neg: Boolean;
begin
  S := '';
  Is_Neg := HugeInt_IsNeg(a);
  if Is_Neg then HugeInt_Min(a);
  repeat
    HugeInt_DivMod100(a, R);
    Insert(Str100(R), S, 1);
  until HugeInt_Zero(a) or (Length(S) = 254);
  while (Length(S) > 1) and (S[1] = '0') do Delete(S, 1, 1);
  if Is_Neg then Insert('-', S, 1);
end;{ HugeInt2String }

procedure String_DivMod256(var S: string; var R: Integer);
{ This works on Positive numbers Only
  10(00)-based division: R := S mod 256; S := S div 256 }
var Q: string;
begin
  FillChar(Q, SizeOf(Q), 0);
  R := 0;
  while S <> '' do
    begin
      R := 10*R + Ord(S[1]) - Ord('0'); Delete(S, 1, 1);
      Q := Q + Chr(R div 256 + Ord('0'));
      R := R  mod 256;
    end;{ while }
```

```
  while (Q <> '') and (Q[1] = '0') do Delete(Q, 1, 1);
  S := Q;
end;{ String_DivMod256 }

procedure String2HugeInt(AString: string; var a: HugeInt);
var
  i, h:   Integer;
  Is_Neg: Boolean;
begin
  if AString = '' then AString := '0';
  Is_Neg := AString[1] = '-';
  if Is_Neg then Delete(Astring, 1, 1);
  i := 0;
  while (AString <> '') and (i <= HugeIntMSB) do
    begin
      String_DivMod256(AString, h);
      a[i] := h;
      Inc(i);
    end;{ while }
  if Is_Neg then HugeInt_Min(a);
end;{ String2HugeInt }

procedure Integer2HugeInt(AInteger: Integer; var a: HugeInt);
var Is_Neg: Boolean;
begin
  Is_Neg := AInteger < 0;
  if Is_Neg then AInteger := -AInteger;
  FillChar(a, SizeOf(HugeInt), 0);
  Move(AInteger, a, SizeOf(Integer));
  if Is_Neg then HugeInt_Min(a);
end;{ Integer2HugeInt }

end.
```

Borland style buttons

The BORBTNS.PAS unit contains two Delphi components, TBorCheck and TBorRadio which implement the BWCC style CheckBox and RadioButton. I loved the BWCC style buttons so I decided to reproduce them in Delphi. I didn't test them very much, but as far as I know they work properly. However I've enclosed the source code, so you can modify them as you prefer.
The GroupIndex property in TBorRadio allow you to define groups: all the buttons which share the same GroupIndex are mutually exclusive.

Freeware, enjoy them!
(and if you're very happy with them, send me a postcard of your city)

Enrico Lodolo
via F.Bolognese 27/3
40129 Bologna
Italy

CompuServe: 100275,1255
Internet: ldlc18k1@bo.nettuno.it

```
{-------------------------------------------------------------------}
{ BORBTNS - BWCC Style CheckBoxes & Radio Buttons for Delphi v 1.01 }
{-------------------------------------------------------------------}
{ v. 1.00 April, 8 1995                                             }
{ v. 1.01  July, 6 1995  Controls refreshed when caption changes    }
{-------------------------------------------------------------------}
{ Copyright Enrico Lodolo                                           }
{ via F.Bolognese 27/3 - 440129 Bologna - Italy                     }
{ CIS 100275,1255 - Internet ldlc18k1@bo.nettuno.it                 }
{-------------------------------------------------------------------}

unit BorBtns;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Menus;

type
  TBorCheck = class(TCustomControl)
  private
    FDown:Boolean;
    FState:TCheckBoxState;
    FFocused:Boolean;
    FCheckColor:TColor;
  protected
    constructor Create(AOwner: TComponent); override;
    procedure Paint; override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
      X, Y: Integer); override;
```

```
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState;
      X, Y: Integer); override;
    procedure MouseMove(Shift: TShiftState;X, Y: Integer);
      override;
    procedure KeyDown(var Key:Word;Shift:TShiftSTate); override;
    procedure KeyUp(var Key:Word;Shift:TShiftSTate); override;
    procedure SetDown(Value:Boolean);
    procedure SetState(Value:TCheckBoxState);
    procedure SetChecked(Value:Boolean);
    function  GetChecked:Boolean;
    procedure SetCheckColor(Value:TColor);
    function  GetCaption: TCaption;
    procedure SetCaption(const Value:TCaption);
    procedure DoEnter; override;
    procedure DoExit; override;
  public
  published
    property Caption:TCaption read GetCaption write SetCaption;
    property CheckColor:TColor read FCheckColor write SetCheckColor
             default clBlack;
    property Checked:Boolean read GetChecked write SetChecked
             default False;
    property Down:Boolean read FDown write SetDown default False;
    property DragCursor;
    property DragMode;
    property Font;
    property ParentFont;
    property PopupMenu;
    property ShowHint;
    property State:TCheckBoxState read FState write SetState
             default cbUnchecked;
    property TabOrder;
    property TabStop;
    property OnClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDrag;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
  end;

type
  TBorRadio = class(TCustomControl)
  private
    FDown:Boolean;
    FChecked:Boolean;
    FFocused:Boolean;
    FCheckColor:TColor;
    FGroupIndex:Byte;
    procedure TurnSiblingsOff;
  protected
    constructor Create(AOwner: TComponent); override;
    procedure Paint; override;
```

```
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState;
      X, Y: Integer); override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState;
      X, Y: Integer); override;
    procedure MouseMove(Shift: TShiftState;X, Y: Integer);
      override;
    procedure KeyDown(var Key:Word;Shift:TShiftSTate); override;
    procedure KeyUp(var Key:Word;Shift:TShiftSTate); override;
    function  GetCaption: TCaption;
    procedure SetCaption(const Value:TCaption);
    procedure SetDown(Value:Boolean);
    procedure SetChecked(Value:Boolean);
    procedure SetCheckColor(Value:TColor);
    procedure DoEnter; override;
    procedure DoExit; override;
  public
  published
    property Caption:TCaption read GetCaption write SetCaption;
    property CheckColor:TColor read FCheckColor write SetCheckColor
             default clBlack;
    property Checked:Boolean read FChecked write SetChecked
             default False;
    property Down:Boolean read FDown write SetDown default False;
    property DragCursor;
    property DragMode;
    property Font;
    property GroupIndex:Byte read FGroupIndex write FGroupIndex
      default 0;
    property ParentFont;
    property PopupMenu;
    property ShowHint;
    property TabOrder;
    property TabStop;
    property OnClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDrag;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
  end;

procedure Register;

implementation

{--------------------------------------------------------------------}
{                          Borland Style CheckBox                    }
{--------------------------------------------------------------------}

constructor TBorCheck.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  Width := 98;
```

```
  Height := 20;
  ParentColor:=False;
  Color:=clBtnFace;
end;

const BW=12;

procedure TBorCheck.Paint;
var BL,BT,BR,BB:Integer;
    TX,TY,TW,TH:Integer;
    Rect:TRect;
begin
  Canvas.Font:=Font;
  with Canvas do
  begin
    BT:=(Height div 2)-(BW div 2);
    BB:=BT+BW;
    BL:=1;
    BR:=BW+1;
    Brush.Color:=clBtnFace;
    if not FDown then
      begin
        Pen.Color:=clBtnFace;
        Rectangle(BL,BT,BR,BB);
        Pen.Color:=clBtnHighLight;
        MoveTo(BL,BB);
        LineTo(BL,BT);
        LineTo(BR,BT);
        Pen.Color:=clBtnShadow;
        LineTo(BR,BB);
        LineTo(BL,BB);
      end
    else
      begin
        Pen.Color:=clBlack;
        Pen.Width:=2;
        Rectangle(BL+1,BT+1,BR+1,BB+1);
        Pen.Width:=1;
      end;
    TX:=BR+5;
    TY:=(Height div 2)+(Font.Height div 2)-1;
    TW:=TextWidth(Caption);
    TH:=TextHeight(Caption);
    TextOut(TX,TY,Caption);
    case State of
      cbChecked:
      begin
       Pen.Color:=FCheckColor;
        Pen.Width:=1;
        Dec(BT);Dec(BB);
        MoveTo(BL+2,BT+BW div 2+1);
        LineTo(BL+2,BB-1);
        MoveTo(BL+3,BT+BW div 2);
        LineTo(BL+3,BB-2);
        MoveTo(BL+2,BB-1);
        LineTo(BR-2,BT+3);
        MoveTo(BL+3,BB-1);
```

```
        LineTo(BR-1,BT+3);
      end;
      cbGrayed:
      begin
       if Down then
        begin
          Pen.Color:=clBtnFace;
          Brush.Color:=clBtnFace;
          Rectangle(BL+2,BT+2,BR-1,BB-1);
        end;
        Brush.Color:=clBtnShadow;
        Rectangle(BL+2,BT+2,BR-1,BB-1);
      end;
    end;
    Brush.Color:=clBtnFace;
    Rect:=Bounds(TX-1,TY,TW+3,TH+1);
    FrameRect(Rect);
    if FFocused then
      DrawFocusRect(Rect);
  end;
end;

function TBorCheck.GetCaption:TCaption;
var Len:Integer;
begin
  Len := GetTextBuf(@Result, 256);
  Move(Result[0], Result[1], Len);
  Result[0] := Char(Len);
end;

procedure TBorCheck.SetCaption(const Value:TCaption);
var Buffer: array[0..255] of Char;
begin
  if GetCaption <> Value then
    SetTextBuf(StrPCopy(Buffer,Value));
  Invalidate;
end;

procedure TBorCheck.SetDown(Value:Boolean);
begin
  if FDown<>Value then
  begin
    FDown:=Value;
    Paint;
  end;
end;

procedure TBorCheck.SetState(Value:TCheckBoxState);
begin
  if FState<>Value then
  begin
    FState:=Value;
    Paint;
    Click;
  end;
end;
```

```
function TBorCheck.GetChecked: Boolean;
begin
  Result:=State=cbChecked;
end;

procedure TBorCheck.SetChecked(Value:Boolean);
begin
  if Value then State := cbChecked
  else State := cbUnchecked;
end;

procedure TBorCheck.SetCheckColor(Value:TColor);
begin
  FCheckColor:=Value;
  Paint;
end;

procedure TBorCheck.DoEnter;
begin
  inherited DoEnter;
  FFocused:=True;
  Paint;
end;

procedure TBorCheck.DoExit;
begin
  inherited DoExit;
  FFocused:=False;
  Paint;
end;

procedure TBorCheck.MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  SetFocus;
  FFocused:=True;
  inherited MouseDown(Button, Shift, X, Y);
  MouseCapture:=True;
  Down:=True;
end;

procedure TBorCheck.MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  MouseCapture:=False;
  Down:=False;
  if (X>=0) and (X<=Width) and (Y>=0) and (Y<=Height) then
    Checked:=not Checked;
  inherited MouseUp(Button, Shift, X, Y);
end;

procedure TBorCheck.MouseMove(Shift: TShiftState;X, Y: Integer);
begin
  if MouseCapture then
    Down:=(X>=0) and (X<=Width) and (Y>=0) and (Y<=Height);
  inherited MouseMove(Shift,X,Y);
end;
```

```
procedure TBorCheck.KeyDown(var Key:Word;Shift:TShiftSTate);
begin
  if Key=vk_Space then Down:=True;
  inherited KeyDown(Key,Shift);
end;

procedure TBorCheck.KeyUp(var Key:Word;Shift:TShiftSTate);
begin
  if Key=vk_Space then
  begin
    Down:=False;
    Checked:=not Checked;
  end;
end;

{-------------------------------------------------------------------------}
{                            Borland Radio Button                         }
{-------------------------------------------------------------------------}

constructor TBorRadio.Create(AOwner: TComponent);

begin
  inherited Create(AOwner);
  Width := 98;
  Height := 20;
  ParentColor:=False;
  Color:=clBtnFace;
end;

procedure TBorRadio.Paint;

var BL,BT,BR,BB,BM:Integer;
    TX,TY,TW,TH:Integer;
    CX,CY:Integer;
    Rect:TRect;

begin
Canvas.Font:=Font;
with Canvas do
  begin
    BM:=BW div 2;
    BT:=(Height div 2)-BM;
    BB:=BT+BW;
    BL:=1;
    BR:=BW+1;
    Brush.Color:=clBtnFace;
    if Down then
    begin
      Pen.Color:=clBlack;
      MoveTo(BL+BM,BT);
      LineTo(BL,BT+BM);
      LineTo(BL+BM,BB);
      LineTo(BR,BT+BM);
      LineTo(BL+BM,BT);
      MoveTo(BL+BM,BT+1);
      LineTo(BL+1,BT+BM);
```

```
      LineTo(BL+BM,BB-1);
      LineTo(BR-1,BT+BM);
      LineTo(BL+BM,BT+1);
    end
    else
    begin
      Pen.Color:=clBtnFace;
      Rectangle(BL,BT,BR,BB);
      if Checked then Pen.Color:=clBtnShadow
                 else Pen.Color:=clBtnHighLight;
      MoveTo(BL+BM,BT);
      LineTo(BL,BT+BM);
      LineTo(BL+BM,BB);
      if Checked then Pen.Color:=clBtnHighLight
                 else Pen.Color:=clBtnShadow;
      LineTo(BR,BT+BM);
      LineTo(BL+BM,BT);
    end;
    if Checked then
    begin
      Pen.Color:=CheckColor;
      CX:=BL+BM;CY:=BT+BM;
      MoveTo(CX-1,CY-1);
      LineTo(CX+2,CY-1);
      MoveTo(CX-2,CY);
      LineTo(CX+3,CY);
      MoveTo(CX-1,CY+1);
      LineTo(CX+2,CY+1);
      MoveTo(CX,CY-2);
      LineTo(CX,CY+3);
    end;
    TX:=BR+5;
    TY:=(Height div 2)+(Font.Height div 2)-1;
    TW:=TextWidth(Caption);
    TH:=TextHeight(Caption);
    TextOut(TX,TY,Caption);
    Brush.Color:=clBtnFace;
    Rect:=Bounds(TX-1,TY,TW+3,TH+1);
    FrameRect(Rect);
    if FFocused then
      DrawFocusRect(Rect);
  end;
end;

function TBorRadio.GetCaption:TCaption;
var Len:Integer;
begin
  Len := GetTextBuf(@Result, 256);
  Move(Result[0], Result[1], Len);
  Result[0] := Char(Len);
end;

procedure TBorRadio.SetCaption(const Value:TCaption);
var Buffer: array[0..255] of Char;
begin
  if GetCaption <> Value then
    SetTextBuf(StrPCopy(Buffer,Value));
```

```
  Invalidate;
end;

procedure TBorRadio.SetDown(Value:Boolean);
begin
  if FDown<>Value then
  begin
    FDown:=Value;
    Paint;
  end;
end;


var i:Integer;
    Sibling: TBorRadio;
begin
  if Parent <> nil then
    for i:=0 to Parent.ControlCount-1 do
      if Parent.Controls[i] is TBorRadio then
      begin
        Sibling:=TBorRadio(Parent.Controls[i]);
        if (Sibling<>Self) and
           (Sibling.GroupIndex=GroupIndex) then
             Sibling.SetChecked(False);
      end;
end;

procedure TBorRadio.SetChecked(Value: Boolean);
begin
  if FChecked <> Value then
  begin
    TabStop:=Value;
    FChecked:=Value;
    if Value then
    begin
      TurnSiblingsOff;
      Click;
    end;
    Paint;
  end;
end;

procedure TBorRadio.SetCheckColor(Value:TColor);
begin
  FCheckColor:=Value;
  Paint;
end;

procedure TBorRadio.DoEnter;
begin
  inherited DoEnter;
  FFocused:=True;
  Checked:=True;
  Paint;
end;

procedure TBorRadio.DoExit;
```

```
begin
  inherited DoExit;
  FFocused:=False;
  Paint;
end;

procedure TBorRadio.MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  SetFocus;
  FFocused:=True;
  inherited MouseDown(Button, Shift, X, Y);
  MouseCapture:=True;
  Down:=True;
end;

procedure TBorRadio.MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  MouseCapture:=False;
  Down:=False;
  if (X>=0) and (X<=Width) and (Y>=0) and (Y<=Height)
    and not Checked then Checked:=True;
  inherited MouseUp(Button, Shift, X, Y);
end;

procedure TBorRadio.MouseMove(Shift: TShiftState;X, Y: Integer);
begin
  if MouseCapture then
    Down:=(X>=0) and (X<=Width) and (Y>=0) and (Y<=Height);
  inherited MouseMove(Shift,X,Y);
end;

procedure TBorRadio.KeyDown(var Key:Word;Shift:TShiftSTate);
begin
  if Key=vk_Space then Down:=True;
  inherited KeyDown(Key,Shift);
end;

procedure TBorRadio.KeyUp(var Key:Word;Shift:TShiftSTate);
begin
  if Key=vk_Space then
  begin
    Down:=False;
    if not Checked then Checked:=True;
  end;
end;

procedure Register;
begin
  RegisterComponents('Samples',[TBorCheck,TBorRadio]);
end;

end.
```

Q: How do I terminate all running tasks?

A: Below is some code that will help if you want to terminate ALL tasks, no questions asked.

A word of caution, before you run this for the first time, make sure that you save it and anything else that may have some pending data.

```
procedure TForm1.ButtonKillAllClick(Sender: TObject);
var
  pTask   : PTASKENTRY;
  Task    : Bool;
  ThisTask: THANDLE;
begin
  GetMem (pTask, SizeOf (TTASKENTRY));
  pTask^.dwSize := SizeOf (TTASKENTRY);
  Task := TaskFirst (pTask);
  while Task do
  begin
    if pTask^.hInst = hInstance then
      ThisTask := pTask^.hTask
    else
      TerminateApp (pTask^.hTask, NO_UAE_BOX);
    Task := TaskNext (pTask);
  end;
  TerminateApp (ThisTask, NO_UAE_BOX);
end;
```

bitmap pasting error

After pasting an image, which was copied to the clipboard from MS PaintBrush, into a TDBImage control, the image is not saved into the table when the record is posted. This is because the TBlobField doesn't support saving of MetaFile's (see explanation below).

Workaround:
Change the code in DBCTRLS.PAS (TDBImage.PasteFromClipboard - approximately line # 2180) which reads

```
Picture.Assign(Clipboard);
```

to instead read

```
Picture.Bitmap.Assign(Clipboard);
```

method assignment

Q: How do I assign a method to the event of a dynamically created object?

A:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  t: TButton;
begin
  t := TButton.create(application);
  t.parent := form1;
  t.caption := 'New Button';
  t.OnClick := Button2.OnClick;
  t.show;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  showMessage((sender as TButton).caption);
end;

end.
```

humor:

Microsoft
Dr. Seuss
Engineering
programming
Misc.

Star Trek Lost Episodes" transcript.

<Pichard> "Mr. LaForge, have you had any success with your attempts at finding a weakness in the Borg? And Mr. Data, have you been able to access their command pathways?"

<Geordi>"Yes, Captain. In fact, we found the answer by searching through our archives on late Twentieth-century computing technology."

<Geordi presses a key, and a logo appears on the computer screen.>

<Riker looks puzzled.> "What the hell is 'Microsoft'?"

<Data turns to answer.> "Allow me to explain. We will send this program, for some reason called 'Windows', through the Borg command pathways. Once inside their root command unit, it will begin consuming system resources at an unstoppable rate."

<Pichard> "But the Borg have the ability to adapt. Won't they alter their processing systems to increase their storage capacity?"

<Data> "Yes, Captain. But when 'Windows' detects this, it creates a new version of itself known as an 'upgrade'. The use of resources increases exponentially with each iteration. The Borg will not be able to adapt quickly enough. Eventually all of their processing ability will be taken over and none will be available for their normal operational functions."

<Pichard> "Excellent work. This is even better than that 'unsolvable geometric shape' idea."

... . . 15 Minutes Later . . .

><Data> "Captain, We have successfully installed the 'Windows' in the command
>unit and as expected it immediately consumed 85% of all resources. We however
>have not received any confirmation of the expected 'upgrade'."

<Geordi> "Our scanners have picked up an increase in Borg storage and
CPU capacity to compensate, but we still have no indication of an
'upgrade' to compensate for their increase."

<Pichard> "Data, scan the history banks again and determine if their is something we have missed."

<Data> "Sir, I believe their is a reason for the failure in the 'upgrade'.
Apparently the Borg have circumvented that part of the plan by not sending in their registration cards.

<Riker> "Captain we have no choice. Requesting permission to begin emergency escape sequence 3F . . ."

<Geordi, excited> "Wait, Captain I just detected their CPU capacity has suddenly dropped to 0% !"

<Pichard> "Data, what does your scanners show?"

<Data> "Apparently the Borg have found the internal 'Windows' module named 'Solitaire' and it has used up all the CPU capacity."

<Pichard> "Lets wait and see how long this 'solitaire' can reduce their functionality."

... . . Two Hours Pass . . .

<Riker> "Geordi what's the status on the Borg?"

<Geordi> "As expected the Borg are attempting to re-engineer to compensate for increased CPU and storage demands, but each time they successfully increase resources I have setup our closest deep space monitor beacon to transmit more 'windows' modules from something called the 'Microsoft fun-pack'.

<Pichard> "How much time will that buy us ?"

<Data> "Current Borg solution rates allow me to predicate an interest time span of 6 more hours."

<Geordi> "Captain, another vessel has entered our sector."

<Pichard> "Identify."

<Data> "It appears to have markings very similar to the 'Microsoft' logo"

<Over the speakers> "THIS IS ADMIRAL BILL GATES OF THE
MICROSOFT FLAGSHIP
MONOPOLY. WE HAVE POSITIVE CONFIRMATION OF UNREGISTERED
SOFTWARE IN THIS
SECTOR. SURRENDER ALL ASSETS AND WE CAN AVOID ANY
TROUBLE. YOU HAVE 10
SECONDS"

<Data> "The alien ship has just opened its forward hatches and released thousands of humanoid shaped objects."

<Pichard> "Magnify forward viewer on the alien craft"

<Riker> "Good God captain! Those are humans floating straight toward the Borg ship with no life support suits ! How can they survive the tortures of deep space ?!"

<Data> "I don't believe that those are humans sir, if you will look closer I believe you will see that they are carrying something recognized by twenty-first century man as doe skin leather briefcases, and wearing Armani suits"

<Riker and Pichard together horrified> "Lawyers !!"

<Geordi> "It can't be. All the Lawyers were rounded up and sent hurtling into the sun in 2017 during the Great Awakening."

<Data> "True, but apparently some must have survived."

<Riker> "They have surrounded the Borg ship and are covering it with all types of papers."

<Data> "I believe that is known in ancient vernacular as 'red tape' it often proves fatal."

<Riker> "They're tearing the Borg to pieces !"

<Pichard> "Turn off the monitors. I can't stand to watch, not even the Borg deserve that."

<Pichard> "Mr. LaForge, what's the current status on the Borg ship?"

<Geordi> "They are still undergoing heavy attack from Microsoft's Shark team. Wait! They lawyers backing off...It looks like the Borg must have negotiated a site wide license."

<Pichard> "Damn! Data, what's your analysis?"

<Data> "Sir, I am reading some interesting program changes in the Borg's command pathways. They are spending an enourmous amount of effort evaluating a program left by Microsoft."

<Riker> "What are you talking about Data?"

<Data> "It appears to be a new program, I am trying to isolate it's description. The Microsoft registry reports the new program as Win stardate 7451332 Build 455."

<Pichard> "Well, that should take care of them."

<Geordi> "I don't think so sir. The Borg have managed to isolate this new program to

only a small part of their collective intelligence. They are referencing this node as an 'evaluation team'."

<Pichard> "How long until the program proliferates Mr. LaForge?"

<Geordi> "It's hard to tell Captain. But from the amount of sub-space communication being transmitted to the Microsoft space station 'Help Desk', I'd say it will be a while before th..."

<Data> "Sir, the Borg ship is rapidly regaining resources. I estimate 2 minutes before they will be able to attack."

<Pichard> "Options"

<Riker> "We could offer this new Windows to other members of the Borg."

<Pichard> "Sell them a Beta version of a product? That goes against the Prime Directive. Besides, Number 1, it would take too long to install...we don't have the time."

<Geordi> "I've got it!! Data, insert this program into the Borg's command pathways."

<Data> "Very interesting...Initiating transfer...Sir, the Borg have completely stopped working on restoring their systems."

<Riker> "What is it? What did you do?"

<Pichard> "It's going to be OK Number 1. <grin> Geordi, did you send them the program I think you sent?"

<Geordi> "Yes sir, Netscape v1.1

MICROSOFT UNVEILS NEW JOE-BOB(tm) SOFTWARE

REDMOND, Wash. -- April 10, 1995 -- Microsoft today announced the release of Joe-Bob(tm), a new software package that the company hopes will open up a huge untapped computer market. With the motto "The software for the rest of y'all(tm)," Joe-Bob reaches out to the same demographic group that buys
4x4s, supports the gun lobby, and drinks Miller Lite.

"Computers have been commonly seen as for leftists and intellectuals," explains Microsoft spokesperson Willy Maclean, "but we've recently seen people like Newt Gingrinch embracing new technology -- the time is right for the rest of America to get wired!"

Instead of a desktop or office metaphor, Joe-Bob(tm) puts the user in a garage. "Click on the Lynyrd Skynyrd tapes, and get a complete music library in digital stereo. Click on the pinups, and get hooked up to the Internet's hottest gifs," the promotional materials explain.

The package does not include a word processor or spreadsheet, but does have software that keeps track of the football season, lists the best roadhouses between Florida and Nevada, and can even order spareribs and beer at the click of a mouse.

"This is righteous software, man," says beta-tester Billy Grugg. "It thinks like I think." Brad Cunningham agrees: "I take it everywhere," he says, pointing to a Pentium laptop racked under his 12-gauge in his pickup truck.
Microsoft is offering desktop users a special clip-on beer holder for their monitors.

"Look at what's popular out there," says Microsoft Chairman Bill Gates.
"Four of the top-10 Usenet newsgroups are about sex, and splatter video games like Doom and Mortal Kombat are bestsellers. We're just catering to a demand, that's all."

Microsoft is reportedly distributing badges and bumper stickers saying things like "Joe-Bob: Make Your Disk Hard," "Go Microsoft -- Go Intel -- Go America," and "QuickTime is for Pinko Hippie Wimps."

Apple declined to comment.

What if people knew (only) as much about the cars they buy, as they know about the computers they buy?

General Motors doesn't have a help line for people who don't know how to drive. Imagine if they did ... (Think of a computer software or hardware helpline)

HelpLine: "General Motors HelpLine, how can I help you?"
Customer: "I got in my car and closed the door and nothing happened!"
HelpLine: "Did you put the key in the ignition slot and turn it?"
Customer: "What's an ignition?"
HelpLine: "It's a starter motor that draws current from your battery and turns over the engine."
Customer: "Ignition? Motor? Battery? Engine? How come I have to know all these technical terms just to use my car?"

HelpLine: "General Motors HelpLine, how can I help you?"
Customer: "My car ran fine for a week and now it won't go anywhere!"
HelpLine: "Is the gas tank empty?"
Customer: "Huh? How do I know?"
HelpLine: "There's a little gauge on the front panel with a needle and markings from 'E' to 'F'. Where is the needle pointing?"
Customer: "It's pointing to 'E'. What does that mean?"
HelpLine: "It means you have to visit a gasoline vendor and purchase some more gasoline. You can install it yourself or pay the vendor to install it for you."
Customer: "What? I paid $12,000 for this car! Now you tell me that I have to keep buying more components? I want a car that comes with everything built in!"

HelpLine: "General Motors HelpLine, how can I help you?"
Customer: "Your cars suck!"
HelpLine: "What's wrong?"
Customer: "It crashed, that's what wrong!"
HelpLine: "What were you doing?"
Customer: "I wanted to run faster, so I pushed the accelerator pedal all the way to the floor. It worked for a while and then it crashed and it won't start now!"
HelpLine: "It's your responsibility if you misuse the product. What do you expect us to do about it?"
Customer: "I want you to send me one of the latest version that doesn't crash any more!"

HelpLine: "General Motors HelpLine, how can I help you?"
Customer: "Hi, I just bought my first car, and I chose your car because it has automatic transmission, cruise control, power steering, power brakes, and power door locks."
HelpLine: "Thanks for buying our car. How can I help you?"
Customer: "How do I work it?"

HelpLine: "Do you know how to drive?"
Customer: "Do I know how to what?"
HelpLine: "Do you know how to drive?"
Customer: "I'm not a technical person. I just want to go places in my car!

Billy the Chilly
(With Apologies to Dr. Seuss)

In a far away island of Redy-Mond-Ross,
Billy the Chilly was king of the DOS.
A nice little DOS with a great big Window
Where programs of all sorts would come and would go.
The vendors flocked to it -- for none were afraid.
From that DOS and that Window much money they made.

They did until Billy, the king of that clutch,
Decided the vendors were making too much.
"I'm ruler," said Billy, "of all that I own.
But I don't own enough," he let out with a groan.
"I own the DOS and the Window -- that's true.
Then how come my spreadsheet is still number two?
I make the most money," he said with no glee.
"But no one should make any money but me.
I must have it all, whether Big Blue or clone.
What a king! I'd be ruler of all that I own."

So Billy the Chilly his minions did hail,
And Billy, the Chilly king, sent some e-mail;
He ordered nine vendors to give him their code --
To put it in DOS, not to lighten their load.
"If you give me your programs," he said with a smile,
"When I ruin your market, I'll do it in style."
Then Billy put all of those programs in DOS,
And said "Of defraggers and backups, I'm Boss."

"All mine!" Billy cried, and he started to sway.
"I'll control all the apps!" And he shouted "OLE!
From former King Blue, finished now my divorce is;
While word and Excel will use all the resources.
I'll buy out that Fox, and I'll reap what he's sown,
For I am the ruler of all that I own."

Then Billy cried "No one can sell a PC,
Unless he is willing to pay me a fee."
But as he was speaking, he heard with great dread
A meek little voice coming from a mild Fed.
"Excuse me, great King, I wish not to alarm,
But I think there's a danger you'll do us all harm.
Be nice, and please tell us you never would cheat,
And that other vendors can truly compete.
Please tell us, King Billy, so we won't think of suing,

That your right hand knows not what your left hand is doing."

"SILENCE!" yelled Billy, his face a bright red.
"I'm king, and you're only a meek little Fed.
We've worked much too hard to let you guys demote us;
I'm bigger than Novell, I'm bigger than Lotus!
Get out of my way; it's a shame you can't see
That your boss was elected to serve men like me.
You'll never get me 'cause my bandwidth is tough.
And I've got the power! Though not near enough."

Then Billy, he smiled and explained what he meant,
"I just want my fair share -- that's one hundred percent.
If it takes a computer, I must have no equal
In spreadsheets, games, CDs, words, BASIC, or SQL.
Home finance is one place where I really blew it --
But that doesn't matter; I'll just buy Intuit."

But that meek little Fed made a meekish attack.
He asked "Have you settled with that fellow, Stac?"
"I have," laughed King Billy, "it ended just fine.
I bought part of Stac; he won't get out of line."
Then the Fed humbly asked as he fell in a swoon
"Can you say why your apps all had OLE so soon?"
Then Billy the Chilly switched to angry mode;
"Are you saying I let myself read my own code?
Such things do not happen, and I don't like your tone.
For I am the ruler of all that I own."

But as he was planning himself to enshrine,
He noticed that millions were going online.
"If they talk will their talk be a squawk that goes my way?
I must buy control of the Info-Net Highway.
I'll build my own turnpike, and I'll charge the toll.
And what is said on it, that I will control.
Millions will use it; my network will thrive,
I'll make it a part of Windows 95."

Then the meek little Fed made a meek little noise.
"Perhaps we'll agree to let you keep your toys.
A big courtroom battle we'd hate to prolong,
So let us just say that you've done nothing wrong."
"I like that," said Bill, "And to make it quite plain,
What I haven't done, I won't do it ... again."

So Billy shook hands with the meek little Fed,

And signed an agreement that left him ahead.
The Fed smiled at Billy and thought them both blessed.
But one little judge found it hard to digest.
He thought about Billy as his stomach, it turned.
And that little judge -- well, his money, he earned.
For that little judge did a curious thing:
He decided,
And thus shook the throne of the king.

And Billy the Chilly, the king of the DOS,
The king of Excel, the NT albatross,
The king of Encarta and that C++ tool ...
Well, that was the end of the Chilly King's rule!
For Billy, he failed, then retried to abort,
Fell out of his Office and plunk into court!

And today the great Billy, who never atones,
Is King of QBASIC, that's all that he owns.
And the vendors and users, well all are now free.
Don't you wish, in this world, that's the way it could be?

Top 20 Engineers' Terminologies

1. A NUMBER OF DIFFERENT APPROACHES ARE BEING TRIED
- We are still spitting in the wind.

2. EXTENSIVE REPORT IS BEING PREPARED ON A FRESH APPROACH TO THE PROBLEM
- We just hired three kids fresh out of college.

3. CLOSE PROJECT COORDINATION
- We know who to blame.

4. MAJOR TECHNOLOGICAL BREAKTHROUGH
- It works OK, but looks very hi-tech.

5. CUSTOMER SATISFACTION IS DELIVERED ASSURED
- We are so far behind schedule the customer is happy to get it delivered.

6. PRELIMINARY OPERATIONAL TESTS WERE INCONCLUSIVE
- The darn thing blew up when we threw the switch.

7. TEST RESULTS WERE EXTREMELY GRATIFYING
- We are so surprised that the stupid thing works.

8. THE ENTIRE CONCEPT WILL HAVE TO BE ABANDONED
- The only person who understood the thing quit.

9. IT IS IN THE PROCESS
- It is so wrapped up in red tape that the situation is about hopeless.

10. WE WILL LOOK INTO IT
- Forget it! We have enough problems for now.

11. PLEASE NOTE AND INITIAL
- Let's spread responsibility for the screw up

12. GIVE US THE BENEFIT OF YOUR THINKING
- We'll listen to what you have to say as long as it doesn't interfere with what we've already done.

13. GIVE US YOUR INTERPRETATION
- I can't wait to hear this BS!

14. SEE ME or LET'S DISCUSS
- Come into my office, I'm lonely.

15. ALL NEW
- Parts not interchangeable with the previous design.

16. RUGGED
- Too damn heavy to lift!

17. LIGHTWEIGHT
- Lighter than RUGGED.

18. YEARS OF DEVELOPMENT
- One finally worked.

19. ENERGY SAVING
- Achieved when the power switch is off.

20. LOW MAINTENANCE
- Impossible to fix if broken.

YET ANOTHER LAYPERSON'S GUIDE TO PROGRAMMING LANGUAGES

C: You shoot yourself in the foot.

C++: You accidentally create a dozen instances of yourself and shoot them all in the foot. Providing emergency assistance is impossible since you can't tell which are bitwise copies and which are just pointing at others and saying, "That's me, over there."

Fortran: You shoot yourself in each toe, iteratively, until you run out of toes, then you read in the next foot and repeat. If you run out of bullets, you continue anyway because you have no exception-handling ability.

Modula-2: After realizing that you can't accomplish anything in this language, you shoot yourself in the head.

COBOL: USEing a COLT 45 HANDGUN, AIM gun at LEG.FOOT, THEN place ARM.HAND.FINGER on HANDGUN.TRIGGER and SQUEEZE, THEN return HANDGUN to HOLDSTER. CHECK whether shoelace needs to be retied.

LISP: You shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds the gun with which you shoot yourself in the appendage which holds ...

BASIC: Shoot yourself in foot with water pistol. On big systems, continue until entire lower body is waterlogged.

FORTH: Foot in yourself shoot.

APL: You shoot yourself in the foot, then spend all day figuring out how to do it in fewer characters.

Pascal: The compiler won't let you shoot yourself in the foot.

SNOBOL: If you succeed, shoot yourself in the left foot. If you fail, shoot yourself in the right foot.

Concurrent Euclid: You shoot yourself in somebody else's foot.

HyperTalk: Put the first bullet of the gun into the left of leg of you. Answer the result.

Motif: You spend days writing a UIL description of your foot, the trajectory, the bullet, and the intricate scrollwork on the ivory handles of the gun. When you finally get around to pulling the trigger, the gun jams.

Unix: % Is foot.c foot.h foot.o toe.c toe.o % rm *.o rm: .o: No such file or directory % is %

DOS: You can't get to either foot from here.

OS/2: Point to Body and click, point to leg and click, point to lower leg and click, point to foot and gun goes click.

Xbase: Shooting yourself is no problem. If you want to shoot yourself in the foot, you'll have to use Clipper.

Paradox: Not only can you shoot yourself in the foot, your users can too.

Revelation: You'll be able to shoot yourself in the foot, just as soon as you figure out what all these bullets are for.

Visual Basic: You'll shoot yourself in the foot, but you'll have so much fun doing it that you don't care.

Prolog: You tell your program you want to be shot in the foot. The program figures out how to do it, but the syntax doesn't allow it to explain.

370 JCL: You send your foot down to MIS with a 4000-page document explaining how you want it to be shot. Three years later, your foot comes back deep-fried.

A Girl's Guide to Geek Guys

So, your crush on the bass player from Vibrating Sandbox has finally died a whimpering death and you're wondering where to go from here. All the sinister dudes are either dating a series of interchangeable high-school riot grrrrls in baby doll dresses and an overdose of manic panic hair dye, or permanently shacked up with some bitter old lady who pays all the bills. Which will it be, a wifely prison or a humiliating one night stand? Into this void of potential mates comes a man you may not have considered before, a man of substance, quietude and stability, a cerebral creature with a culture all his own. In short, a geek.

Why Geek Dudes Rule

- They are generally available.
- Other women will tend not to steal them.
- They can fix things.
- Your parents will love them.
- They're smart.

Where The Geek Dude Lurks

While they are often into alternative music, geek dudes tend not to go to shows too often. Instead you'll find them hanging out with their friends, discussing the latest hardware revolution or perfecting their Bill Gates impressions. You know how some people wear t-shirts with their favorite bands on them, thus showing that they went to certain shows? Well, geek dudes wear t-shirts with the logos of different software companies on them, thus showing that they are up on the latest, um, releases. A small, though convivial, rivalry may be detected here amongst the geek dudes. Try wearing one yourself and see if he strikes up a conversation.

Of course the best way to meet a geek dude is through the Internet. All geeks harbor a secret fantasy about meeting some girl in cyberspace, carrying on an e-mail romance in which he has the chance to combine an activity he is comfortable with, computing, with one he is very uncomfortable with, socializing. To many geek dudes, cyberdating is just an advanced form of some kind of video game, but they are frustrated by a lack of players. Their lack is your strength.

Imprinting

You might notice that these men harbor some strange ideas about how the world works and some particularly strange ideas about women. There is a reason for this. Because they've had limited interpersonal experience, geek dudes must look elsewhere for behavior models. Lacking a real world social milieu, geeks often go through a transference stage with such narratives, and try to model their interactions on them. Thus, certain media images and themes come to have an overly cathected, metaphorized reality to them, while the rest of us view such programming as mere entertainment. Case in point, our next topic...

The Trek factor

If you're not up on your Star Trek, you can forget about getting or keeping a geek dude. And I'm not just talking vintage-era Captain Kirk and Spock either. You've got to be up on your The Next Generation, your Deep Space Nine, your Babylon 5, and let us not forget Voyager. Armed with your own knowledge of Federation policies, you can better gauge when and how to act. The sexual politics of Star Trek are pretty blunt: the men run the technology and the ship, and the women are caretakers (a doctor and a counselor). Note the sexual tensions on the bridge of the Enterprise: the women, in skin tight uniforms, and with luxuriant, flowing hair. The men, often balding, and sporting some sort of permanently attached computer auxiliary. This world metaphorizes the fantasies of the geek dude, who sees himself in the geeky-but-heroic male officers and who secretly desires a sexy, smart, Deanna or Bev to come along and deferentially accept him for who he is. If you are willing to accept that this is his starting point for reality, you are ready for a geek relationship.

Once You've Nabbed Him

Of course, catching that geek guy is only half the battle. Keeping him by your side is another story altogether. I was privileged to speak with Miss Victoria Maat, who not only got herself a geek guy but was also clever enough to marry him just a few short months ago. She interrupted her newlywed bliss to give us a few tips on the care and feeding of a geek man:

Geeks are sensitive and caring lovers and husbands. If you can hang with the techno-lifestyle, they make the best mates. They are the most attractive people, not flashy or hunky, but the kind who get cuter and more alluring over time (I told you she was a newlywed). Definitely give geeks a chance.

Geek Cuisine

Geeks tend towards packaged, junk foods since they prefer to work and think and aren't all that into cooking for themselves. Make sure that your geek understands that you are not merely a replicator, and provide him with home cooked food. A batch of

chocolate chip cookies will let him know that you love him. You do have to monitor your geek for weight gain; however, remember that most of their days are spent sitting and staring at a monitor.

Geek Lifestyle

The geek dude has long work habits and tends to bring his work home with him. He seems permanently connected to his hard disk. You must at least appear interested in his work. Generally, a solid understanding of the computer is a must; if you cannot master this, you should at least be able to talk the talk. Remember most geeks are anal and they get stressed about details which appear insignificant. Be understanding, put on your best Deanna Troi face (see above) and empathize.

To relax, geeks love to play the latest computer games. Let him play Myst or Chuck Yeager's Air Combat for hours if he wants to. Act concerned if he's stuck or has just been ambushed by three MiGs. My geek loves to try to help people on the Internet who say that they are stuck in Myst. He comes up with clever riddles instead of directing them point blank. Geeks also like to go to sci-fi and Japanese animated movies, again, a basically harmless vent for your man.

Geek Buddies

Many geeks extend their work friendships into what they jokingly refer to as RL (Real Life, also known as "that big room with the ceiling that is sometimes blue and sometimes black with little lights"). The greatest thing about your geek's buddies is that you can feel secure in setting them up with your girlfriends. They may feel awkward around females at first, so don't overwhelm them. In time they will come out of their shell and realize that you are into the same things they are.

Post-It Note

I thank Victoria for the above advice. I must say that when she read my draft of the piece, before writing her section, she asked her husband which one he thought she was more like, Deanna or Beverly. Howard, the devil, immediately replied that he had always thought Victoria was actually most like Ensign Ro Laren, a cute character with a slight authority problem who is always in trouble (this is fairly apt). This exchange is interesting for several reasons:

- o Howard had already thought about who she was most like.
- o He could summon up characters from seasons past with ease.
- o Victoria actually knew who he meant.

Folks, I think this marriage will last.

One Last Thing

Because they have been so abused and ignored by society, many geeks have gone underground. You may actually know some and just haven't noticed them. They often feel resentful, and misunderstood, and it is important to realize this as you grow closer to them. Don't ever try to force the issue, or make crazy demands that he choose between his computer and you. Remember, his computer has been there for him his whole life; you are a new interloper he hasn't quite grasped yet.

Geek dudes thrive on mystery and love challenges and intellectual puzzles. Don't you consider yourself one? Wouldn't you like a little intellectual stimulation or your own? We thought so.

A Grandchild's guide to using Grandpa's computer
(Ode to Dr. Seuss, with apologies to 'Fox in Sox')

Bits. Bytes. Chips. Clocks.
Bits in bytes on chips in box.
Bytes with bits and chips with clocks.
Chips in box on ether-docks.

Chips with bits come.
Chips with bytes come.
Chips with bits and bytes and clocks come.

Look, sir. Look, sir.
Read the book, sir.
Let's do tricks with bits and bytes, sir.
Let's do tricks with chips and clocks, sir.

First, I'll make a quick trick bit stack. Then I'll make a quick trick byte stack.
You can make a quick trick chip stack.
You can make a quick trick clock stack.

And here's a new trick on the scene.
Bits in bytes for your machine.
Bytes in words to fill your screen.

Now we come to ticks and tocks, sir.
Try to say this by the clock, sir.

Clocks on chips tick.
Clocks on chips tock.
Eight byte bits tick.
Eight bit bytes tock.
Clocks on chips with eight bit bytes tick Chips with clocks and eight byte bits tock.

Here's an easy game to play.
Here's an easy thing to say.

If a packet hits a pocket on a socket on a port,
And the bus is interrupted as a very last resort,
And the address of the memory makes your floppy disk abort,
Then the socket packet pocket has an error to report!

If your cursor finds a menu item followed by a dash,
And the double-clicking icon puts your window in the trash,
And your data is corrupted 'cause the index doesn't hash,
Then your situation's hopeless and your system's gonna crash.

You can't say this?
What a shame, sir!
We'll find you another game, sir.

If the label on the cable on the table at your house
Says the network is connected to the button on your mouse,
But your packets want to tunnel on another protocol,
That's repeatedly rejected by the printer down the hall,
And your screen is all distorted by the side effects of gauss
So your icons in the window are as wavy as a souse,
Then you may as well reboot and go out with a bang,
'Cause as sure as I'm a poet, the sucker's gonna hang!

When the copy of your floppy's getting sloppy on the disk,
And the microcode instructions cause unnecessary risc,
Then you have to flash your memory and you'll want to ram your ROM.
Quickly turn off your computer and be sure to tell your mom!

Heaven's donuts are jelly donuts. The blend of texture, from the cool, sweet ooze of the jelly, mined with tiny rasberry seeds, to the firm, spongy cake, so lightly encrusted in a thin glaze of sugar, that cracks and flakes as you gingerly tear off small pieces of delight, is certainly the greatest experience a humble man can afford.

I was eating a jelly donut when he first appeared in my office, smelling slightly of gunpowder. He was tall and gaunt, with deep-set eyes and crooked teeth, long, delicate fingers, and sloped shoulders.

He wore a black Ozzy Osborne concert t-shirt, frayed black jeans, and dusty black high-tops, unlaced. He smiled at me in an ugly way. I put down my donut and glanced at my watch. 7:00 PM.

"You're David Webster."

I nodded.

"You're a programmer for Core."

I nodded again. Not only was I a programmer for Core--I was the best damn programmer this group had ever or would ever see. I suppose
I should introduce myself. I am David Elijah Webster, master programmer. I'm not just blowing smoke here either. I'm the best damn programmer to come out of MIT since code was constructed one bit at a time. I can do it all: C, LISP, assembly--even the languages no self-respecting programmer would deign to look at. I can do it all in no time flat, with the most elegant of style. Code sprinkled with glistening semicolons and flowing rivers of indentation. Lesser programmers avert their eyes when I enter the room.

"They say you're the best, and I'm here to challenge you."

I sized this guy up again. He had the right shape. The pot-belly, the greasy hair, parted with percision. The fingers. And the funny smell.

I told him I didn't have time.

"I'll make it worth your while," he said. "I have something you might be interested in. Follow me."

I grabbed my box of donuts, and followed him down the hall and into the elevator. He pressed a button and the elevator descended into the basement. I'd never been in the basement before. For that matter, I didn't even recall that the building had a basement. Nonetheless, the elevator chimed, the doors opened, and we stepped out into a wide room that was entirely featureless. That is, except for the fog on the floor and two workstations that were set up, side by side. One of the workstations was mine. The other was a workstation like none other that I had seen before. It was magnificent.

It was matte black. More than an object, it looked like a hole in space. The monitor it sported was the biggest I had ever seen, and the keyboard was a flow of liquid lines, containing a field of keys of different sizes and shapes, packed in like cobblestones. The mouse floated above the table, and had no wire. Next to the computer was a box with a small chute coming out of one side, and a large red button on the top. The monitor was flanked by two gigantic speakers, and I could see a sub-woofer poking up out of the fog. It hummed. It steamed. It was the most beautiful computer I had ever seen.

"You approve," said the stranger.

I swallowed and said, "It is beyond description."

"It's a custom job. And it's yours. If," he said, "If you can beat me in a coding

contest."

I looked at him incredulously. "What's in it for you?"

"I will have defeated the greatest coder in the world, and thus, I can claim that title. AND, I get to keep your immortal soul."

He smiled the ugly smile again.

Here was a dilemma. I was dealing with the Devil. There was no doubt about that. And he was no doubt very good. I am somewhat attached to my soul, but oh, the prizes. The glory. I can easily claim to be the best coder in the company, in the Bay Area, probably on the whole planet, but if I pulled this off, I will have shown myself to be the best coder in this entire theology! Vanity got the better part of me.

"What's the contest?" I asked.

I won't bore you with the details, but it was seriously ugly. Ugly in a way that makes the most arrogant of coders cringe and causes managers to pad development schedules into the next century. It had to run in any language, including the nasty chicken-scratch ones. It had to be backward compatible all the way to the ENIAC. And it had to run on Windows. I cringed.

But vanity won. I signed the forms, agreed on a deadline of midnight, and we sat down at our machines and started to code.

My watch said 8:00 PM, and I started warming up. Class definitions flew off my fingertips like throwing stars. Structures and declarations grew like perfect crystals, and I didn't even break a sweat. True to the task, I soon lost myself in an endless cycle of postulate, create, instantiate and verify. Bits grew to bytes, to K, to Megs, and finally to Gigs. By 11:00 PM it had come to that crucial point. With an hour to go, I had to put all the peices together. It wasn't going to be easy. It was going to take all the concentration I had.

Then I hit the first bug.

At first, I wasn't sure where it was coming from, but then I spotted it. It wasn't mine. It was bug in Windows. Even worse, it was a bug in Windows that stemmed from a timing problem with the system clock itself. I couldn't see a workaround. I was stymied. I genuflected and called Microsoft support.

"Hello, and welcome to the Microsoft help line. Please enter your 64 digit user identification number, followed by your 32 digit password."

While I frantically typed number after number, trying to navigate through layer upon layer of phone menu, I heard him pick up his phone and call a number.

"Hello, is Bill in? ... I don't care, wake him up ... Tell him it's
Mr. Black ... Hey Bill, what's shakin'? Listen, I needed to know a workaround to one of your bugs ... Yes, I know what time it is
... Yes, I know ... Bill ... Bill! You remember our little deal?
... That's right. Now be a dear and give me that workaround ... Mm-hm
... Right ... Thank you, Bill. I'll be seeing you."

I was shocked. It was obviously pointless continuing my desperate journey through Microsoft's Help line. I needed immediate genius! I scarfed down a grape jelly. Sugar shock engulfed me, and my vision tunneled. I shuddered once, something clicked, and I determined the answer I needed--I could use the clock on the sound chip to get my timings.

I dove back into the code, and was quickly integrating modules when I hit bug number

two. It was even uglier than the first. In fact, it was the ugliest bug I had ever seen. It was a problem with C. With the language itself. There's no way fix a broken hammer using the same hammer.

I wracked my brains. I clenched and grunted and sweated and thought and Thought and THOUGHT, but to no avail. Over my shoulder, I could hear Him chime in, "Bugger, isn't it? I remember putting that one in back when I was working on the Unix kernal. Did you really think there was a Kernighan and Ritchie? Rearrange the letters in their names and you'll discover an interesting anagram."

I ignored him and continued thinking. My mind went deeper and deeper into the problem at hand--my senses dulled, my breathing grew shallow.
My eyes rolled back and sweat beaded on my forehead. Clumsily, blindly, my hand pawed it's way to the box on my desk, containing my last jelly donut. It raised slowly to my lips, and I bit.

Pounding waves of sugar induced euphoria washed through my mind. I felt my brain hum and crackle. My hands trembled, my body shuddered, and I began to type. I was a man possessed. Complex topographical math equations formed on my screen. Klien bottles and hypercubes locked neatly into place like pieces of a puzzle. Beyond my control, a complex mathematical world formed in my computer, with additional dimensions unimaginable.

I felt a small pop, and I came to. I looked at my screen. I had worked around the bug.

My watch read 11:45. Frantically I continued putting all the modules into place. Glancing for a moment at my rival, I could see I had him worried. He was typing furiously. Smoke poured from his ears, and flames licked around his collar.

Then I hit the third bug.

It was not so much a bug, it was a limit. I only had 4 Gigabytes of memory, and I had used it all. There wasn't a bit left. I had compressed data to a point so fine that it was in danger of collapsing into a black hole. I was storing memory in every conceivable way, including keeping a chain of sound waves running between the speaker and the microphone. There was no memory left to be had.

Frantic, I reached into my box of donuts, and my heart sank into my stomach when I realized that I had eaten the last one. I glanced at my watch, but it was too late. I was sunk. I had done the best that I could, and I had nothing more to give.

The Devil laughed, and grinning cruelly, he reached over to the box with the chute and the button. Remember the box? Slowly, firmly, his hand pressed the red button, and a jelly donut slid down the chute and onto the table.

My jaw dropped. "What...is...that?" I asked.

He languorously chewed as he replied, "The Box of Eternal Donuts."

"The Box of Eternal Donuts!?"

"Yes," he said.

"It never runs out?"

"Never," he said.

"It's mine if I win?!?!"

"If you can win, it is entirely yours," he replied, grinning cockily.

My mind reeled. The Box of Eternal Donuts. The Box of Eternal Donuts! My eyes darted everywhere, my jaw hung slack, and my throat emitted strange animal-like

noises. Anything. I would do anything to win! I just needed the smallest amount of memory. But where could I get it from? I glanced at my watch again, and a plan came into my mind. A beautiful, devious plan.

I went quickly upstairs and retrieved the emergency toolkit that we keep in the medicine cabinet. I ripped the case off my computer, and quickly scanned for the right connections. I pulled two wires, and unscrewed the back of my watch. The Devil's eyes widened and he desparately started coding again, but it was too late. I got the last of the memory I needed out of my watch, and pressed the ENTER key seconds before he did.

The watch burst into flames. Sparks flew from the disk drives and the monitor glowed and throbbed, finally melting into a puddle of glass. The computer exploded in a shower of sparks, and then there was absolute silence.

There was a pause, and both of us turned as the printer started, slowly emitting a single sheet that wafted gently into the out bin. I nonchalantly strolled over, and held up to the Devil's scowling face, a sheet imprinted with two words. "Hello World".

Nothing more needs to be told, other than, as I write this, I am sitting in front of my new computer, munching on what is undoubtedly the best jelly donut I have ever eaten.

It was a typical morning at the office: 53 new e-mail messages, 86 games of telephone tag lobbed into my court, and a mass of Post-its reminding me of about 18 missed deadlines. So I knew exactly what to expect when my boss appeared.

"Good morning," she said cheerfully. "I need complete information on how soy cheese is affecting Wisconsin's trucking industry. Drop everything else until you get this done." She disappeared as quickly as she had materialized.

I considered my situation. I would have to put off finishing the Forbin Project, for which I'd dropped the Morbius Proposition the day before. Of course, Morbius had forced me to set aside the Kinsey Report, for which I had delayed ...? I had trouble remembering farther back than that.

Luckily, I had a new weapon to help in my research project: The World Wide Web. With the help of my new Web browser, Odysseus, I knew I could sneak into any server, siphon off the key data I needed, then find my way home in no time.

I launched Odysseus, entered my password, clicked the button to go online, and waited as my modem dialed, made contact, hissed at my Internet server, and exchanged more civilized protocols. Then I repeated the process, using the right password.

The second time around, Odysseus successfully made contact with the World Wide Web, affording me a chance to wait some more. In the upper left hand corner, a small icon of a Bronze Age ship circled the Mediterranean, while the text for Odysseus's own Hollow Horse home page wrote itself out. Then, line by line, twin murals of the Iliad and Odyssey formed on-screen.

HOME RUNT But since I didn't want to read about the latest offerings from Mythological Software, the Hollow Horse home page was not where I wanted to be. So I pressed my Hot button (I love saying that) and selected the Brobdingnagian Black Widow Web Searcher page from the University of Michigan in Copenhagen.

Net traffic must have been light that day, because in less than four minutes I had a window full of instructions, prompts and the requisite cartoon of a giant spider attacking the world. I set up my search criteria "Soy AND trucking AND Wisconsin," pressed the Search button (I don't like saying that nearly so much), and got up for a coffee break.

Three cups and a few revelations about office romances later, I returned to my PC just as the search was finishing. There were 83 hits, the most promising of which was "Truckers, Soy Beans, and Wisconsin," a page emanating from a data-processing plant in Honolulu. I clicked on it, and five minutes later found myself staring at a photograph of three cats named Truckers, Soy Beans, and Wisconsin. The accompanying text filled me in on their favorite foods and pastimes.

After jotting down a few notes on cat care that might one day prove useful, I backtracked to my search results, and started working my way through the other 82 hits. After 14 additional pet pages, 3 obscene illustrations, one trap set up to steal my password, and 42 "Address Wrong or Go Back to America Online" error messages, I found something interesting: a discussion on the use of beans in the Wisconsin area's pre-Columbian art. I leaned forward and started reading.

It was fascinating stuff. Did you know that there is absolutely no evidence that beans were used in the Wisconsin area's pre-Columbian art? That's the sort of information you can only find on the Web.

At the bottom of the article was a link to the Artifacts of Ancient Civilizations That May or May Not Have Existed home page. Who could resist? I clicked, then got up and jogged around the block.

RAMBLIN' ON By the time I got back, the page was just beginning to appear. I browsed a few topics on the similarities between Mayan and Martian cultures before discovering the Foods of the World home page. This was amazing! I could actually click on a carrot and watch it grow. Or place an order with a pizzeria with a fax-back service. Not that everything was this exciting; there was even an article on how soy cheese is affecting Wisconsin's trucking industry, but I abandoned it for the piece on eggplant as an aphrodisiac.

I'm not sure how the connection was made, but somehow I found myself at the U.S. Census site, examining data tables of people, broken down by age and sex. Finding myself on both lists, I clicked on the first available link, which brought me to a catalog selling replacement parts for 19th century steam engines.

Odysseus was just finishing displaying the catalog's pictures when I returned from lunch. Soon I was pondering exactly where in my office I'd put a 400-pound water pump, and how many people would get access to my credit card number if I ordered it.

I was deep in thought when I realized someone was standing behind me. It was my boss. I flicked on my screen saver and spun around to face her, hoping she hadn't seen my screen.

"Hello," she said cheerily. "I need to know how rainfall in British Columbia is affecting the tourist trade in Malaysia. Drop everything else until you get this done."

```
MyWordVar := MyTObjectDescendant.InstanceSize;
{This is for descendants of TObject.}
```

Q: How do I read and write to a com-port?

A:

```
program PortAccess;

var
  port: TextFile;
  x: char;

begin
  Assign(port, 'COM2');
  Rewrite(port);
  write(port, 'AS5', #13);
  { sample uses:
  read(port, x);
  write(port, x);
  }
  close(port);
end.
```

Q: How do I iterate through tabbed notebook pages to see each object?

A: Here is a procedure that will iterate through all tabbed notebook pages and add the object name and type under the page's name in an outline.

```
procedure TForm1.Button2Click(Sender: TObject);
var
  cmpnts, pg: word;
  MyPageObj: TObject;
  OutlineIdx: longint;
begin
  for pg := 0 to TabbedNotebook1.pages.count - 1 do begin
    MyPageObj := TabbedNotebook1.pages.objects[pg];
    OutlineIdx := outline1.add(0, TabbedNotebook1.pages[pg]);
    for cmpnts := 0 to componentCount - 1 do
      if (components[cmpnts] as TControl).parent = MyPageObj then
        outline1.AddChild(outlineIdx, components[cmpnts].name +
          ' [' + components[cmpnts].ClassName + ']');
  end;
end;
```

As it turns out, there is a slight problem with this code. If a page and its components are added dynamically, this code will not find it. That is because the new component is added to the page's component list and not the form's list. Here is a way around that one:

```
var
  cmpnts, pg: word;
  MyPageObj: TWinControl;
  OutlineIdx: longint;
begin
  for pg := 0 to TabbedNotebook1.pages.count - 1 do begin
    MyPageObj := (TabbedNotebook1.pages.objects[pg]) as TWinControl;
    OutlineIdx := outline1.add(0, TabbedNotebook1.pages[pg]);
    with MyPageObj do
      for cmpnts := 0 to ControlCount - 1 do
        outline1.AddChild(outlineIdx, Controls[cmpnts].name +
          ' [' + Controls[cmpnts].ClassName + ']');
  end;
end;
```

Q: How can I parse a PChar?

A: This reads the autoexec.bat file into a memory block referenced by a PChar. Then, it is parsed, line by line, into a list box.

(Yes. I know that `listbox1.items.LoadFromFile('c:\autoexec.bat');` is simpler, but this is an exercise in PChar use.)

```
procedure TForm1.Button1Click(Sender: TObject);
var
  f: file;
  pBeginString, pEndString, pTemp: PChar;
  scratch: array[0..255] of char; {Automatically gets memory allocated for
it.}
  LengthOfFile: integer;
begin
  {Get the information.}
  AssignFile(f, 'c:\autoexec.bat');
  {Because this is not a text file type, the record size is 1 (char)}
  Reset(f, 1);
  LengthOfFile := FileSize(f) + 1; {Add one for the null terminator.}
  pBeginString := AllocMem(LengthOfFile); {Zeros the memory also.}
  BlockRead(f, pBeginString^, LengthOfFile - 1);
  CloseFile(f);

  pTemp := pBeginString;
  inc(pTemp, LengthOfFile);

  {Parse the strings into the Listbox.}
  repeat
    pEndString := StrPos(pBeginString, #13#10);  {carriage return/line feed}
    listBox1.items.add(StrPas(StrLCopy(scratch,
    pBeginString, pEndString - pBeginString)));
    inc(pBeginString, pEndString - pBeginString + 2);
  until pBeginString >= pTemp - 2;
  dec(pTemp, LengthOfFile);
  FreeMem(pTemp, LengthOfFile);
end;
```

Q: How do I pass variables to Report Smith?

A: The important part of the code is line to concatenate the single quotes to the string. (If it is just a string that is being passed, you don't need the embedded quotes. That is for a date string.)

In Report Smith, the REP VAR is

Name: tday
Type: DATE
Entry: Type-in

(Note: The values passed to RS are case sensitive.)

```
procedure PassVars;
var s: string;
begin
  s := DateToStr(date);
  s := '''' + s + ''''; {This is not needed for regular strings.}
  report1.InitialValues.add('@tday=<'+s+'>');
  report1.run;
end;
```

How do I pass variables to Report Smith?
How can I get rid of the ReportSmith about box splash screen when I run a report?
How do I connect to TReport?

DbiGetNetUserName
DbiGetErrorString

Using DbiGetNetUserName:

```
uses
  DbiTypes, DbiProcs, DbiErrs;

procedure Whatever;
var
  szVar: array[0..200] of char;
begin
  DbiGetNetUserName(szVar);
  edit1.text := szVar;
end;
```

Using DbiGetErrorString:

```
uses
  DbiTypes, DbiProcs, DbiErrs;

procedure Whatever;
var
  rslt: DbiResult;
  szVar: array[0..200] of char;
begin
  rslt := DbiGetNetUserName(szVar); {...or whatever you're checking.}
  DbiGetErrorString(rslt, szVar);
  edit2.text := szVar; {sample display}
end;
```

Q: How do I populate a popup menu on the fly?

A:

```
var
  NewItem: TMenuItem;
  i: integer;
begin
  for i := 0 to listBox1.items.count - 1 do
  begin
    NewItem := TMenuItem.Create(Self);
    NewItem.Caption := listBox1.items[i];
    PopupMenu1.items.Add(NewItem);
  end;
end;
```

Note: There needs to be some way to free the resources, but I haven't gotten around to that yet.

Q: How can I write my Delphi program to detect if there is already another copy running and exit if so?

A: Here is some code from Pat Ritchey that works great. Create a unit called PrevInst and add it to your uses clause. Here's the code:

```
unit PrevInst;

interface

uses
 WinTypes, WinProcs, SysUtils;

type
 PHWND = ^HWND;
 function EnumFunc(Wnd:HWND; TargetWindow:PHWND): bool; export;
 procedure GotoPreviousInstance;

implementation

 function EnumFunc(Wnd:HWND; TargetWindow:PHWND): bool;
 var
   ClassName : array[0..30] of char;
 begin
   Result := true;
   if GetWindowWord(Wnd,GWW_HINSTANCE) = hPrevInst then
      begin
      GetClassName(Wnd,ClassName,30);
      if StrIComp(ClassName,'TApplication') = 0 then
         begin
         TargetWindow^ := Wnd;
         Result := false;
         end;
      end;
 end;

 procedure GotoPreviousInstance;
 var
   PrevInstWnd : HWND;
 begin
   PrevInstWnd := 0;
   EnumWindows(@EnumFunc,longint(@PrevInstWnd));
   if PrevInstWnd <> 0 then
      if IsIconic(PrevInstWnd) then
         ShowWindow(PrevInstWnd,SW_RESTORE)
      else
         BringWindowToTop(PrevInstWnd);
 end;

 end.
```

And then make the main block of your *.DPR file look something like this--

```
if hPrevInst <> 0 then
   GotoPreviousInstance
```

```
else
begin
    Application.CreateForm(MyForm, MyForm);
    Application.Run;
end;
```

Q: I need to process certain files (*.ssd) in a user selected directory and every nested directory thereafter using Delphi. Anyone has any pointers to give me? What calls to use (my main concern is how to recognize a subdirectory while scanning a directory), may be a fragment of code that does something similar?

A: FindFirst and FindNext are the key functions.

Here is the short version. It is written in a generic way so that you can use the FileFind() procedure from any unit.

unit Findfile;

interface

```
uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    Button1: TButton;
    Edit1: TEdit;
    Label1: TLabel;
    Edit2: TEdit;
    Label2: TLabel;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure FileFind(StartingDirectory, FileName: string; FilesFound:
TStringList);

  procedure SearchTree;
  var
    SearchRec: TSearchRec;
    DosError: integer;
    dir: string;
  begin
    GetDir(0, dir);
    if dir[length(dir)] <> '\' then dir := dir + '\';
    DosError := FindFirst(FileName, 0, SearchRec);
    while DosError = 0 do begin
      try
```

```
        FilesFound.add(dir + SearchRec.name);
      except
        on EOutOfResources do begin
          ShowMessage('Too many files.');
          abort;
        end;
      end;
      DosError := FindNext(SearchRec);
    end;
    {Now that we have all the files we need, lets go to a subdirectory.}
    DosError := FindFirst('*.*', faDirectory, SearchRec);
    while DosError = 0 do begin
      {If there is one, go there and search.}
      if ((SearchRec.attr and faDirectory = faDirectory) and
        (SearchRec.name <> '.') and (SearchRec.name <> '..')) then begin
        ChDir(SearchRec.name);
        SearchTree; {Time for the recursion!}
        ChDir('..'); {Down one level.}
      end;
      DosError := FindNext(SearchRec); {Look for another subdirectory}
    end;
  end; {SearchTree}

begin
  FilesFound.clear;
  ChDir(StartingDirectory);
  SearchTree;
end; {FileFind}

procedure TForm1.Button1Click(Sender: TObject);
var
  t: TStringList;
begin
  t := TStringList.create;
  FileFind(edit2.text, edit1.text, t);
  listbox1.items.assign(t);
  t.free;
end;

end.
```

Here is an example using a slightly different format:

```
unit Dirlist1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    Edit1: TEdit;
```

```
    Label1: TLabel;
    Label2: TLabel;
    Edit2: TEdit;
    Button1: TButton;
    Button2: TButton;
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    Function LogFiles( Const path: String; Const SRec: TSearchRec ): Boolean;
    { Private-Declaration }
  public
    { Public-Declaration }
  end;

var
  Form1: TForm1;

implementation

Type
  TLogFunct = Function( Const path: String; Const SRec: TSearchRec ): Boolean
              of Object;
{$R *.DFM}

Procedure FindRecursive( Const path: String; Const mask: String;
                         LogFunction: TLogFunct );
  Var
    fullpath: String;
  Function Recurse( Var path: String; Const mask: String ): Boolean;
    Var
      SRec: TSearchRec;
      retval: Integer;
      oldlen: Integer;
    Begin
      Recurse := True;
      oldlen := Length( path );
      (* phase 1, look for normal files *)
      retval := FindFirst( path+mask, faAnyFile, SRec );
      While retval = 0 Do Begin
        If (SRec.Attr and (faDirectory or faVolumeID)) = 0 Then
          (* we found a file, not a directory or volume label,
             log it. Bail out if the log function returns false. *)
          If not LogFunction( path, SRec ) Then Begin
            Result := False;
            Break;
          End;
        retval := FindNext( SRec );
      End;
      FindClose( SRec );
      If not Result Then Exit;

      (* Phase II, look for subdirectories and recurse thru them *)
      retval := FindFirst( path+'*.*', faDirectory, SRec );
      While retval = 0 Do Begin
        If (SRec.Attr and faDirectory) <> 0 Then (* we have a directory *)
          If (SRec.Name <> '.') and (SRec.Name <> '..') Then Begin
            path := path + SRec.Name + '\';
```

```
            If not Recurse( path, mask ) Then Begin
              Result := False;
              Break;
            End;
            Delete( path, oldlen+1, 255 );
          End;
        retval := FindNext( SRec );
      End;
      FindClose( SRec );
    End;
  Begin
    If path = '' Then
      GetDir(0, fullpath)
    Else
      fullpath := path;
    If fullpath[Length(fullpath)] <> '\' Then
      fullpath := fullpath + '\';
    If mask = '' Then
      Recurse( fullpath, '*.*' )
    Else
      Recurse( fullpath, mask );
  End;


Function TForm1.LogFiles( Const path: String; Const SRec: TSearchRec ):
Boolean;
  Begin
    Listbox1.Items.Add( path+SRec.Name );
    Result := True;   (* proceeed with recursion *)
  End;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Application.Terminate;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  ListBox1.Clear;
  Listbox1.Perform( WM_SETREDRAW, 0, 0 );
  FindRecursive( Edit1.Text, Edit2.Text, LogFiles );
  Listbox1.Perform( WM_SETREDRAW, 1, 0 );
  Listbox1.Refresh;
end;

end.
```

---- dirlist1.dfm -----

```
object Form1: TForm1
  Left = 260
  Top = 222
  Width = 642
  Height = 300
  Caption = 'Recursive Directory Scan'
  Font.Color = clWindowText
```

```
Font.Height = -17
Font.Name = 'System'
Font.Style = []
PixelsPerInch = 120
Position = poScreenCenter
TextHeight = 20
object Label1: TLabel
  Left = 480
  Top = 24
  Width = 116
  Height = 20
  Caption = '&Path to search'
  FocusControl = Edit1
end
object Label2: TLabel
  Left = 480
  Top = 96
  Width = 74
  Height = 20
  Caption = '&File Mask'
  FocusControl = Edit2
end
object ListBox1: TListBox
  Left = 16
  Top = 24
  Width = 449
  Height = 225
  ItemHeight = 20
  TabOrder = 0
end
object Edit1: TEdit
  Left = 480
  Top = 48
  Width = 137
  Height = 29
  TabOrder = 1
end
object Edit2: TEdit
  Left = 480
  Top = 120
  Width = 137
  Height = 29
  TabOrder = 2
end
object Button1: TButton
  Left = 480
  Top = 168
  Width = 137
  Height = 33
  Caption = '&Search'
  Default = True
  TabOrder = 3
  OnClick = Button1Click
end
object Button2: TButton
  Left = 480
  Top = 216
```

```
    Width = 137
    Height = 33
    Caption = 'Close'
    TabOrder = 4
    OnClick = Button2Click
  end
end
```

From an ex-field sales/support survivor:

I used to work in a computer store and one day we had a gentleman call in with a smoking power supply. The service rep was having a bit of trouble convincing this guy that he had a hardware problem.Service Rep: Sir, something has burnt within your power supply.

> Customer: I bet that there is some command that I can put into the Autoexec.bat that will take care of this.
> Service Rep: There is nothing that software can do to help you with this problem.
> Customer: I know that there is something that I can put in... some command... maybe it should go into the Config.sys.
>
> [After a few minutes of going round and round]

> Service Rep: Okay, I am not supposed to tell anyone this but there is a hidden command in some versions of DOS that you can use. I want you to edit your Autoexec.bat and add the last line as C:\DOS\NOSMOKE and reboot your computer.

> [Customer does this]
> Customer: It is still smoking.
> Service Rep: I guess you need to call MicroSoft and ask them for a patch for the NOSMOKE.EXE.
>
[The customer then hung up. We thought that we had heard the last of this guy but NO... he calls back four hours later]
>

Service Rep: Hello Sir, how is your computer?
>
Customer: I call MicroSoft and they said that my Power Supply is incompatible with their NOSMOKE.EXE and that I need to get a new one. I was wondering when I can have that done and how much it will cost...
>
Moral: Remember those hidden DOS commands!

Q: How can I detect the presense of a DLL that may or may not be loaded?

A: If you mean "presence in memory", then use GetModuleHandle with the DLLs module name. If it returns 0, the module is not loaded. When you get a valid handle better check the filename with GetModuleFilename since you may have a freak match with an EXE module name (its rampand chaos out there as far as module names are considered).

If you mean "presence on system", just try to LoadLibrary the DLL.

Q: How can my component tell if I'm running via the IDE or the EXE?

A: If the component is being used in the IDE the following test will evaluate as true:

```
csDesigning in ComponentState
```

file locking

Q: How do I change an icon to bitmap?

A: Here is the idea in short form:

```
VAR
 Pic : TPicture;
 TI : TIcon;
BEGIN
...
  TI := TIcon.Create;
  TI.Handle := ExtractIcon(HInstance, FileNameBuf, 0);
  Pic := TPicture.Create;
  Pic.Icon := TI;
  Image1.Picture := Pic; {TImage}
  BitBtn1.Glyph := TBitmap.Create;
  WITH BitBtn1.Glyph DO
    BEGIN
      width := TI.Width;
      Height := TI.Height;
      Canvas.Draw(0, 0, Pic.Icon);
    END;
....
END;
```

I added a TBitBtn and a TImage to the Form. You will see that TImage looks OK but sometimes not the TBitBtn.
This is due too that Delphi's Glyph-drawing code checks the lower Left Corner pixel and uses that as the transparent color. I don't know why they changed that during development cycle. But of course you can change that pixel yourself.

Here is the idea in long form. It does much more.

This code take a 32x32 icon and runs it through 2 passes. First it makes it a 16x32 bitmap. The second takes it to 16x16. In the reduction I take special care of edges (often black or a single pixel width color that needs to be maintained) and patterns. By patterns I am referring to dithering by alternating 2 colors to give a third. An example of this is using dark yellow and lite gray to give the color of a folder. These patterns are maintained even after the reduction. I also give special weight to certain colors in the seperate passes. If none of these conditions apply I averaging.

This procedure will give almost but not exactly the same results as MS in Win95. By the way This is the same technique they use I found out later. Also of interest is the fact that this is rather slow so in my Win95 TaskBar replacement (that sure could use Delphi32 by the way so as to by mutithreaded and allow me to do the last thing that I can't do in 16 bit land, that is implement the notification area) I cache the 16x16 bitmaps.

```
procedure TTask.scaleaicon(i:integer);

var
```

```
  m,n,p,q                 :integer;
  ibitmap,ibitmap1,
  oldbitmap,oldbitmap1,
  oldbitmap2              :hbitmap;
  icolor                  :tcolorref;
  pc                      :array[1..4] of tcolorref;
  r,g,b                   :byte;
  memdc,memdc1,memdc2     :hdc;
  isedge,dither           :boolean;

begin
  memdc:=createcompatibledc(canvas.handle);
  memdc1:=createcompatibledc(canvas.handle);
  memdc2:=createcompatibledc(canvas.handle);
  ibitmap:=createcompatiblebitmap(canvas.handle,32,32);
  ibitmap1:=createcompatiblebitmap(canvas.handle,16,32);
  oldbitmap:=selectobject(memdc,ibitmap);
  oldbitmap1:=selectobject(memdc1,ibitmap1);
  oldbitmap2:=selectobject(memdc2,lapp[i].ibitmap);
  selectobject(memdc,getstockobject(ltgray_brush));
  selectobject(memdc1,getstockobject(ltgray_brush));
  selectobject(memdc2,getstockobject(ltgray_brush));
  patblt(memdc,0,0,32,32,patcopy);
  patblt(memdc1,0,0,16,32,patcopy);
  patblt(memdc2,0,0,16,16,patcopy);
  drawicon(memdc,0,0,apicon);

  m:=0;p:=0;
  n:=0;q:=0;
  while m<32 do
  begin
    while n<32 do
    begin
      dither:=false;
      icolor:=getpixel(memdc,n,m);
      pc[1]:=icolor;
      icolor:=getpixel(memdc,n+1,m);
      pc[2]:=icolor;
      if (n>0) and (n<30) then
      begin
        icolor:=getpixel(memdc,n+2,m);
        pc[3]:=icolor;
        icolor:=getpixel(memdc,n+3,m);
        pc[4]:=icolor;
        if (pc[1]=pc[3]) and (pc[2]=pc[4]) then
        begin
          dither:=true;
          setpixel(memdc1,q,p,pc[1]);
          setpixel(memdc1,q+1,p,pc[2]);
          n:=n+4;
          q:=q+2;
        end;
      end;
      if not dither then
      begin
        isedge:=false;
        if (n=0) then
```

```
        begin
          if (pc[1]=0) or (pc[2]=0) then
          begin
            isedge:=true;
            setpixel(memdc1,q,p,0);
            n:=n+2;
            q:=q+1;
          end else
          if (pc[1]=8421504) or (pc[2]=8421504) then
          begin
            isedge:=true;
            setpixel(memdc1,q,p,8421504);
            n:=n+2;
            q:=q+1;
          end;
        end;
        if (n=30) then
        begin
          if (pc[1]=0) or (pc[2]=0) then
          begin
            isedge:=true;
            setpixel(memdc1,q,p,0);
            n:=n+2;
            q:=q+1;
          end else
          if (pc[1]=8421504) or (pc[2]=8421504) then
          begin
            isedge:=true;
            setpixel(memdc1,q,p,8421504);
            n:=n+2;
            q:=q+1;
          end;
        end;
        if not isedge then
        begin
          if ((pc[1]=12632256) and (pc[2]=8421504)) or ((pc[1]=8421504) and
(pc[2]=12632256)) then
          begin
            r:=128;g:=128;b:=128;
          end else
          if ((pc[1]=16777215) and (pc[2]=12632256)) or ((pc[1]=12632256) and
(pc[2]=16777215)) then
          begin
            r:=192;g:=192;b:=192;
          end else
          if (pc[1]=0) or (pc[2]=0) then
          begin
            r:=0;g:=0;b:=0;
          end else
          begin
            r:=byte(round((getrvalue(pc[1])+getrvalue(pc[2]))/2));
            g:=byte(round((getgvalue(pc[1])+getgvalue(pc[2]))/2));
            b:=byte(round((getbvalue(pc[1])+getbvalue(pc[2]))/2));
          end;
          setpixel(memdc1,q,p,rgb(r,g,b));
          n:=n+2;
          q:=q+1;
```

```
      end;
    end;
  end;
  m:=m+1;
  p:=p+1;
  n:=0;
  q:=0;
end;

m:=0;p:=0;
n:=0;q:=0;
while n<16 do
begin
  while m<32 do
  begin
    dither:=false;
    icolor:=getpixel(memdc1,n,m);
    pc[1]:=icolor;
    icolor:=getpixel(memdc1,n,m+1);
    pc[2]:=icolor;
    if (m>0) and (m<30) then
    begin
      icolor:=getpixel(memdc1,n,m+2);
      pc[3]:=icolor;
      icolor:=getpixel(memdc1,n,m+3);
      pc[4]:=icolor;
      if (pc[1]=pc[3]) and (pc[2]=pc[4]) then
      begin
        dither:=true;
        setpixel(memdc2,q,p,pc[1]);
        setpixel(memdc2,q,p+1,pc[2]);
        m:=m+4;
        p:=p+2;
      end;
    end;
    if not dither then
    begin
      isedge:=false;
      if (m=0) then
      begin
        if (pc[1]=0) or (pc[2]=0) then
        begin
          isedge:=true;
          setpixel(memdc2,q,p,0);
          m:=m+2;
          p:=p+1;
        end else
        if (pc[1]=8421504) or (pc[2]=8421504) then
        begin
          isedge:=true;
          setpixel(memdc2,q,p,8421504);
          m:=m+2;
          p:=p+1;
        end;
      end;
      if (m=30) then
      begin
```

```
          if (pc[1]=0) or (pc[2]=0) then
          begin
            isedge:=true;
            setpixel(memdc2,q,p,0);
            m:=m+2;
            p:=p+1;
          end else
          if (pc[1]=8421504) or (pc[2]=8421504) then
          begin
            isedge:=true;
            setpixel(memdc2,q,p,8421504);
            m:=m+2;
            p:=p+1;
          end;
        end;
        if not isedge then
        begin
          if ((pc[1]=12632256) and (pc[2]=8421504)) or ((pc[1]=8421504) and
(pc[2]=12632256)) then
          begin
            r:=128;g:=128;b:=128;
          end else
          if ((pc[1]=16777215) and (pc[2]=12632256)) or ((pc[1]=12632256) and
(pc[2]=16777215)) then
          begin
            r:=192;g:=192;b:=192;
          end else
          if ((pc[1]=0) and (pc[2]=8421504)) or ((pc[1]=8421504) and
(pc[2]=0)) then
          begin
            r:=128;g:=128;b:=128;
          end else
          if (pc[1]=8421504) or (pc[2]=8421504) then
          begin
            r:=128;g:=128;b:=128;
          end else
          begin
            r:=byte(round((getrvalue(pc[1])+getrvalue(pc[2]))/2));
            g:=byte(round((getgvalue(pc[1])+getgvalue(pc[2]))/2));
            b:=byte(round((getbvalue(pc[1])+getbvalue(pc[2]))/2));
          end;
          setpixel(memdc2,q,p,rgb(r,g,b));
          m:=m+2;
          p:=p+1;
        end;
      end;
    end;
    n:=n+1;
    q:=q+1;
    m:=0;
    p:=0;
  end;
  selectobject(memdc,oldbitmap);
  selectobject(memdc1,oldbitmap1);
  selectobject(memdc2,oldbitmap2);
  deleteobject(ibitmap);
  deleteobject(ibitmap1);
```

```
  deletedc(memdc);
  deletedc(memdc1);
  deletedc(memdc2);
end;
```

Q: How do I fill a TMemo from a PChar?

A:

```
procedure PutPCharIntoBlob( p: PChar; mf: TBlobField );
var
  bs: TBlobStream;
begin
  bs := TBlobStream.Create( mf, bmWrite );
  bs.Write( p^, StrLen( p ) );
  bs.Free;
end;
```

Q: How can I reference a field name with a space in a query?

A:

```
select * from MyTable
where MyTable."field with spaces" = 123
```

Q: How do I pass a variable to a query?

A: First, you must write a query that uses a variable.

```
Select Test."FName", Test."Salary Of Employee"
From Test
Where Test."Salary of Employee" > :val
```

Note: If you just write the field name as `"Salary of Employee"` you will get a **Capability Not Supported** error. It must be `Test."Salary of Employee"`.

In this can the variable name is "val", but it can be whatever you want (of course). Then, you go to the TQuery's params property and set the "val" param to whatever the appropriate type is. In our example here we will call it an integer.

Next, you write the code that sets the param's value. We will be setting the value from a TEdit box.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with Query1 do
  begin
    Close;
    ParamByName('val').AsInteger := StrToInt(Edit1.Text);
    Open;
  end;
end;
```

Note: you may want to place this code in a try..except block as a safety precaution.

If you want to use a LIKE in your query, you can do it this way:

```
Select * From customer
Where company like :CompanyName

procedure TForm1.Button1Click(Sender: TObject);
begin
  with Query1 do
  begin
    Close;
    ParamByName('CompanyName').AsString := Edit1.Text + '%';
    Open;
  end;
end;
```

The trick is in the concatenating of the percentage sign at the end of the parameter.

Q: How do I copy a file?

A: Here are four ways:

```
{This way uses a File stream.}
Procedure FileCopy( Const sourcefilename, targetfilename: String );
Var
  S, T: TFileStream;
Begin
  S := TFileStream.Create( sourcefilename, fmOpenRead );
  try
    T := TFileStream.Create( targetfilename, fmOpenWrite or fmCreate );
    try
      T.CopyFrom(S, S.Size ) ;
    finally
      T.Free;
    end;
  finally
    S.Free;
  end;
End;

{This way uses memory blocks for read/write.}
procedure FileCopy(const FromFile, ToFile: string);
 var
  FromF, ToF: file;
  NumRead, NumWritten: Word;
  Buf: array[1..2048] of Char;
begin
  AssignFile(FromF, FromFile);
  Reset(FromF, 1);            { Record size = 1 }
  AssignFile(ToF, ToFile);    { Open output file }
  Rewrite(ToF, 1);            { Record size = 1 }
  repeat
    BlockRead(FromF, Buf, SizeOf(Buf), NumRead);
    BlockWrite(ToF, Buf, NumRead, NumWritten);
  until (NumRead = 0) or (NumWritten <> NumRead);
  System.CloseFile(FromF);
  System.CloseFile(ToF);
end;

{This one uses LZCopy, which USES LZExpand.}
procedure CopyFile(FromFileName, ToFileName: string);
var
  FromFile, ToFile: File;
begin
  AssignFile(FromFile, FromFileName); { Assign FromFile to FromFileName }
  AssignFile(ToFile, ToFileName);     { Assign ToFile to ToFileName }
  Reset(FromFile);                    { Open file for input }
  try
    Rewrite(ToFile);                  { Create file for output }
    try
      { copy the file an if a negative value is returned raise an exception }
      if LZCopy(TFileRec(FromFile).Handle, TFileRec(ToFile).Handle) < 0 then
        raise Exception.Create('Error using LZCopy')
```

```
    finally
      CloseFile(ToFile);  { Close ToFile }
    end;
  finally
    CloseFile(FromFile);  { Close FromFile }
  end;
end;
```

This one is from Dr. Bob (Swart). The point of this one is that it contains a callback function that gives you the ability to callback. This can be used for progress bars and the like. Groetjes, Dr. Bob!

```
{$A+,B-,D-,F-,G+,I+,K+,L-,N+,P+,Q-,R-,S+,T+,V-,W-,X+,Y-}
unit FileCopy;
(*
  FILECOPY 1.5 (Public Domain)
  Borland Delphi 1.0
  Copr. (c) 1995-08-27 Robert E. Swart (100434.2072@compuserve.com)
                       P.O. box 799
                       5702 NP  Helmond
                       The Netherlands
  ------------------------------------------------------------------
  This unit implements a FastFileCopy procedure that is usable from
  Borland Pascal (real mode, DPMI or Windows) and Borland Delphi. A
  callback routine (or nil) can be given as extra argument.

  Example of usage:

  {$IFDEF WINDOWS}
   uses FileCopy, WinCrt;
  {$ELSE}
   uses FileCopy, Crt;
  {$ENDIF}

     procedure CallBack(Position, Size: LongInt); far;
     var i: Integer;
     begin
       { do you stuff here... }
       GotoXY(1,1);
       for i:=1 to (80 * Position) div Size do write('X')
     end {CallBack};

   begin
     FastFileCopy('C:\AUTOEXEC.BAT', 'C:\AUTOEXEC.BAK', nil);
     FastFileCopy('C:\CONFIG.SYS', 'C:\CONFIG.BAK', CallBack)
   end.
*)
interface

Type
  TCallBack = procedure (Position, Size: LongInt); { export; }

  procedure FastFileCopy(Const InFileName, OutFileName: String;
```

```
                          CallBack: TCallBack);

implementation
{$IFDEF VER80}
uses SysUtils;
{$ELSE}
  {$IFDEF WINDOWS}
  uses WinDos;
  {$ELSE}
  uses Dos;
  {$ENDIF}
{$ENDIF}

  procedure FastFileCopy(Const InFileName, OutFileName: String;
                         CallBack: TCallBack);
  Const BufSize = 8*4096; { 32Kbytes gives me the best results }
  Type
    PBuffer = ^TBuffer;
    TBuffer = Array[1..BufSize] of Byte;
  var Size: Word;
      Buffer: PBuffer;
      infile,outfile: File;
      SizeDone,SizeFile,TimeDateFile: LongInt;
  begin
    if (InFileName <> OutFileName) then
    begin
      Buffer := nil;
      Assign(infile,InFileName);
      System.Reset(infile,1);
      {$IFDEF VER80}
      try
      {$ELSE}
      begin
      {$ENDIF}
        SizeFile := FileSize(infile);
        Assign(outfile,OutFileName);
        System.Rewrite(outfile,1);
        {$IFDEF VER80}
        try
        {$ELSE}
        begin
        {$ENDIF}
          SizeDone := 0;
          New(Buffer);
          repeat
            BlockRead(infile,Buffer^,BufSize,Size);
            Inc(SizeDone,Size);
            if (@CallBack <> nil) then
              CallBack(SizeDone,SizeFile);
            BlockWrite(outfile,Buffer^,Size)
          until Size < BufSize;
          {$IFDEF VER80}
          FileSetDate(TFileRec(outfile).Handle,
            FileGetDate(TFileRec(infile).Handle));
          {$ELSE}
          GetFTime(infile, TimeDateFile);
          SetFTime(outfile, TimeDateFile);
```

```
          {$ENDIF}
        {$IFDEF VER80}
        finally
        {$ENDIF}
          if Buffer <> nil then Dispose(Buffer);
          System.close(outfile)
        end;
      {$IFDEF VER80}
      finally
      {$ENDIF}
        System.close(infile)
      end
    end
    {$IFDEF VER80}
    else
      Raise EInOutError.Create('File cannot be copied onto itself')
    {$ENDIF}
  end {FastFileCopy};
end.
```

Q: How do I detect whether a drive exists or not?

A: Here are some different ways to do it:

```
function DoesDriveExist(DriveLetter: char): string;
var i: integer;
begin
  if DriveLetter in ['A'..'Z'] then {Make it lower case.}
    DriveLetter := chr(ord(DriveLetter) or $20);
  i := GetDriveType(ord(DriveLetter) - ord('a'));
  case i of
    DRIVE_REMOVABLE: result := 'floppy';
    DRIVE_FIXED: result := 'hard disk';
    DRIVE_REMOTE: result := 'network drive';
    else result := 'does not exist';
  end;
end;



function DoesDriveExist(DriveLetter: char): boolean;
var
  drives: TDriveComboBox;
  i: integer;
begin
  result := false;
  drives := TDriveComboBox.create(application);
  drives.parent := form1;
  form1.listbox1.items := drives.items;
  for i := drives.items.count - 1 downto 0 do {Note: this is case sensitive:
lower case.}
    if drives.items.strings[i][1] = DriveLetter then result := true;
  drives.free; {...so that the combobox doesn't show.}
end;
```

Also, `DiskFree()` will return -1 if the drive does not exist.

Neil Rubenking wrote this code --

```
function DirExists(const S : String): Boolean;
VAR
  OldMode : Word;
  OldDir  : String;
BEGIN
  Result := True;
  GetDir(0, OldDir); {save old dir for return}
  OldMode := SetErrorMode(SEM_FAILCRITICALERRORS); {if drive empty, except}
  try try
    ChDir(S);
  except
    ON EInOutError DO Result := False;
  end;
  finally
    ChDir(OldDir); {return to old dir}
```

```
    SetErrorMode(OldMode); {restore old error mode}
  end;
END;
```

```
for i := 1 to ParamCount do listbox1.items.add(paramStr(i));
```

Q: How do I set and reset the canvas.font.style property?

A:

```
canvas.font.style := [fsUnderline];  {set}
canvas.font.style := [];  {reset}
```

Because the property uses a list, sending an empty list clears it out.

What If Dr. Seuss Did Technical Writing?

Here's an easy game to play.
Here's an easy thing to say:

If a packet hits a pocket on a socket on a port,
And the bus is interrupted as a very last resort,
And the address of the memory makes your floppy disk abort,
Then the socket packet pocket has an error to report!

If your cursor finds a menu item followed by a dash,
And the double-clicking icon puts your window in the trash,
And your data is corrupted 'cause the index doesn't hash,
Then your situation's hopeless, and your system's gonna crash!

You can't say this?
What a shame sir!
We'll find you
Another game sir.

If the label on the cable on the table at your house,
Says the network is connected to the button on your mouse,
But your packets want to tunnel on another protocol,
That's repeatedly rejected by the printer down the hall,

And your screen is all distorted by the side effects of gauss
So your icons in the window are as wavy as a souse,
Then you may as well reboot and go out with a bang,
'Cause as sure as I'm a poet, the sucker's gonna hang!

When the copy of your floppy's getting sloppy on the disk,
And the microcode instructions cause unnecessary risc,
Then you have to flash your memory and you'll want to RAM your ROM.
Quickly turn off the computer and be sure to tell your mom!

PAYMENT( )

Returns the periodic amount required to repay a debt.

```
function payment(princ, int, term: double): double;
var temp: double;
begin
  int := int / 100;
  temp := exp(ln(int + 1) * term);
  result := princ * ((int * temp) / (temp - 1));
end;
```

Syntax

PAYMENT(<principal expN>, <interest expN>, <term expN>)

<principal expN>

The original amount to be repaid over time.

<interest expN>

The interest rate per period expressed as a positive decimal number. Specify the interest rate in the same time increment as the term. It is to be expressed as a percentage. The number is divided by 100 inside the function.

<term expN>

The number of payments. Specify the term in the same time increment as the interest.

Description

Use PAYMENT( ) to calculate the periodic amount (payment) required to repay a loan or investment of <principal expN> amount in <term expN> payments. PAYMENT( ) returns a numeric value based on a fixed interest rate compounding over a fixed length of time. If <principal expN> is positive, PAYMENT( ) returns a positive number. If <principal expN> is negative, PAYMENT( ) returns a negative number. Express the interest rate as a decimal. For example, if the annual interest rate is 9.5%, <interest expN> is 9.5 for payments made annually.

Express <interest expN> and <term expN> in the same time increment. For example, if the payments are monthly, express the interest rate per month, and the number of payments in months. You would express an annual interest rate of 9.5%, for example, as 9.5/12, which is the 9.5% divided by 12 months. The formula used to calculate PAYMENT( ) is as follows:

```
                    term
int*(1 + int)^
```

```
pmt = princ * --------------------
                        term
              (1 + int)^     -  1
```

where int = rate / 100 (as a percentage).

For the monthly payment required to repay a principal amount of $16860.68 in five years, at 9% interest, the formula expressed as a dBASE expression looks like this:

```
MyVar := PAYMENT(16860.68, 9/12, 60)       {Returns 350.00}
```

Q: How can I use a TList to hold variables?

A:

```
implementation
type
  pLongInt = ^LongInt;

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  t: tlist;
  l: longint;
begin
  t := tlist.create;
  l := 123;
  t.add(@l);
  caption := IntToStr(pLongInt(t.items[0])^);
  t.free;
end;
```

Q: How do I determine if two strings sound alike?

A: Soundex function--determines whether two words sound alike. Written after reading an article in PC Magazine about the Soundex algorithm. Pass the function a string. It returns a Soundex value string. This value can be saved in a database or compared to another Soundex value. If two words have the same Soundex value, then they sound alike (more or less).

Note that the Soundex algorithm ignores the first letter of a word. Thus, "won" and "one" will have different Soundex values, but "Won" and "Wunn" will have the same values.

Soundex is especially useful in databases when one does not know how to spell a last name.

```
Function Soundex(OriginalWord: string): string;
var
  Tempstring1, Tempstring2: string;
  Count: integer;
begin
  Tempstring1 := '';
  Tempstring2 := '';
  OriginalWord := Uppercase(OriginalWord); {Make original word uppercase}
  Appendstr(Tempstring1, OriginalWord[1]); {Use the first letter of the word}
  for Count := 2 to length(OriginalWord) do
    {Assign a numeric value to each letter, except the first}
    case OriginalWord[Count] of
      'B','F','P','V': Appendstr(Tempstring1, '1');
      'C','G','J','K','Q','S','X','Z': Appendstr(Tempstring1, '2');
      'D','T': Appendstr(Tempstring1, '3');
      'L': Appendstr(Tempstring1, '4');
      'M','N': Appendstr(Tempstring1, '5');
      'R': Appendstr(Tempstring1, '6');
      {All other letters, punctuation and numbers are ignored}
    end;
  Appendstr(Tempstring2, OriginalWord[1]);
  {Go through the result removing any consecutive duplicate numeric values.}
  for Count:=2 to length(Tempstring1) do
    if Tempstring1[Count-1]<>Tempstring1[Count] then
        Appendstr(Tempstring2,Tempstring1[Count]);
  Soundex:=Tempstring2; {This is the soundex value}
end;
```

SoundAlike--pass two strings to this function. It returns True if they sound alike, False if they don't. Simply calls the Soundex function.

```
Function SoundAlike(Word1, Word2: string): boolean;
begin
  if (Soundex(Word1) = Soundex(Word2)) then result := True
  else result := False;
end;
```

Q: How can I tell the length in bytes of a memo field?

Background: I have been using the memo field and have been using the getTextLen to get the size I need to set my buffer before getting the information out of my large memo field. However, I have notice, that if the Memo field is larger than 256 character, the getTexLen will return a number only 0-255. How am I going to set my buffer to use GetTextBuf?

A: The lines property of a memo field is a TStrings. You could try something like this:

```
function GetMemoSize(TheMemo: TObject): integer;
var i: integer;
begin
  result := 0;
  with (TheMemo as TMemo).lines do
    for i := count - 1 downto 0 do
      result := result + length(strings[i]);
end;
```

This can be called with `IntVariable := GetMemoSize(memo1);`

Q: How can I get the windows or dos versions?

A: The API call of GetVersions will do it, but the information is encrypted into a longint. Here is how to get and decrypt the information:

```
Type TGetVer = record WinVer, WinRev, DosRev, DosVer: Byte; end;

procedure TForm1.Button1Click(Sender: TObject);
var AllVersions: longint;
begin
  AllVersions := GetVersion;
  edit1.text := IntToStr(TGetVer(AllVersions).WinVer) + '.' +
                IntToStr(TGetVer(AllVersions).WinRev);
  edit2.text := IntToStr(TGetVer(AllVersions).DosVer) + '.' +
                IntToStr(TGetVer(AllVersions).DosRev);
end;
```

Note: The values that windows displays for the versions and the values that it returns through its API call are not always the same. e.g. The workgroup version displays as 3.10 rather than 3.11.

Q: How do I detect for a co-processor?
Q: How can I tell which CPU is being used?

A: Here is the short version. The problem here is that it doesn't detect the pentium.

```
var winFlags: LongInt;
begin
  winFlags := GetWinFlags;
  { Get math coprocessor status }
  If winFlags And WF_80x87 > 0 Then Caption := 'Present'
  Else Caption := 'Not Present';

  { Get CPU type }
  If winFlags And WF_CPU486 > 0 Then edit1.text := '486' {also pentium}
  else If winFlags And WF_CPU386 > 0 Then edit1.text := '386'
  else If winFlags And WF_CPU286 > 0 Then edit1.text := '286';
end;
```

Here is a version that will work with the pentium:

```
{ This code comes from Intel, and has been modified for Delphi's
  inline assembler.
}

unit Cpu;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons;

type
  { All the types currently known.  As new types are created,
    add suitable names, and extend the case statement in
    the GetCpuType function.
  }
  TCPUType = (i8086CPU, i286CPU, i386CPU, i486CPU, iPentiumCPU);

  TForm1 = class(TForm)
    Edit1: TEdit;
    Label1: TLabel;
    BitBtn1: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Return the type of the current CPU }
    function CpuType: TCPUType;
    { Return the type as a string }
    function GetCPUType: String;
  public
  end;

var
```

```
  Form1: TForm1;
  { Define the winFlags variable for 286 check }
  winFlags: Longint;

implementation

{$R *.DFM}


{ Get CPU type }
function TForm1.GetCPUType: String;
var
  kind: TCPUType;
begin
  if winFlags and WF_CPU286 > 0 then
     Result := '80286'
  else
    begin
      kind := CpuType;
      case kind of
      i8086CPU:
        Result := '8086';
      i386CPU:
        Result := '80386';
      i486CPU:
        Result := '80486';
      iPentiumCPU:
        Result := 'Pentium';
      else
        { Try to be flexible for future cpu types, e.g., P6. }
        Result := Format('P%d', [Ord(kind)]);
      end;
    end;
end;

{ Assembly function to get CPU type including Pentium and later }
function TForm1.CpuType: TCPUType; assembler;
asm
  push DS
  { First check for an 8086 CPU }
  { Bits 12-15 of the FLAGS register are always set on the }
  { 8086 processor. }
  pushf                          { save EFLAGS }
  pop       bx                { store EFLAGS in BX }
  mov       ax,0fffh          { clear bits 12-15 }
  and       ax,bx             { in EFLAGS }
  push      ax                       { store new EFLAGS value on stack }
  popf                           { replace current EFLAGS value }
  pushf                          { set new EFLAGS }
  pop       ax                { store new EFLAGS in AX }
  and       ax,0f000h         { if bits 12-15 are set, then CPU }
  cmp       ax,0f000h         { is an 8086/8088 }
  mov       ax, i8086CPU             { turn on 8086/8088 flag }
  je        @@End_CpuType

  { 80286 CPU check }
  { Bits 12-15 of the FLAGS register are always clear on the }
```

```
  { 80286 processor. }
  { Commented out because 'pop ax' crashes it to the DOS prompt when
running }
  { with a Delphi form on some Machines.}
  {  or          bx,0f000h    }   { try to set bits 12-15 }
  {  push   bx              }
  {  popf                     }
  {  pushf                    }
  {  pop         ax           }     { This crashes Delphi programs on
some machines }
  {  and         ax,0f000h    }     { if bits 12-15 are cleared, CPU=80286 }
  {  mov    ax, i286CPU       }     { turn on 80286 flag }
  {  jz          @@End_CpuType    }

  { To test for 386 or better, we need to use 32 bit instructions,
    but the 16-bit Delphi assembler does not recognize the 32 bit opcodes
    or operands.  Instead, use the 66H operand size prefix to change
    each instruction to its 32-bit equivalent. For 32-bit immediate
    operands, we also need to store the high word of the operand immediately
    following the instruction.  The 32-bit instruction is shown in a comment
    after the 66H instruction.
  }

  { i386 CPU check }
  { The AC bit, bit #18, is a new bit introduced in the EFLAGS }
  { register on the i486 DX CPU to generate alignment faults. }
  { This bit can not be set on the i386 CPU. }

  db 66h                    { pushfd }
  pushf
  db 66h                    { pop eax }
  pop ax                  { get original EFLAGS }
  db 66h                    { mov ecx, eax }
  mov cx,ax           { save original EFLAGS }
  db 66h                    { xor eax,40000h }
  xor ax,0h             { flip AC bit in EFLAGS }
  dw 0004h
  db 66h                    { push eax }
  push ax             { save for EFLAGS }
  db 66h                    { popfd }
  popf                      { copy to EFLAGS }
  db 66h                    { pushfd }
  pushf                     { push EFLAGS }
  db 66h                    { pop eax }
  pop ax              { get new EFLAGS value }
  db 66h                    { xor eax,ecx }
  xor ax,cx           { can't toggle AC bit, CPU=Intel386 }
  mov ax, i386CPU           { turn on 386 flag }
  je @@End_CpuType

  { i486 DX CPU / i487 SX MCP and i486 SX CPU checking }
  { Checking for ability to set/clear ID flag (Bit 21) in EFLAGS }
  { which indicates the presence of a processor }
  { with the ability to use the CPUID instruction. }
  db 66h                    { pushfd }
  pushf                     { push original EFLAGS }
  db 66h                    { pop eax }
```

```
  pop ax              { get original EFLAGS in eax }
  db 66h                   { mov ecx, eax }
  mov cx,ax           { save original EFLAGS in ecx }
  db 66h                   { xor eax,200000h }
  xor ax,0h             { flip ID bit in EFLAGS }
  dw 0020h
  db 66h                   { push eax }
  push ax             { save for EFLAGS }
  db 66h                   { popfd }
  popf                     { copy to EFLAGS }
  db 66h                   { pushfd }
  pushf                    { push EFLAGS }
  db 66h                   { pop eax }
  pop ax              { get new EFLAGS value }
  db 66h                   { xor eax, ecx }
  xor ax, cx
  mov ax, i486CPU          { turn on i486 flag }
  je @@End_CpuType         { if ID bit cannot be changed, CPU=486 }
                      { without CPUID instruction functionality }

  { Execute CPUID instruction to determine vendor, family, }
  { model and stepping.  The use of the CPUID instruction used }
  { in this program can be used for B0 and later steppings }
  { of the P5 processor. }
  db 66h                  { mov eax, 1 }
  mov ax, 1            { set up for CPUID instruction }
  dw 0
  db 66h                  { cpuid }
  db  0Fh              { Hardcoded opcode for CPUID instruction }
  db  0a2h
  db 66h                  { and eax, 0F00H }
  and ax, 0F00H             { mask everything but family }
  dw 0
  db 66h                  { shr eax, 8 }
  shr ax, 8               { shift the cpu type down to the low byte }
  sub ax, 1               { subtract 1 to map to TCpuType }

@@End_CpuType:
  pop ds
end;


{ Get the Windows Flags to check for 286.  The 286 assembly code
  crashes due to a problem when using with Delphi Forms on some machines.
This
  method is safer.
}
procedure TForm1.FormCreate(Sender: TObject);
begin
  winFlags := GetWinFlags;
end;

{ Call the CPU function and assign it to the Edit box }
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Edit1.Text := GetCPUType;
end;
```

```
end.
```

Q: If you have the date 1/1/2000 and run the following code: `DateToStr(StrToDate('1/1/2000'))` the year gets changed from 2000 to 1900. What's the best way to handle this?

A: The value of the typed constant ShortDateTime in WIN.INI determines how DateToStr and StrToDate convert strings. The default is dd/mm/yy - two digit year. I believe setting ShortDateTime := 'dd/mm/yyyy' will solve your problem.

Q: How can I make interbase run faster?

A: When using Interbase, the screen tries to update for every update. If you disable the connections while performing actions like lookups or GotoKey, it will be much faster. e.g.

```
dbLookupList1.Enabled := false;
dbGrid1.DataSource := nil;
dbLookupList1.Enabled := false;
with table1 do
begin
  setkey;
  fields[0].AsString := edit1.text;
  DisableControls;
  GotoKey;
  EnableControls;
end;
dbLookupList1.Enabled := true;
dbgrid1.DataSource := DataSource1;
LockWindowUpdate(0);
```

ARRAY: searching and sorting routines:

Here is the UNIT with the code.
Here is the manual that describes the code.
Here is a program to compare the sorting algorithms.
Here is a program to test the sorts.

```
select * from MyTable
where field1 = (select max(field1) from MyTable)
```

Q: How do I highlight selected fields on a TStringGrid?

A:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;
begin
  stringGrid1.DefaultDrawing := false;
  with StringGrid1 do
  begin
    colCount := 26;
    RowCount := 99;
    for i := 1 to 20 do
      cells[i, 0] := chr(i + 64);
    for i := 0 to 99 do
      cells[0, i + 1] := IntToStr(i);
  end;
end;

procedure TForm1.StringGrid1DrawCell(Sender: TObject; Col, Row: Longint;
  Rect: TRect; State: TGridDrawState);
begin
  if (stringGrid1.cells[col, row][1] = '1') and (Col = 0) then
  begin
    stringGrid1.canvas.brush.color := clMaroon;
    stringGrid1.canvas.font.color := clWhite;
  end
  else
  begin
    stringGrid1.canvas.brush.color := clWhite;
    stringGrid1.canvas.font.color := clMaroon;
  end;
  stringGrid1.canvas.fillRect(rect);
  stringGrid1.canvas.TextRect(Rect, Rect.Left + 3, Rect.Top + 3,
StringGrid1.cells[col, row]);
end;
```

Q: How do I access hardware memory directly?

A: In real mode, to access the shift states you would use the following code.

```
var
   ShiftStates: Word;

begin
  ShiftStates := MemW[$0040: $0017];
  :
  :
end;
```

In protected mode, it is necessary to set up a selector to access memory directly.

```
var
  ShiftSel: Word;
  ShiftStates: Word;
begin
  ShiftSel := AllocSelector(DSeg);
  SetSelectorBase(ShiftSel, $00400);
  SetSelectorLimit(ShiftSel, $10000);
  ShiftStates := MemW[ShiftSel: $0017];
  :
  :
end;
```

Notice that in SetSelectorBase the value $00400 is used instead of $0040. The value represents a complete 20-bit linear address. For instance, if you needed to set a selector to point to the real mode segment $D000, you would use the value $D0000 in the call to SetSelectorBase.

Q: How do I put a repeating bitmap on the background of an MDI main form.

A: The basic technique involved subclassing the MDI client window (ClientHandle property) and responding to WM_ERASEBKGND by tiling the bitmap on the client window. However, there were a couple of problems; scrolling the main window to bring an off-screen child into view would screw up the background, and the background didn't get painted correctly behind the child window icons.

Well, HOORAY! I think I've whupped both those problems. Here's the code, for those who are interested. I'll start with the child form's code, followed by the main form's (units are named MDIWAL2U.PAS and MDIWAL1U.PAS). The main form is assumed to have the desired bitmap in a TImage named Image1.

```
  ...
  private
    { Private declarations }
    procedure WMIconEraseBkgnd(VAR Message: TWMIconEraseBkgnd);
      message WM_ICONERASEBKGND;
 ...
USES MdiWal1u;
procedure TForm2.WMIconEraseBkgnd(VAR Message: TWMIconEraseBkgnd);
BEGIN
  TForm1(Application.Mainform).PaintUnderIcon(Self, Message.DC);
  Message.Result := 0;
END;

================================================================
   ...
    { Private declarations }
    bmW, bmH : Integer;
    FClientInstance,
    FPrevClientProc : TFarProc;
    PROCEDURE ClientWndProc(VAR Message: TMessage);
  public
    PROCEDURE PaintUnderIcon(F: TForm; D: hDC);
...
PROCEDURE TForm1.PaintUnderIcon(F: TForm; D: hDC);
VAR
  DestR, WndR : TRect;
  Ro, Co,
  xOfs, yOfs,
  xNum, yNum  : Integer;
BEGIN
  {calculate number of tilings to fill D}
  GetClipBox(D, DestR);
  WITH DestR DO
    BEGIN
      xNum := Succ((Right-Left) DIV bmW);
      yNum := Succ((Bottom-Top) DIV bmW);
    END;
  {calculate offset of image in D}
  GetWindowRect(F.Handle, WndR);
  WITH ScreenToClient(WndR.TopLeft) DO
    BEGIN
      xOfs := X MOD bmW;
```

```
      yOfs := Y MOD bmH;
    END;
  FOR Ro := 0 TO xNum DO
    FOR Co := 0 TO yNum DO
      BitBlt(D, Co*bmW-xOfs, Ro*bmH-Yofs, bmW, bmH,
        Image1.Picture.Bitmap.Canvas.Handle,
        0, 0, SRCCOPY);
END;

PROCEDURE TForm1.ClientWndProc(VAR Message: TMessage);
VAR Ro, Co : Word;
begin
  with Message do
    case Msg of
      WM_ERASEBKGND:
        begin
          FOR Ro := 0 TO ClientHeight DIV bmH DO
            FOR Co := 0 TO ClientWIDTH DIV bmW DO
              BitBlt(TWMEraseBkGnd(Message).DC,
                Co*bmW, Ro*bmH, bmW, bmH,
                Image1.Picture.Bitmap.Canvas.Handle,
                0, 0, SRCCOPY);
          Result := 1;
        end;
      WM_VSCROLL,
      WM_HSCROLL :
        begin
          Result := CallWindowProc(FPrevClientProc,
            ClientHandle, Msg, wParam, lParam);
          InvalidateRect(ClientHandle, NIL, True);
        end;
    else
      Result := CallWindowProc(FPrevClientProc,
        ClientHandle, Msg, wParam, lParam);
    end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  bmW := Image1.Picture.Width;
  bmH := Image1.Picture.Height;
  FClientInstance := MakeObjectInstance(ClientWndProc);
  FPrevClientProc := Pointer(
    GetWindowLong(ClientHandle, GWL_WNDPROC));
  SetWindowLong(ClientHandle, GWL_WNDPROC,
    LongInt(FClientInstance));
end;
```

Q: How do I do bit-wise manipulation?

A:

```
{****************************************
TheBit parameter is counted from 0..31
****************************************}

unit Bitwise;

interface
  function IsBitSet(const val: longint; const TheBit: byte): boolean;
  function BitOn(const val: longint; const TheBit: byte): LongInt;
  function BitOff(const val: longint; const TheBit: byte): LongInt;
  function BitToggle(const val: longint; const TheBit: byte): LongInt;


implementation

function IsBitSet(const val: longint; const TheBit: byte): boolean;
begin
  result := (val and (1 shl TheBit)) <> 0;
end;

function BitOn(const val: longint; const TheBit: byte): LongInt;
begin
  result := val or (1 shl TheBit);
end;

function BitOff(const val: longint; const TheBit: byte): LongInt;
begin
  result := val and not (1 shl TheBit);
end;

function BitToggle(const val: longint; const TheBit: byte): LongInt;
begin
  result := val xor (1 shl TheBit);
end;

end.
```

Turn off compile optimizations. Use w8loss.exe on the EXE file instead.

```
with memo1.lines do
  begin
    add('');
    delete(count - 1);
  end;

  memo1.perform(em_LineScroll, 0, memo1.lines.count - 1);
```

Outline topic

How do I do outline drag and drop?
Here is how to fill an outline component from a table.

Using the methods

AddChildObject
GetDataItem

How to use AddChildObject and GetDataItem

```
unit U_outl;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Grids, Outline, Spin;

type
  TForm1 = class(TForm)
    Outline1: TOutline;
    SpinEdit1: TSpinEdit;
    procedure FormCreate(Sender: TObject);
    procedure SpinEdit1Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
const
  NumberOfItems = 4;
var
  l: array[1..NumberOfItems] of integer;

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
var i, indx: integer;
begin
  with outline1 do
  begin
    indx := add(0, 'level 1');
    for i := 1 to NumberOfItems do
    begin
      l[i] := i;
      AddChildObject(indx, 'level 2 item ' + IntToStr(i), @l[i]);
    end;
  end;
  SpinEdit1.MaxValue := NumberOfItems;
  SpinEdit1Change(Sender);
end;

procedure TForm1.SpinEdit1Change(Sender: TObject);
begin
  with outline1 do
    Caption := Items[GetDataItem(@l[SpinEdit1.value])].Text;
end;
```

```
end.
```

Q: How can I use huge arrays? (i.e. > 64K)

A: Unit provides support for arrays and huge (>64K) arrays of data. Each item size stored in the array must be a multiple of 2 (2,4,8,16,32..) in order to work using the huge arrays and matrices. This is needed so that you don't straddle the segment boundaries.

To use these objects merely create the object using the appropriate constructor. The object will be created with it's data initialized to zeros (GMEM_ZEROINIT). to access an element of the object use the AT() method. This will return you a pointer to the element you specified. for huge objects it will do the segment math correctly. Then you can dereference the pointer and work with the data.

This unit merely simplifies the use of huge type arrays and matrices. It does not do much more.

```
Unit Arrays;

{Author ROBERT WARREN CIS ID 70303,537}

interface

uses WObjects,WinTypes,WinProcs;

type

PArray = ^TArray;
TArray = object(TObject)
  Handle: THandle;
  ItemSize: Word;
  Limit: LongInt;
  Address: Pointer;
  constructor Init(aItemSize: Word; aLimit: LongInt);
  destructor done; virtual;
  function At(index: LongInt): Pointer; virtual;
 end;

PMatrix = ^TMatrix;
TMatrix = object(TObject)
  Handle: THandle;
  ItemSize: Word;
  Rows,Cols: LongInt;
  Address: Pointer;
  constructor Init(aItemSize: Word; aRows,aCols: LongInt);
  destructor done; virtual;
  function At(aRow,aCol: LongInt): Pointer; virtual;
 end;

PHugeMatrix = ^THugeMatrix;
THugeMatrix = object(TMatrix)
  SegIncr : Word;
  constructor Init(aItemSize: Word; aRows,aCols: LongInt);
  function At(aRow,aCol: LongInt): Pointer; virtual;
 end;
```

```
PHugeArray = ^THugeArray;
THugeArray = object(TArray)
  SegIncr: Word;
  constructor Init(aItemSize: Word; aLimit: LongInt);
  function At(index: LongInt): Pointer; virtual;
end;

function NewArray(aItemSize: Word; aLimit: LongInt): PArray;
function NewMatrix(aItemSize: Word; aRows,aCols: LongInt): PMatrix;

implementation

{
 returns a pointer to an Array if small enough otherwise a HugeArray
}
function NewArray(aItemSize: Word; aLimit: LongInt): PArray;
var
 TempArrayPtr: PArray;
begin
 TempArrayPtr:=New(PArray,Init(aItemSize,aLimit));
 if TempArrayPtr = nil then
  TempArrayPtr:=New(PHugeArray,Init(aItemSize,aLimit));
 NewArray:=TempArrayPtr;
end;

{
 returns a pointer to an Matrix if small enough otherwise a HugeMatrix
}
function NewMatrix(aItemSize: Word; aRows,aCols: LongInt): PMatrix;
var
 TempMatrixPtr: PMatrix;
begin
 TempMatrixPtr:=New(PMatrix,Init(aItemSize,aRows,aCols));
 if TempMatrixPtr = nil then
  TempMatrixPtr:=New(PHugeMatrix,Init(aItemSize,aRows,aCols));
 NewMatrix:=TempMatrixPtr;
end;

procedure AHIncr; far; external 'KERNEL' index 114;

{ ---------------------------------------------
        TMatrix
  --------------------------------------------- }

constructor TMatrix.Init(aItemSize: Word; aRows,aCols: LongInt);
var
 InitSize: LongInt;
begin
  TObject.Init;
  Rows:=aRows;
  Cols:=aCols;
  ItemSize:=aItemSize;
  InitSize:=LongInt(ItemSize * Rows * Cols);
  if InitSize > $FFFF then fail;
  Handle:=GlobalAlloc(GMEM_MOVEABLE or GMEM_ZEROINIT,ItemSize * Rows * Cols);
  if handle = 0 then fail;
```

```
  Address:=GlobalLock(Handle);
end;

destructor TMatrix.done;
begin
 GlobalUnlock(Handle);
 GlobalFree(Handle);
end;

function TMatrix.At(aRow,aCol: LongInt): Pointer;
var
 pos: Word;
begin
 pos:=(aRow * Cols * ItemSize) + (ACol * ItemSize);
 At:=Pointer(MakeLong(pos,HiWord(LongInt(Address))));
end;

{ ---------------------------------------------
        THugeMatrix
  --------------------------------------------- }

constructor THugeMatrix.Init(aItemSize: Word; aRows,aCols: LongInt);
begin
  TObject.Init;
  Rows:=aRows;
  Cols:=aCols;
  ItemSize:=aItemSize;

  Handle:=GlobalAlloc(GMEM_MOVEABLE or GMEM_ZEROINIT,LongInt(ItemSize * Rows
* Cols));
  if handle = 0 then fail;
  Address:=GlobalLock(Handle);
  SegIncr:=Ofs(AHIncr);
end;

function THugeMatrix.At(aRow,aCol: LongInt): Pointer;
var
 Segs,Offs: Word;
 Pos: LongInt;
begin
  pos:=(aRow * Cols * ItemSize) + (ACol * ItemSize);
  Segs:=Pos div $FFFF;
  Offs:=Pos mod $FFFF;
  At:=Pointer(MakeLong(Offs,((Segs*SegIncr)+(HiWord(LongInt(Address))))));
end;


{ ---------------------------------------------
        TArray
  --------------------------------------------- }

constructor TArray.Init(aItemSize: Word; aLimit: LongInt);
var
 InitSize: LongInt;
begin
  TObject.Init;
  ItemSize:=aItemSize;
```

```
  Limit:=aLimit;
  InitSize:=ItemSize * Limit;
  if InitSize > $FFFF then fail;
  Handle:=GlobalAlloc(GMEM_MOVEABLE or GMEM_ZEROINIT,InitSize);
  if handle = 0 then fail;
  Address:=GlobalLock(Handle);
end;

destructor TArray.Done;
begin
 TObject.Done;
 GlobalUnlock(Handle);
 GlobalFree(Handle);
end;

function TArray.At(index: LongInt): Pointer;
begin
  At:=Pointer(LongInt(ItemSize * index) + LongInt(Address));
end;

{ ---------------------------------------------
        THugeArray
  --------------------------------------------- }

constructor THugeArray.Init(aItemSize: Word; aLimit: LongInt);
begin
  TObject.Init;
  ItemSize:=aItemSize;
  Limit:=aLimit;
  Handle:=GlobalAlloc(GMEM_MOVEABLE or GMEM_ZEROINIT,ItemSize * Limit);
  if handle = 0 then fail;
  Address:=GlobalLock(Handle);
  SegIncr:=Ofs(AHIncr);
end;

function THugeArray.At(index: LongInt): Pointer;
var
 Segs,Offs: Word;
 Pos: LongInt;
begin
  Pos:=Index * ItemSize;
  Segs:=Pos div $FFFF;
  Offs:=Pos mod $FFFF;
  At:=Pointer(MakeLong(Offs,((Segs*SegIncr)+(HiWord(LongInt(Address))))));
end;

begin
end.
```

{ Here is a slider custom component. }

```
unit Slider;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs;

type
  TSliderOrientation = (slHoriz, slVertical);
  TSlider = class(TCustomControl)
  private
    Thumb : TRect;
    MemDC : HDC;
    Bitmap : HBitmap;

    capture : boolean;
    capturePoint : TPoint;
    captureValue : Integer;

    fTrackWidth : Integer;
    fTrackColor : TColor;
    fOrientation : TSliderOrientation;
    fThumbHeight : Integer;
    fThumbColor : TColor;
    fMin : Integer;
    fMax : Integer;
    fValue : Integer;
    fValueChange : TNotifyEvent;
    fCtl3D : boolean;
    procedure SetTrackWidth (value : Integer);
    procedure SetTrackColor (value : TColor);
    procedure SetOrientation (value : TSliderOrientation);
    procedure SetThumbHeight (value : Integer);
    procedure SetThumbColor (value : TColor);
    procedure SetMin (v : Integer);
    procedure SetMax (v : Integer);
    procedure SetValue (value : Integer);
    procedure SetCtl3D (value : boolean);
  protected
    procedure Paint; override;
    procedure MouseDown (Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseUp (Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
    procedure MouseMove (Shift: TShiftState; X, Y: Integer); override;
    procedure DrawThumb; virtual;
  public
    constructor Create (AOwner : TComponent); override;
    destructor Destroy; override;
  published
    property TrackWidth : Integer read fTrackWidth write SetTrackWidth;
    property TrackColor : TColor read fTrackColor write SetTrackColor;
    property ThumbHeight : Integer read fThumbHeight write SetThumbHeight;
```

```
    property ThumbColor : TColor read fThumbColor write SetThumbColor;
    property Orientation : TSliderOrientation read fOrientation write
SetOrientation;
    property Minimum : Integer read fMin write SetMin;
    property Maximum : Integer read fMax write SetMax;
    property Value : Integer read fValue write SetValue;
    property Ctl3D : boolean read fCtl3D write SetCtl3D;
    property OnValueChange : TNotifyEvent read fValueChange write
fValueChange;

    property Color;
    property Enabled;
    property HelpContext;
    property Hint;
    property ParentShowHint;
    property ShowHint;
    property Tag;
    property Visible;

    property OnClick;
    property OnDragDrop;
    property OnDragOver;
    property OnEndDrag;
    property OnEnter;
    property OnExit;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;

  end;

procedure Register;

implementation

constructor TSlider.Create (AOwner : TComponent);
begin
  inherited Create (AOwner);
  Width := 50;
  Height := 200;
  fTrackWidth := 10;
  fOrientation := slVertical;
  fTrackColor := clBtnFace;
  fThumbColor := clBtnFace;
  fMin := 0;
  fMax := 100;
  fValue := 0;
  fThumbHeight := 20;
  fValueChange := Nil;
  fCtl3D := True;
  capture := False;
  thumb.left := -1;
end;

destructor TSlider.Destroy;
begin
  if Bitmap <> 0 then DeleteObject (Bitmap);
```

```
  if MemDC <> 0 then DeleteDC (MemDC);
  inherited Destroy
end;

procedure TSlider.SetTrackWidth (value : Integer);
begin
  if fTrackWidth <> value then
  begin
    fTrackWidth := value;
    Invalidate
  end
end;

procedure TSlider.SetOrientation (value : TSliderOrientation);
begin
  if value <> fOrientation then
  begin
    fOrientation := value;
    Invalidate
  end
end;

procedure TSlider.SetTrackColor (value : TColor);
begin
  if value <> fTrackColor then
  begin
    fTrackColor := value;
    Invalidate
  end
end;

procedure TSlider.SetThumbHeight (value : Integer);
begin
  if value <> fThumbHeight then
  begin
    fThumbHeight := value;
    Invalidate
  end
end;

procedure TSlider.SetThumbColor (value : TColor);
begin
  if value <> fThumbColor then
  begin
    fThumbColor := value;
    Invalidate
  end
end;

procedure TSlider.SetMin (v : Integer);
begin
  if v <> fMin then
  begin
    fMin := V;
    if Value < fMin then Value := fMin;
    Invalidate
  end
```

```
end;

procedure TSlider.SetMax (v : Integer);
begin
  if v <> fMax then
  begin
    fMax := V;
    if Value > fMax then Value := fMax;
    Invalidate
  end
end;

procedure TSlider.SetValue (value : Integer);
begin
  if value < Minimum then value := Minimum
  else if value > Maximum then value := Maximum;

  if value <> fValue then
  begin
    fValue := Value;
    if Assigned (fValueChange) then OnValueChange (self);
    DrawThumb
  end
end;

procedure TSlider.SetCtl3D (value : boolean);
begin
  if value <> fCtl3D then
  begin
    fCtl3D := value;
    Invalidate
  end
end;


procedure TSlider.Paint;
var Rect : TRect;
begin
  with Canvas do
  begin
    if MemDC = 0 then MemDC := CreateCompatibleDC (Canvas.Handle);

    if fOrientation = slVertical then
    begin
      if Bitmap = 0 then
            Bitmap := CreateCompatibleBitmap (Canvas.Handle, Width,
ThumbHeight);
      Rect.top := 0;
      Rect.bottom := Height;
      Rect.left := (Width - TrackWidth) div 2;
      Rect.Right := Rect.Left + TrackWidth
    end
    else
    begin
      if Bitmap = 0 then
            Bitmap := CreateCompatibleBitmap (Canvas.Handle, ThumbHeight,
Height);
```

```
      Rect.top := (Height - TrackWidth) div 2;
      Rect.bottom := Rect.Top + TrackWidth;
      Rect.left := 0;
      Rect.Right := Width
    end;

    Brush.Color := TrackColor;
    if Ctl3D then
    begin
      Pen.Color := clBtnHighlight;
      with Rect do
      begin
        Rectangle (left, top, right, bottom);
        Pen.Color := clBtnShadow;
        MoveTo (left, top);
        LineTo (right, top);
        MoveTo (left, top);
        LineTo (left, bottom)
      end
    end
    else FillRect (Rect);
    DrawThumb;

  end
end;

procedure TSlider.DrawThumb;
var
  basePos : Integer;
  rc : bool;
  oldBmp : HBitmap;
  oldThumb : TRect;
begin
  if csLoading in ComponentState then Exit;
  oldBmp := SelectObject (MemDC, Bitmap);

  if Enabled then Canvas.Brush.Color := ThumbColor
  else Canvas.Brush.Color := clGray;
  if Ctl3D then Canvas.Pen.Color := clBtnHighlight
  else Canvas.Pen.Color := clBlack;
  oldThumb := Thumb;

  if Orientation = slVertical then
  begin
    basePos := (Height - ThumbHeight) * (Value - Minimum)
      div (Maximum - Minimum);
    Thumb.left := 0;
    Thumb.right := Width;
    Thumb.Bottom := Height - BasePos;
    Thumb.top := Thumb.Bottom - ThumbHeight;
    if oldThumb.left <> -1 then with oldThumb do
      BitBlt (Canvas.Handle, Left, Top, Width, ThumbHeight, MemDC, 0, 0,
SRCCOPY);

    with Thumb do
      rc := BitBlt (MemDC, 0, 0, Width, ThumbHeight, Canvas.Handle, Left,
Top, SRCCOPY);
```

```
  end
  else
  begin
    basePos := (Width - ThumbHeight) * (Value - Minimum) div (Maximum -
Minimum);
    Thumb.left := basePos;
    Thumb.Right := Thumb.left + ThumbHeight;
    Thumb.Top := 0;
    Thumb.Bottom := Height;
    if oldThumb.left <> -1 then with oldThumb do
      BitBlt (Canvas.Handle, Left, Top, ThumbHeight, Height, MemDC, 0, 0,
SRCCOPY);

    with Thumb do
      rc := BitBlt (MemDC, 0, 0, ThumbHeight, Height, Canvas.Handle, Left,
Top, SRCCOPY);
  end;

  with Canvas do
  begin
    with Thumb do if Ctl3D then
    begin
      Rectangle (left, top, right-1, bottom-1);
      Pen.Color := clBtnShadow;
      MoveTo (Left + 1, Bottom - 3);
      LineTo (Left + 1, Top+1);
      LineTo (Right - 2, Top+1);
      MoveTo (Left, Bottom - 1);
      LineTo (Right-1, Bottom - 1);
      LineTo (Right-1, Top - 1)
    end
    else
      Rectangle (left, top, right, bottom);
  end;

  SelectObject (MemDC, OldBmp);
end;

procedure TSlider.MouseDown (Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  inherited MouseDown (Button, Shift, X, Y);
  if (Button = mbLeft) and PtInRect (Thumb, Point (X, Y)) then
  begin
    capture := True;
    capturePoint := Point (X, Y);
    captureValue := value;
  end;
end;

procedure TSlider.MouseUp (Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  inherited MouseUp (Button, Shift, X, Y);
  if (Button = mbLeft) then capture := False
end;
```

```
procedure TSlider.MouseMove (Shift: TShiftState; X, Y: Integer);
begin
  inherited MouseMove (shift, X, Y);
  if capture then
    if Orientation = slVertical then
      value := captureValue + Minimum + (Maximum - Minimum) * (capturePoint.Y
- Y) div (Height - ThumbHeight)
    else
      value := captureValue + Minimum + (Maximum - Minimum) * (X -
capturePoint.X) div (Width - ThumbHeight);
end;

procedure Register;
begin
  RegisterComponents('Samples', [TSlider]);
end;

end.
```

TSlider Component Notes

By Colin Wilson - woozle@cix.compulink.co.uk

The TSlider object is a representation of a Slider - as used in mixers, lighting control units, etc.

It defines the following new public properties:

property TrackWidth : Integer

The width of the slider track.

property TrackColor : TColor

The slider track color.

property ThumbHeight : Integer

The height of the Thumb (the bit that slides). The thumb is always as wide as the component - so can be controlled by the Width property.

property ThumbColor : TColor

The thumb colour.

property Orientation : TSliderOrientation

slHorizontal or slVertical. Controls whether the slider slider left/right or up/down.

property Minimum : Integer

The minimum slider value.

property Maximum : Integer

The maximum slider value.

property Value : Integer

The current slider value.

The following new protected procedure is defined:

procedure DrawThumb; virtual;

Can be overriden to draw custom thumbs or thumbs with legends, bitmaps, etc.

The following new event is defined:

property OnValueChange : TNotifyEvent;

Called whenever the value changes

JPEG support is not genericly included in Delphi. Contact Jan Dekkers CIS[72130,353] for an affordable (49$) JPEG/PCX component! You can download a shareware-version from the Delphi forum, too.

Q: How do I manipulate (enable/disable) the 'close' item on the default system menu (i.e. the menu which drops down from the system box when the user clicks it or presses the ALT-SPACEBAR key combination?

A:

```
procedure TMainForm.WMInitMenuPopup(var Msg : TWMInitMenuPopup);
begin
   if ( Msg.SystemMenu ) then
      EnableMenuItem(Msg.MenuPopup, SC_Close, MF_ByCommand or MF_Grayed);
end;
```

TEdit

Q: How do I get everything on the command line? In other words I would like a piece of code that would do the following....

Given the DOS execute line of

utils.exe c:\oscar d:\colpas "oscar colpas"

I would like to get the three parameters into variables in my pascal program to have these string values....

a = "c:\oscar"
b = "d:\colpas"
c = "oscar colpas"

Using ParamStr would ignore the spaces in between oscar and colpas.

A:

A: Delphis System Unit has a variable called CmdLine which points to the full command line (zero-terminated). You can copy it into a Pascal String with StrPas(CmdLine);

Q: How do I call a "C" DLL from Delphi?

A: If you have a DLL written in C++ like this:

```
#define  STRICT
#include <windows.h>
#define DEFINE_EXPORT
#include "lloyd.h"

//****************************************************************
int _EXP MyFunc (int i)
{
  return (i+10);
}
//****************************************************************

#pragma argsused
int FAR PASCAL LibMain( HINSTANCE hInstance,
                        WORD wDataSegment,
                        WORD wHeapSize,
                        LPSTR lpszCmdLine )
{
    if ( wHeapSize != 0 )
         UnlockData( 0 );
    return 1;   // Indicate that the DLL was initialized successfully.
}

#pragma argsused
int FAR PASCAL WEP ( int bSystemExit )
{
    return 1;
}
```

This won't work. There are two ways to fix it. One is to declare the function in question as EXTERN. e.g.

```
extern "C" {
int _EXP MyFunc (int i)
{
  return (i+10);
}
}
```

In this case you need to use cdelc when you prototype the function. Something like this:

```
function MyFunc(i: integer): integer; cdecl; far; external 'MYDLL' index 12;
```

OR

you can declare the functions with FAR PASCAL and then call the function in the normal way

If there are problems, make sure that you declare the DLL functions with "_loadds". Remember that, normally, a DLL function runs on the caller's DS, unless you specifically load the correct one.

Q: How do I make a component that uses the built in editor for a TStrings property?

A: The biggest trick is that when you create the memory for the TStrings property, it must be created as a TStringList even though it must be a TStrings in the class itself. Here is an example of what I mean:

```
unit TestEdit;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TTestEditorClass = class(TListBox)
  private
    { Private declarations }
    FStringListValues: TStrings;
  protected
    { Protected declarations }
  public
    { Public declarations }
    constructor create(AOwner: TComponent); overRide;
    procedure SetStringListValues(Value: TStrings);
  published
    { Published declarations }
    property StringListValues: tstrings read FStringListValues write
SetStringListValues;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Samples', [TTestEditorClass]);
end;

constructor TTestEditorClass.create(AOwner: TComponent);
begin
  inherited create(AOwner);
  FStringListValues := TStringList.create;
end;

procedure TTestEditorClass.SetStringListValues(Value: TStrings);
begin
  StringListValues.Assign(Value);
end;

end.
```

```
table1.FieldDefs.items[SomeValue].name;
```

Note: If you get a "List Index out of bounds" error, it is probably because the TTable object is not active.

Oxymorons

advanced basic
airline food (also: junk food, hospital food)
alone together
amateur expert
baby grand (piano)
black light
Brief survey
civil war
Congressional ethics
criminal justice
crisis management
Deafening silence
Death Benefits
divorce court
down escalator
dry wine
elementary calculus
evaporated milk
extra time
fast idle
flexible freeze (economics)
freezer burn
fresh frozen
friendly arguement
gourmet pizza
governmental efficiency
great depression
guest host
half full (also half empty)
Honest Crook
hopelessly optimistic
idiot savant
industrial park
irate patient
jumbo shrimp
Lean pork
Light-Heavyweight
live recording
mandatory option
marijuana initiative
Medium Large
Mild Abrasive
minor disaster
Mobil station

No-good Do-gooder!
non-stop flight
old news
Organized mess
original copy
Oxymorons in vehicle names: Dodge Ram
Oxymorons in vehicle names: Cherokee Pioneer
partially completed
passive agression
peacekeeper missile
perfect idiot
plastic glasses
pretty ugly
random logic (also fuzzy logic)
rap music
Renegade Lawmakers (from CNN during the battle in Moscow.)
scheduled spontaneity (From a Franklin time management class.)
second best
singular relationship
standard deviation
student teacher
sure bet
sweet sorrow
terribly nice
Turbo Diesel
unacceptable solution
work party
working vacation

Once upon a midnight dreary,
Fingers cramped and vision bleary,
System manuals piled high and wasted paper on the floor,
Longing for the warmth of bedsheets,
Still I sat there, doing spreadsheets;
Having reached the bottom line I took a floppy from the drawer.
Typing with a steady hand,
I then invoked the SAVE command
But got instead a reprimand: it read "Abort, Retry, Ignore?"

Was this some occult illusion?
Some maniacal intrusion?
There were choices Solomon himself had never faced before.
Carefully I weighed the options.
These three seemed to be the top ones.
Clearly, now I must now adopt one: Choose: "Abort, Retry, Ignore?"

With my fingers pale and trembling
Slowly toward the keyboard bending,
Longing for a happy ending, hoping all would be restored,
Praying for some guarantee
Finally I pressed a key ----
But on the screen what did I see? Again: "Abort, Retry, Ignore?"

I tried to catch the chips off-guard---
I pressed again but twice as hard
Luck was just not in the cards. I saw what I had seen before.
Now I typed in desperation
Trying random combinations
Still there came the incantation: Choose: "Abort, Retry, Ignore?"

There I sat, distraught, exhausted
by my own machine accosted
Getting up I turned away and paced across the office floor.
And then I saw an awful sight:
A bold and blinding flash of light---
A lightning bolt had cut the night and shook me to my core.
I saw the screen collapse and die;
"No No my database", I cried
I thought I heard a voice reply, "You'll see your data NEVERMORE!"

To this day I do not know
The place to which lost data goes
I bet it goes to heaven where the angels have it stored.
But as for productivity, well
I fear that it goes straight to hell

And that's the tale I have to tell. Your choice: "Abort, Retry, Ignore?"

* * * * * * * * * * * * * * * * * * *
WHY DID THE CHICKEN CROSS THE ROAD ?
* * * * * * * * * * * * * * * * * * *

****
The Star Trek Answers
****

Chakotay: Whatever its reason, whatever its goals, we should respect its right to cross the road and seek its own spiritual awareness.

Neelix: Actually, Captain, I'm not really familiar with the chickens in this system. But, if you can catch it, I can cook it.

Riker: I don't know why, but I know how: with pleasure, sir.

Worf: I don't know. KLINGON chickens do NOT cross roads.

HoloDoc: How should I know? No one tells me anything around here. I didn't even know we added chickens to the crew. All I know is that it would have been nice, BEFORE the chicken went off to the cross the road, if it had remembered to turn me off!

Dr. Crusher: If there's nothing wrong with the chicken, there must be something wrong with the universe.

Dr. Soran: His heart just wasn't in it. (Scenes of chicken torture with nanoprobes have been edited out.)

Scotty: Because she couldna take much morrrrrre.

Odo: I don't know, but I'm sure it must be Quark's fault.

Quark: Who, me?

Charlie X: Because it didn't want to STAY...STAY...STAY...

Kirk: You chicken bastard, you killed my son...YOU chicken BASTARD, you killed...my SON...you CHICKEN bastard....youkilledmy...son!

Troi: I feel the chicken's pain!

Kira: It was probably being chased by those cursed Cardassians.

Bones: Dammit, I'm a doctor, not an ornithologist!

Data: The chicken, in observing that it was on the opposite side of the 20th century Terran paved roadway, was aware that its immediate goal should have been to traverse the distance without interception by an kind of combustion-propelled personal trans perambulate upon a conveyance normally reserved for the usage of...yes, sir.

Dr. Bashir: It probably heard about my amazing medical skills not to mention my sexual prowess and came to get some pointers.

The Borg: Crossing the road is irrelevant. The chicken will be assimilated.

Hugh the Borg: Maybe it just needed a big hug!

B'Elanna: I'm sure it felt suffocated by all the bleeping regulations of bleeping Starfleet and just couldn't stand it any longer!

Picard: There are four lights!

Q: Wouldn't you like to know? Too bad your puny human brain wouldn't be able to comprehend the answer.

Uhura: Shall I open hailing frequencies so you can ask it, sir?

Tasha: That depends...was it fully functional?

Chekov: It must have been on its way to assist in saving my life for the billionth time..did I scream this time?

Khan: With my last breath I spit at the chicken...

Harry: I don't know, it's my first mission.

Paris: Well, I think that...say, that's a lovely shirt you're wearing.

Harvey Mudd: Chicken? I don't remember any chicken. No no no, there's been a terrible misunderstanding.

Janeway: Its primary goal was no doubt to get back to the Alpha Quadrant...and it probably misses its dog.

Nurse Chapel: Oh, Spock!

Lwaxana: Oh, Jean-Luc!

Spock: Fascinating, Captain.

V'Ger: To join with the Creator.

The Grand Nagus: Stupid chicken! You don't cross the road all at once! You sneak across it quietly, without anyone noticing!

Gul Dukat: Well, that's a very interesting question...I'm sure we can work out some kind of arrangement to obtain that information that will be to everyone's satisfaction.

Kes: It was remembering back to the times when its ancestors crossed roads all the time! They lost those abilities because they stopped using them!

O'Brien: No problem, Commander, I'll get right on it.

Wesley: I'm not sure, but I can figure it out if I reroute these systems and reconfigure the warp field and run a complete internal whootchacallit on the computers and...

Sisko: It was seeking deeper meaning. Jake, do you see what we've learned from all this?

Jake: Check out the babe that just came off that transport!

Geordi: Well, wherever it's going, I'm sure it'll have more luck with women than I do.

Sulu: Don't call me Tiny!

Sarek: Sometimes logic fails me where chickens are concerned.

Mr. Homn:

Dax: To get to the other side. Kurzon might have disagreed with me, Tobin I'm sure wouldn't have had a clue,and then there's...

Tuvok: That's not a question we'd prefer to hear from a senior officer. It makes the junior officers nervous.

****
Other's Answers
****

Plato: For the greater good.

Karl Marx: It was a historical inevitability.

Hamlet: Because 'tis better to suffer in the mind the slings and arrows of outrageous road maintenance than to take arms against a sea of oncoming vehicles.

Timothy Leary: Because that's the only kind of trip the Establishment would let it take.

Douglas Adams: Forty-two.

Nietzsche: Because if you gaze too long across the Road, the Road gazes also across you.

Oliver North: National Security was at stake.

Gary Gygax: Because I rolled a 64 on the "Chicken Random Behaviors" chart on page 497 of the Dungeon Master's Guide.

Trent Reznor: Because the world is F----- UP and it HATES ITSELF for being such a PITIFUL WHINY USELESS S---!

Jean-Luc Picard: To see what's out there.

Darth Vader: Because it could not resist the power of the Dark Side.

Albert Einstein: Whether the chicken crossed the road or the road crossed the chicken depends upon your frame of reference.

Roseanne Barr: Urrrrrp. What chicken?

Buddha: If you ask this question, you deny your own chicken-nature.

Salvador Dali: The Fish.

Darwin: It was the logical next step after coming down from the trees.

Bob Dylan: How many roads must one chicken cross?

Gerald R. Ford: It probably fell from an airplane and couldn't stop its forward momentum.

Sigmund Freud: The chicken obviously was female and obviously interpreted the pole on which the crosswalk sign was mounted as a phallic symbol of which she was envious, selbstverstaendlich.

Saddam Hussein: This was an unprovoked act of rebellion and we were quite justified in dropping 50 tons of nerve gas on it.

Lee Iacocca: It found a better car, which was on the other side of the road.

Martin Luther King: It had a dream.

James Tiberius Kirk: To boldly go where no chicken has gone before.

Stan Laurel: I'm sorry, Ollie. It escaped when I opened the run.

Groucho Marx: Chicken? What's all this talk about chicken? Why, I had an uncle who thought he was a chicken. My aunt almost divorced him, but we needed the eggs.

Karl Marx: To escape the bourgeois middle-class struggle.

Sir Isaac Newton: Chickens at rest tend to stay at rest. Chickens in motion tend to cross the road.

Jack Nicholson: 'Cause it (censored) wanted to. That's the (censored) reason.

Thomas Paine: Out of common sense.

Michael Palin: Nobody expects the banished inky chicken!

Ronald Reagan: I forget.

John Sununu: The Air Force was only too happy to provide the transportation, so quite understandably the chicken availed himself of the opportunity.

The Sphinx: You tell me.

Mae West: I invited it to come up and see me sometime.

Mr. Scott: 'Cos ma wee transporter beam was na functioning properly. Ah canna work miracles, Captain!

Howard Cosell: It may very well have been one of the most astonishing events to grace the annals of history. An historic, unprecedented avian biped with the temerity to attempt such an herculean achievement formerly relegated to homo sapien pedestrians is truly a remarkable occurrence.

Troutman's Laws of Computer Programming

1. Any running program is obsolete.
2. Any planned program costs more and takes longer.
3. Any useful program will have to be changed.
4. Any useless program will have to be documented.
5. The size of a program expands to fill all available memory.
6. The value of a program is inversely proportional to the weight of output
7. The complexity of a program grows until it exceeds the capability of the maintainers.
8. Information necessitating a change in design is always conveyed to the implementors after the code is written. Corollary: Given a simple choice between one obviously right way and one obviously wrong way, it is often wiser to choose the wrong way, so as to expedite subsequent revision.
9. The more innocuous a modification appears, the more code it will require rewriting.
10. If a test installation functions perfectly, all subsequent systems will malfunction.
11. Not until a program has been in production for at least six months will the most harmful error be discovered.
12. Interchangeable modules won't.
13. Any system that relies on computer reliability is unreliable.
14. Any system that relies on human reliability is unreliable.
15. Investment in reliability increases until it exceeds the probable cost of errors, or until someone insists on getting some useful work done.
16. Adding manpower to a late software project makes it later.
17. There's always one more bug.

Real Programmers Don't Write Specs

Real Programmers don't write specs -- users should consider themselves lucky to get any programs at all and take what they get.

Real Programmers don't comment their code. If it was hard to write, it should be hard to understand and even harder to modify.

Real Programmers don't write application programs; they program right down on the bare metal. Application programming is for feebs who can't do systems programming.

Real Programmers don't eat quiche. In fact, real programmers don't know how to SPELL quiche. They eat Twinkies, and Szechwan food.

Real Programmers don't write in COBOL. COBOL is for wimpy applications programmers.

Real Programmers' programs never work right the first time. But if you throw them on the machine they can be patched into working in "only a few" 30-hour debugging sessions.

Real Programmers don't write in FORTRAN. FORTRAN is for pipe stress freaks and crystallography weenies.

Real Programmers never work 9 to 5. If any real programmers are around at 9 AM, it's because they were up all night.

Real Programmers don't write in BASIC. Actually, no programmers write in BASIC, after the age of 12.

Real Programmers don't write in PL/I. PL/I is for programmers who can't decide whether to write in COBOL or FORTRAN.

Real Programmers don't play tennis, or any other sport that requires you to change clothes. Mountain climbing is OK, and real programmers wear their climbing boots to work in case a mountain should suddenly spring up in the middle of the machine room.

Real Programmers don't document. Documentation is for simps who can't read the listings or the object deck.

Real Programmers don't write in PASCAL, or BLISS, or ADA, or any of those pinko computer science languages. Strong typing is for people with weak memories.

Real Programmers only write specs for languages that might run on future hardware. Noboby trusts them to write specs for anything homo sapiens will ever be able to fit on a single planet.

Real Programmers don't play tennis or any other sport which requires a change of clothes. Mountain climbing is ok, and real programmers often wear climbing boots to work in case a mountain should suddenly spring up in the middle of the machine room.

Real Programmers spend 70\% of their work day fiddling around and then get more done in the other 30\% than a user could get done in a week.

Real Programmers are surprised when the odometers in their cars don't turn from 99999 to 9999A.

Real Programmers are concerned with the aesthetics of their craft; they will writhe in pain at shabby workmanship in a piece of code.

Real Programmers will defend to the death the virtues of a certain piece of peripheral equipment, especially their lifeline, the terminal.

Real Programmers never use hard copy terminals, they never use terminals that run at less than 9600 baud, they never use a terminal at less than its maximum practical speed.

Real Programmers think they know the answers to your problems, and will happily tell them to you rather than answer your questions.

Real Programmers never program in COBOL, money is no object.

Real Programmers never right justify text that will be read on a fixed-character-width medium.

Real Programmers wear hiking boots only when it's much too cold to wear sandals. When it's only too cold, they wear socks with their sandals.

Real Programmers don't think that they should get paid at all for their work, but they know that they're worth every penny that they do make.

Real Programmers log in first thing in the morning, last thing before they go to sleep, and stay logged in for lots of time in between.

Real programmers don't draw flowcharts. Flowcharts are after all, the illerate's form of documentation.

Real Programmers don't use Macs. Computers which draw cute little pictures are for wimps.

Real Programmers don't read manuals. Reliance on a reference is the hallmark of a novice and a coward.

Real Programmers don't write in COBOL. COBOL is for gum chewing twits who maintain ancient payroll programs.

Real Programmers don't write in FORTRAN. FORTRAN is for wimpy engineers who wear white socks. The get excited over finite state analysis and nuclear reactor simulations.

Real Programmers don't write in Modula-2. Modula-2 is for insecure analretentives who can't choose between Pascal and COBOL.

Real Programmers don't write in APL, unless the whole program can be written on one line.

Real Programmers don't write in Lisp. Only effeminate programmers use more parentheses than actual code.

Real Programmers don't write in Pascal, Ada or any of those other pinko computer science languages. Strong variable typing is for people with weak memories.

Real Programmers distain structured programming. Structured programming is for compulsive neurotics who were prematurely toilet trained. They wear neckties and carefully line up sharp pencils on an otherwise clear desk.

Real Programmers scorn floating point arithmetic. The decimal point was invented for pansy bedwetters who are unable to think big.

Real Programmers know every nuance of every instruction and use them all in every Real Program. Some candyass architectures won't allow EXECUTE instructions to address another EXECUTE instruction as the target instruction. Real Programmers despise petty restrictions.

Real Programmers Don't use PL/I. PL/I is for insecure momma's boys who can't choose between Cobol and Fortran.

Real Programmers don't like the team programming concept. Unless, of course, they are the Chief Programmer.

Real Programmers have no use for managers. Managers are sometimes a necessary evil. Managers are good for dealing with personnel bozos, bean counters, senior planners and other mental defectives.

Real programmers ignore schedules.

Real Programmers don't bring brown bag lunches to work. If the vending machine sells it, they eat it. If the vending machine doesn't sell it, they don't eat it.

Real Programmers think better when playing Adventure or Rogue.

Real Programmers use C since it's the easiest language to spell.

Real Programmers don't use symbolic debuggers, who needs symbols.

Real Programmers only curse at inanimate objects.

Menu Help

How do I assign a method to a dynamically created menu item?
What is the easiest way to change the control menu of a form based application ?
How do I manipulate (enable/disable) the 'close' item on the default system menu?

Q: How do I assign a method to a dynamically created menu item?

A:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Menus;

type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    one1: TMenuItem;
    Button1: TButton;
    N111: TMenuItem;
    N121: TMenuItem;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure MonkeyBoy(sender: TObject);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  m: TMenuItem;
begin
  m := TMenuItem.create(self);
  with m do
  begin
    name := 'Lloyd1';
    caption := '1-3';
    OnClick := MonkeyBoy;
  end;
  one1.add(m);
end;

procedure TForm1.MonkeyBoy(sender: TObject);
begin
  ShowMessage('Lloyd is the greatest!!!');
end;


end.
```

Sendkeys

Here is the poor man's version:

```
 procedure keybd_Event; external 'USER' index 289;

 procedure PostVKey(bVirtKey: byte; Up: Boolean);
 var
  AXReg,BXReg : Word;
  AXHigh, AXLow, BXHigh, BXLow : Byte;

function MakeWord(L,H: Byte): Word;
begin
  MakeWord := (H shl 8) + L;
end;

begin
  AXLow := bVirtKey;
  if up then AXHigh := $80 else AXHigh := $0;
  AXreg := MakeWord(AXLow,AXHigh);
  BXLow := VkKeyScan(bVirtKey);
  BXHigh := 0;
  BXReg := MakeWord(BXLow,BXHigh);
  asm
    mov bx,BXreg;
    mov ax,AXReg;
  end;
  Keybd_Event;
end;
```

then to simulate Shift+Ins you need:-

```
PostVKey(VK_Shift,false);
PostVKey(VK_Insert,false);
PostVKey(VK_Insert,True);
PostVKey(VK_Shift,True);
```

Here is the Rolls-Royce version:

Note: This is commercial and copyrighted code. The source code may not be sold for profit (unless Steve is doing the selling).

```
{This unit is to be included in the app that you are running.}
unit SKeys;

interface

type
  { Return values for SendKeys function }
  TSendKeyError = (sk_None, sk_FailSetHook, sk_InvalidToken,
sk_UnknownError);
```

```
function SendKeys(S: String): TSendKeyError;

implementation

function SendKeys; external 'SendKey' index 2;

end.

(*****************************************)

{Here is the DLL that is used.}

library SendKey;

uses
 SysUtils, WinTypes, WinProcs, Messages, Classes, KeyDefs;

type
  { Error codes }
  TSendKeyError = (sk_None, sk_FailSetHook, sk_InvalidToken,
sk_UnknownError);

  { exceptions }
  ESendKeyError = class(Exception);
  ESetHookError = class(ESendKeyError);
  EInvalidToken = class(ESendKeyError);

  { a TList descendant that know how to dispose of its contents }

  TMessageList = class(TList)
  public
    destructor Destroy; override;
  end;

destructor TMessageList.Destroy;
var
  i: longint;
begin
  { deallocate all the message records before discarding the list }
  for i := 0 to Count - 1 do
    Dispose(PEventMsg(Items[i]));
  inherited Destroy;
end;

var
  { variables global to the DLL }
  MsgCount: word;
  MessageBuffer: TEventMsg;
  HookHandle: hHook;
  Playing: Boolean;
  MessageList: TMessageList;
  AltPressed, ControlPressed, ShiftPressed: Boolean;

  NextSpecialKey: TKeyString;

function MakeWord(L, H: Byte): Word;
```

```
{ macro creates a word from low and high bytes }
inline(
  $5A/            { pop dx }
  $58/            { pop ax }
  $8A/$E2);       { mov ah, dl }

procedure StopPlayback;
{ Unhook the hook, and clean up }
begin
  { if Hook is currently active, then unplug it }
  if Playing then
    UnhookWindowsHookEx(HookHandle);
  MessageList.Free;
  Playing := False;
end;

function Play(Code: integer; wParam: word; lParam: Longint): Longint; export;

{ This is the JournalPlayback callback function.  It is called by Windows }
{ when Windows polls for hardware events.  The code parameter indicates
what }
{ to do. }
begin
  case Code of

    hc_Skip: begin
    { hc_Skip means to pull the next message out of our list. If we }
    { are at the end of the list, it's okay to unhook the JournalPlayback }
    { hook from here. }
      { increment message counter }
      inc(MsgCount);
      { check to see if all messages have been played }

      if MsgCount >= MessageList.Count then
        StopPlayback
      else
      { copy next message from list into buffer }
      MessageBuffer := TEventMsg(MessageList.Items[MsgCount]^);
      Result := 0;
    end;

    hc_GetNext: begin
    { hc_GetNext means to fill the wParam and lParam with the proper }
    { values so that the message can be played back.  DO NOT unhook }
    { hook from within here.  Return value indicates how much time until }
    { Windows should playback message.  We'll return 0 so that it's }

    { processed right away. }
      { move message in buffer to message queue }
      PEventMsg(lParam)^ := MessageBuffer;
      Result := 0  { process immediately }
    end

    else
      { if Code isn't hc_Skip or hc_GetNext, then call next hook in chain }
      Result := CallNextHookEx(HookHandle, Code, wParam, lParam);
  end;
```

```
end;

procedure StartPlayback;
{ Initializes globals and sets the hook }
begin
  { grab first message from list and place in buffer in case we }

  { get a hc_GetNext before and hc_Skip }
  MessageBuffer := TEventMsg(MessageList.Items[0]^);
  { initialize message count and play indicator }
  MsgCount := 0;
  { initialize Alt, Control, and Shift key flags }
  AltPressed := False;
  ControlPressed := False;
  ShiftPressed := False;
  { set the hook! }
  HookHandle := SetWindowsHookEx(wh_JournalPlayback, Play, hInstance, 0);
  if HookHandle = 0 then
    raise ESetHookError.Create('Couldn''t set hook')
  else

    Playing := True;
end;

procedure MakeMessage(vKey: byte; M: word);
{ procedure builds a TEventMsg record that emulates a keystroke and }
{ adds it to message list }
var
  E: PEventMsg;
begin
  New(E);                                  { allocate a message record }
  with E^ do begin
    Message := M;                          { set message field }
    { high byte of ParamL is the vk code, low byte is the scan code }
    ParamL := MakeWord(vKey, MapVirtualKey(vKey, 0));

    ParamH := 1;                           { repeat count is 1 }
    Time := GetTickCount;                  { set time }
  end;
  MessageList.Add(E);
end;

procedure KeyDown(vKey: byte);
{ Generates KeyDownMessage }
begin
  { don't generate a "sys" key if the control key is pressed (Windows
quirk) }
  if (AltPressed and (not ControlPressed) and (vKey in [Ord('A')..Ord('Z')]))
or
     (vKey = vk_Menu) then
    MakeMessage(vKey, wm_SysKeyDown)
  else
    MakeMessage(vKey, wm_KeyDown);

end;

procedure KeyUp(vKey: byte);
```

```
{ Generates KeyUp message }
begin
  { don't generate a "sys" key if the control key is pressed (Windows
quirk) }
  if AltPressed and (not ControlPressed) and (vKey in [Ord('A')..Ord('Z')])
then
    MakeMessage(vKey, wm_SysKeyUp)
  else
    MakeMessage(vKey, wm_KeyUp);
end;

procedure SimKeyPresses(VKeyCode: Word);
{ This function simulates keypresses for the given key, taking into }
{ account the current state of Alt, Control, and Shift keys }

begin
  { press Alt key if flag has been set }
  if AltPressed then
    KeyDown(vk_Menu);
  { press Control key if flag has been set }
  if ControlPressed then
    KeyDown(vk_Control);
  { if shift is pressed, or shifted key and control is not pressed... }
  if (((Hi(VKeyCode) and 1) <> 0) and (not ControlPressed)) or ShiftPressed
then
    KeyDown(vk_Shift);    { ...press shift }
  KeyDown(Lo(VKeyCode));  { press key down }
  KeyUp(Lo(VKeyCode));    { release key }

  { if shift is pressed, or shifted key and control is not pressed... }
  if (((Hi(VKeyCode) and 1) <> 0) and (not ControlPressed)) or ShiftPressed
then
    KeyUp(vk_Shift);      { ...release shift }
  { if shift flag is set, reset flag }
  if ShiftPressed then begin
    ShiftPressed := False;
  end;
  { Release Control key if flag has been set, reset flag }
  if ControlPressed then begin
    KeyUp(vk_Control);
    ControlPressed := False;
  end;
  { Release Alt key if flag has been set, reset flag }

  if AltPressed then begin
    KeyUp(vk_Menu);
    AltPressed := False;
  end;
end;

procedure ProcessKey(S: String);
{ This function parses each character in the string to create the message
list }
var
  KeyCode: word;
  Key: byte;
  index: integer;
```

```
  Token: TKeyString;
begin
  index := 1;
  repeat
    case S[index] of

      KeyGroupOpen : begin
      { It's the beginning of a special token! }
        Token := '';
        inc(index);
        while S[index] <> KeyGroupClose do begin

          { add to Token until the end token symbol is encountered }
          Token := Token + S[index];
          inc(index);
          { check to make sure the token's not too long }
          if (Length(Token) = 7) and (S[index] <> KeyGroupClose) then
            raise EInvalidToken.Create('No closing brace');
        end;
        { look for token in array, Key parameter will }
        { contain vk code if successful }
        if not FindKeyInArray(Token, Key) then

          raise EInvalidToken.Create('Invalid token');
        { simulate keypress sequence }
        SimKeyPresses(MakeWord(Key, 0));
      end;

      AltKey : begin
        { set Alt flag }
        AltPressed := True;
      end;

      ControlKey : begin
        { set Control flag }
        ControlPressed := True;
      end;

      ShiftKey : begin
        { set Shift flag }
        ShiftPressed := True;
      end;

      else begin
      { A normal character was pressed }

        { convert character into a word where the high byte contains }
        { the shift state and the low byte contains the vk code }
        KeyCode := vkKeyScan(MakeWord(Byte(S[index]), 0));
        { simulate keypress sequence }
        SimKeyPresses(KeyCode);
      end;
    end;
    inc(index);
  until index > Length(S);
end;
```

```
function SendKeys(S: String): TSendKeyError; export;
{ This is the one entry point.  Based on the string passed in the S  }
{ parameter, this function creates a list of keyup/keydown messages, }

{ sets a JournalPlayback hook, and replays the keystroke messages.   }
var
  i: byte;
begin
  try
    Result := sk_None;                    { assume success }
    MessageList := TMessageList.Create;   { create list of messages }
    ProcessKey(S);                        { create messages from string }
    StartPlayback;                        { set hook and play back messages }
  except
    { if an exception occurs, return an error code, and clean up }
    on E:ESendKeyError do begin

      MessageList.Free;
      if E is ESetHookError then
        Result := sk_FailSetHook
      else if E is EInvalidToken then
        Result := sk_InvalidToken;
    end
    else
      { Catch-all exception handler ensures than an exception }
      { doesn't walk up into application stack }
      Result := sk_UnknownError;
  end;
end;

exports
  SendKeys index 2;

begin
end.

(*****************************************)

unit Keydefs;

interface

uses WinTypes;

const
  MaxKeys = 24;
  ControlKey = '^';
  AltKey = '@';
  ShiftKey = '~';
  KeyGroupOpen = '{';
  KeyGroupClose = '}';

type
  TKeyString = String[7];

  TKeyDef = record
    Key: TKeyString;
```

```
    vkCode: Byte;
  end;

const
  KeyDefArray : array[1..MaxKeys] of TKeyDef = (
    (Key: 'F1';     vkCode: vk_F1),
    (Key: 'F2';     vkCode: vk_F2),
    (Key: 'F3';     vkCode: vk_F3),
    (Key: 'F4';     vkCode: vk_F4),
    (Key: 'F5';     vkCode: vk_F5),

    (Key: 'F6';     vkCode: vk_F6),
    (Key: 'F7';     vkCode: vk_F7),
    (Key: 'F8';     vkCode: vk_F8),
    (Key: 'F9';     vkCode: vk_F9),
    (Key: 'F10';    vkCode: vk_F10),
    (Key: 'F11';    vkCode: vk_F11),
    (Key: 'F12';    vkCode: vk_F12),
    (Key: 'INSERT'; vkCode: vk_Insert),
    (Key: 'DELETE'; vkCode: vk_Delete),
    (Key: 'HOME';   vkCode: vk_Home),
    (Key: 'END';    vkCode: vk_End),
    (Key: 'PGUP';   vkCode: vk_Prior),
    (Key: 'PGDN';   vkCode: vk_Next),

    (Key: 'TAB';    vkCode: vk_Tab),
    (Key: 'ENTER';  vkCode: vk_Return),
    (Key: 'BKSP';   vkCode: vk_Back),
    (Key: 'PRTSC';  vkCode: vk_SnapShot),
    (Key: 'SHIFT';  vkCode: vk_Shift),
    (Key: 'ESCAPE'; vkCode: vk_Escape));


function FindKeyInArray(Key: TKeyString; var Code: Byte): Boolean;

implementation

uses SysUtils;

function FindKeyInArray(Key: TKeyString; var Code: Byte): Boolean;
{ function searches array for token passed in Key, and returns the }

{ virtual key code in Code. }
var
  i: word;
begin
  Result := False;
  for i := Low(KeyDefArray) to High(KeyDefArray) do
    if UpperCase(Key) = KeyDefArray[i].Key then begin
      Code := KeyDefArray[i].vkCode;
      Result := True;
      Break;
    end;
end;

end.
```

Q: How can I use the aggregate functions (avg, sum, count, max, min) with a table?

A:

```
select customer.company, Sum(orders.AmountPaid)
from customer, orders
where (customer.CustNo = orders.CustNo)
group by customer.company
order by customer.company
```

Here are some examples of how to use the TBlobStream to get text from a memofield.

```
{This one copies from one memo field to another.}

procedure TForm1.Button2Click(Sender: TObject);
var
  b1, b2: TBlobStream;
begin
  b1 := TBlobStream.create(table1notes, bmRead);
  table2.edit;
  b2 := TBlobStream.create(table2mymemo, bmReadWrite);
  b2.CopyFrom(b1, b1.size);
  b1.free;
  b2.free;
  table2.post;
end;


{Write to a stream.}
procedure TForm1.Button1Click(Sender: TObject);
var
  Stream: TBlobStream;
  s: string;
begin
  table1.edit;
  Stream := TBlobStream.Create(Table1Notes, bmReadWrite);
  Stream.Seek(0, 2); {Seek 0 bytes from the stream's end point (2).}
  s := 'Lloyd is really cool.';
  Stream.write(s[1], length(s));
  Stream.Free;
  table1.post;
end;

{Read from a stream.}
procedure TForm1.Button1Click(Sender: TObject);
var
  Buffer: PChar;
  Size: Integer;
  Stream: TBlobStream;
begin
  Stream := TBlobStream.Create(Table1MemoField, bmRead);

{ Here is how it is done from a query: }
{  Stream := TBlobStream.Create(query1.FieldByName('MemoField') as
TBlobField, bmRead);}

  Size := Stream.Seek(0, 2);
  Stream.Seek(0, 0);
  Inc(Size);
  GetMem(Buffer, Size);
  FillChar(Buffer^, Size, #0);
  Stream.Read(Buffer^, Size);
  Memo1.SetTextBuf(Buffer);
  FreeMem(Buffer, Size);
  Stream.Free;
end;
```

See also:
How do I fill a TMemo from a PChar using a TBlobStream?

Q: How do I make tables in a loop?

A:

```
procedure TForm1.Button1Click(Sender: TObject);
const
  MaxTableCount = 100;
  MaxFieldCount = 50;
var
  t: tTable;
  TableCount, xFieldCount: integer;
begin
  for TableCount := 1 to MaxTableCount do begin
    t := tTable.create(self);
    with t do begin
      DatabaseName := 'Lloyd';
      TableName := 'hoser' + IntToStr(TableCount);
      TableType := ttParadox;

      with FieldDefs do begin
        Clear;
        for xFieldCount := 1 to MaxFieldCount do
          Add('Field' + IntToStr(xFieldCount), ftInteger, 0, false);
      end;

      with IndexDefs do begin
        Clear;
        Add('Field1Index', 'Field1', [ixPrimary, ixUnique]);
      end;

      CreateTable;
      close;
      free;
    end;
  end;
end;
```

Q: How do I disable and enable the keyboard ?

A: Here is a DLL to do it:

```
Library KillKB;

Uses Wintypes, WinProcs
{$IFNDEF VER80}
  ,Win31
{$ENDIF}
  ;
Var
  oldHook: HHook;

Function KbHook( code: Integer; wparam: Word; lparam: LongInt ): LongInt;
  export;
Begin
  If code < 0 Then
    KbHook := CallNextHookEx( oldHook, code, wparam, lparam )
  Else
    KbHook := 1;
End; { KbHook }

Function DisableKeyboard: Boolean; export;
Begin
  oldHook := SetWindowsHookEx( WH_KEYBOARD, KbHook, HInstance, 0 );
  DisableKeyboard := oldHook <> 0;
End;

Procedure EnableKeyboard; export;
Begin
  If oldHook <> 0 Then Begin
    UnhookWindowshookEx( oldHook );
    oldHook := 0;
  End; { If }
End;

exports
DisableKeyboard index 1,
EnableKeyboard index 2;

Begin
  oldHook := 0;
End.
```

Note: There are a few key combinations that are not passed on to apps at all so they cannot be trapped by a hook. Ctrl-Alt-Del may be one of them but I'm not sure. Just try to see if you can get the blue screen when you have disabled the keyboard via the DLL. Other dubious candiates are Alt-Tab and Ctrl-Esc.

Q: How do I save an object to a disk file?

A: Use a stream to write to a disk file. The object must be a component and is written to the stream like this:

```
var Stream : TFileStream ;
begin
  Stream := TFileStream.Create( 'AFile', fmCreate ) ;
  try
    Stream.WriteComponent( Button1 ) ;
    Stream.WriteComponent( Grid1 ) ; etc.
  finally
    Stream.Free ;
  end ;
end ;
```

To read it back, do this:

```
var Stream : TFileStream ;
    Button2 : TButton ;
    Grid2 : TStringGrid ;
begin
  Stream := TFileStream.Create( 'AFile', fmOpenRead ) ;
  try
    Button2 := Stream.ReadComponent( nil ) as TButton ;
    Stream.WriteComponent( Grid1 ) ; etc.
  finally
    Stream.Free ;
  end ;
end ;
```

At some point you need to register the classes you're going to write and read. For example, you could put the following in the forms OnCreate handler:

```
  RegisterClass( TButton ) ;
  RegisterClass( TStringGrid ) ;
```

If you don't register the classes you'll get a 'Class not found' error when you try to read the object back.

TReport

How do I connect to TReport?

Connect (TReport)

```
var
  DatabaseName, ServerName, UserName, ThePassword: string;
begin
  ServerName := 'BigRedS';
  UserName := 'SYSDBA';
  ThePassword := 'masterkey';
  if LocalInterbase then DatabaseName := 'd:\delphi\work\bigreds.gdb'
  else DatabaseName := 'conan@vol1:\delphi\work\bigreds.gdb';

  if Report1.Connect(ctIDAPIInterBase, ServerName, UserName, ThePassword,
DatabaseName) then Report1.Run;
end;
```

The Server type is in reports.pas. Here are the constants:

```
const
  ctDBase = 2;
  ctExcel = 3;
  ctParadox = 4;
  ctAscii = 5;
  ctSqlServer = 6;
  ctOracle = 7;
  ctDB2 = 8;
  ctNetSQL = 9;
  ctSybase = 10;
  ctBtrieve = 11;
  ctGupta = 12;
  ctIngres = 13;
  ctWatcom = 14;
  ctOcelot = 15;
  ctTeraData = 16;
  ctDB2Gupta = 17;
  ctAS400 = 18;
  ctUnify = 19;
  ctQry = 20;
  ctMinNative = 2;
  ctMaxNative = 20;
  ctODBCDBase = 40;
  ctODBCExcel = 41;
  ctODBCParadox = 42;
  ctODBCSqlServer = 43;
  ctODBCOracle = 44;
  ctODBCDB2 = 45;
  ctODBCNetSql = 46;
  ctODBCSybase = 47;
  ctODBCBtrieve = 48;
  ctODBCGupta = 49;
  ctODBCIngres = 50;
  ctODBCDB2Gupta = 51;
  ctODBCTeraData = 52;
  ctODBCAS400 = 53;
  ctODBCDWatcom = 54;
  ctODBCDefault = 55;
  ctODBCUnify = 56;
```

```
ctMinODBC = 40;
ctMaxODBC = 56;
ctIDAPIStandard = 60;
ctIDAPIParadox = 61;
ctIDAPIDBase = 62;
ctIDAPIAscii = 63;
ctIDAPIOracle = 64;
ctIDAPISybase = 65;
ctIDAPINovSql = 66;
ctIDAPIInterbase = 67;
ctIDAPIIBMEE = 68;
ctIDAPIDB2 = 69;
ctIDAPIInformix = 70;
ctMinIDAPI = 60;
ctMaxIDAPI = 70;
```

Q: How do I have lines in a listbox as different colors?

A: Neil Rubenking posted this code about a week ago, and I have altered it a little. It should draw all the entries in a listbox as red. (Don't forget to change the Listbox's style to lbOwnerDrawFixed.)

```
procedure TForm1.ListBox1DrawItem(Control: TWinControl; Index: Integer;
  Rect: TRect; State: TOwnerDrawState); VAR
  S : String;
  Temp: array[0..255] of Char;
begin
  WITH Control AS TListBox, Canvas DO
    BEGIN
      S := Items[Index];
      FillRect(Rect);
      MoveTo(Rect.Left+2, Rect.Top);
      SetTextAlign(Canvas.Handle, TA_LEFT OR TA_UPDATECP);
      Font.Color := clRed;
      StrPCopy(Temp, S);
      WinProcs.TextOut(Canvas.Handle, 0, 0, Temp, StrLen(Temp));
    END;
end;

{******************************}
```

Here is a version that does not use the API call, but does the same thing:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    procedure FormActivate(Sender: TObject);
    procedure ListBox1DrawItem(Control: TWinControl; Index: Integer;
      Rect: TRect; State: TOwnerDrawState);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
```

```
procedure TForm1.FormActivate(Sender: TObject);
begin
  listbox1.items.LoadFromFile('c:\autoexec.bat');
end;

procedure TForm1.ListBox1DrawItem(Control: TWinControl; Index: Integer;
  Rect: TRect; State: TOwnerDrawState);
begin
  WITH Control AS TListBox, Canvas DO BEGIN
    FillRect(Rect);
    MoveTo(Rect.Left+2, Rect.Top);
    SetTextAlign(Canvas.Handle, TA_LEFT OR TA_UPDATECP);
    Font.Color := clGreen;
    TextOut(0, 0, Items[Index]);
  END;
end;

end.
```

TSpeedButton

How do I paste a bitmap or glyph on a speedbutton without losing the background color?
How can I write a function that generically responds to all Speedbutton clicks?
Is there an easy way to assign speedbutton properties via iteration at runtime?

TListBox

How can I get a horizontal scrollbar on a list box?
How do you fill a listbox with multiple lines?
How do I make a TListBox with displayable tabs?
How do I have lines in a listbox as different colors?
How do I fill a listbox from a memo field in a table?
How do I do click and drag in a TListbox?

Tabbed Notebook

How do I add items to a tabbed notebook at runtime?
How do I iterate through tabbed notebook pages to see each object?
How do I fake TTabbedNotebook with multiple forms?
How do I draw on the pages of a TTabbedNotebook?

Q: I get an "Error creating cursor handle" message. Here is my SQL statement:

```
update "parts.db"
set "parts.db"."Retail Price" =
(("parts.db"."Retail Price" * :percentage) + "parts.db"."Retail Price")
where "parts.db"."Part Number" like :pre
and "parts.db"."Retail Price" < :high
and "parts.db"."Retail Price" > :low
```

A: Update queries do not return any information. They just update. You used `open` or `active := true` when you called the SQL statement. That expects a cursor to be returned. Use `ExecSQL` instead.

ODBC

How do I update Access records?

Q: How do I update Access records?

A: The ODBC driver provided with Access 2.0 (ODBCJT16.DLL with a file size of <65k) is designed to work only within the Microsoft Office environment. To work with ODBC/Access in Delphi, you need the Microsoft ODBC Desktop Driver (ODBCJT16.DLL, internal version number 02.00.23.17, with a file size of approx 260k) kit, part# 273-054-030 available from Microsoft Direct for $10.25US (post on WINEXT for where to get it in your country if you are not in the US). It is also available on the Jan. MSDN, Level 2 (Development Platform) CD4 \ODBC\X86 as part of the ODBC 2.1 SDK. Be aware that your redistribution rights for the Desktop Drivers are pretty restricted by Microsoft. For info on (and objections to) the restrictions post on the WINEXT forum.

You also need the following ODBC files.

Minimum:
ODBC.DLL 03.10.1994, Version 2.00.1510
ODBCINST.DLL 03.10.1994, Version 2.00.1510
ODBCINST.HLP 11.08.1993
ODBCADM.EXE 11.08.1993, Version 1.02.3129

Better:
ODBC.DLL 12.07.1994, Version 2.10.2401
ODBCINST.DLL 12.07.1994, Version 2.10.2401
ODBCINST.HLP 12.07.1994
ODBCADM.EXE 12.07.1994, Version 2.10.2309

The following steps will get you started in Delphi

1. Using the ODBC Administrator, set-up a datasource for your database. Be sure to specify a path to your mdb file. For the purposes of this explanation we'll say that the datasource name is MYDSN. The correct driver name is:

Microsoft Access Driver (*.mdb)

2. Load the BDE Configuration utility.

3. Select New Driver.

4. Give the driver a name (call it ODBC_MYDSN).

Q: Crystal Reports does not let the user select a printer at run-time. How do I provide this functionality.

1) Include CRPE and PRINTERS in 'uses' (CRPE.pas is packaged with Crystal Reports)

2) Add a TPrinterSetupDialog obect to the form (to allow the user to select a different printer)

3) Create printer variables:

```
Var
  JobHandle : Integer
  ADevice, ADriver, APort : Array[0..30] of Char;
  ADeviceMode : THandle;
```

4) Use engine calls to run the report. Issue the following functions before issuing a PEStartPrintJob:

```
{ Open print job }
PEOpenEngine()
JobHandle = PEOpenPrintJob(<filename>)
.
.
.
{ Change printer }
Printer.GetPrinter(ADevice, ADriver, APort, ADeviceMode);
PESelectPrinter(JobHandle, ADriver, ADevice, Aport,
  PDevMode(PTR(ADeviceMode,0))^);
.
.
.
{ Start print job }
PEStartPrintJob(JobHandle, True);
PECloseEngine();
```

Q: How do I fill a listbox from a table?

A: Use a memo field as an intermediate step.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  memo1.lines.assign(table1.fieldByName('notes'));
  listbox1.items.assign(memo1.lines);
end;
```

You can also use a `table1.fieldByName('notes').GetData(buffer)`, but that is extra work. This is easier.

Q: How to I create a Paradox table with an Auto Increment type field programatically? I'm using TTable.CreateTable, but TFieldType doesn't include this type.

A: Use a TQuery and SQL CREATE TABLE statement. For example:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  with Query1 do
  begin
    DatabaseName := 'DBDemos';
    with SQL do
    begin
      Clear;
      Add('CREATE TABLE "PDoxTbl.db" (ID AUTOINC,');
      Add('Name CHAR(255),');
      Add('PRIMARY KEY(ID))');
      ExecSQL;
      Clear;
      Add('CREATE INDEX ByName ON "PDoxTbl.db" (Name)');
      ExecSQL;
    end;
  end;
end;
```

Q: How can I find out what all of the available fonts are?

A: You can use EnumFontFamilies, a Win API function. That uses a callback that is handled each font in turn.

The following sample uses an object method as the callback.

```
unit Fontenum;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    LstFamilies: TListBox;
    Label1: TLabel;
    Label2: TLabel;
    MemVariations: TMemo;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure LstFamiliesDblClick(Sender: TObject);
  private
    { Private declarations }
    Function DoEnum( Var lf: TEnumLogFont; Var tm: TNewTextMetric;
                     fonttype: Integer ): Integer; export;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

Function TForm1.DoEnum( Var lf: TEnumLogFont; Var tm: TNewTextMetric;
                        fonttype: Integer ): Integer;
Begin
  Result := 1;
  If (fonttype and TRUETYPE_FONTTYPE) <> 0 Then
    Memvariations.Lines.Add( StrPas( lf.elfFullName ))
  Else
    MemVariations.Lines.Add( StrPas( lf.elfLogFont.lfFacename ));
End;

procedure TForm1.Button1Click(Sender: TObject);
begin
  Application.terminate;
end;
```

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  LstFamilies.Items.Assign( Screen.Fonts );
end;

procedure TForm1.LstFamiliesDblClick(Sender: TObject);
Var
  familyname: Array [0..40] of Char;
begin
  MemVariations.Clear;
  With Sender As TListbox Do
    If ItemIndex >=0 Then Begin
       StrPCopy( familyname, Items[ItemIndex] );
       EnumFontFamilies( Canvas.Handle, familyname,
                         @TForm1.DoEnum, Pointer(Self));
    End;
end;

end.

-------- form file for this unit ---------
object Form1: TForm1
  Left = 310
  Top = 229
  Width = 497
  Height = 300
  Caption = 'EnumFontFamilies Demo'
  Font.Color = clWindowText
  Font.Height = -17
  Font.Name = 'System'
  Font.Style = []
  PixelsPerInch = 120
  OnCreate = FormCreate
  TextHeight = 20
  object Label1: TLabel
    Left = 16
    Top = 8
    Width = 105
    Height = 20
    Caption = '&Font Families'
    FocusControl = LstFamilies
  end
  object Label2: TLabel
    Left = 184
    Top = 8
    Width = 81
    Height = 20
    Caption = 'Variations'
  end
  object LstFamilies: TListBox
    Left = 16
    Top = 32
    Width = 153
    Height = 217
    ItemHeight = 20
    TabOrder = 0
    OnDblClick = LstFamiliesDblClick
```

```
  end
  object MemVariations: TMemo
    Left = 184
    Top = 32
    Width = 185
    Height = 217
    Lines.Strings = (
      '')
    ReadOnly = True
    TabOrder = 1
  end
  object Button1: TButton
    Left = 384
    Top = 32
    Width = 89
    Height = 33
    Caption = 'Close'
    TabOrder = 2
    OnClick = Button1Click
  end
end
```

Q: How do you tell which record and which field of a TDBGrid is current?

A: Here is a method to keep track of the current column and row. The following code in the method MyDBGridDrawDataCell updates the variables Col and Row (which must not be local to the method) every time the grid is redrawn. Using this code you can assume that Col and Row point to the current column and row respectively.

```
var
  Col, Row: Integer;

procedure TForm1.MyDBGridDrawDataCell(Sender: TObject; const Rect: TRect;
Field: TField; State: TGridDrawState);
var
  RowHeight: Integer;
begin
  if gdFocused in State then
  begin
    RowHeight := Rect.Bottom - Rect.Top;
    Row := (Rect.Top div RowHeight) - 1;
    Col := Field.Index;
  end;
end;
```

Q: How do I change the color of a grid cell in a TDBGrid?

A: Enter the following code in the TDBGrid's OnDrawDataCell event:

```
Procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
Field: TField; State: TGridDrawState);
begin
  If gdFocused in State then
  with (Sender as TDBGrid).Canvas do
  begin
    Brush.Color := clRed;
    FillRect(Rect);
    TextOut(Rect.Left, Rect.Top, Field.AsString);
  end;
end;
```

Set the Default drawing to true. With this, it only has to draw the highlighted cell. If you set DefaultDrawing to false, you must draw all the cells yourself with the canvas properties.

Q: How can I view dBASE records marked for deletion?

A: Call the following function on the AfterOpen event of the table. You must include DBITYPES, DBIERRS, DBIPROCS in the uses clause. To call, send as arguments name of TTable and TRUE/FALSE depending to show/not show deleted records.

Example:

```
procedure TForm1.Table1AfterOpen(DataSet: TDataset);
begin
  SetDelete(Table1, TRUE);
end;

procedure SetDelete(oTable:TTable; Value: Boolean);
var
  rslt: DBIResult;
  szErrMsg: DBIMSG;
begin
  try
    oTable.DisableControls;
    try
      rslt := DbiSetProp(hDBIObj(oTable.Handle), curSOFTDELETEON,
      LongInt(Value));
      if rslt <> DBIERR_NONE then
      begin
        DbiGetErrorString(rslt, szErrMsg);
        raise Exception.Create(StrPas(szErrMsg));
      end;
    except
      on E: EDBEngineError do ShowMessage(E.Message);
      on E: Exception do ShowMessage(E.Message);
    end;
  finally
    oTable.Refresh;
    oTable.EnableControls;
  end;
end;
```

Here is a custom TStringGrid that will allow for inserting a whole row at a time.

```
unit Gridmv;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Grids;

type

  TGridMV = class(TStringGrid)
  public
    procedure RowMoved(FromIndex, ToIndex: Longint); override;
    procedure Insert(ToIndex: Longint);
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Samples', [TGridMV]);
end;

procedure TGridMv.RowMoved(FromIndex, ToIndex: Longint);
begin
  inherited RowMoved(FromIndex, ToIndex);
end;

{This will insert a row before the row selected.}
procedure TGridMv.Insert(ToIndex: Longint);
begin
  RowCount := RowCount + 1;
  RowMoved(RowCount, ToIndex);
end;

end.
```

Q: "How can I determine the current record number for a dataset?"

A: If the dataset is based upon a Paradox or dBASE table then the record number can be determined with a couple of calls to the BDE (as shown below). The BDE doesn't support record numbering for datasets based upon SQL tables, so if your server supports record numbering you will need to refer to its documentation.

The following function takes as its parameter any component derived from TDataset (i.e. TTable, TQuery, TStoredProc) and returns the current record number (greater than zero) if it is a Paradox or dBASE table. Otherwise, the function returns zero.

NOTE: for dBASE tables the record number returned is always the physical record number. So, if your dataset is a TQuery or you have a range set on your dataset then the number returned won't necessarily be relative to the dataset being viewed, rather it will be based on the record's physical position in the underlying dBASE table.

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, DBCtrls, Grids, DBGrids, DB, DBTables,
  DbiProcs, DbiTypes, DbiErrs;

type
  TForm1 = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    DBGrid1: TDBGrid;
    DBNavigator1: TDBNavigator;
    function RecordNumber(Dataset: TDataset): Longint;
    procedure DataSource1DataChange(Sender: TObject; Field: TField);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

function TForm1.RecordNumber(Dataset: TDataset): Longint;
var
  CursorProps: CurProps;
  RecordProps: RECProps;

begin
```

```
  { Return 0 if dataset is not Paradox or dBASE }
  Result := 0;

  with Dataset do
  begin
    { Is the dataset active? }
    if State = dsInactive then
      ShowMessage('Cannot perform this operation on a closed dataset');

    { We need to make this call to grab the cursor's iSeqNums }
    Check(DbiGetCursorProps(Handle, CursorProps));

    { Synchronize the BDE cursor with the Dataset's cursor }
    UpdateCursorPos;

    { Fill RecordProps with the current record's properties }
    Check(DbiGetRecord(Handle, dbiNOLOCK, nil, @RecordProps));

    { What kind of dataset are we looking at? }
    case CursorProps.iSeqNums of
      0: Result := RecordProps.iPhyRecNum;  { dBASE   }
      1: Result := RecordProps.iSeqNum;     { Paradox }
    end;
  end;
end;

procedure TForm1.DataSource1DataChange(Sender: TObject; Field: TField);
begin
  if (Sender as TDataSource).State = dsBrowse then
    caption := 'Record number = ' +
IntToStr(RecordNumber(Datasource1.Dataset));
end;

end.
```

Here are some math functions that do not come with Delphi that may come in handy:

```
unit Math;

interface

function exponent(BaseNumber, ToThePowerOf: real): real;
function tan(TheVal: real): real;
function ArcSin(TheVal: real): real;
function ArcCos(TheVal: real): real;
function IntToBinaryStr(TheVal: LongInt): string;

implementation

function exponent(BaseNumber, ToThePowerOf: real): real;
begin
  result := Exp(Ln(BaseNumber) * ToThePowerOf);
end;

function tan(TheVal: real): real;
begin
  result := sin(TheVal) / cos(TheVal);
end;

function ArcSin(TheVal: real): real;
begin
  result := ArcTan(TheVal / sqrt(1 - sqrt(TheVal)));
end;

function ArcCos(TheVal: real): real;
begin
  result := ArcTan(sqrt(1 - sqrt(TheVal)) / TheVal);
end;

function IntToBinaryStr(TheVal: LongInt): string;
var
  counter: LongInt;
begin
{This part is here because we remove leading zeros.  That
means that a zero value would return an empty string.}
  if TheVal = 0 then begin
    result := '0';
    exit;
  end;

  result := '';
  counter := $80000000;

  {Suppress leading zeros}
  while  ((counter and TheVal) = 0) do begin
    counter := counter shr 1;
    if (counter = 0) then break; {We found our first "1".}
  end;

  while counter > 0 do begin
    if (counter and TheVal) = 0 then result := result + '0'
```

```
    else  result := result + '1';
    counter := counter shr 1;
  end;
end;

end.
```

Q: "How can I determine when the current record in a dataset has changed?"

A: Check the DataSource's State property in the OnDataChanged event. The State property will be set to dsBrowse if the record position has changed. The following example will display a message box every time the record position has changed in MyDataSource:

```
procedure TMyForm.MyDataSourceDataChange(Sender: TObject; Field: TField);
begin
  if (Sender as TDataSource).State = dsBrowse then
    ShowMessage('Record Position Changed');
end;
```

The compuserve version: OnCreate, OnShow, OnPaint, OnActivate, OnResize and OnPaint again.

Roland's version:

Create
Show - Visible
ReSize
Activate - Visible
Paint - Visible
Close query
Close
Deactivate
Hide
Destroy

Q: How do perform something on the form's OnActivate method? (It doesn't always fire. e.g. if you ALT-TAB to it, then OnActivate doesn't fire.)

A:

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  application.OnActivate := FormActivate;
end;
```

Q: How can I get rid of the ReportSmith about box splash screen when I run a report.

A: Add the following line in [RS_RunTime] section of the RS_RUN.INI file to not have the ReportSmith about box appear when a report is ran.

```
ShowAboutBox=0
```

Q: Is it possible to access components by their name property (i.e. 'SpeedButton' + IntToStr(i))

A: Yes it's possible. The following example uses the FindComponent method of Form1 to disable the first 10 SpeedButtons by name.

```
for I := 1 to 10 do
  with Form1.FindComponent('SpeedButton' + IntToStr(i)) as TSpeedButton do
    Enabled := False;
```

Here is an example that prints columns that are right, left, and center justified. There are headers, footers, and, generally, a bunch o' things here. This app encapsulates functionality to print text, lines, boxes and shaded boxes. Text can be left or right justified and centered. Columns can be created and text can be left or right justified within the columns or text can be centered. Lines of any thickness can be drawn. Boxes can be drawn with any thickness. The boxes can be shaded if desired. Headers and footers can be created and the header/footer areas can be shaded if desired. Page numbering can contain custom text and can be placed anywhere desired.

```
{******* prnMain.pas *******}

unit Prnmain;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, ExtCtrls, Printers;

const
  HeaderLines = 5;                        { Number of allowable header
lines }
  FooterLines = 5;                        { Number of allowable footer
lines }
  Columns = 20;                           { Number of allowable columns }

type
  THeaderRecord = Record
     Text: String[240];                   { Header text }
     YPosition: Single;                   { Inches from the top }
     Alignment: Integer;                  { 0:Left 1:Center 2:Right }
     FontName: String[80];                { Font name }
     FontSize: Integer;                   { Font size }
     FontStyle: TFontStyles;              { Font style }
     End;

  TFooterRecord = Record
     Text: String[240];                   { Footer text }
     YPosition: Single;                   { Inches from the top }
     Alignment: Integer;                  { 0:Left 1:Center 2:Right }
     FontName: String[80];                { Font name }
     FontSize: Integer;                   { Font size }
     FontStyle: TFontStyles;              { Font style }
     End;

  THeaderCoordinates = Record
     XTop: Single;
     YTop: Single;
     XBottom: Single;
     YBottom: Single;
     Boxed: Boolean;
     Shading: Word;
     LineWidth: Word;
```

```
     End;

  TFooterCoordinates = Record
     XTop: Single;
     YTop: Single;
     XBottom: Single;
     YBottom: Single;
     Boxed: Boolean;
     Shading: Word;
     LineWidth: Word;
     End;

  TPageNumberRecord = Record
     YPosition: Single;
     Text: String[240];
     Alignment: Word;
     FontName: String[80];
     FontSize: Word;
     FontStyle: TFontStyles;
     End;

  TColumnInformationRecord = Record
     XPosition: Single;
     Length: Single;
     End;

  TPrintObject = class
     private
        TopMargin: Integer;               { Top margin in pixels }
        BottomMargin: Integer;            { Bottom margin in pixels }
        LeftMargin: Integer;              { Left margin in pixels }
        RightMargin: Integer;             { Right margin in pixels }
        PixelsPerInchVertical: Integer;   { Number of pixels per inch along Y
axis }
        PixelsPerInchHorizontal: Integer; { Number of pixels per inch along X
axis }
        TotalPageWidthPixels: Integer;    { Full width of page in pixels -
includes gutters }
        TotalPageHeightPixels: Integer;   { Full height of page in pixels -
includes gutters }
        TotalPageHeightInches: Single;    { Height of page in inches }
        TotalPageWidthInches: Single;     { Width of page in inches }
        GutterLeft: Integer;              { Unprintable area on left }
        GutterRight: Integer;             { Unprintable area on right }
        GutterTop: Integer;               { Unprintable area on top }
        GutterBottom: Integer;            { Unprintable area on bottom }
        DetailTop: Single;                { Inches from the top where the
detail section starts }
        DetailBottom: Single;             { Inches from the top where the
detail section ends }
        LastYPosition: Single;            { The Y position where the last
write occurred }
        AutoPaging: Boolean;              { Are new pages automatically
generated? }
        CurrentTab: Single;               { The value of the current tab }
        CurrentFontName: String[30];
        CurrentFontSize: Integer;
```

```
        CurrentFontStyle: TFontStyles;
        TextMetrics: TTextMetric;
        Header: Array[1..HeaderLines] of THeaderRecord;
        Footer: Array[1..FooterLines] of TFooterRecord;
        ColumnInformation: Array[1..Columns] of TColumnInformationRecord;
        PageNumber: TPageNumberRecord;
        HeaderCoordinates: THeaderCoordinates;
        FooterCoordinates: TFooterCoordinates;
        function CalculateLineHeight: Integer;
        function InchesToPixelsHorizontal( Inches: Single ): Integer;
        function InchesToPixelsVertical( Inches: Single ): Integer;
        function PixelsToInchesHorizontal( Pixels: Integer ): Single;
        function PixelsToInchesVertical( Pixels: Integer ): Single;
        function LinesToPixels( Line:Integer ): Integer;
        procedure CalculateMeasurements;
        procedure _DrawBox( XTop:Word; YTop:Word; XBottom:Word; YBottom:Word;
LineWidth:Word; Shading:Word );
     public
        procedure Start;
        procedure Quit;
        procedure Abort;
        procedure SetMargins( Top:Single; Bottom:Single; Left:Single;
Right:Single );
        procedure SetFontInformation( Name:String; Size:Word; Style:
TFontStyles );
        procedure WriteLine( X:Single; Y:Single; Text:String );
        procedure WriteLineRight( Y:Single; Text:String );
        procedure WriteLineCenter( Y:Single; Text:String );
        procedure WriteLineColumnRight( ColumnNumber:Word; Y:Single;
Text:String );
        procedure WriteLineColumnCenter( ColumnNumber:Word; Y:Single;
Text:String );
        procedure DrawLine( TopX:Single; TopY:Single; BottomX:Single;
BottomY:Single; LineWidth:Word );
        procedure SetLineWidth( Width:Word );
        function  GetLineWidth: Word;
        procedure SetTab( Inches:Single );
        procedure NewPage;
        function  GetLinesPerPage: Integer;
        procedure GetPixelsPerInch( var X:Word; var Y:Word );
        procedure GetPixelsPerPage( var X:Word; var Y:Word );
        procedure GetGutter( var Top:Word; var Bottom:Word; var Left:Word;
var Right:Word );
        function  GetTextWidth( Text:String ): Integer;
        function  GetLineHeightPixels: Word;
        function  GetLineHeightInches: Single;
        function  GetPageNumber:Integer;
        function  GetColumnsPerLine: Integer;
        procedure SetOrientation( Orient: TPrinterOrientation );
        procedure SetHeaderInformation( Line:Integer; YPosition: Single;
Text:String; Alignment:Word;
                    FontName:String; FontSize: Word; FontStyle:
TFontStyles );
        procedure SetFooterInformation( Line:Integer; YPosition: Single;
Text:String; Alignment:Word;
                    FontName:String; FontSize: Word; FontStyle:
TFontStyles );
```

```
        procedure WriteHeader;
        procedure WriteFooter;
        procedure SaveCurrentFont;
        procedure RestoreCurrentFont;
        procedure SetDetailTopBottom( Top: Single; Bottom: Single );
        procedure SetAutoPaging( Value: Boolean );
        procedure SetPageNumberInformation( YPosition:Single; Text:String;
Alignment:Word; FontName:String;
                     FontSize:Word; FontStyle:TFontStyles );
        procedure WritePageNumber;
        procedure WriteLineColumn( ColumnNumber:Word; Y:Single;
Text:String );
        procedure DrawBox( XTop:Single; YTop:Single; XBottom:Single;
YBottom:Single; LineWidth:Word );
        procedure DrawBoxShaded( XTop:Single; YTop:Single; XBottom:Single;
YBottom:Single; LineWidth:Word; Shading:Word );
        procedure SetHeaderDimensions( XTop:Single; YTop:Single;
XBottom:Single; YBottom:Single;
                  Boxed: Boolean; LineWidth:Word; Shading:Word );
        procedure SetFooterDimensions( XTop:Single; YTop:Single;
XBottom:Single; YBottom:Single;
                  Boxed: Boolean; LineWidth:Word; Shading:Word );
        procedure CreateColumn( Number:Word; XPosition:Single;
Length:Single );
        procedure SetYPosition( YPosition:Single );
        function  GetYPosition: Single;
        procedure NextLine;
        function  GetLinesLeft: Word;
        function  GetLinesInDetailArea: Word;
        procedure SetTopOfPage;
        procedure NewLines( Number:Word );
        function GetFontName: String;
        function GetFontSize: Word;
   End;

implementation

procedure TPrintObject.Start;

   { This function MUST be called first before any other printing function }

   var
      Top,Bottom,Left,Right: Single;
      I: Integer;

   Begin
   Printer.BeginDoc;

   AutoPaging := True;

   CalculateMeasurements;

   PageNumber.Text := '';

   Top := PixelsToInchesVertical( GutterTop );
   Bottom := PixelsToInchesVertical( GutterBottom );
   Left := PixelsToInchesHorizontal( GutterLeft );
```

```
   Right := PixelsToInchesHorizontal( GutterRight );
   SetMargins( Top,Bottom,Left,Right );

   For I := 1 To HeaderLines Do
      Header[I].Text := '';
   HeaderCoordinates.Boxed := False;
   HeaderCoordinates.Shading := 0;
   For I := 1 To FooterLines Do
      Footer[I].Text := '';
   FooterCoordinates.Boxed := False;
   FooterCoordinates.Shading := 0;

   CurrentTab := 0.0;

   LastYPosition := 0.0;
   End;

procedure TPrintObject.Quit;

   { 'Quit' must always be called when printing is completed }

   Begin
   WriteHeader;
   WriteFooter;
   WritePageNumber;

   Printer.EndDoc
   End;

procedure TPrintObject.SetMargins( Top:Single; Bottom:Single; Left:Single;
Right:Single );

   { Set the top, bottom, left and right margins in inches }

   var
      Value: Single;
      Buffer: String;

   Begin
   { If the sum of the left and right margins exceeds the width of the page,
     set the left margin to the value of 'GutterLeft' and set the right
     margin to the value of 'GutterRight' }
   If ( Left + Right >= TotalPageWidthInches ) Then
      Begin
      Left := GutterLeft;
      Right := GutterRight;
      End;
   If ( Left <= 0 ) Then
      Left := GutterLeft;
   If ( Right <= 0 ) Then
      Right := GutterRight;

   { If the sum of the top and bottom margins exceeds the height of the
     page, set the top margin to the value of 'GutterTop' and set the
     bottom margin to the value of 'GutterBottom' }
   If ( Top + Bottom >= TotalPageHeightInches ) Then
      Begin
```

```
      Top := GutterTop;
      Bottom := GutterBottom;
      End;
   If ( Top <= 0 ) Then
      Top := GutterTop;
   If ( Bottom <= 0 ) Then
      Bottom := GutterBottom;

   { Convert everything to pixels }
   TopMargin := InchesToPixelsVertical( Top );
   If ( TopMargin < GutterTop ) Then
      TopMargin := GutterTop;

   BottomMargin := InchesToPixelsVertical( Bottom );
   If ( BottomMargin < GutterBottom ) Then
      BottomMargin := GutterBottom;

   LeftMargin := InchesToPixelsHorizontal( Left );
   If ( LeftMargin < GutterLeft ) Then
      LeftMargin := GutterLeft;

   RightMargin := InchesToPixelsHorizontal( Right );
   If ( RightMargin < GutterRight ) Then
      RightMargin := GutterRight;
   End;

procedure TPrintObject.WriteLine( X:Single; Y:Single; Text:String );

   { Write some text.  The parameters represent inches from the left ('X')
     and top ('Y') margins. }

   var
      XPixels: Integer;
      YPixels: Integer;

   Begin
   { How many pixels are there in the inches represented by 'X'? }
   If ( X >= 0.0 ) Then
      XPixels := InchesToPixelsHorizontal( X )
   Else
      XPixels := LeftMargin;
   If ( XPixels < GutterLeft ) Then
      XPixels := GutterLeft;

   { If there is a tab set, increase 'XPixels' by the amount of the tab }
   If ( CurrentTab > 0.0 ) Then
      Inc( XPixels,InchesToPixelsHorizontal(CurrentTab) );

   { How many pixels are there in the inches represented by 'Y'? }
   If ( Y > -0.01 ) Then
      { Printing will occur at an absolute location from the top of the
        page. }
      Begin
      YPixels := InchesToPixelsVertical( Y );
      If ( YPixels < GutterTop ) Then
         YPixels := GutterTop;
      If ( YPixels > TotalPageHeightPixels ) Then
```

```
         YPixels := TotalPageHeightPixels - GutterBottom;

      LastYPosition := Y;
      End;
   If ( Y = -1.0 ) Then
      { Write the text at the next line }
      Begin
      If ( AutoPaging = True ) Then
         Begin
         { If the next line we're going to write to exceeds beyond the
           bottom of the detail section, issue a new page }
         If ( LastYPosition + GetLineHeightInches > DetailBottom ) Then
            NewPage;
         End;
      YPixels := InchesToPixelsVertical( LastYPosition +
GetLineHeightInches );
      LastYPosition := LastYPosition + GetLineHeightInches;
      End;
   If ( Y = -2.0 ) Then
      { Write the text on the current line }
      YPixels := InchesToPixelsVertical( LastYPosition );

   Printer.Canvas.TextOut( XPixels-GutterLeft,YPixels-GutterTop,Text );
   End;

procedure TPrintObject.WriteLineColumn( ColumnNumber:Word; Y:Single;
Text:String );

   { Write text, left aligned against the column represented by
     'ColumnInformation[ColumnNumber]' }

   Begin
   WriteLine( ColumnInformation[ColumnNumber].XPosition,Y,Text );
   End;

procedure TPrintObject.WriteLineColumnRight( ColumnNumber:Word; Y:Single;
Text:String );

   { Write text, right aligned against the column represented by
     'ColumnInformation[ColumnNumber]' }

   var
      PixelLength: Word;
      StartPixel: Word;

   Begin
   { How many pixels does the text in 'Text' require? }
   PixelLength := Printer.Canvas.TextWidth( Text );

   { Calculate where printing should start }
   StartPixel :=
InchesToPixelsHorizontal( ColumnInformation[ColumnNumber].XPosition +
      ColumnInformation[ColumnNumber].Length ) - PixelLength;

   SetTab( 0.0 );
   WriteLine( PixelsToInchesHorizontal(StartPixel),Y,Text );
   SetTab( CurrentTab );
```

```
   End;

procedure TPrintObject.WriteLineRight( Y:Single; Text:String );

   { Print a line of text right justified 'Y' inches from the top }

   var
      PixelLength: Word;
      StartPixel: Word;

   Begin
   { How many pixels does the text in 'Text' require? }
   PixelLength := Printer.Canvas.TextWidth( Text );

   { Calculate where printing should start }
   StartPixel := (TotalPageWidthPixels-GutterLeft-GutterRight) - PixelLength;

   SetTab( 0.0 );
   WriteLine( PixelsToInchesHorizontal(StartPixel),Y,Text );
   SetTab( CurrentTab );
   End;

procedure TPrintObject.WriteLineCenter( Y:Single; Text:String );

   { Print a line of text centered at Y inches from the top }

   var
      PixelLength: Integer;
      StartPixel: Integer;

   Begin
   { How many pixels does the text in 'Text' require? }
   PixelLength := Printer.Canvas.TextWidth( Text );

   { Calculate where printing should start }
   StartPixel := ((GutterLeft+(TotalPageWidthPixels-GutterRight)) Div 2) -
(PixelLength Div 2);

   SetTab( 0.0 );
   WriteLine( PixelsToInchesHorizontal(StartPixel),Y,Text );
   SetTab( CurrentTab );
   End;

procedure TPrintObject.WriteLineColumnCenter( ColumnNumber:Word; Y:Single;
Text:String );

   { Print a line of text centered within the column number represented by
     'ColumnNumber', at Y inches from the top }

   var
      PixelLength: Integer;
      StartPixel: Integer;
      Pixels: Integer;

   Begin
   { How many pixels does the text in 'Text' require? }
   PixelLength := Printer.Canvas.TextWidth( Text );
```

```
   { Calculate where printing should start }
   Pixels := InchesToPixelsHorizontal( ColumnInformation[ColumnNumber].Length
);
   StartPixel :=
(InchesToPixelsHorizontal( ColumnInformation[ColumnNumber].Length ) Div 2) +
      InchesToPixelsHorizontal(ColumnInformation[ColumnNumber].XPosition) -
(PixelLength Div 2);

   SetTab( 0.0 );
   WriteLine( PixelsToInchesHorizontal(StartPixel),Y,Text );
   SetTab( CurrentTab );
   End;

procedure TPrintObject.DrawLine( TopX:Single; TopY:Single; BottomX:Single;
BottomY:Single; LineWidth:Word );

   { Draw a line beginning at a particular X,Y coordinate and ending at a
     particular X,Y coordinate. }

   var
      TopXPixels, BottomXPixels, TopYPixels, BottomYPixels: Integer;

   Begin
   TopXPixels := InchesToPixelsHorizontal( TopX );
   BottomXPixels := InchesToPixelsHorizontal( BottomX );
   TopYPixels := InchesToPixelsVertical( TopY );
   BottomYPixels := InchesToPixelsVertical( BottomY );

   Dec( TopXPixels,GutterLeft );
   Dec( BottomXPixels,GutterLeft );
   Dec( TopYPixels,GutterTop );
   Dec( BottomYPixels,GutterTop );

   Printer.Canvas.Pen.Width := LineWidth;

   Printer.Canvas.MoveTo( TopXPixels,TopYPixels );
   Printer.Canvas.LineTo( BottomXPixels,BottomYPixels );
   End;

procedure TPrintObject.SetFontInformation( Name:String; Size:Word; Style:
TFontStyles );

   { Change the current font information }

   Begin
   Printer.Canvas.Font.Name := Name;
   Printer.Canvas.Font.Size := Size;
   Printer.Canvas.Font.Style := Style;

   CalculateMeasurements;
   End;

function TPrintObject.GetFontName: String;

   { Return the current font name }
```

```
   Begin
   Result := Printer.Canvas.Font.Name;
   End;

function TPrintObject.GetFontSize: Word;

   { Return the current font size }

   Begin
   Result := Printer.Canvas.Font.Size;
   End;

procedure TPrintObject.SetOrientation( Orient: TPrinterOrientation );

   Begin
   Printer.Orientation := Orient;

   CalculateMeasurements;
   End;

function TPrintObject.CalculateLineHeight: Integer;

   { Calculate the height of a line plus the normal amount of space between
     each line }

   Begin
   Result := TextMetrics.tmHeight + TextMetrics.tmExternalLeading;
   End;

procedure TPrintObject.NewPage;

   { Issue a new page }

   Begin
   WriteHeader;
   WriteFooter;
   WritePageNumber;
   LastYPosition := DetailTop - GetLineHeightInches;

   Printer.NewPage;
   End;

function TPrintObject.GetPageNumber;

   { Return the current page number }

   Begin
   Result := Printer.PageNumber;
   End;

function TPrintObject.GetTextWidth( Text:String ): Integer;

   { Return the width of the text contained in 'Text' in pixels }

   Begin
   Result := Printer.Canvas.TextWidth( Text );
   End;
```

```
function TPrintObject.GetLineHeightPixels: Word;

   Begin
   Result := CalculateLineHeight;
   End;

function TPrintObject.GetLineHeightInches: Single;

   Begin
   Result := PixelsToInchesVertical( GetLineHeightPixels );
   End;

procedure TPrintObject._DrawBox( XTop:Word; YTop:Word; XBottom:Word;
YBottom:Word; LineWidth:Word; Shading:Word );

   { The low level routine which actually draws the box and shades it as
     desired. The paramaters are in pixels and not inches. }

   Begin
   Printer.Canvas.Pen.Width := LineWidth;
   Printer.Canvas.Brush.Color := RGB( Shading,Shading,Shading );

   Printer.Canvas.Rectangle( XTop,YTop,XBottom,YBottom );
   End;

procedure TPrintObject.DrawBox( XTop:Single; YTop:Single; XBottom:Single;
YBottom:Single; LineWidth:Word );

   { Draw a box at the X,Y coordinates passed in the parameters }

   var
      BLinePixels,BColPixels,ELinePixels,EColPixels: Integer;

   Begin
   BLinePixels := InchesToPixelsVertical( YTop ) - GutterTop;
   ELinePixels := InchesToPixelsVertical( YBottom ) - GutterTop;

   BColPixels := InchesToPixelsHorizontal( XTop ) - GutterLeft;
   EColPixels := InchesToPixelsHorizontal( XBottom ) - GutterLeft;

   _DrawBox( BColPixels,BLinePixels,EColPixels,ELinePixels,LineWidth,255 );
   End;

procedure TPrintObject.DrawBoxShaded( XTop:Single; YTop:Single;
XBottom:Single; YBottom:Single; LineWidth:Word; Shading:Word );

   { Draw a box at the X,Y coordinates passed in the parameters }

   var
      BLinePixels,BColPixels,ELinePixels,EColPixels: Integer;

   Begin
   BLinePixels := InchesToPixelsVertical( YTop ) - GutterTop;
   ELinePixels := InchesToPixelsVertical( YBottom ) - GutterTop;

   BColPixels := InchesToPixelsHorizontal( XTop ) - GutterLeft;
```

```
   EColPixels := InchesToPixelsHorizontal( XBottom ) - GutterLeft;


_DrawBox( BColPixels,BLinePixels,EColPixels,ELinePixels,LineWidth,Shading );
   End;

function TPrintObject.GetLinesPerPage: Integer;

   { Return the number of lines on the entire page }

   Begin
   Result := (TotalPageHeightPixels - GutterTop - GutterBottom) Div
CalculateLineHeight;
   End;

function TPrintObject.GetLinesInDetailArea: Word;

   { Return the number of lines in the detail area }

   Begin
   Result := InchesToPixelsVertical( DetailBottom - DetailTop ) Div
CalculateLineHeight;
   End;

procedure TPrintObject.GetPixelsPerInch( var X:Word; var Y:Word );

   Begin
   X := PixelsPerInchHorizontal;
   Y := PixelsPerInchVertical;
   End;

procedure TPrintObject.GetPixelsPerPage( var X:Word; var Y:Word );

   Begin
   X := TotalPageWidthPixels - GutterLeft - GutterRight;
   Y := TotalPageHeightPixels - GutterTop - GutterBottom;
   End;

procedure TPrintObject.GetGutter( var Top:Word; var Bottom:Word; var
Left:Word; var Right:Word );

   Begin
   Top := GutterTop;
   Bottom := GutterBottom;
   Left := GutterLeft;
   Right := GutterRight;
   End;

procedure TPrintObject.Abort;

   Begin
   Printer.Abort;
   End;

function TPrintObject.GetColumnsPerLine: Integer;

   { How many columns are there in a Line? }
```

```
   var
      Pixels: Integer;

   Begin
   Pixels := TotalPageWidthPixels - GutterLeft - GutterRight;

   Result := Pixels Div Printer.Canvas.TextWidth( 'B' );
   End;

function TPrintObject.InchesToPixelsHorizontal( Inches: Single ): Integer;

   { Convert the horizontal inches represented in 'Inches' to pixels }

   var
      Value: Single;
      Buffer: String;
      I: Integer;

   Begin
   Value := Inches * PixelsPerInchHorizontal;
   Buffer := FloatToStr( Value );

   { If there is a decimal point in 'Buffer', remove it. }
   I := 1;
   While( (Buffer[I] <> '.') And (I <= Length(Buffer)) ) Do
      Inc( I );
   Buffer[0] := Chr( I-1 );

   Result := StrToInt( Buffer );
   End;

function TPrintObject.InchesToPixelsVertical( Inches: Single ): Integer;

   { Convert the vertical inches represented in 'Inches' to pixels }

   var
      Value: Single;
      Buffer: String;
      I: Integer;

   Begin
   Value := Inches * PixelsPerInchVertical;
   Buffer := FloatToStr( Value );

      { If there is a decimal point in 'Buffer', remove it. }
   I := 1;
   While( (Buffer[I] <> '.') And (I <= Length(Buffer)) ) Do
      Inc( I );
   Buffer[0] := Chr( I-1 );

   Result := StrToInt( Buffer );
   End;

function TPrintObject.PixelsToInchesHorizontal( Pixels: Integer ): Single;

   Begin
```

```
   Result := Pixels / PixelsPerInchHorizontal;
   End;

function TPrintObject.PixelsToInchesVertical( Pixels: Integer ): Single;

   Begin
   Result := Pixels / PixelsPerInchVertical;
   End;

function TPrintObject.LinesToPixels( Line:Integer ): Integer;

   { Calculate the number of vertical pixels in 'Line' }

   Begin
   If ( Line <= 0 ) Then
      Line := 1;

   Result := (Line-1) * CalculateLineHeight;
   End;

procedure TPrintObject.SetLineWidth( Width:Word );

   Begin
   Printer.Canvas.Pen.Width := Width;
   End;

function TPrintObject.GetLineWidth: Word;

   Begin
   Result := Printer.Canvas.Pen.Width;
   End;

procedure TPrintObject.CalculateMeasurements;

   { Calculate some necessary measurements.  Thanks to Robert Fabiszak
     CompuServe: 70304,2047 for the Escape() Windows API calls. }

   var
      pt: TPoint;

   Begin
   { Call the Windows API function GetTextMetrics() to get the specifics
     of the particular font. }
   GetTextMetrics( Printer.Canvas.Handle,TextMetrics );

   { Calculate the number of pixels per inch vertical and horizontal.
     'GetDeviceCaps' is a Windows API call. }
   PixelsPerInchVertical := GetDeviceCaps( Printer.Handle,LOGPIXELSY );
   PixelsPerInchHorizontal := GetDeviceCaps( Printer.Handle,LOGPIXELSX );

   { Get the gutter on the left and top.  'Escape' is a Windows API
     call. }
   Escape( Printer.Canvas.Handle,GETPRINTINGOFFSET,0,Nil,@pt );
   GutterLeft := pt.X;
   GutterTop := pt.Y;

   Escape( Printer.Canvas.Handle,GETPHYSPAGESIZE,0,Nil,@pt );
```

```
   TotalPageWidthPixels := pt.X;
   TotalPageHeightPixels := pt.Y;
   TotalPageWidthInches := pt.X / PixelsPerInchHorizontal;
   TotalPageHeightInches := pt.Y / PixelsPerInchVertical;

   GutterRight := TotalPageWidthPixels - GutterLeft - Printer.PageWidth;
   GutterBottom := TotalPageHeightPixels - GutterTop - Printer.PageHeight;

   If ( TopMargin < GutterTop ) Then
      TopMargin := GutterTop;
   If ( BottomMargin < GutterBottom ) Then
      BottomMargin := GutterBottom;
   If ( LeftMargin < GutterLeft ) Then
      LeftMargin := GutterLeft;
   If ( RightMargin < GutterRight ) Then
      RightMargin := GutterRight;
   End;

procedure TPrintObject.SetHeaderInformation( Line:Integer; YPosition: Single;
Text:String; Alignment:Word;
   FontName:String; FontSize: Word; FontStyle: TFontStyles );

   Begin
   If ( Line > HeaderLines ) Then
      Exit;

   Header[Line].Text := Text;
   Header[Line].YPosition := YPosition;
   Header[Line].Alignment := Alignment;
   Header[Line].FontName := FontName;
   Header[Line].FontSize := FontSize;
   Header[Line].FontStyle := FontStyle;
   End;

procedure TPrintObject.SetFooterInformation( Line:Integer; YPosition: Single;
Text:String; Alignment:Word;
   FontName:String; FontSize: Word; FontStyle: TFontStyles );

   Begin
   If ( Line > FooterLines ) Then
      Exit;

   Footer[Line].Text := Text;
   Footer[Line].YPosition := YPosition;
   Footer[Line].Alignment := Alignment;
   Footer[Line].FontName := FontName;
   Footer[Line].FontSize := FontSize;
   Footer[Line].FontStyle := FontStyle;
   End;

procedure TPrintObject.WriteHeader;

   { If any headers are defined, write them }

   var
      I: Integer;
```

```
   Begin
   SaveCurrentFont;
   For I := 1 To HeaderLines Do
      Begin
      If ( Length(Header[I].Text) > 0 ) Then
         Begin
         With Header[I] Do
            Begin
            SetFontInformation( FontName,FontSize,FontStyle );
            If ( Alignment = 0 ) Then
               WriteLine( LeftMargin, YPosition, Text );
            If ( Alignment = 1 ) Then
               WriteLineCenter( YPosition, Text );
            If ( Alignment = 2 ) Then
               WriteLineRight( YPosition, Text );
            End;
         End;

      RestoreCurrentFont;
      End;

   { Does the user desire a box around the header? }
   If ( HeaderCoordinates.Boxed = True ) Then
      Begin
      If ( HeaderCoordinates.Shading > 0 ) Then

DrawBoxShaded( HeaderCoordinates.XTop,HeaderCoordinates.YTop,HeaderCoordinate
s.XBottom,

HeaderCoordinates.YBottom,HeaderCoordinates.LineWidth,HeaderCoordinates.Shadi
ng )
      Else

DrawBox( HeaderCoordinates.XTop,HeaderCoordinates.YTop,HeaderCoordinates.XBot
tom,
            HeaderCoordinates.YBottom,HeaderCoordinates.LineWidth );
      End;
   End;

procedure TPrintObject.WriteFooter;

   { If any footers are defined, write them }

   var
      I: Integer;
      Temp: Boolean;

   Begin
   SaveCurrentFont;

   { Set 'AutoPaging' off.  Otherwise the footer will not get written
     correctly. }
   Temp := AutoPaging;
   AutoPaging := False;

   For I := 1 To FooterLines Do
      Begin
```

```
      If ( Length(Footer[I].Text) > 0 ) Then
         Begin
         With Footer[I] Do
            Begin
            SetFontInformation( FontName,FontSize,FontStyle );
            If ( Alignment = 0 ) Then
               WriteLine( LeftMargin, YPosition, Text );
            If ( Alignment = 1 ) Then
               WriteLineCenter( YPosition, Text );
            If ( Alignment = 2 ) Then
               WriteLineRight( YPosition, Text );
            End;
         End;

      RestoreCurrentFont;
      End;

   { Does the user desire a box around the footer? }
   If ( FooterCoordinates.Boxed = True ) Then
      Begin
      If ( FooterCoordinates.Shading > 0 ) Then

DrawBoxShaded( FooterCoordinates.XTop,FooterCoordinates.YTop,FooterCoordinate
s.XBottom,

FooterCoordinates.YBottom,FooterCoordinates.LineWidth,FooterCoordinates.Shadi
ng )
      Else

DrawBox( FooterCoordinates.XTop,FooterCoordinates.YTop,FooterCoordinates.XBot
tom,
            FooterCoordinates.YBottom,FooterCoordinates.LineWidth );
      End;

   AutoPaging := Temp;
   End;

procedure TPrintObject.SaveCurrentFont;

   Begin
   CurrentFontName := Printer.Canvas.Font.Name;
   CurrentFontSize := Printer.Canvas.Font.Size;
   CurrentFontStyle := Printer.Canvas.Font.Style;
   End;

procedure TPrintObject.RestoreCurrentFont;

   Begin
   SetFontInformation( CurrentFontName,CurrentFontSize,CurrentFontStyle );
   End;

procedure TPrintObject.SetDetailTopBottom( Top: Single; Bottom: Single );

   Begin
   DetailTop := Top;
   DetailBottom := Bottom;
```

```
   LastYPosition := Top - GetLineHeightInches;
   End;

procedure TPrintObject.SetAutoPaging( Value: Boolean );

   Begin
   AutoPaging := Value;
   End;

procedure TPrintObject.SetPageNumberInformation( YPosition:Single;
Text:String; Alignment:Word; FontName:String;
   FontSize:Word; FontStyle:TFontStyles );

   Begin
   PageNumber.Text := Text;
   PageNumber.YPosition := YPosition;
   PageNumber.Alignment := Alignment;
   PageNumber.FontName := FontName;
   PageNumber.FontSize := FontSize;
   PageNumber.FontStyle := FontStyle;
   End;

procedure TPrintObject.WritePageNumber;

   var
      Buffer: String;
      Temp: Boolean;

   Begin
   Buffer := Format( PageNumber.Text,[Printer.PageNumber] );

   SaveCurrentFont;

SetFontInformation( PageNumber.FontName,PageNumber.FontSize,PageNumber.FontSt
yle );

   Temp := AutoPaging;
   AutoPaging := False;

   If ( PageNumber.Alignment = 0 ) Then
      WriteLine( LeftMargin, PageNumber.YPosition, Buffer );
   If ( PageNumber.Alignment = 1 ) Then
      WriteLineCenter( PageNumber.YPosition, Buffer );
   If ( PageNumber.Alignment = 2 ) Then
      WriteLineRight( PageNumber.YPosition, Buffer );

   AutoPaging := Temp;

   RestoreCurrentFont;
   End;

procedure TPrintObject.SetTab( Inches:Single );

   Begin
   CurrentTab := Inches;
   End;
```

```
procedure TPrintObject.SetHeaderDimensions( XTop:Single; YTop:Single;
XBottom:Single; YBottom:Single;
   Boxed: Boolean; LineWidth:Word; Shading:Word );

   Begin
   HeaderCoordinates.XTop := XTop;
   HeaderCoordinates.XBottom := XBottom;
   HeaderCoordinates.YTop := YTop;
   HeaderCoordinates.YBottom := YBottom;
   HeaderCoordinates.Boxed := Boxed;
   HeaderCoordinates.LineWidth := LineWidth;
   HeaderCoordinates.Shading := Shading;
   End;

procedure TPrintObject.SetFooterDimensions( XTop:Single; YTop:Single;
XBottom:Single; YBottom:Single;
   Boxed: Boolean; LineWidth:Word; Shading:Word );

   Begin
   FooterCoordinates.XTop := XTop;
   FooterCoordinates.XBottom := XBottom;
   FooterCoordinates.YTop := YTop;
   FooterCoordinates.YBottom := YBottom;
   FooterCoordinates.Boxed := Boxed;
   FooterCoordinates.LineWidth := LineWidth;
   FooterCoordinates.Shading := Shading;
   End;

procedure TPrintObject.CreateColumn( Number:Word; XPosition:Single;
Length:Single );

   Begin
   ColumnInformation[Number].XPosition := XPosition;
   ColumnInformation[Number].Length := Length;
   End;

procedure TPrintObject.SetYPosition( YPosition:Single );

   Begin
   LastYPosition := YPosition;
   End;

function TPrintObject.GetYPosition: Single;

   Begin
   Result := LastYPosition;
   End;

procedure TPrintObject.NextLine;

   Begin
   LastYPosition := LastYPosition + GetLineHeightInches;
   End;

function TPrintObject.GetLinesLeft: Word;

   { Return the number of lines left in the detail area }
```

```
   var
      Lines: Single;
      Buffer: String[20];
      I: Word;

   Begin
   Lines := (DetailBottom - LastYPosition) / GetLineHeightInches;
   Buffer := FloatToStr( Lines );

   { Buffer contains the number of lines left as a floating point number.
     Find the decimal and truncate the string at that point.  So, if there
     are 2.99 lines left, 2 will be returned.  Better to be conservative. }
   For I := 1 To Length(Buffer) Do
      Begin
      If ( Buffer[I] = '.' ) Then
         Begin
         Buffer[0] := Chr(I-1);
         Break;
         End;
      End;

   Result := StrToInt( Buffer );
   End;

procedure TPrintObject.SetTopOfPage;

   Begin
   LastYPosition := DetailTop;
   End;

procedure TPrintObject.NewLines( Number:Word );

   { Generate the number of line feeds represented in 'Number' }

   var
      I: Word;

   Begin
   For I := 1 To Number Do
      NextLine;
   End;

end.
```

{******* demo.pas *******}

```
unit Demo;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, StdCtrls, Mask, DBCtrls, Menus, PrnMain;

const
```

```
  LeftMargin = 0.5;
  RightMargin = 0.5;
  TopMargin = 0.5;
  BottomMargin = 0.5;

type
  TPrintForm = class(TForm)
   Button1: TButton;
   Button2: TButton;
   PixelsPerInch: TPanel;
   PixelsPerPage: TPanel;
   Gutters: TPanel;
   LineHeight: TPanel;
   FontInformation: TPanel;
   LinesInDetailArea: TPanel;
   procedure Button1Click(Sender: TObject);
   procedure FormCreate(Sender: TObject);
   procedure Button2Click(Sender: TObject);
  private
   { Private declarations }
  public
   { Public declarations }
  end;

var
  PrintForm: TPrintForm;
  Prn: TPrintObject;

implementation

{$R *.DFM}

procedure TPrintForm.Button1Click(Sender: TObject);

  var
     Buffer: String;
     Code: String[10];
     ECHOCode: String[10];
     HeaderLine: Boolean;
     I: Word;

  Begin
  { Define the dimensions of the header area.  I want the header area
    lightly shaded.  If I wanted no shading, the last parameter would be
    255. }
  with prn do
  begin
    SetHeaderDimensions( 0.25,0.25,8.25,1.25,True,0,225 );

    { Define two header lines }
    SetHeaderInformation( 1,0.5,'This is header line number 1',1,'Arial',14,
[fsBold] );
    SetHeaderInformation( 2,1.0,DateToStr(Date),1,'Arial',11,[] );

    { Define the dimensions of the footer area.  I want the footer area
      lightly shaded.  If I wanted no shading, the last parameter would be
      255. }
```

```
    SetFooterDimensions( 0.25,9.40,8.25,10.20,True,0,225 );

    { Define two footer lines }
    SetFooterInformation( 1,9.5,'This is footer line number 1',1,'Arial',14,
[fsBold] );
    SetFooterInformation( 2,9.85,'This is footer line number 2',1,'Arial',12,
[fsBold] );

    { I would like page numbering, right justified on the very bottom of the
      page. }
    SetPageNumberInformation( 10.25,'Page: %d',2,'Arial',9,[fsBold] );

    { Set the current position to the top of the detail area }
    SetTopOfPage;

    { Write three lines, the first left justified, the second centered and
      the third right justified.  The first line gets printed two inches
      from the top.  The next two lines get printed at the next line from
      the previous line. The '-1' for the first parameter indicates that
      printing should be on the next line.  If '-2' is passed as a
      parameter, printing would occur on the current line. }
    WriteLine( -1.0,2.0,'This is a line left justified' );
    WriteLineCenter( -1.0,'This is a line centered' );
    WriteLineRight( -1.0,'This is a line right justified' );

    { Create five columns.  The first parameter is the column number, the
      second parameter is the location in inches from the left and the third
      parameter is the length in inches. }
    CreateColumn( 1,0.25,1.5 );
    CreateColumn( 2,1.80,1.5 );
    CreateColumn( 3,3.35,1.5 );
    CreateColumn( 4,4.90,1.5 );
    CreateColumn( 5,6.50,1.5 );

    { Start writing column text (left justified) at three inches from the
      top }
    SetYPosition( 3.0 );
    For I := 1 To 10 Do
       Begin
       { The first parameter of 'WriteLineColumn' is the column number and
         the second parameter indicates that printing should occur on the
         current line (in this case, three inches from the top).  If the
         second parameter was -1, printing would occur on the next line. }
       WriteLineColumn( 1,-2,Format('Column 1, Line %d',[I]) );
       WriteLineColumn( 2,-2,Format('Column 2, Line %d',[I]) );
       WriteLineColumn( 3,-2,Format('Column 3, Line %d',[I]) );
       WriteLineColumn( 4,-2,Format('Column 4, Line %d',[I]) );
       WriteLineColumn( 5,-2,Format('Column 5, Line %d',[I]) );
       { Generate a line feed }
       NextLine;
       End;

    { Start writing column text (right justified) at six inches from the
      top }
    SetYPosition( 5.0 );
    For I := 1 To 10 Do
       Begin
```

```
      WriteLineColumnRight( 1,-2,Format('Column 1, Line %d',[I]) );
      WriteLineColumnRight( 2,-2,Format('Column 2, Line %d',[I]) );
      WriteLineColumnRight( 3,-2,Format('Column 3, Line %d',[I]) );
      WriteLineColumnRight( 4,-2,Format('Column 4, Line %d',[I]) );
      WriteLineColumnRight( 5,-2,Format('Column 5, Line %d',[I]) );
      NextLine;
      End;

   { Start writing column text (centered) at seven inches from the
     top }
   SetYPosition( 7.0 );
   For I := 1 To 10 Do
      Begin
      WriteLineColumnCenter( 1,-2,Format('Column 1, Line %d',[I]) );
      WriteLineColumnCenter( 2,-2,Format('Column 2, Line %d',[I]) );
      WriteLineColumnCenter( 3,-2,Format('Column 3, Line %d',[I]) );
      WriteLineColumnCenter( 4,-2,Format('Column 4, Line %d',[I]) );
      WriteLineColumnCenter( 5,-2,Format('Column 5, Line %d',[I]) );
      NextLine;
      End;

   { Start a new page }
   NewPage;

   { Change the font information }
   SetFontInformation( 'Courier',20,[fsBold,fsUnderline] );

   For I := 1 To 10 Do
      WriteLine( LeftMargin,-1,Format('This is line %d',[I]) );

   { Set a tab of .5 inches }
   SetTab( 0.5 );

   { Change the font information }
   SetFontInformation( 'Arial',10,[fsItalic] );
   NextLine;
   For I := 1 To 10 Do
      { Since a tab of .5 is set, this text will actually get printed at
        1.0 inches from the left }
      WriteLine( LeftMargin,-1,Format('This is line %d',[I]) );

   { Draw some lines of varying thickness }
   DrawLine( 2.5,5.0,6.0,8.5,5 );
   DrawLine( 6.2,5.2,3.0,8.7,20 );

   { We're all done.  Always call 'Quit' }
   Quit;
   Free;
   Exit;
  end;
End;

procedure TPrintForm.FormCreate(Sender: TObject);
var
  X,Y: Word;
  Top,Bottom,Left,Right: Word;
```

```
Begin
    { Create a TPrintObject }
    Prn := TPrintObject.Create;
      with prn do
    begin

    { Must always call 'Start' first thing }
    Start;

    { Set left, right, top and bottom margins - in inches }
    SetMargins( LeftMargin,RightMargin,TopMargin,BottomMargin );

    { Define what the 'detail' section dimensions will be.  The detail
section
      is the space between the header and the footer areas. }
    SetDetailTopBottom( 1.4,9.4 );

    { Set default information }
    SetFontInformation( 'Arial',11,[] );

    GetPixelsPerInch( X,Y );
    PixelsPerInch.Caption := Format( 'Pixels Per Inch      X: %d  Y: %d',
[X,Y] );

    GetPixelsPerPage( X,Y );
    PixelsPerPage.Caption := Format( 'Pixels Per Page      X: %d  Y: %d',
[X,Y] );

    GetGutter( Top,Bottom,Left,Right );
    Gutters.Caption := Format( 'Gutters     Top: %d   Bottom: %d   Left: %d
Right: %d',[Top,Bottom,Left,Right] );

    LineHeight.Caption := Format( 'Height of Each Line:   %d',
[GetLineHeightPixels] );

    FontInformation.Caption := Format( 'Font Name: %s     Font Size: %d',
[GetFontName,GetFontSize] );

    LinesInDetailArea.Caption := Format( 'Lines in Detail Area: %d',
[GetLinesInDetailArea] );
      end; {with}
End;

procedure TPrintForm.Button2Click(Sender: TObject);

  Begin
  Close;
  Halt;
  End;

end.
```

{******* project.dpr *******}

```
program Project;
```

```
uses
  Forms,
  Prnmain in 'PRNMAIN.PAS',
  Demo in 'DEMO.PAS' {PrintForm};

{$R *.RES}

begin
  Application.CreateForm(TPrintForm, PrintForm);
  Application.Run;
end.
```

{******* demo.dfm *******}

```
object PrintForm: TPrintForm
  Left = 104
  Top = 90
  BorderIcons = [biSystemMenu]
  BorderStyle = bsDialog
  Caption = 'Print Demonstration'
  ClientHeight = 317
  ClientWidth = 427
  Color = clSilver
  Font.Color = clWindowText
  Font.Height = -13
  Font.Name = 'System'
  Font.Style = []
  PixelsPerInch = 96
  Position = poScreenCenter
  OnCreate = FormCreate
  TextHeight = 16
  object Button1: TButton
    Left = 276
    Top = 270
    Width = 61
    Height = 33
    Caption = '&Print'
    TabOrder = 0
    OnClick = Button1Click
  end
  object Button2: TButton
    Left = 342
    Top = 270
    Width = 61
    Height = 33
    Cancel = True
    Caption = '&Cancel'
    ModalResult = 2
    TabOrder = 1
    OnClick = Button2Click
  end
  object PixelsPerInch: TPanel
    Left = 6
    Top = 12
```

```
    Width = 415
    Height = 25
    TabOrder = 2
  end
  object PixelsPerPage: TPanel
    Left = 6
    Top = 42
    Width = 415
    Height = 25
    TabOrder = 3
  end
  object Gutters: TPanel
    Left = 6
    Top = 72
    Width = 415
    Height = 25
    TabOrder = 4
  end
  object LineHeight: TPanel
    Left = 6
    Top = 102
    Width = 415
    Height = 25
    TabOrder = 5
  end
  object FontInformation: TPanel
    Left = 6
    Top = 132
    Width = 415
    Height = 25
    TabOrder = 6
  end
  object LinesInDetailArea: TPanel
    Left = 6
    Top = 162
    Width = 415
    Height = 25
    TabOrder = 7
  end
end
```

String manipulation and parsing is often a place where we wish that we had more functions and methods to do our work for us. There are many existing methods that cover our work-a-day needs, but every now and then we need something more.

In Paradox for Windows, there is an immensely useful function called BreakApart(). It can do so many things that I have written one (or two) for Delphi. Here is how it works: You pass the base string that you want to parse and the string that you want to use for the BreakApart(). The function then separates the string into several pieces and fills a TStringList with their values.

Example:

Base String: "Here we go."
Break String: " "

Resulting array:
TStringList[0]: "Here"
TStringList[1]: "we"
TStringList[2]: "go."

Note that the break string, in this case spaces, is not to be seen anywhere in the output.

Now that we know what is done, lets take a look at how to do it.

First, lets look at the string version.

```
function sBreakApart(BaseString, BreakString: string; StringList:
TStringList): TStringList;
var
  EndOfCurrentString: byte;
  TempStr: string;
begin
  repeat
    EndOfCurrentString := Pos(BreakString, BaseString);
    if EndOfCurrentString = 0 then
      StringList.add(BaseString)
    else
      StringList.add(Copy(BaseString, 1, EndOfCurrentString - 1));
    BaseString := Copy(BaseString, EndOfCurrentString + length(BreakString),
length(BaseString) - EndOfCurrentString);
  until EndOfCurrentString = 0;
  result := StringList;
end;
```

This is a fairly straight forward version (as compared with the PChar version). We look for the break string. If we don't find it at all, then we just assign it to the TStringList and go on our merry way. (Note: The TStringList must be created outside the function or there will be problems when the variable goes out of scope.) If we do find it, then we want to extract that portion of the string and assign it to the next available position in the

TStringList. Note that the string is updated each step of the way. This is done just because it simplifies the code. There are several possible ways of doing this. I just picked one.

Let me answer some questions that I hear you asking.

1. Why didn't I make this a procedure? It is true that the TStringList would reflect all changes without a value returned, but I left it as a function because it allows for use in another function that uses the TStringList. I use the function in this way later in the article.

E.g. Listbox1.items.assign(sBreakApart(...));

2. It doesn't interfere with calling it as if it was a procedure. (I.e. You don't need to catch the result if you don't want to code it that way.)

Now that we have seen that it does something, how about if we have it do something useful.

Here is a search and replace that uses the string version of the BreakApart() function.

```
function ReplaceStr(BaseString, ReplaceThis, WithThis: string): string;
var
  t: TStringList;
  i: integer;
begin
  t := TStringList.create;
  sBreakApart(BaseString, ReplaceThis, t);
  if t.count > 1 then
  begin
    result := '';
    for i := 0 to t.count - 2 do
      result := result + t[i] + WithThis;
    result := result + t[i + 1];
  end
  else result := BaseString;
  t.free;
end;
```

This example requires a form with a pushbutton and 3 edit boxes. You can call this function like this:

```
edit1.text := ReplaceStr(edit1.text, edit2.text, edit3.text);
```

It replaces all occurrences of edit2.text in edit1.text with edit3.text.

I told you that this was simple and useful!

Now lets take a look at the PChar version. Since we have an idea of how this works now, I'll show the code first.

```
function pBreakApart(BaseString, BreakString: PChar; StringList:
TStringList): TStringList;
var
  BreakStringLength: word;
  pEndOfCurrentString, pEndOfBaseString: PChar;
{Automatically gets memory allocated for it.}
  temp: array[0..255] of char;
begin
{Initialize the pointers.}
  BreakStringLength := StrLen(BreakString);
  pEndOfBaseString := BaseString;
  inc(pEndOfBaseString, StrLen(BaseString));
  repeat
    pEndOfCurrentString := StrPos(BaseString, BreakString);
    StringList.add(StrPas(StrLCopy(temp, BaseString, pEndOfCurrentString -
BaseString)));
    inc(BaseString, pEndOfCurrentString - BaseString + BreakStringLength);
  until BaseString >= pEndOfBaseString - BreakStringLength;
  result := StringList;
end;
```

This takes a different approach to solving the same problem. Since this is done with a PChar that can be of greatly varying size, it was best to do it with pointers and pointer arithmetic. Since we are not changing the value passed in (which would be bad as it is passed by reference and that would change the values in their original memory locations) we need a way to keep track of just where we are in the current part of the process.

A word about pointer arithmetic... The Inc() and Dec() functions have an undocumented feature that allows for pointer arithmetic. The relevant feature is that the function "knows" the size of the object pointed to and increments the pointer the correct number of bytes.

Here is an example that uses the PChar version:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  f: file;
  pStr: PChar;
  LengthOfFile: integer;
  t: TStringList;
begin
  {Get the information.}
  AssignFile(f, 'c:\autoexec.bat');
  {Because this is not a text file type, the record size is 1 (char)}
  Reset(f, 1);
  LengthOfFile := FileSize(f) + 1; {Add one for the null terminator.}
  pStr := AllocMem(LengthOfFile); {Zeros the memory also.}
  BlockRead(f, pStr^, LengthOfFile - 1);
  CloseFile(f);
  t := TStringList.create;
  listBox1.items.assign(pBreakApart(pStr, #13#10, t));
```

```
  t.free;
  FreeMem(pStr, LengthOfFile);
end;
```

This example requires a form that has a listbox and a pushbutton. It reads the autoexec.bat file into memory in a single gulp using Blockread(). There is just enough memory allocated to get the job done. This is done by using the file's size as the basis. The PChar version is called and assigned directly, and the file is "broken apart" by carriage return/line feed. (Note: I know that a LoadFromFile() will do this also, but this is an exercise in memory juggling.) Then memory clean up is performed. The contents of the autoexec.bat file are then displayed in the listbox line by line.

The uses for this are many and varied. If you used this on a filename with the full path and did a BreakApart() on "\", you would have element 0 of the list as the drive, and the last element would be the file name. You could break that apart on the ".", and get the separated file name and extension.

I did not include error checking here. A string can only hold 255 characters. It is possible that your users might try to put more than that in there. If you want to do the error checking for them, then I wish you well. I thought that it would be beyond the scope of this short article and left it to the reader.

Q: How do I do a search and replace?

A: This uses the sBreakApart() function.

```
function ReplaceStr(BaseString, ReplaceThis, WithThis: string): string;
var
  t: TStringList;
  i: integer;
begin
  t := TStringList.create;
  sBreakApart(BaseString, ReplaceThis, t);
  if t.count > 1 then
  begin
    result := '';
    for i := 0 to t.count - 2 do
      result := result + t[i] + WithThis;
    result := result + t[i + 1];
  end
  else result := BaseString;
  t.free;
end;
```

Q: How do I surface the MouseDown Event on a TDBGrid?

A: Here is a component that will do it:

```
unit Lloyd;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls, Forms,
Dialogs, Grids, DBGrids;

type
  TMouseGrid = class(TDBGrid)
  private
    { Private declarations }
    FOnMouseDown: TMouseEvent;
  protected
    { Protected declarations }
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer); override;
  public
    { Public declarations }
  published
    { Published declarations }
    property OnMouseDown read FOnMouseDown write FOnMouseDown;
  end;

procedure Register;

implementation

procedure TMouseGrid.MouseDown(Button: TMouseButton; Shift: TShiftState;
  X, Y: Integer);
begin
  if Assigned(FOnMouseDown) then FOnMouseDown(Self, Button, Shift, X, Y);
  inherited MouseDown(Button, Shift, X, Y);
end;

procedure Register;
begin
  RegisterComponents('Custom', [TMouseGrid]);
end;

end.
```

BDE error codes and descriptions:

The code listed at the end of this topic creates ERRLIST.TXT containing a list of error codes, along with the descriptions returned by DbiGetErrorString. This is essentially a dump; it tries to get every possible error code. Most return null strings and are filtered out. Some return valid strings that aren't real error messages.

```
     0  0000  Successful completion.
    33  0021  System Error
    34  0022  Object of Interest Not Found
    35  0023  Physical Data Corruption
    36  0024  I/O Related Error
    37  0025  Resource or Limit Error
    38  0026  Data Integrity Violation
    39  0027  Invalid Request
    40  0028  Lock Violation
    41  0029  Access/Security Violation
    42  002A  Invalid Context
    43  002B  OS Error
    44  002C  Network Error
    45  002D  Optional Parameter
    46  002E  Query Processor
    47  002F  Version Mismatch
    48  0030  Capability Not Supported
    49  0031  System Configuration Error
    50  0032  Warning
    51  0033  Miscellaneous
    52  0034  Compatibility Error
    62  003E  Driver Specific Error
    63  003F  Internal Symbol
   256  0100  KEYVIOL
   257  0101  PROBLEMS
   258  0102  CHANGED
   512  0200  Production Index file missing, corrupt or cannot interpret
index key
   513  0201  Open Read Only
   514  0202  Open the table in read only mode
   515  0203  Open and Detach
   516  0204  Open the table and detach the Production Index file
   517  0205  Fail Open
   518  0206  Do not open the table
   519  0207  Convert Non-dBase Index
   520  0208  Convert production index to dBase format
   521  0209  BLOB file not found
   522  020A  Open without blob file
   523  020B  Open the table without the blob file
   524  020C  Empty all blob fields
   525  020D  Reinitialize BLOB file and LOSE all blobs
   526  020E  Fail Open
   527  020F  Do not open the table
   528  0210  Import Non-dBASE BLOB file
   529  0211  Import BLOB file to dBASE format
   530  0212  Open as Non-dBASE table
   531  0213  Open Table and BLOB file in its native format
```

532	0214	Production Index Language driver mismatch
533	0215	Production Index damaged
534	0216	Rebuild Production Index
535	0217	Rebuild all the Production Indexes
1024	0400	Lookup table not found or corrupt
1025	0401	Blob file not found or corrupt
1026	0402	Open Read Only
1027	0403	Open the table in read only mode
1028	0404	Fail Open
1029	0405	Do not open the table
1030	0406	Remove lookup
1031	0407	Remove link to lookup table
2048	0800	Reading records
2049	0801	Sorting records
2050	0802	Writing records
2051	0803	Merging
2052	0804	Steps Completed
2053	0805	Packing records
2309	0905	LIKE
2310	0906	NOT
2320	0910	INSERT
2321	0911	DELETE
2322	0912	CHANGETO
2323	0913	CHANGE
2324	0914	TO
2325	0915	FIND
2326	0916	CALC
2327	0917	COUNT
2328	0918	SUM
2329	0919	AVERAGE
2330	091A	MAX
2331	091B	MIN
2332	091C	ALL
2333	091D	UNIQUE
2334	091E	BLANK
2335	091F	TODAY
2336	0920	AS
2337	0921	DESCENDING
2338	0922	OR
2339	0923	ONLY
2340	0924	EVERY
2341	0925	NO
2342	0926	EXACTLY
2343	0927	SET
2347	092B	%time
2348	092C	%date
2353	0931	%lower
2354	0932	%upper
2355	0933	%trim
2356	0934	%substring
2364	093C	__QB0000
2365	093D	ANSWER
2366	093E	DELETED
2367	093F	INSERTED
2368	0940	CHANGED
2369	0941	ERRORDEL
2370	0942	ERRORINS

2371	0943	ERRORCHG
2372	0944	__XLTTMP
2373	0945	__QBEDIC
2405	0965	JAN
2406	0966	FEB
2407	0967	MAR
2408	0968	APR
2409	0969	MAY
2410	096A	JUN
2411	096B	JUL
2412	096C	AUG
2413	096D	SEP
2414	096E	OCT
2415	096F	NOV
2416	0970	DEC
2423	0977	INSERTED.DB
2424	0978	CHANGED.DB
2425	0979	DELETED.DB
2426	097A	ANSWER.DB
2427	097B	blank
2428	097C	Sum of
2429	097D	Average of
2430	097E	Count of
2431	097F	Max of
2432	0980	Min of
2433	0981	GROUPBY
2434	0982	FIELDORDER
2435	0983	SORT
2436	0984	ANSWER
2437	0985	TYPE
2438	0986	OPTIONS
2440	0988	GENERATE AUXILIARY TABLES
2441	0989	NO AUXILIARY TABLES
2442	098A	SERVER
2443	098B	LOCAL
2444	098C	CANNED
2445	098D	LIVE
2446	098E	SPEED
2447	098F	%extract
2448	0990	DATE
2449	0991	TIME
2450	0992	YEAR
2451	0993	MONTH
2452	0994	DAY
2453	0995	HOUR
2454	0996	MINUTE
2455	0997	SECOND
8449	2101	Cannot open a system file.
8450	2102	I/O error on a system file.
8451	2103	Data structure corruption.
8452	2104	Cannot find Engine configuration file.
8453	2105	Cannot write to Engine configuration file.
8454	2106	Cannot initialize with different configuration file.
8455	2107	System has been illegally re-entered.
8456	2108	Cannot locate IDAPI01.DLL
8457	2109	Cannot load IDAPI01.DLL
8458	210A	Cannot load an IDAPI service library

8705	2201	At beginning of table.
8706	2202	At end of table.
8707	2203	Record moved because key value changed.
8708	2204	Record/Key deleted.
8709	2205	No current record.
8710	2206	Could not find record.
8711	2207	End of BLOB.
8712	2208	Could not find object.
8713	2209	Could not find family member.
8714	220A	BLOB file is missing.
8715	220B	Could not find language driver.
8961	2301	Corrupt table/index header.
8962	2302	Corrupt file - other than header.
8963	2303	Corrupt Memo/BLOB file.
8965	2305	Corrupt index.
8966	2306	Corrupt lock file.
8967	2307	Corrupt family file.
8968	2308	Corrupt or missing .VAL file.
8969	2309	Foreign index file format.
9217	2401	Read failure.
9218	2402	Write failure.
9219	2403	Cannot access directory.
9220	2404	File Delete operation failed.
9221	2405	Cannot access file.
9222	2406	Access to table disabled because of previous error.
9473	2501	Insufficient memory for this operation.
9474	2502	Not enough file handles.
9475	2503	Insufficient disk space.
9476	2504	Temporary table resource limit.
9477	2505	Record size is too big for table.
9478	2506	Too many open cursors.
9479	2507	Table is full.
9480	2508	Too many sessions from this workstation.
9481	2509	Serial number limit (Paradox).
9482	250A	Some internal limit (see context).
9483	250B	Too many open tables.
9484	250C	Too many cursors per table.
9485	250D	Too many record locks on table.
9486	250E	Too many clients.
9487	250F	Too many indexes on table.
9488	2510	Too many sessions.
9489	2511	Too many open databases.
9490	2512	Too many passwords.
9491	2513	Too many active drivers.
9492	2514	Too many fields in Table Create.
9493	2515	Too many table locks.
9494	2516	Too many open BLOBs.
9495	2517	Lock file has grown too large.
9496	2518	Too many open queries.
9498	251A	Too many BLOBs.
9729	2601	Key violation.
9730	2602	Minimum validity check failed.
9731	2603	Maximum validity check failed.
9732	2604	Field value required.
9733	2605	Master record missing.
9734	2606	Master has detail records. Cannot delete or modify.
9735	2607	Master table level is incorrect.

9736	2608	Field value out of lookup table range.
9737	2609	Lookup Table Open operation failed.
9738	260A	Detail Table Open operation failed.
9739	260B	Master Table Open operation failed.
9740	260C	Field is blank.
9741	260D	Link to master table already defined.
9742	260E	Master table is open.
9743	260F	Detail table(s) exist.
9744	2610	Master has detail records. Cannot empty it.
9745	2611	Self referencing referential integrity must be entered one at a time with no other changes to the table
9746	2612	Detail table is open.
9747	2613	Cannot make this master a detail of another table if its details are not empty.
9748	2614	Referential integrity fields must be indexed.
9749	2615	A table linked by referential integrity requires password to open.
9750	2616	Field(s) linked to more than one master.
9985	2701	Number is out of range.
9986	2702	Invalid parameter.
9987	2703	Invalid file name.
9988	2704	File does not exist.
9989	2705	Invalid option.
9990	2706	Invalid handle to the function.
9991	2707	Unknown table type.
9992	2708	Cannot open file.
9993	2709	Cannot redefine primary key.
9994	270A	Cannot change this RINTDesc.
9995	270B	Foreign and primary key do not match.
9996	270C	Invalid modify request.
9997	270D	Index does not exist.
9998	270E	Invalid offset into the BLOB.
9999	270F	Invalid descriptor number.
10000	2710	Invalid field type.
10001	2711	Invalid field descriptor.
10002	2712	Invalid field transformation.
10003	2713	Invalid record structure.
10004	2714	Invalid descriptor.
10005	2715	Invalid array of index descriptors.
10006	2716	Invalid array of validity check descriptors.
10007	2717	Invalid array of referential integrity descriptors.
10008	2718	Invalid ordering of tables during restructure.
10009	2719	Name not unique in this context.
10010	271A	Index name required.
10011	271B	Invalid session handle.
10012	271C	invalid restructure operation.
10013	271D	Driver not known to system.
10014	271E	Unknown database.
10015	271F	Invalid password given.
10016	2720	No callback function.
10017	2721	Invalid callback buffer length.
10018	2722	Invalid directory.
10019	2723	Translate Error. Value out of bounds.
10020	2724	Cannot set cursor of one table to another.
10021	2725	Bookmarks do not match table.
10022	2726	Invalid index/tag name.
10023	2727	Invalid index descriptor.

10024	2728	Table does not exist.
10025	2729	Table has too many users.
10026	272A	Cannot evaluate Key or Key does not pass filter condition.
10027	272B	Index already exists.
10028	272C	Index is open.
10029	272D	Invalid BLOB length.
10030	272E	Invalid BLOB handle in record buffer.
10031	272F	Table is open.
10032	2730	Need to do (hard) restructure.
10033	2731	Invalid mode.
10034	2732	Cannot close index.
10035	2733	Index is being used to order table.
10036	2734	Unknown user name or password.
10037	2735	Multi-level cascade is not supported.
10038	2736	Invalid field name.
10039	2737	Invalid table name.
10040	2738	Invalid linked cursor expression.
10041	2739	Name is reserved.
10042	273A	Invalid file extension.
10043	273B	Invalid language Driver.
10044	273C	Alias is not currently opened.
10045	273D	Incompatible record structures.
10046	273E	Name is reserved by DOS.
10047	273F	Destination must be indexed.
10048	2740	Invalid index type
10049	2741	Language Drivers of Table and Index do not match
10050	2742	Filter handle is invalid
10051	2743	Invalid Filter
10052	2744	Invalid table create request
10053	2745	Invalid table delete request
10054	2746	Invalid index create request
10055	2747	Invalid index delete request
10056	2748	Invalid table specified
10058	274A	Invalid Time.
10059	274B	Invalid Date.
10060	274C	Invalid Datetime
10061	274D	Tables in different directories
10062	274E	Mismatch in the number of arguments
10063	274F	Function not found in service library.
10064	2750	Must use baseorder for this operation.
10065	2751	Invalid procedure name
10241	2801	Record locked by another user.
10242	2802	Unlock failed.
10243	2803	Table is busy.
10244	2804	Directory is busy.
10245	2805	File is locked.
10246	2806	Directory is locked.
10247	2807	Record already locked by this session.
10248	2808	Object not locked.
10249	2809	Lock time out.
10250	280A	Key group is locked.
10251	280B	Table lock was lost.
10252	280C	Exclusive access was lost.
10253	280D	Table cannot be opened for exclusive use.
10254	280E	Conflicting record lock in this session.
10255	280F	A deadlock was detected.
10256	2810	A user transaction is already in progress.

 10257 2811 No user transaction is currently in progress.
 10258 2812 Record lock failed.
 10259 2813 Couldn't perform the edit because another user changed the record.
 10260 2814 Couldn't perform the edit because another user deleted or moved the record.
 10497 2901 Insufficient field rights for operation.
 10498 2902 Insufficient table rights for operation. Password required.
 10499 2903 Insufficient family rights for operation.
 10500 2904 This directory is read only.
 10501 2905 Database is read only.
 10502 2906 Trying to modify read-only field.
 10503 2907 Encrypted dBASE tables not supported.
 10504 2908 Insufficient SQL rights for operation.
 10753 2A01 Field is not a BLOB.
 10754 2A02 BLOB already opened.
 10755 2A03 BLOB not opened.
 10756 2A04 Operation not applicable.
 10757 2A05 Table is not indexed.
 10758 2A06 Engine not initialized.
 10759 2A07 Attempt to re-initialize Engine.
 10760 2A08 Attempt to mix objects from different sessions.
 10761 2A09 Paradox driver not active.
 10762 2A0A Driver not loaded.
 10763 2A0B Table is read only.
 10764 2A0C No associated index.
 10765 2A0D Table(s) open. Cannot perform this operation.
 10766 2A0E Table does not support this operation.
 10767 2A0F Index is read only.
 10768 2A10 Table does not support this operation because it is not uniquely indexed.
 10769 2A11 Operation must be performed on the current session.
 10770 2A12 Invalid use of keyword.
 10771 2A13 Connection is in use by another statement.
 10772 2A14 Passthrough SQL connection must be shared
 11009 2B01 Invalid function number.
 11010 2B02 File or directory does not exist.
 11011 2B03 Path not found.
 11012 2B04 Too many open files. You may need to increase MAXFILEHANDLE limit in IDAPI configuration.
 11013 2B05 Permission denied.
 11014 2B06 Bad file number.
 11015 2B07 Memory blocks destroyed.
 11016 2B08 Not enough memory.
 11017 2B09 Invalid memory block address.
 11018 2B0A Invalid environment.
 11019 2B0B Invalid format.
 11020 2B0C Invalid access code.
 11021 2B0D Invalid data.
 11023 2B0F Device does not exist.
 11024 2B10 Attempt to remove current directory.
 11025 2B11 Not same device.
 11026 2B12 No more files.
 11027 2B13 Invalid argument.
 11028 2B14 Argument list is too long.
 11029 2B15 Execution format error.
 11030 2B16 Cross-device link.

 11041 2B21 Math argument.
 11042 2B22 Result is too large.
 11043 2B23 File already exists.
 11047 2B27 Unknown internal operating system error.
 11058 2B32 Share violation.
 11059 2B33 Lock violation.
 11060 2B34 Critical DOS Error.
 11061 2B35 Drive not ready.
 11108 2B64 Not exact read/write.
 11109 2B65 Operating system network error.
 11110 2B66 Error from NOVELL file server.
 11111 2B67 NOVELL server out of memory.
 11112 2B68 Record already locked by this workstation.
 11113 2B69 Record not locked.
 11265 2C01 Network initialization failed.
 11266 2C02 Network user limit exceeded.
 11267 2C03 Wrong .NET file version.
 11268 2C04 Cannot lock network file.
 11269 2C05 Directory is not private.
 11270 2C06 Multiple .NET files in use.
 11271 2C07 Unknown network error.
 11272 2C08 Not initialized for accessing network files.
 11273 2C09 SHARE not loaded. It is required to share local files.
 11274 2C0A Not on a network. Not logged in or wrong network driver.
 11275 2C0B Lost communication with SQL server.
 11521 2D01 Optional parameter is required.
 11522 2D02 Invalid optional parameter.
 11777 2E01 obsolete
 11778 2E02 obsolete
 11779 2E03 Ambiguous use of ! (inclusion operator).
 11780 2E04 obsolete
 11781 2E05 obsolete
 11782 2E06 A SET operation cannot be included in its own grouping.
 11783 2E07 Only numeric and date/time fields can be averaged.
 11784 2E08 Invalid expression.
 11785 2E09 Invalid OR expression.
 11786 2E0A obsolete
 11787 2E0B bitmap
 11788 2E0C CALC expression cannot be used in INSERT, DELETE, CHANGETO and
SET rows.
 11789 2E0D Type error in CALC expression.
 11790 2E0E CHANGETO can be used in only one query form at a time.
 11791 2E0F Cannot modify CHANGED table.
 11792 2E10 A field can contain only one CHANGETO expression.
 11793 2E11 A field cannot contain more than one expression to be inserted.
 11794 2E12 obsolete
 11795 2E13 CHANGETO must be followed by the new value for the field.
 11796 2E14 Checkmark or CALC expressions cannot be used in FIND queries.
 11797 2E15 Cannot perform operation on CHANGED table together with a
CHANGETO query.
 11798 2E16 chunk
 11799 2E17 More than 255 fields in ANSWER table.
 11800 2E18 AS must be followed by the name for the field in the ANSWER
table.
 11801 2E19 DELETE can be used in only one query form at a time.
 11802 2E1A Cannot perform operation on DELETED table together with a
DELETE query.

 11803 2E1B Cannot delete from the DELETED table.
 11804 2E1C Example element is used in two fields with incompatible types or with a BLOB.
 11805 2E1D Cannot use example elements in an OR expression.
 11806 2E1E Expression in this field has the wrong type.
 11807 2E1F Extra comma found.
 11808 2E20 Extra OR found.
 11809 2E21 One or more query rows do not contribute to the ANSWER.
 11810 2E22 FIND can be used in only one query form at a time.
 11811 2E23 FIND cannot be used with the ANSWER table.
 11812 2E24 A row with GROUPBY must contain SET operations.
 11813 2E25 GROUPBY can be used only in SET rows.
 11814 2E26 Use only INSERT, DELETE, SET or FIND in leftmost column.
 11815 2E27 Use only one INSERT, DELETE, SET or FIND per line.
 11816 2E28 Syntax error in expression.
 11817 2E29 INSERT can be used in only one query form at a time.
 11818 2E2A Cannot perform operation on INSERTED table together with an INSERT query.
 11819 2E2B INSERT, DELETE, CHANGETO and SET rows may not be checked.
 11820 2E2C Field must contain an expression to insert (or be blank).
 11821 2E2D Cannot insert into the INSERTED table.
 11822 2E2E Variable is an array and cannot be accessed.
 11823 2E2F Label
 11824 2E30 Rows of example elements in CALC expression must be linked.
 11825 2E31 Variable name is too long.
 11826 2E32 Query may take a long time to process.
 11827 2E33 Reserved word or one that can't be used as a variable name.
 11828 2E34 Missing comma.
 11829 2E35 Missing).
 11830 2E36 Missing right quote.
 11831 2E37 Cannot specify duplicate column names.
 11832 2E38 Query has no checked fields.
 11833 2E39 Example element has no defining occurrence.
 11834 2E3A No grouping is defined for SET operation.
 11835 2E3B Query makes no sense.
 11836 2E3C Cannot use patterns in this context.
 11837 2E3D Date does not exist.
 11838 2E3E Variable has not been assigned a value.
 11839 2E3F Invalid use of example element in summary expression.
 11840 2E40 Incomplete query statement. Query only contains a SET definition.
 11841 2E41 Example element with ! makes no sense in expression.
 11842 2E42 Example element cannot be used more than twice with a ! query.
 11843 2E43 Row cannot contain expression.
 11844 2E44 obsolete
 11845 2E45 obsolete
 11846 2E46 No permission to insert or delete records.
 11847 2E47 No permission to modify field.
 11848 2E48 Field not found in table.
 11849 2E49 Expecting a column separator in table header.
 11850 2E4A Expecting a column separator in table.
 11851 2E4B Expecting column name in table.
 11852 2E4C Expecting table name.
 11853 2E4D Expecting consistent number of columns in all rows of table.
 11854 2E4E Cannot open table.
 11855 2E4F Field appears more than once in table.
 11856 2E50 This DELETE, CHANGE or INSERT query has no ANSWER.

 11857 2E51 Query is not prepared. Properties unknown.
 11858 2E52 DELETE rows cannot contain quantifier expression.
 11859 2E53 Invalid expression in INSERT row.
 11860 2E54 Invalid expression in INSERT row.
 11861 2E55 Invalid expression in SET definition.
 11862 2E56 row use
 11863 2E57 SET keyword expected.
 11864 2E58 Ambiguous use of example element.
 11865 2E59 obsolete
 11866 2E5A obsolete
 11867 2E5B Only numeric fields can be summed.
 11868 2E5C Table is write protected.
 11869 2E5D Token not found.
 11870 2E5E Cannot use example element with ! more than once in a single row.
 11871 2E5F Type mismatch in expression.
 11872 2E60 Query appears to ask two unrelated questions.
 11873 2E61 Unused SET row.
 11874 2E62 INSERT, DELETE, FIND, and SET can be used only in the leftmost column.
 11875 2E63 CHANGETO cannot be used with INSERT, DELETE, SET or FIND.
 11876 2E64 Expression must be followed by an example element defined in a SET.
 11877 2E65 Lock failure.
 11878 2E66 Expression is too long.
 11879 2E67 Refresh exception during query.
 11880 2E68 Query canceled.
 11881 2E69 Unexpected Database Engine error.
 11882 2E6A Not enough memory to finish operation.
 11883 2E6B Unexpected exception.
 11884 2E6C Feature not implemented yet in query.
 11885 2E6D Query format is not supported.
 11886 2E6E Query string is empty.
 11887 2E6F Attempted to prepare an empty query.
 11888 2E70 Buffer too small to contain query string.
 11889 2E71 Query was not previously parsed or prepared.
 11890 2E72 Function called with bad query handle.
 11891 2E73 QBE syntax error.
 11892 2E74 Query extended syntax field count error.
 11893 2E75 Field name in sort or field clause not found.
 11894 2E76 Table name in sort or field clause not found.
 11895 2E77 Operation is not supported on BLOB fields.
 11896 2E78 General BLOB error.
 11897 2E79 Query must be restarted.
 11898 2E7A Unknown answer table type.
 11926 2E96 Blob cannot be used as grouping field.
 11927 2E97 Query properties have not been fetched.
 11928 2E98 Answer table is of unsuitable type.
 11929 2E99 Answer table is not yet supported under server alias.
 11930 2E9A Non-null blob field required. Can't insert records
 11931 2E9B Unique index required to perform changeto
 11932 2E9C Unique index required to delete records
 11933 2E9D Update of table on the server failed.
 11934 2E9E Can't process this query remotely.
 11935 2E9F Unexpected end of command.
 11936 2EA0 Parameter not set in query string.
 11937 2EA1 Query string is too long.

 12033 2F01 Interface mismatch. Engine version different.
 12034 2F02 Index is out of date.
 12035 2F03 Older version (see context).
 12036 2F04 .VAL file is out of date.
 12037 2F05 BLOB file version is too old.
 12038 2F06 Query and Engine DLLs are mismatched.
 12289 3001 Capability not supported.
 12290 3002 Not implemented yet.
 12291 3003 SQL replicas not supported.
 12292 3004 Non-blob column in table required to perform operation.
 12293 3005 Multiple connections not supported.
 12545 3101 Invalid database alias specification.
 12546 3102 Unknown database type.
 12547 3103 Corrupt system configuration file.
 12548 3104 Network type unknown.
 12549 3105 Not on the network.
 12550 3106 Invalid configuration parameter.
 12801 3201 Object implicitly dropped.
 12802 3202 Object may be truncated.
 12803 3203 Object implicitly modified.
 12804 3204 Should field constraints be checked?
 12805 3205 Validity check field modified.
 12806 3206 Table level changed.
 12807 3207 Copy linked tables?
 12809 3209 Object implicitly truncated.
 12810 320A Validity check will not be enforced.
 12811 320B Multiple records found, but only one was expected.
 12812 320C Field will be trimmed, cannot put master records into PROBLEM
table.
 13057 3301 File already exists.
 13058 3302 BLOB has been modified.
 13059 3303 General SQL error.
 13060 3304 Table already exists.
 13061 3305 Paradox 1.0 tables are not supported.
 13313 3401 Different sort order.
 13314 3402 Directory in use by earlier version of Paradox.
 13315 3403 Needs Paradox 3.5-compatible language driver.
 14849 3A01 SYSTEM
 14850 3A02 DRIVERS
 14851 3A03 DATABASES
 14853 3A05 VERSION
 14854 3A06 NET TYPE
 14855 3A07 NET DIR
 14856 3A08 LOCAL SHARE
 14857 3A09 LANGDRIVER
 14858 3A0A LANGDRVDIR
 14859 3A0B MINBUFSIZE
 14860 3A0C MAXBUFSIZE
 14861 3A0D LOCKRETRY
 14862 3A0E SYSFLAGS
 14864 3A10 SQLQRYMODE
 14865 3A11 LOW MEMORY USAGE LIMIT
 14868 3A14 VERSION
 14869 3A15 TYPE
 14870 3A16 LANGDRIVER
 14871 3A17 FILL FACTOR
 14872 3A18 BLOCK SIZE

```
14873  3A19  LOCKPROTOCOL
14874  3A1A  LEVEL
14875  3A1B  DRIVER FLAGS
14878  3A1E  MEMO FILE BLOCK SIZE
14879  3A1F  MDX BLOCK SIZE
14888  3A28  INIT
14889  3A29  DB CREATE
14890  3A2A  DB OPEN
14891  3A2B  TABLE CREATE
14892  3A2C  TABLE OPEN
14898  3A32  DB INFO
14908  3A3C  TYPE
14909  3A3D  PATH
14910  3A3E  DEFAULT DRIVER
14918  3A46  INIT
14919  3A47  TYPE
14920  3A48  STANDARD
14921  3A49  TRUE
14922  3A4A  FALSE
14923  3A4B  OPEN MODE
14924  3A4C  READ/WRITE
14925  3A4D  READ ONLY
14926  3A4E  SHARE MODE
14927  3A4F  EXCLUSIVE
14928  3A50  SHARED
14929  3A51  USER NAME
14930  3A52  SERVER NAME
14931  3A53  DATABASE NAME
14932  3A54  SCHEMA CACHE SIZE
14933  3A55  STRICTINTEGRTY
14938  3A5A  ORACLE
14939  3A5B  1.0
14940  3A5C  SERVER
14941  3A5D  NET PROTOCOL
14942  3A5E  DECNET
14943  3A5F  NETBIOS
14944  3A60  NAMED PIPES
14945  3A61  SPX/IPX
14946  3A62  TCP/IP
14947  3A63  3270
14948  3A64  VINES
14949  3A65  APPC
14950  3A66  ASYNC
14958  3A6E  SYBASE
14959  3A6F  1.0
14960  3A70  SERVER
14961  3A71  BLOB EDIT LOGGING
14962  3A72  CONNECT TIMEOUT
14963  3A73  TIMEOUT
14964  3A74  DATE MODE
14965  3A75  DATE SEPARATOR
14966  3A76  DECIMAL SEPARATOR
14968  3A78  INTRBASE
14969  3A79  1.0
14970  3A7A  SERVER
14978  3A82  FORMATS
14979  3A83  DATE
```

14980	3A84	TIME
14981	3A85	NUMBER
14988	3A8C	SEPARATOR
14989	3A8D	MODE
14990	3A8E	FOURDIGITYEAR
14991	3A8F	YEARBIASED
14992	3A90	LEADINGZEROM
14993	3A91	LEADINGZEROD
14994	3A92	TWELVEHOUR
14995	3A93	AMSTRING
14996	3A94	PMSTRING
14997	3A95	SECONDS
14998	3A96	MILSECONDS
15008	3AA0	DECIMALSEPARATOR
15009	3AA1	THOUSANDSEPARATOR
15010	3AA2	DECIMALDIGITS
15011	3AA3	LEADINGZERON
15013	3AA5	ascii
15014	3AA6	DB437US0
15018	3AAA	/
15019	3AAB	0
15020	3AAC	FALSE
15021	3AAD	TRUE
15022	3AAE	TRUE
15023	3AAF	TRUE
15024	3AB0	TRUE
15025	3AB1	AM
15026	3AB2	PM
15027	3AB3	TRUE
15028	3AB4	FALSE
15029	3AB5	.
15030	3AB6	,
15031	3AB7	TRUE
15873	3E01	Wrong driver name.
15874	3E02	Wrong system version.
15875	3E03	Wrong driver version.
15876	3E04	Wrong driver type.
15877	3E05	Cannot load driver.
15878	3E06	Cannot load language driver.
15879	3E07	Vendor initialization failed.
16129	3F01	Query By Example
16130	3F02	SQL Generator
16131	3F03	IDAPI
16132	3F04	Lock Manager
16133	3F05	SQL Driver
16134	3F06	IDAPI Services
16135	3F07	dBASE Driver
16138	3F0A	Token
16140	3F0C	Table
16141	3F0D	Field
16142	3F0E	Image
16143	3F0F	User
16144	3F10	File
16145	3F11	Index
16146	3F12	Directory
16147	3F13	Key
16148	3F14	Alias

```
 16149  3F15  Drive
 16150  3F16  Server error
 16151  3F17  Server message
 16152  3F18  Line Number
 16153  3F19  Capability
 16154  3F1A  Limit
 16239  3F6F

{ ErrList - Copyright 1995 Robert Arnson }

program ErrList;

uses DbiTypes, DbiProcs, DbiErrs, SysUtils;

var
  Category: byte;
  Code: byte;
  ResultCode: word;
  ErrorString: array[0..DBIMAXMSGLEN + 1] of char;
  Output: text;
  OutString: string;

begin
  DbiInit(nil);

  Assign(Output, 'ErrList.Txt');
  Rewrite(Output);

  for Category := ERRCAT_NONE to ERRCAT_RC do
    for Code := 0 to 255 do
    begin
      ResultCode := (Category shl 8) + Code;
      DbiGetErrorString(ResultCode, ErrorString);
      if StrLen(ErrorString) > 0 then
      begin
        OutString := Format('%6d  %0.4x  %s', [ResultCode, ResultCode,
ErrorString]);
        WriteLn(Output, OutString);
      end;
    end;

  DbiExit;
end.
```

Q: How do I get the result from a TStoredProc?

A:

```
var
  sp : TStoredProc;
begin
  sp := TStoredProc.Create(nil);
  with sp do
  begin
    DatabaseName := 'test';
    StoredProcName := 'dbwintech1.johnt;1';
    ExecProc;
    Caption := IntToStr(Params[0].AsInteger);
    Free;
  end;
end;
```

TStoredProc

How do I get the result from a TStoredProc?

TDBNavigator

How do I determine if a particular button on a TDBNavigator control is enabled?
Navigator control use

Q: How can I restore a window to its last state when I run it again?

A: Here is WindowRestorer - a window size and state restorer

DESCRIPTION: Ever notice how professional programs seem to remember in what condition and location you left them and their child windows? Ever notice how most RAD apps don't? You can take that ragged edge off your program with this unit. It Allows apps to save the location, size, and state of windows so that when the user reopens them, they will look as the user left them.

USE: Put WINRSTOR in the uses of clause of your main form and any forms that will be saving or restoring their own state, size, or location. (If you will be doing all the saving and restoring using WinSaveChildren and WinRestoreChildren from the main form, you only need reference it in the main form's uses clause.)

In MainForm.Create, initialize the global WinRestorer object as follows (it's already declared in this file, but needs to be allocated): GlobalWinRestorer := TWinRestorer.create(Application, TRUE, WHATSAVE_ALL); Which is the same as: GlobalWinRestorer := TWinRestorer.create(Application, TRUE, [location, size, state]); Then, in MainForm.Destroy, deallocate the global WinRestorer object as follows: GlobalWinRestorer.free;

A good place to save a form's status is in the queryclose event or else attached to a button or menu item. I usually create an item in the File Menu captioned 'Save &Workspace' which does: GlobalWinRestorer.SaveChildren(Self, [default]); And under main form's Close event I put: GlobalWinRestorer.SaveWin(Self, [WHATSAVE_ALL]);

I have tended to restore the children's status in their own show events like this: GlobalWinRestorer.RestoreWin(Self, [default]); though I am moving toward putting in the main form's show event: GlobalWinRestorer.RestoreWin(Self, [default]); GlobalWinRestorer.RestoreChildren(Self, [default]);

HINTS: If you set TForm.Position to poScreenCenter or anything fancy, this unit won't do what you expect. poDesigned seems to work fairly well. I could have raised an exception if you try to set top and left of a poScreenCentere'd form, but then you have to be careful using WinRestoreChildren. I opted not to check the position property and leave that up to individual developers.

```
unit WinRstor;

INTERFACE

USES SysUtils, Forms;

TYPE {============================================================}

{------------------------------------------------------------------
```

```
Windows restorer object class and related types.
----------------------------------------------------------------------}
EWinRestorer = class( Exception);
TWhatSave = (default, size, location, state);
STWhatSave = set of TWhatSave;
TWinRestorer = class(TObject)
 protected
  mIniFile: string;
  mIniSect: string[80];
  mIsInitialized: boolean;
  mDefaultWhat: STWhatSave;
 public
  constructor Create( TheApp: TApplication;
    LocalDir: boolean; DefaultWhatSave: STWhatSave);
    {If localDir is true, ini dir is the app dir.  Else, ini dir is the
windows dir.}
  procedure SaveWin(TheForm: TForm; What: STWhatSave);
  procedure SaveChildren(TheMDIForm: TForm; What: STWhatSave);
  procedure RestoreWin( TheForm: TForm; What: STWhatSave);
  procedure RestoreChildren(TheMDIForm: TForm; What: STWhatSave);
  property IniFileName: string  read mIniFile;
end;

CONST
  WHATSAVE_ALL = [size, location, state];

VAR
GlobalWinRestorer: TWinRestorer;

IMPLEMENTATION

Uses IniFiles;

constructor TWinRestorer.create;
var fname, path: string[100];
begin
  inherited create;
{Calculate ini file name}
  if default in DefaultWhatSave then
    raise EWinRestorer.create(
     'Attempt to initialize default window position paramaters with set ' +
     ' containing [default] item.  ' +
     'Default params may contain only members of [size, location, state].  ')
  else mDefaultWhat := DefaultWhatSave;
  fname := ChangeFileExt( ExtractFileName( TheApp.exeName), '.INI');
  if LocalDir then begin {parse out path and add to file name}
    path := ExtractFilePath(TheApp.exeName);
    if path[length(path)] <> '\' then
      path := path + '\';
    fname := path + fname;
  end;
{fill object fields}
  mIniFile := fname;
  mIniSect := 'WindowsRestorer';
{It'd be nice to write some notes to a section called [WinRestorer Notes]}
end;
```

```
procedure TWinRestorer.RestoreWin;
var FormNm, SectionNm: string[80];   ini: TIniFile;
  n,l,t,w,h: integer; {Left, Top Width, Height}
begin
  ini := TIniFile.create( mIniFile);
  TRY
    SectionNm := mIniSect;
    FormNm := TheForm.classname;
    if default in What then What := mDefaultWhat;
{Update Window State if Necessary}
    if state in What then
      n := ini.ReadInteger( SectionNm, FormNm + '_WindowState', 0);
      case  n of
        1:   TheForm.WindowState := wsMinimized;
        2:  TheForm.WindowState := wsNormal;
        3:   TheForm.WindowState := wsMaximized;
      end;
{Update Size and Location if necessary.}
    with TheForm do begin l:=left; t:=top; h:=height; w:=width; end; {Save
current vals.}
    if size in What then begin
      w := ini.ReadInteger( SectionNm, FormNm + '_Width', w);
      h := ini.ReadInteger( SectionNm, FormNm + '_Height', h);
    end;
    if location in What then begin
      t := ini.ReadInteger( SectionNm, FormNm + '_Top', t);
      l := ini.ReadInteger( SectionNm, FormNm + '_Left', l);
    end;
    TheForm.SetBounds(l,t,w,h);
  FINALLY
    ini.free;
  END;
end;

procedure TWinRestorer.RestoreChildren;
var i: integer;
begin
  if TheMDIForm.formstyle <> fsMDIForm then
    raise EWinRestorer.create('Attempting to save window sizes of children
for a non MDI parent window.')
  else
    for i := 0 to TheMDIForm.MDIChildCount - 1 do
      RestoreWin( TheMDIForm.MDIChildren[i], what);
end;

procedure TWinRestorer.SaveWin;
var FormNm, SectionNm: string[80];   w : STWhatsave; ini: TIniFile;
begin
  ini := TIniFile.create( mIniFile);
  TRY
    SectionNm := mIniSect;
    FormNm := TheForm.ClassName;
    if default in What then w := mDefaultWhat else w := mDefaultWhat;
    if size in w then begin
      ini.WriteInteger( SectionNm, FormNm + '_Width', TheForm.Width);
      ini.WriteInteger( SectionNm, FormNm + '_Height', TheForm.Height);
    end;
```

```
    if location in w then begin
      ini.WriteInteger( SectionNm, FormNm + '_Top', TheForm.Top);
      ini.WriteInteger( SectionNm, FormNm + '_Left', TheForm.Left);
    end;
    if state in w then
      case TheForm.WindowState of
        wsMinimized:   ini.WriteInteger( SectionNm, FormNm + '_WindowState',
1);
        wsNormal:     ini.WriteInteger( SectionNm, FormNm + '_WindowState',
2);
        wsMaximized:   ini.WriteInteger( SectionNm, FormNm + '_WindowState',
3);
      end;
  FINALLY
    ini.free;
  END;
end;

procedure TWinRestorer.SaveChildren;
var i: integer;
begin
  if TheMDIForm.formstyle <> fsMDIForm then
    raise EWinRestorer.create('Attempting to restore window sizes of children
for a non MDI parent window.')
  else
    for i := 0 to TheMDIForm.MDIChildCount - 1 do
      SaveWin( TheMDIForm.MDIChildren[i], what);
end;

INITIALIZATION
END.
```

Q: I'm getting a GPF (Message is: 'Stack Fault in module <DLL name> at <address>') when my Delphi app calls a function in a DLL. (It's probably a C or C++ DLL) The function returns the value of a floating-point variable in another application, and is described as follows:

```
float <function name>(ACCID accID, HPT hPt)
```

Return value is a 32-bit IEEE floating point number.

The return value is read into a variable of type single. The DLL vendor tells me that the floating point value returned might be incompatible with Borland's floating point format. Any idea as to why this is happening?

A: It's not a question of the floating-point format. Instead, the problem is that Microsoft compilers return floating-point values in a different manner from that of Borland compilers. Here's Pat's standard MS vs. Borland FP result message (it talks about Excel, but it applies to MS compilers in general):

When a "B" format is specified as the function result for a registered function in Excel, Excel passes a near pointer (SS relative) that is the location to store the result of function. In addition Excel expects a far pointer to be returned that specifies the address of the result. So for a function registered as "BBB" where you THINK the function heading SHOULD be:

```
function MyFunc(d1,d2 : double) : double; export;
```

the actual Turbo Pascal function actually is:

```
function MyFunc(d1,d2 : double; NearPtr : word) : pointer; export;
```

So that results are returned as expected, a temporary pointer variable is used with the address established as outline above. The body of the function would be:

```
var
  Temp : ^double;
begin
  MyFunc := Ptr(SSeg,NearPtr); { what Excel expects to be returned }
  Temp := Ptr(SSeg,NearPtr);
  Temp^ := d1+d2; { or whatever your function needs to do }
end;
```

TCanvas

How do I set and reset the canvas.font.style property?
How do I do a TextOut() so that I don't get the white space around the text?
How do I paint with a cross-hatched brush?

Q: Is there ANY way of calling a dll function at runtime if the DLL is not known about at compile time???

I am trying to call a DLL function at runtime. I know the Name of the DLL, of the FUNCTION, and the PARAMETERS which need to be passed to it. The only problem is that I do NOT know these items at the compile time of the App. They will be retrieved from an INI file.

So far: I used LoadLibrary to get a handle to the DLL, I used GetProcAddress to get the address of the Function. Now I have a TFarProc which is pointing to the function and I am STUCK!!!! :-(

A:

What you are trying to do is hard in a compiled language. I think you will have to drop to assembler for the actual function call.

I would go at it this way:

Do the LoadLibrary and GetProcAddress stuff as you do now, that is the easy part <g>. Then you need to evaluate the parameters and construct an image of the call stack frame in a buffer. That can be done in Pascal code without problem. Wrap the whole mess into a TStack object, for example, that has a Push method like

```
 Procedure TStack.Push( Var data; datasize: Word );
```

A quick sketch of such a beast would perhaps look like this:

```
Type
  TStack= Class
  private
    FPtr: Pointer;
    FSize: Word;
    FStackTop: Pointer;
    FUsed: Word;
    Function GetFree: Word;
  public
    Constructor Create( size: Word );
    Destructor Destroy; override;
    Procedure Push( Var data; datasize: Word );

    Property StackTop: Pointer read FStackTop;
    Property Free: Word read GetFree;
    Property Used: Word read FUsed
  end;

{  Methods of TStack  }
Function TStack.GetFree: Word;
  Begin
    Result := FSize - FUsed;
  End; { TStack.GetFree }
```

```
Constructor TStack.Create( size: Word );
  Begin
    inherited Create;
    FSize := size;
    GetMem( FPtr, size );
    FStackTop := FPtr;
    Inc( PtrRec( FStackTop ).ofs, size );
  End; { TStack.Create }

Destructor TStack.Destroy;
  Begin
    FreeMem( FPtr, size );
  End; { TStack.Destroy }

Procedure TStack.Push( Var data; datasize: Word );
  Begin
    If Free >= datasize Then Begin
      Dec( PtrRec( FStackTop ).ofs, datasize );
      Move( data, FStackTop^, datasize );
      Inc( FUsed, datasize );
    End { If }
    Else
      raise Exception.Create( 'Stack full!' );
  End; { TStack.Push }
```

So you create an instance of this animal and Push your parameters onto the stack. The actual call would then look something like this:

```
 Var
   Stack: TStack;
   size : Word;
   Top  : Pointer;
 Begin
   ..create stack and fill it
   size := Stack.Used;
   Top  := Stack.StackTop;
   asm
     mov cx, size
     jcxz @noParams
     mov ax, ss
     mov es, ax
     sub sp, cx
     mov di, sp
     push ds
     lds si, Top
     cld
     rep movsb
     pop ds
   @noParams:
     call DLLProc
     { add sp, size  only if DLLProc is cdecl and size <> 0! }
   end;
end;
```

One caveat: TStack.Push should raise an exception if you try to push data with an odd datasize; the CPU stack uses WORD as the smallest unit so only data with an even datasize can be pushed; odd size items like Char and Byte have to be extended to Words first!

Murphy's Laws and Other Observations

MURPHY'S LAW
If anything can go wrong, it will.

MURPHY'S FIRST COROLLARY
Nothing is as easy as it looks.

MURPHY'S SECOND COROLLARY
Everything takes longer than you think.

MURPHY'S THIRD COROLLARY
If there is a possibility of several things going wrong, the one that will cause the most damage will be the one to go wrong.

MURPHY'S FOURTH COROLLARY
Whenever you set out to do something, something else must be done first.

MURPHY'S FIFTH COROLLARY
Every solution breeds new problems.

MURPHY'S SIXTH COROLLARY
It is impossible to make anything foolproof because fools are so ingenious.

MURPHY'S SEVENTH COROLLARY
Nature always sides with the hidden flaw.

MURPHY'S EIGHTH COROLLARY
Left to themselves, things tend to go from bad to worse.

MURPHY'S CONSTANT
Matter will be damaged in direct proportion to its value.

HILL'S COMMENTARIES ON MURPHY'S LAW
1. If we have much to lose by having things go wrong, take all possible care.
2. If we have nothing to lose, relax.
3. If we have everything to gain, relax.
4. If it doesn't matter, it doesn't matter.

BOLING'S POSTULATE
If you're feeling good, don't worry. you'll get over it.

MURPHY'S LAW OF MULTIPLES
If you percieve that there are four possible ways in which a procedure can go wrong, and circumvent these, then a fifth way will promptly develop.

THE ORDERING PRINCIPLE

The supplies necessary for yesterday's work must be ordered no later then noon tomorrow.

CHISHOLM'S LAW

When things just can't get any worse, they will.

CHISHOLM'S COMMENTARY

Anytime things appear to be going better, you have overlooked something.

SCOTT'S FIRST LAW

No matter what goes wrong, it will probably look right.

SCOTT'S SECOND LAW

When an error has been detected and corrected it will be found to have been correct in the first place.

FINAGLE'S FIRST LAW

If an experiment works, something has gone wrong

FINAGLE'S SECOND LAW

No matter what the anticipated result,
there will always be someone eager to
(a) misinterpret it,
(b) fake it, or
(c) believe it happened to his own pet theory.

FINAGLE'S THIRD LAW

In any collection of data, the figure most obviously correct, beyond all need of checking, is the mistake.

COROLLARIES

1. No one whom you ask for help will see it.
2. Everyone who stops by with unsought advice will see it immediately.

FINAGLE'S FOURTH LAW

Once a job is fouled up, anything done to improve it only makes it worse.

WINGO'S AXIOM

All Finagle's Laws may be bypassed by learning the simple art of doing without thinking.

SIMON'S LAW

Everything put togeather falls apart sooner or later.

GUMPERSON'S LAW

The probability of anything happening is in inverse ratio to its desirability.

ISSAWI'S LAWS OF PROGRESS
The Course of Progress: Most things get steadily worse.
The Path of Progress: A shortcut is the longest distance between two points.

MURPHY'S LAW OF THERMODYNAMICS
Things get worse under pressure.

THE UNSPEAKABLE LAW
As soon as you mention something....
... if it's good, it goes away.
... if it's bad, it happens.

STURGEON'S LAW
90% of everything is crud.

STOCKMAYER'S THEOREM
If it looks easy, it's tough.
If it looks tough, it's impossible.

COMMONER'S SECOND LAW OF ECOLOGY
Nothing ever goes away.

HOWE'S LAW
Everyone has a scheme that will not work.

GINSBERG'S THEOREM
1. You can't win
2. You can't break even.
3. You can't even quit the game.

RUDIN'S LAW
In crises that force people to choose among
alternative courses of action, most people will choose the
worst one possible.

ZYMURGY'S FIRST LAW OF EVOLVING
(System Dynamics:)
Once you open a can of worms, the only way to recan them
is to use a larger can.

NON-RECIPROCAL LAWS OF EXPECTATIONS
Negative expectations yield negative results.
Positive expectations yield negative results.

HARPER'S MAGAZINE LAW

You never find an article until
you replace it.

RICHARD'S COMPLIMENTARY RULES OF OWNERSHIP

1. If you keep anything long enough, you can throw it away.
2. If you throw it away, you will need it the next day.

LEWIS' LAW

No matter how long or how hard you shop for an item,
after you've bought it, it will be on sale somewhere cheaper.

PERLSWEIG'S LAW

People who can least afford to pay rent, pay rent. People who can most afford to pay rent, build up equity.

FIRST LAW OF BICYCLING

No matter which way you ride, it's uphill and against the wind.

THE AIRPLANE LAW

When the plane you are on is late, the plane you want to transfer to is on time.

LIEBERMAN'S LAW

Everybody lies; but it doesn't matter since nobody listens.

OSBORN'S LAW

Variables won't; constants aren't.

ILES'S LAW

There is always an easier way to do it.

COROLLARY

When looking directly at the easier way, especially for long periods, you will not see it.

SATTINGER'S LAW

It works better if you plug it in.

LAW OF THE LOST INCH

In designing any type of construction, no overall dimension can be totalled correctly after 4:40p.m. Friday

COROLLARY

The correct total will become self-evident at 9:01 a.m. on Monday.

EHRMAN'S COMMENTARY

1. Things get worse before they get better.
2. Who said things would get better?

SHAW'S PRINCIPAL

Build a system that even a fool can use, and only a fool will want to use it.

LUBARSKY'S LAW OF CYBERNETIC ENTOMOLOGY

There is always one more bug.

ANTHONY'S LAW OF THE WORKSHOP

Any tool, when dropped,will roll into the least accessible corner of the workshop.

HORNER'S FIVE THUMB POSTULATE

Experience varies directly with the equipment ruined.

MAIER"s LAW

If the facts do not conform to the theory, they must be disposed of.

WILLIAMS AND HOLLAND'S LAW

If enough data is collected, anything can be proven by statistical methods.

YOUNG'S LAW

All great discoveries are made by mistake.

COROLLARY

The greater the funding, the longer it takes to make the mistake.

RULE OF ACCURACY

When working toward the solution of a problem, it always helps if you know the answer.

MR. COOPER'S LAW

If you do not understand a particular word in a piece of technical writing, ignore it. The piece will make perfect sense without it.

TRUMAN'S LAW

If you cannot convince them, confuse them.

FIRST LAW OF DEBATE

Never argue with a fool - people might not know the difference.

WORKER'S DILEMMA

1. No matter how much you do, you'll never do enough.
2. What you don't do is always more important than what you do.

RULE OF THE GREAT

When somebody you greatly admire and respect appears to be thinking deep thoughts, they are probably thinking about lunch.

MILLER'S LAW

You can't tell how deep a puddle is until you step in it.

WELLINGTON'S LAW OF COMMAND

The cream rises to the top.
So does the scum.

COLE'S LAW

Thinly sliced cabbage.

PATTON'S LAW

A good plan today is better than a perfect plan tomorrow.

WESTHEIMER'S RULE

To estimate the time it takes to do a task, estimate the time you think it should take, multiply by 2, and change the unit of measure to the next highest unit. Thus we allocate 2 days for a one-hour task.

WIKER'S LAW

Government expands to absorb revenue and then some.

PARKINSON'S LAW OF DELAY

Delay is the deadliest form of denial.

NINETY-NINETY RULE OF PROJECT SCHEDULES

The first ninety percent of the task takes ninety percent of the time, and the last ten percent takes the other ninety percent.

LEVY'S NINTH LAW

Only God can make a random selection.

JUHANI'S LAW

The compromise will always be more expensive than either of the suggestions it is compromising.

JOHN'S COLLATERAL COLLARY

In order to get a loan you must first prove you don't need it.

MALEK'S LAW

Any simple idea will be worded in the most complicated way.

WEILER'S LAW

Nothing is impossible for the man who doesn't have to do it himself.

WEINBERG'S SECOND LAW

If builders built buildings the way programmers wrote programs, then the first

woodpecker that came along would destroy civilization.

MR. COLE'S AXIOM
The sum of the intelligence on the planet is a constant; the population is growing.

CANADA BILL JONES' MOTTO
It's morally wrong to allow suckers to keep their money.

JONES' MOTTO
Friends come and go, but enemies accumulate.

McCLAUGHRY'S CODICIL ^TO JONES' MOTTO
To make an enemy, do someone a favor.

WALKER'S LAW OF THE HOUSEHOLD
There is always more dirty laundry than clean laundry.

CLIVE'S REBUTTAL TO WALKER'S LAW
If it's clean, it isn't laundry.

LYNCH'S LAW
When the going gets tough, everybody leaves.

DUCHARME'S PRECEPT
Opportunity always knocks at the least opportune moment.

FURGUSON'S PRECEPT
A crisis is when you can't say let's forget the whole thing.

IMBESI'S LAW OF THE CONSERVATION OF FILTH
In order for something to become clean, something else must become dirty.
FREEMAN'S EXTENSION
... but you can get everything dirty without getting anything clean.

FLUGG'S RULE
The slowest checker is always at the quick- check-out lane.

MURRAY'S FIRST RULE OF THE ARENA
Nothing is ever so bad it can't be made worse by firing the coach.

HARRISON'S POSTULATE
For every action, there is an equal and opposite criticism.

PERKINS' POSTULATE
The bigger they are, the harder they hit.

ROGER'S RULE

Authorization for a project will be granted only when none of the authorizers can be blamed if the project fails but when all of the authorizers can claim credit if it succeeds.

MOLLISOMN'S BUREAUCRACY HYPOTHESIS

If an idea can survive a bureaucratic review and be implemented, it wasn't worth doing.

GOURD'S AXIOM

A meeting is an event at which the minutes are kept and the hours are lost.

WHISTLER'S LAW

You never know who's right, but you always know who's in charge.

LOFTUS' THEORY ON PERSONNEL RECRUITMENT

Far away talent always seems better then home-developed talent.

FIRST RULE OF NEGATIVE ANTICIPATION

You will save yourself a lot of needless worry if you don't burn your bridges until you come to them.

SPENCER'S LAWS OF DATA

1. Anyone can make a decision given enough facts.
2. A good manager can make a decision without enough facts.
3. A perfect manager can operate in perfect ignorance.

DREW'S LAW OF PROFESSIONAL PRACTICE

The client who pays the least complains the most.

LAW OF PROBABLE DISPERSAL

Whatever hits the fan will not be evenly distributed.

KITMAN'S LAW

Pure drivel tends to drive ordinary drivel off the TV screen.

SWIPPLE RULE OF ORDER

He who shouts loudest has the floor.

ROBERT'S AXIOM

Only errors exist.

BERMAN'S COROLLARY TO ROBERT'S AXIOM

One man's error is another man's data.

FIFTH LAW OF UNRELIABILITY

To err is human, but to really foul things up requires a computer.

DEAL'S FIRST LAW OF SAILING

The amount of wind will vary inversely with the number and experience of the people you take on board.

LERMAN'S LAW OF TECHNOLOGY

Any technical problem can be overcome given enough time and money.

LERMAN'S COROLLARY

You are never given enough time or money.

LAW OF LIFE'S HIGHWAY

If everything is coming your way, you're in the wrong lane.

BROMBERG'S FIRST LAW OF AUTO REPAIR

When the need arises, any tool or object closest to you becomes a hammer.

JOHNSON'S THIRD LAW

If you miss one issue of any magazine, it will be the issue which contained the article, story or installment you were most anxious to read.

COROLLARY

All of your friends either missed it, lost it, or threw it out.

KOVAC'S CONUNDRUM

When you dial a wrong number, you never get a busy signal.

SHIRLEY'S LAW

Most people deserve each other.

ARTHUR'S FIRST LAW OF LOVE

People to whom you are attracted invariably think you remind them of someone else.

GILLENSON'S LAW OF EXPECTATION

Never get excited about a blind date because of how it sounds over the phone.

CHEIT'S LAMENT

If you help a friend in need, he is sure to remember you - the next time he's in need.

DENNISTON'S LAW

Virtue is its own punishment.

DENNISTON'S COROLLARY

If you do something right once, someone will ask you to do it again.

JACOB'S LAW

To err is human - to blame it on someone else is even more human.

BALLANCE'S LAW OF RELIABILITY

How long a minute is depends on which side of the bathroom door you're on.

RUAN'S APPLICATION OF PARKINSON'S LAW

Possessions increase to fill the space available for their storage.

CORNUELLE'S LAW

Authority tends to assign jobs to those least able to do them.

ZYMURGY'S LAW OF VOLUNTEER LABOR

People are always available for work in the past tense.

CONWAY'S LAW

In any organization there will always be one person who knows what is going on. This person must be fired.

SEIT'S LAW OF HIGHER EDUCATION

The one course you must take to graduate will not be offered during your last semester.

THIRD LAW OF APPLIED TERROR

80% of the final exam will be based on the one lecture you missed about the one book you didn't read.

FOURTH LAW OF APPLIED TERROR

Every instructor assumes that you have nothing else to do except study for that instructor's course.

FIFTH LAW OF APPLIED TERROR

If you are given an open-book exam, you will forget the book.

COROLLARY

If you are given a take-home exam, you will forget where you live.

SIXTH LAW OF APPLIED TERROR

At the end of the semester you will recall having enrolled in a course at the beginning of the semester and never attending.

DUGGAN'S LAW OF SCHOLARLY RESEARCH

The most valuable quotation will be the one for which you cannot determine the source.

ROMINGER'S RULES FOR STUDENTS

1. The more general the title of a course, the less you will learn from it.
2. The more specific a title is, the less you will be able to apply it later.

WALLACE'S OBSERVATION

Everything is in a state of utter dishevelment.

TERMAN'S LAW OF INNOVATION

If you want a track team to win the high jump, you find one person who can jump seven feet, not seven people who can jump one foot.

CHURCHILL'S COMMENTARY ON MAN

Man will ocassionally stumble over the truth, but most of the time he will pick himself up and continue on.

MATZ'S MAXIM

A conclusion is the place where you get tired of thinking.

MERKIN'S MAXIM

When in doubt, predict that the trend will continue.

MEYER'S LAW

It is a simple task to make things complex, but a complex task to make them simple.

HLADE'S LAW

If you have a difficult task give it to a lazy man, he will find an easier way to do it.

MATZ'S RULE REGARDING MEDICATIONS

A drug is that substance which, when injected into a rat, will produce a scientific report.

GROUND RULE FOR LABORATORY WORKERS

When you do not know what you are doing, do it neatly.

LEE'S LAW

In any dealings with a collective body of people, the people will always be more tacky than originally expected.

SPENCER'S LAWS OF ACCOUNTABILITY

1. Trial balances don't.
2. Working capitol doesn't.
3. Liquidity tends to run out.
4. Return on investments won't.

O'BRIEN'S LAW

Nothing is ever done for the right reasons.

WARREN'S RULE

To spot the expert, pick the one who predicts the job will take the longest and cost the most.

MARS' RULE

An expert is anyone from out of town.

GREEN'S LAW OF DEBATE
Anything is possible if you don't know what you're talking about.

LAW OF REVELATION
The hidden flaw never remains hidden.

FIRST PRINCIPLE FOR PATIENTS
Just because your doctor has a name for your condition doesn't mean he knows what it is.

SECOND PRINCIPLE FOR PATIENTS
The more boring and out-of-date the magazines in the waiting room, the longer you will have to wait for your scheduled appointment.

THIRD PRINCIPLE FOR PATIENTS
Only adults have difficulty with child-proof bottler.

FOURTH PRINCIPLE FOR PATIENTS
you never have the right number of pills left on the last day od a prescription.

FIFTH PRINCIPLE FOR PATIENTS
If your condition seems to be getting better, it's probably your doctor getting sick.

TELESCO'S SECOND LAW OF NURSING
There are two kinds of adhesive tape the one that won't stay on and the one that won't come off.

FISH'S FIRST LAW OF ANIMAL BEHAVIOR
The probability of a cat eating its dinner has absolutely nothing to do with the price of the food placed before it.

JAFFE'S PRECEPT
There are some things which are impossible to know - but it is impossible to know which things these are.

THE SAUSAGE PRINCIPLE
People who love sausage and respect the law should never watch either one being made.

THE WATERGATE PRINCIPLE
government corruption is always reported in the past tense.

TODD'S FIRST TWO PRINCIPLES
1. No matter what they're telling you, they're not telling you the whole truth.

2. No matter what they're talking about, they're taking about money.

SPARK'S FIRST RULE FOR THE PROJECT MANAGER
Strive to look tremendously important.

SPARK'S SECOND RULE FOR MANAGERS
Attempt to be seen with important people.

SPARK'S THIRD RULE FOR MANAGERS
Speak with authority; however, only expound on the obvious and proven facts.

SPARK'S FOURTH RULE FOR MANAGERS
Don't engage in arguments, but if cornered, ask an irrelevant question and lean back with a satisfied grin while your opponent tries to figure out what's going on - then quickly change the subject.

SPARK'S FIFTH RULE FOR MANAGERS
Always keep the office door closed. This puts visitors on the defensive and also makes it look as if you are always in an important conference.

JAY'S FIRST LAW OF LEADERSHIP
Changing things is central to leadership, and changing them before anyone else is creativeness.

JACQUIN'S POSTULATE ON DEMOCRATIC GOVERNMENT
No man's life, liberty, ot property are safe while legislature is in session.

TRISCHMANN'S PARADOX
A pipe gives a wise man time to think and a fool something to stick in his mouth.

BUCY'S LAW
Nothing is ever accomplished by a reasonable man. (i.e. If he is reasonable, he can live with things as they are. He adapts to his environment. The unreasonable man will adapt his environment to himself. Therefore, progress is made by unreasonable men.)

LLOYD LINKLATER'S CORROLARY
You can't be excellent while being just like everyone else.

TROUTMANN'S PROGRAMMING POSTULATE #2
Profanity is the one language all programmers know best.

GLIB'S LAW OF UNRELIABILITY
Computers are unreliable, but humans are even more unreliable.

LAW OF THE PERVERSITY OF NATURE
You cannot sucessfully determine beforehand which side of the bread to butter.

HELLER'S LAW
The first myth of management is that it exists.

FREEMAN'S RULE
Circumstances can force a generalized incompetent to become competent, at least in a specialized field.

Mc GOWAN'S CHRISTMAS SHOPPING AXIOM
If an item is advertised as "under $50.00", you can bet it's not $19.95.

BROOK'S LAWS OF RETAILING
Security isn't.
Management can't.
Sale promotions don't.
Consumer assistance doesn't.
Workers won't.

HERSHISER'S SECOND RULE
The label "NEW" and/or "IMPROVED" means the price went up.

HERSHISER'S THIRD RULE
The label "ALL NEW","COMPLETELY NEW" or "GREAT NEW" means the price went way up.

LEWIS' LAW
People will buy anything that's one to a customer.

CHISHOLM'S FIRST COROLLARY
If you do something which you are sure will meet with everybody's approval, somebody won't like it.

COOPER'S METALAW
A proliferation of new laws creates a proliferation of new loopholes.

Weiler's Law

Nothing is impossible for the man who doesn't have to do it himself.

The Laws of Computer Programming

1. Any given program, when running, is obsolete.
2. Any given program costs more and takes longer each time it is run.
3. If a program is useful, it will have to be changed.
4. If a program is useless, it will have to be documented.
5. Any given program will expand to fill all the available memory.

6. The value of a program is inversely proportional to the weight of its output.
7. Program complexity grows until it exceeds the capability of the programmer who must maintain it.

Pierce's Law

In any computer system, the machine will always misinterpret, misconstrue, misprint, or not evaluate any math or subroutines or fail to print any output on at least the first run through.

Corollary to Pierce's Law

When a compiler accepts a program without error on the first run, the program will not yield the desired output.

Addition to Murphy's Laws

In nature, nothing is ever right. Therefore, if everything is going right ... something is wrong.

Brook's Law

If at first you don't succeed, transform your data set!

Grosch's Law

Computing power increases as the square of the cost.

Golub's Laws of Computerdom

1. Fuzzy project objectives are used to avoid embarrassment of estimating the corresponding costs.
2. A carelessly planned project takes three times longer to complete than expected; a carefully planned project takes only twice as long.
3. The effort required to correct course increases geometrically with time.
4. Project teams detest weekly progress reporting because it so vividly manifests their lack of progress.

Osborn's Law

Variables won't; constants aren't.

Gilb's Laws of Unreliability

1. Computers are unreliable, but humans are even more unreliable.
2. Any system that depends upon human reliability is unreliable.

3. Undetectable errors are infinite in variety, in contrast to detectable errors, which by definition are limited.
4. Investment in reliability will increase until it exceeds the probable cost of errors, or until someone insists on getting some useful work done.

Lubarsky's Law of Cybernetic Entomology

There's always one more bug.

Troutman's Postulate

1. Profanity is the one language understood by all programmers.
2. Not until a program has been in production for six months will the most harmful error be discovered.
3. Job control cards that positively cannot be arranged in improper order will be.
4. Interchangeable tapes won't.
5. If the input editor has been designed to reject all bad input, an ingenious idiot will discover a method to get bad data past it.
6. If a test installation functions perfectly, all subsequent systems will malfunction.

Weinberg's Second Law

If builders built buildings the way programmers wrote programs, then the first woodpecker that came along would destroy civilization.

Gumperson's Law

The probability of anything happening is in inverse ratio to its desirability.

Gummidge's Law

The amount of expertise varies in inverse ratio to the number of statements understood by the general public.

Zymurgy's First Law of Evolving System Dynamics

Once you open a can of worms, the only way to recan them is to use a larger can (old worms never die, they just worm their way into larger cans).

Harvard's Law, as Applied to Computers

Under the most rigorously controlled conditions of pressure, temperature, volume, humidity and other variables, the computer will do as it damn well pleases.

Sattinger's Law

It works better if you plug it in.

Jenkinson's Law

It won't work.

Horner's Five Thumb Postulate

Experience varies directly with equipment ruined.

Cheop's Law

Nothing ever gets build on schedule or within budget.

Rule of Accuracy

When working toward the solution of a problem, it always helps if you know the answer.

Zymurg's Seventh Exception to Murphy's Law

When it rains, it pours.

Pudder's Laws

1. Anything that begins well ends badly.
2. Anything that begins badly ends worse.

Westheimer's Rule

To estimate the time it takes to do a task: estimate the time you think it should take, multiply by two and change the unit of measure to the next highest unit. Thus, we allocate two days for a one hour task.

Stockmayer's Theorem

If it looks easy, it's tough. If it looks tough, it's damn near impossible.

Atwoods Corollary

No books are lost by lending except those you particularly wanted to keep.

Johnson's Third Law

If you miss one issue of any magazine, it will be the issue that contains the article, story or installment you were most anxious to read.

Corollary to Johnson's Third Law

All of your friends either missed it, lost it or threw it out.

Harper's Magazine Law

You never find the article until you replace it.

Brooke's Law

Adding manpower to a late software makes it later.

Finagle's Fourth Law

Once a job is fooled up, anything done to improve it will only make it worse.

Featherkile's Rule

Whatever you did, that's what you planned.

Flap's Law

Any inanimate object, regardless of its position, configuration or purpose, may be expected to perform at any time in a totally unexpected manner for reasons that are either entirely obscure or else completely mysterious.

Q: How do I make a virtual table (in memory)?

A:

```
{

 This is an InMemoryTable example. Free for anyone to use, modify and do
 whatever else you wish.

 Just like all things free it comes with no guarantees. I cannot be
responsible
 for any damage this code may cause. Let me repeat this:

 WARNING! THIS CODE IS PROVIDED AS IS WITH NO GUARANTEES OF ANY KIND!
 USE THIS AT YOUR OWN RISK - YOU ARE THE ONLY PERSON RESPONSIBLE FOR
 ANY DAMAGE THIS CODE MAY CAUSE - YOU HAVE BEEN WARNED!

 THANKS to Steve Garland <72700.2407@compuserve.com> for his help. He
 created his own variation of an in-memory table component and I used it
 to get started.

 InMemory tables are a feature of the Borland Database Engine (BDE).
 InMemory tables are created in RAM and deleted when you close them. They
 are much faster and are very useful when you need fast operations on small
 tables. This example uses the DbiCreateInMemoryTable DBE function call.

 This object should work just like a regular table, except InMemory tables
 do not support certain features (like referntial integrity, secondary
 indexes and BLOBs) and currently this code doesn't do anything to
 prevent you from trying to use them. You will probably get some error if
 you try to create a memo field.

 If you have comments - please contact me at INTERNET:grisha@mira.com

 Happy hacking!

 Gregory Trubetskoy
 http://www.mira.com/home/grisha

}

unit Inmem;

interface

uses DBTables, WinTypes, WinProcs, DBITypes, DBIProcs, DB, SysUtils;


type TInMemoryTable = class(TTable)
  private
    hCursor: hDBICur;
    procedure EncodeFieldDesc(var FieldDesc: FLDDesc;
      const Name: string; DataType: TFieldType; Size: Word);
    function CreateHandle: HDBICur; override;
  public
    procedure CreateTable;
```

```
  end;

implementation

{ luckely this function is virtual - so I could override it. In the original
  VCL code for TTable this function actually opens the table - but since
  we already have the handle to the table - we just return it }

function TInMemoryTable.CreateHandle;
begin
  Result := hCursor;
end;

{ This function is cut-and-pasted from the VCL source code. I had to do this
  because it is declared private in the TTable component so I had no access
  to it from here. }

procedure TInMemoryTable.EncodeFieldDesc(var FieldDesc: FLDDesc;
  const Name: string; DataType: TFieldType; Size: Word);
const
  TypeMap: array[TFieldType] of Byte = (
    fldUNKNOWN, fldZSTRING, fldINT16, fldINT32, fldUINT16, fldBOOL,
    fldFLOAT, fldFLOAT, fldBCD, fldDATE, fldTIME, fldTIMESTAMP, fldBYTES,
    fldVARBYTES, fldBLOB, fldBLOB, fldBLOB);
begin
  with FieldDesc do
  begin
    AnsiToNative(Locale, Name, szName, SizeOf(szName) - 1);
    iFldType := TypeMap[DataType];
    case DataType of
      ftString, ftBytes, ftVarBytes, ftBlob, ftMemo, ftGraphic:
        iUnits1 := Size;
      ftBCD:
        begin
          iUnits1 := 32;
          iUnits2 := Size;
        end;
    end;
    case DataType of
      ftCurrency:
        iSubType := fldstMONEY;
      ftBlob:
        iSubType := fldstBINARY;
      ftMemo:
        iSubType := fldstMEMO;
      ftGraphic:
        iSubType := fldstGRAPHIC;
    end;
  end;
end;

{ This is where all the fun happens. I copied this function from the VCL
source
  and then changed it to use DbiCreateInMemoryTable instead of
DbiCreateTable.
  Since InMemory tables do not support Indexes - I took all of the index-
related
```

```
  things out }

procedure TInMemoryTable.CreateTable;
var
  I: Integer;
  pFieldDesc: pFLDDesc;
  szTblName: DBITBLNAME;
  iFields: Word;
  Dogs: pfldDesc;
begin
  CheckInactive;
  if FieldDefs.Count = 0 then
    for I := 0 to FieldCount - 1 do
      with Fields[I] do
        if not Calculated then
          FieldDefs.Add(FieldName, DataType, Size, Required);
  pFieldDesc := nil;
  SetDBFlag(dbfTable, True);
  try
    AnsiToNative(Locale, TableName, szTblName, SizeOf(szTblName) - 1);
    iFields := FieldDefs.Count;
    pFieldDesc := AllocMem(iFields * SizeOf(FLDDesc));
    for I := 0 to FieldDefs.Count - 1 do
      with FieldDefs[I] do
      begin
        EncodeFieldDesc(PFieldDescList(pFieldDesc)^[I], Name,
          DataType, Size);
      end;
      { the driver type is nil = logical fields }
      Check(DbiTranslateRecordStructure(nil, iFields, pFieldDesc,
        nil, nil, pFieldDesc));
      { here we go - this is where hCursor gets its value }
    Check(DbiCreateInMemTable(DBHandle, szTblName, iFields, pFieldDesc,
hCursor));
  finally
    if pFieldDesc <> nil then FreeMem(pFieldDesc, iFields * SizeOf(FLDDesc));
    SetDBFlag(dbfTable, False);
  end;
end;

end.
```

Q: How do I get the julian date?

A:

```
unit JDates;

{ A unit providing Julian day numbers and date manipulations.

  NOTE:
   The range of Dates this unit will handle is 1/1/1900 to 1/1/2078

  Version 1.00 - 10/26/1987 - First general release

  Scott Bussinger
  Professional Practice Systems
  110 South 131st Street
  Tacoma, WA  98444
  (206)531-8944
  Compuserve 72247,2671

  Version 1.01 - 10/09/1995 - Updated for use with Delphi v1.0
                  Lets see some other code last this long without change

  Dennis Passmore
  1929 Mango Tree Drive
  Edgewater Fl, 32141
  Compuserve 71240,2464 }

interface
uses
  Sysutils;

const
  BlankDate = $FFFF;                          { Constant for Not-a-real-Date }

type TDate = Word;
     TDay = (Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday);
     TDaySet = set of TDay;

procedure GetDate(var Year,Month,Day,Wday: Word);
  { replacement for old WINDOS proc }

procedure GetTime(var Hour,Min,Sec,MSec: Word);
  { replacement for old WINDOS proc }

function  CurrentJDate: Tdate;

function  ValidDate(Day,Month,Year: Word): boolean;
  { Check if the day,month,year is a real date storable in a Date variable }

procedure DMYtoDate(Day,Month,Year: Word;var Julian: TDate);
  { Convert from day,month,year to a date }

procedure DateToDMY(Julian: TDate;var Day,Month,Year: Word);
  { Convert from a date to day,month,year }
```

```
function BumpDate(Julian: TDate;Days,Months,Years: Integer): TDate;
  { Add (or subtract) the number of days, months, and years to a date }

function DayOfWeek(Julian: TDate): TDay;
  { Return the day of the week for the date }

function DayString(WeekDay: TDay): string;
  { Return a string version of a day of the week }

function MonthString(Month: Word): string;
  { Return a string version of a month }

function DateToStr(Julian: TDate): string;
  { Convert a date to a sortable string }

function StrToDate(StrVar: string): TDate;
  { Convert a sortable string form to a date }

implementation

procedure GetDate(var Year,Month,Day,Wday: Word);
var
  td: TDatetime;
begin
  td := Date;
  DeCodeDate(td,Year,Month,Day);
  Wday := sysutils.DayofWeek(td);
end;

procedure GetTime(var Hour,Min,Sec,MSec: Word);
var
  td: TDatetime;
begin
  td := Now;
  DecodeTime(td,Hour,Min,Sec,MSec);
end;

function  CurrentJdate: Tdate;
var
 y,m,d,w: word;
 jd: TDate;
begin
  GetDate(y,m,d,w);
  DMYtoDate(d,m,y,jd);
  CurrentJDate:= jd;
end;

function ValidDate(Day,Month,Year: Word): boolean;
  { Check if the day,month,year is a real date storable in a Date variable }
begin
  if {(Day<1) or }(Year<1900) or (Year>2078) then
    ValidDate := false
  else
    case Month of
      1,3,5,7,8,10,12: ValidDate := Day <= 31;
      4,6,9,11: ValidDate := Day <= 30;
      2: ValidDate := Day <= 28 + ord((Year mod 4)=0)*ord(Year<>1900)
```

```
      else ValidDate := false
    end
end;

procedure DMYtoDate(Day,Month,Year: Word;var Julian: TDate);
  { Convert from day,month,year to a date }
  { Stored as number of days since January 1, 1900 }
  { Note that no error checking takes place in this routine -- use
ValidDate }
begin
if (Year=1900) and (Month<3) then
  if Month = 1 then
    Julian := pred(Day)
  else
    Julian := Day + 30
else
  begin
    if Month > 2 then
      dec(Month,3)
    else
      begin
        inc(Month,9);
        dec(Year)
      end;
    dec(Year,1900);
    Julian := (1461*longint(Year) div 4) + ((153*Month+2) div 5) + Day + 58;
  end
end;

procedure DateToDMY(Julian: TDate;var Day,Month,Year: Word);
  { Convert from a date to day,month,year }
var
  LongTemp: longint;
      Temp: Word;
begin
  if Julian <= 58 then
    begin
      Year := 1900;
      if Julian <= 30 then
        begin
          Month := 1;
          Day := succ(Julian)
        end
      else
        begin
          Month := 2;
          Day := Julian - 30
        end
    end
  else
    begin
      LongTemp := 4*longint(Julian) - 233;
      Year := LongTemp div 1461;
      Temp := LongTemp mod 1461 div 4 * 5 + 2;
      Month := Temp div 153;
      Day := Temp mod 153 div 5 + 1;
      inc(Year,1900);
```

```
      if Month < 10 then
        inc(Month,3)
      else
        begin
          dec(Month,9);
          inc(Year)
        end
    end
end;

function BumpDate(Julian: TDate;Days,Months,Years: Integer): TDate;
  { Add (or subtract) the number of days, months, and years to a date }
  { Note that months and years are added first before days }
  { Note further that there are no overflow/underflow checks }
var Day: Word;
    Month: Word;
    Year: Word;
begin
  DateToDMY(Julian,Day,Month,Year);
  Month := Month + Months - 1;
  Year := Year + Years + (Month div 12) - ord(Month<0);
  Month := (Month + 12000) mod 12 + 1;
  DMYtoDate(Day,Month,Year,Julian);
  BumpDate := Julian + Days
end;

function DayOfWeek(Julian: TDate): TDay;
  { Return the day of the week for the date }
begin
  DayOfWeek := TDay(succ(Julian) mod 7)
end;

function DayString(WeekDay: TDay): string;
  { Return a string version of a day of the week }
const DayStr: array[Sunday..Saturday] of string[9] =

('Sunday','Monday','Tuesday','Wednesday','Thursday','Friday','Saturday');
begin
  DayString := DayStr[WeekDay]
end;

function MonthString(Month: Word): string;
  { Return a string version of a month }
  const MonthStr: array[1..12] of string[9] =
     ('January','February','March','April','May','June','July','August',

'September','October','November','December');
begin
  MonthString := MonthStr[Month]
end;

function DateToStr(Julian: TDate): string;
  { Convert a date to a sortable string - NOT displayable }
const tResult: record
                 case integer of
                   0: (Len: byte;  W: word);
                   1: (Str: string[2])
```

```
                end = (Str:'  ');
begin
  tResult.W := swap(Julian);
  DateToStr := tResult.Str
end;

function StrToDate(StrVar: string): TDate;
  { Convert a sortable string form to a date }
var Temp: record
            Len: byte;
              W: word
          end absolute StrVar;
begin
  StrToDate := swap(Temp.W)
end;

end.
```

Q: How do I pack a Paradox table?

A:

```
{ METHOD: PackParadoxTable
  PURPOSE: Pack the currently opened paradox table.
}
procedure TTableEnhanced.PackParadoxTable;
var
  { Specific information about the table structure, indexes, etc. }
  TblDesc: CRTblDesc;
  { Uses as a handle to the database }
  hDb: hDbiDb;
  { Path to the currently opened table }
  TablePath: array[0..dbiMaxPathLen] of char;

begin
  hDb := nil;
  { Initialize the table descriptor }
  FillChar(TblDesc, SizeOf(CRTblDesc), 0);
  with TblDesc do
  begin
    { Place the table name in descriptor }
    StrPCopy(szTblName, TableName);
    { Place the table type in descriptor }
    StrCopy(szTblType, GetTableType);
    { Set the packing option to true }
    bPack := True;
  end;
  { Get the current table's directory.  This is why the table MUST be
    opened until now }
  Chk(DbiGetDirectory(DBHandle, True, TablePath));
  { Close the table }
  Close;
  { NOW: since the DbiDoRestructure call needs a valid DB handle BUT the
    table cannot be opened, call DbiOpenDatabase to get a valid handle.
    Setting TTable.Active = FALSE does not give you a valid handle }
  Chk(DbiOpenDatabase(nil, 'STANDARD', dbiReadWrite, dbiOpenExcl, nil,
                      0, nil, nil, hDb));
  { Set the table's directory to the old directory }
  Chk(DbiSetDirectory(hDb, TablePath));
  { Pack the PARADOX table }
  Chk(DbiDoRestructure(hDb, 1, @TblDesc, nil, nil, nil, FALSE));
  { Close the temporary database handle }
  Chk(DbiCloseDatabase(hDb));
  { Re-Open the table }
  Open;
end;
```

Error Message: connection is in use by another statement

Resolution: (from a compuserve thread)

The 'connection is in use by another statement' message is due to a limitation within Sybase's DB-Library. Basically, DB-Library won't allow you to initiate a new query over a given connection while results are still pending from a previous one. A DB-Library app that fails to respect this receives Sybase's infamous 'attempt to initiate query with results pending' message. You're seeing the Delphi equivalent.

The answer to this is to either ensure that all results are read when a query is initiated, or to open separate physical connections into the server for each query, as you've done. Usually, the latter is unnecessary. To avoid it, just issue a call to the Last method after you open a TQuery or TStoredProc. You can then issue a call to First to return to the top of the result set with no adverse effects. This is usually a workable solution for small to medium sized result sets.

Thanks for jumping in here. I do recall having a similar problem with VB and SQL Server, so I looked it up in MS DevNet and found tech report Q119023 which echoes what you have said.

However, in addition to using Last to flush each query, I'm seeing this problem occur with two executable stored procs (where I can't use last). Here's the scenario:

A) StoredProc1 inserts an audit trail record into a table. It is called with ExecProc. ExecProc fails because of an error during insert (a column defined as not null is null).

B) StoredProc2 inserts an audit trail into a different table. It is called after StoredProc1 (whether is fails or not). When StoredProc2.ExecProc executes, BAM! "connection is in use by another statement". When StoredProc1 is successful, StoredProc2 does *not* produce the error.

Forcing a StoredProc1.Close doesn't help, but forcing a StoredProc1.UnPrepare *does* prevent the "connection is in use..." error.

Q: How do I to one character type-ahead in a TDBGrid?

A: Note: Be sure that the table's selected index is the one that you are doing the type-ahead with. Also, be aware that this code will prevent you from editing directly into the TDBGrid. You may want to add a flag that lets you pick and choose between type-ahead and editing.

```
procedure TForm1.DBGrid1KeyPress(Sender: TObject; var Key: Char);
var s: string;
begin
  with table1 do
  begin
    s := FieldByName('company').AsString;
    if uppercase(s[1]) = uppercase(key) then next
    else FindNearest([key]);
  end;
  key := #0; {Gets rid of the beep.}
end;
```

Best file compression around: "DEL *.*" = 100% compression

The Definition of an Upgrade: Take old bugs out, put new ones in.

BREAKFAST.COM Halted...Cereal Port Not Responding

The name is Baud....., James Baud.

BUFFERS=20 FILES=15 2nd down, 4th quarter, 5 yards to go!

Access denied--nah nah na nah nah!

C:\ Bad command or file name! Go stand in the corner.

Bad command. Bad, bad command! Sit! Stay! Staaay..

Why doesn't DOS ever say "EXCELLENT command or filename!"

As a computer, I find your faith in technology amusing.

Backups? We don' *NEED* no steenking backups.

E Pluribus Modem

... File not found. Should I fake it? (Y/N)

Ethernet (n): something used to catch the etherbunny

A mainframe: The biggest PC peripheral available.

An error? Impossible! My modem is error correcting.

CONGRESS.SYS Corrupted: Re-boot Washington D.C (Y/n)?

Does fuzzy logic tickle?

A computer's attention span is as long as it's power cord.

11th commandment - Covet not thy neighbor's Pentium.

24 hours in a day...24 beers in a case...coincidence?

Disinformation is not as good as datinformation.

Windows: Just another pane in the glass.

SENILE.COM found . . . Out Of Memory . . .

Who's General Failure & why's he reading my disk?

Ultimate office automation: networked coffee.

RAM disk is *not* an installation procedure.

Shell to DOS...Come in DOS, do you copy? Shell to DOS...

All computers wait at the same speed.

DEFINITION: Computer - A device designed to speed and automate errors.

Press <CTRL-<ALT-<DEL to continue ...

Smash forehead on keyboard to continue.....

Enter any 11-digit prime number to continue...

ASCII stupid question, get a stupid ANSI!

E-mail returned to sender -- insufficient voltage.

Help! I'm modeming... and I can't hang up!!!

All wiyht. Rho sritched mg kegtops awound?

Error: Keyboard not attached. Press F1 to continue.

"640K ought to be enough for anybody." - Bill Gates, 1981

DOS Tip #17: Add DEVICE=FNGRCROS.SYS to CONFIG.SYS

Hidden DOS secret: add BUGS=OFF to your CONFIG.SYS

Press any key... no, no, no, NOT THAT ONE!

Press any key to continue or any other key to quit...

Excuse me for butting in, but I'm interrupt-driven.

REALITY.SYS corrupted: Reboot universe? (Y/N/Q)

Sped up my XT; ran it on 220v! Works greO?_~"

Error reading FAT record: Try the SKINNY one? (Y/N)

Read my chips: No new upgrades!

Hit any user to continue.

2400 Baud makes you want to get out and push!!

I hit the CTRL key but I'm still not in control!

Will the information superhighway have any rest stops?

Disk Full - Press F1 to belch.

Backup not found: (A)bort (R)etry (T)hrowup

Backup not found: (A)bort (R)etry (P)anic

(A)bort, (R)etry, (T)ake down entire network?

(A)bort, (R)etry, (G)et a beer?

If debugging is the process of removing bugs, then programming must be the process of putting them in.

Programmers don't die, they just GOSUB without RETURN.

Programmer - A red-eyed, mumbling mammal capable of conversing with inanimate objects.

Beware of programmers who carry screwdrivers.

Relax, its only ONES and ZEROS!

GetMinMax

```
unit Msg;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private declarations }
    procedure WMGetMinMaxInfo(var MSG: Tmessage); message WM_GetMinMaxInfo;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMGetMinMaxInfo(var MSG: Tmessage);
Begin
  inherited;
  with PMinMaxInfo(MSG.lparam)^ do
  begin
    with ptMinTrackSize do
    begin
      X := 300;
      Y := 150;
    end;
    with ptMaxTrackSize do
    begin
      X := 350;
      Y := 250;
    end;
  end;
end;

end.
```

Q: How do I make characters in a string in a TStringGrid different colors?

A:

```
unit Strgr;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Grids, StdCtrls, DB;

type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
    procedure StringGrid1DrawCell(Sender: TObject; Col, Row: Longint;
      Rect: TRect; State: TGridDrawState);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.StringGrid1DrawCell(Sender: TObject; Col, Row: Longint;
  Rect: TRect; State: TGridDrawState);
const
  CharOffset = 3; {Keep away from the edge of the rect.}
begin
  with StringGrid1.canvas do
  begin
    font.color := clMaroon;
    textout(rect.left + CharOffset, rect.top + CharOffset, 'L');
    font.color := clNavy;
    textout(rect.left + CharOffset + TextWidth('L'), rect.top + CharOffset,
'loyd');
  end;
end;

end.
```

"Memo of the Month," From The Washington Monthly, January/February 1991, page 24:

"This is an actual alert to IBM Field Engineers that went out to all IBM Branch Offices. The person who wrote it was very serious. The rest of us find it rather funny.

"Abstract: Mouse Balls Available as FRU (Field Replacement Unit)

"Mouse balls are now available as FRU. Therefore, if a mouse fails to operate or should it perform erratically, it may need a ball replacement. Because of the delicate nature of this procedure, replacement of mouse
balls should only be attempted by properly trained personnel.

"Before proceeding, determine the type of mouse balls by examining the underside of the mouse. Domestic balls will be larger and harder than foreign balls. Ball removal procedures differ depending upon manufacturer
of the mouse. Foreign balls can be replaced using the pop-off method. Domestic balls are replaced using the twist-off method. Mouse balls are not usually static sensitive. However, excessive handling can result in
sudden discharge. Upon completion of ball replacement, the mouse may be used immediately.

"It is recommended that each replacer have a pair of spare balls for maintaining optimum customer satisfaction, and that any customer missing his balls should suspect local personnel of removing these necessary items.

"To re-order, specify one of the following:

"P/N 33F8462 - Domestic Mouse Balls
"P/N 33F8461 - Foreign Mouse Balls"

Q: How do I drag and move components in runtime?

A: This will work for a TPanel.

```
procedure TForm1.Panel1MouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
const
  SC_DragMove = $F012;
begin
  ReleaseCapture;
  (sender as TWinControl).perform(WM_SysCommand, SC_DragMove, 0);
end;
```

Q: Is there a way to get the integer value of a month given only the long form of a month name?

A: Yes. You can loop throught the global array called LongMonthNames:

```
function GetMonthNumber (month: string): integer;
begin
  for result := 1 to 12 do
    if month = LongMonthNames[result] then exit;
  result := 0;
end;
```

Q: How do I translate this 'C' delaration to ObjectPascal?

```
typedef void (_far _pascal * REMOTEPROC)(void);
```

A:

```
type REMOTEPROC = procedure;
```

The "far" is assumed for procedure variables. The "pascal" indicates the order in which arguments are placed on the stack (somewhat immaterial when there are no arguments).

Q: How do I print a bitmap to a specific size (i.e. stretch to fit)?

A: Here is the unit from one that I did:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, printers, ExtCtrls, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Image1: TImage;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  bmp: TBitmap;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var r: TRect;
begin
  with r do
  begin
    left := 0;
    top := 0;
    right := 2000;
    bottom := 2000;
  end;

  printer.begindoc;
  printer.canvas.stretchDraw(r, image1.picture.graphic);
  printer.EndDoc;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  bmp := TBitmap.create;
  bmp.width := image1.width;
  bmp.height := image1.height;
  image1.picture.graphic := bmp;
end;
```

```
procedure TForm1.FormActivate(Sender: TObject);
begin
  with image1.canvas do
  begin
    brush.color := clMaroon;
    pen.width := 5;
    pen.color := clNavy;
    rectangle(10, 10, 150, 150);
  end;
end;

end.
```

Q: How do I calculate the width of a TDBGrid so that it will display all the fields?

A: I've got some code that calculates the width of a TDBGrid so that it will display all the fields, but no horizontal scrollbar. It takes into account the width of the vertical scrollbar (which changes based on the screen resolution) and the indicator. If you're not working with a DBGrid, perhaps something in here will be helpful for the other types of grids.

```
function NewTextWidth(fntFont : TFont; const sString : OpenString) :
  integer;
var
  fntSave : TFont;
begin
  result := 0;
  fntSave := Application.MainForm.Font;
  Application.MainForm.Font := fntFont;
  try
    result := Application.MainForm.Canvas.TextWidth(sString);
  finally
    Application.MainForm.Font := fntSave;
  end;
end;



{ calculate the width of the grid needed to exactly display with no   }
{ horizontal scrollbar and with no extra space between the last       }
{ column and the vertical scrollbar.  The grid's datasource must be   }
{ properly set and the datasource's dataset must be properly set,     }
{ though it need not be open.  Note:  this width includes the width   }
{ of the vertical scrollbar, which changes based on screen            }
{ resolution.  These changes are compensated for.                     }

function iCalcGridWidth
  (
  dbg : TDBGrid { the grid to meaure }
  )
  : integer; { the "exact" width }

const
  cMEASURE_CHAR   = '0';
  iEXTRA_COL_PIX  = 4;
  iINDICATOR_WIDE = 11;
var
  i, iColumns, iColWidth, iTitleWidth, iCharWidth : integer;
begin
  iColumns := 0;
  result := GetSystemMetrics(SM_CXVSCROLL);
  iCharWidth := NewTextWidth(dbg.Font, cMEASURE_CHAR);
  with dbg.dataSource.dataSet do
    for i := 0 to FieldCount - 1 do with Fields[i] do
      if visible then
      begin
```

```
        iColWidth := iCharWidth * DisplayWidth;
        if dgTitles in dbg.Options then
        begin
          iTitleWidth := NewTextWidth(dbg.TitleFont, DisplayLabel);
          if iColWidth < iTitleWidth then iColWidth := iTitleWidth;
        end;
        inc(iColumns, 1);
        inc(result, iColWidth + iEXTRA_COL_PIX);
      end;
  if dgIndicator in dbg.Options then
  begin
    inc(iColumns, 1);
    inc(result, iINDICATOR_WIDE);
  end;
  if dgColLines in dbg.Options
    then inc(result, iColumns)
    else inc(result, 1);
end;
```

I had to use the function NewTextWidth, rather than the Grid's Canvas.TextWith as the Canvas of the Grid may not initialized when you need to call iCalcGridWidth.

Q: How do I use a resource (.RES) file?

A:

In all cases the RES file must be included in the program's build by adding a line like this:

```
{$R c:\programs\delphi\MyFile.res}
```

cursor loading:

```
const PutTheCursorHere_Dude = 1; {arbitrary number}

procedure stuff;
begin
  screen.cursors[PutTheCursorHere_Dude] := LoadCursor(hInstance,
pChar('cursor_1'));
  screen.cursor := PutTheCursorHere_Dude;
end;
```

Bitmap loading:

```
with bitbtn1 do
  begin
    glyph.handle := LoadBitmap(hInstance, 'bitmap_2');
    refresh;
  end;
```

String Lists: The strings are referenced by number like this:

```
listbox1.items.add(LoadStr(101));
```

Note: You may want to have constants that will replace the hard-coded values and make the code mroe readable.

The Evolution of a Programmer

High School/Jr.High

==========

```
10 PRINT "HELLO WORLD"
20 END
```

First year in College

==========

```
  program Hello(input, output)
    begin
      writeln('Hello World')
    end.
```

Senior year in College

==========

```
  (defun hello
    (print
      (cons 'Hello (list 'World))))
```

New professional

==========

```
#include <stdio.h>
void main(void)
{
    char *message[] = {"Hello ", "World"};
    int i;
    for(i = 0; i < 2; ++i)
        printf("%s", message[i]);
        printf("\n");
    }
```

Seasoned professional

==========

```
#include <iostream.h>
#include <string.h>

class string
{
private:
  int size;
  char *ptr;
public:
   string() : size(0), ptr(new char('\0')) {}
   string(const string &s) : size(s.size)
   {
         ptr = new char[size + 1];
         strcpy(ptr, s.ptr);
   }
   ~string()
    {
      delete [] ptr;
    }
```

```
    friend ostream &operator <<(ostream &, const string &);

   string &operator=(const char *);
};

ostream &operator<<(ostream &stream, const string &s)
{
     return(stream << s.ptr);
}

string &string::operator=(const char *chrs)
{
    if (this != &chrs)
    {
      delete [] ptr;
      size = strlen(chrs);
      ptr = new char[size + 1];
      strcpy(ptr, chrs);
    }
    return(*this);
}

int main()
{
    string str;
    str = "Hello World";
    cout << str << endl;
    return(0);
}
```

Master Programmer

```
==================
  [
  uuid(2573F8F4-CFEE-101A-9A9F-00AA00342820)
  ]
  library LHello
  {
    // bring in the master library
      importlib("actimp.tlb");
      importlib("actexp.tlb");
            // bring in my interfaces
      #include "pshlo.idl"
     [
      uuid(2573F8F5-CFEE-101A-9A9F-00AA00342820)
     ]

   cotype THello
   {
     interface IHello;
     interface IPersistFile;
   };
};
  [
  exe,
  uuid(2573F890-CFEE-101A-9A9F-00AA00342820)
  ]
```

```
  module CHelloLib
  {
      // some code related header files
     importheader(<windows.h>);
     importheader(<ole2.h>);
     importheader(<except.hxx>);
     importheader("pshlo.h");
     importheader("shlo.hxx");
     importheader("mycls.hxx");
      // needed typelibs
    importlib("actimp.tlb");
    importlib("actexp.tlb");
    importlib("thlo.tlb");
     [
      uuid(2573F891-CFEE-101A-9A9F-00AA00342820),
      aggregatable
     ]
     coclass CHello
     {
         cotype THello;
     };
};

  #include "ipfix.hxx"
  extern HANDLE hEvent;
  class CHello : public CHelloBase
  {
  public:
     IPFIX(CLSID_CHello);
     CHello(IUnknown *pUnk);
     ~CHello();
     HRESULT  __stdcall PrintSz(LPWSTR             pwszString);
  private:
     static int cObjRef;
};

#include <windows.h>
#include <ole2.h>
#include <stdio.h>
#include <stdlib.h>
#include "thlo.h"
#include "pshlo.h"
#include "shlo.hxx"
#include "mycls.hxx"
  int CHello::cObjRef = 0;
  CHello::CHello(IUnknown *pUnk) : CHelloBase(pUnk)
{
    cObjRef++;
    return;
}
HRESULT  __stdcall  CHello::PrintSz(LPWSTR pwszString)
{
    printf("%ws\n", pwszString);
    return(ResultFromScode(S_OK));
}
CHello::~CHello(void)
{
```

```
  // when the object count goes to zero, stop the server
  cObjRef--;
  if( cObjRef == 0 )
      PulseEvent(hEvent);
  return;
}


#include <windows.h>
#include <ole2.h>
#include "pshlo.h"
#include "shlo.hxx"
#include "mycls.hxx"
  HANDLE hEvent;
  int _cdecl main(
  int argc,
  char * argv[])
{
  ULONG ulRef;
  DWORD dwRegistration;
  CHelloCF *pCF = new CHelloCF();
  hEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
  // Initialize the OLE libraries
  CoInitializeEx(NULL, COINIT_MULTITHREADED);
  CoRegisterClassObject(CLSID_CHello, pCF, CLSCTX_LOCAL_SERVER,
      REGCLS_MULTIPLEUSE, &dwRegistration);
  // wait on an event to stop
  WaitForSingleObject(hEvent, INFINITE);
  // revoke and release the class object
  CoRevokeClassObject(dwRegistration);
  ulRef = pCF->Release();
  // Tell OLE we are going away.
  CoUninitialize();
  return(0);
}

  extern CLSID CLSID_CHello;
  extern UUID LIBID_CHelloLib;
  CLSID CLSID_CHello = { /*  > 573F891-CFEE-101A-9A9F-00AA00342820 */
      0x2573F891,
      0xCFEE,
      0x101A,
      { 0x9A, 0x9F, 0x00, 0xAA, 0x00, 0x34, 0x28, 0x20 }
  };
  UUID LIBID_CHelloLib = { /*  > 573F890-CFEE-101A-9A9F-00AA00342820 */
      0x2573F890,
      0xCFEE,
      0x101A,
     { 0x9A, 0x9F, 0x00, 0xAA, 0x00, 0x34, 0x28, 0x20 }
};

#include <windows.h>
#include <ole2.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "pshlo.h"
```

```
#include "shlo.hxx"
#include "clsid.h"
  int _cdecl main(
  int argc,
  char * argv[]) {
  HRESULT  hRslt;
  IHello        *pHello;
  ULONG  ulCnt;
  IMoniker * pmk;
  WCHAR  wcsT[_MAX_PATH];
  WCHAR  wcsPath[2 * _MAX_PATH];
  // get object path
  wcsPath[0] = '\0';
  wcsT[0] = '\0';
  if( argc > 1) {
     mbstowcs(wcsPath, argv[1], strlen(argv[1]) + 1);
     wcsupr(wcsPath);
     }
  else {
     fprintf(stderr, "Object path must be specified\n");
      return(1);
      }
  // get print string
  if(argc > 2)
      mbstowcs(wcsT, argv[2], strlen(argv[2]) + 1);
  else
     wcscpy(wcsT, L"Hello World");
  printf("Linking to object %ws\n", wcsPath);
  printf("Text String %ws\n", wcsT);
  // Initialize the OLE libraries
  hRslt = CoInitializeEx(NULL, COINIT_MULTITHREADED);
  if(SUCCEEDED(hRslt)) {
      hRslt = CreateFileMoniker(wcsPath, &pmk);
   if(SUCCEEDED(hRslt))
   hRslt = BindMoniker(pmk, 0, IID_IHello, (void **)&pHello);
     if(SUCCEEDED(hRslt)) {
   // print a string out
   pHello->PrintSz(wcsT);
   Sleep(2000);
   ulCnt = pHello->Release();
   }
      else
   printf("Failure to connect, status: %lx", hRslt);
      // Tell OLE we are going away.
      CoUninitialize();
      }
  return(0);
}
```

Apprentice Hacker

```
====================
#!/usr/local/bin/perl
$msg="Hello, world.\n";
if ($#ARGV >= 0) {
while(defined($arg=shift(@ARGV))) {
$outfilename = $arg;
```

```
open(FILE, ">" . $outfilename) || die "Can't write $arg:  > !\n";
print (FILE $msg);
close(FILE) || die "Can't close $arg: $!\n";
    }
  } else {
    print ($msg);
  }
  1;
```

Experienced Hacker

```
#include <stdio.h>
#define S "Hello, World\n"
main(){exit(printf(S) == strlen(S) ? 0 : 1);}
```

Seasoned Hacker

```
% cc -o a.out ~/src/misc/hw/hw.c
% a.out
```

Guru Hacker

```
% cat
Hello, world.
^D
```

New Manager

```
10 PRINT "HELLO WORLD"
20 END
```

Middle Manager

```
mail -s "Hello, world." bob@b12
Bob, could you please write me a program that prints "Hello,  > orld."?
I need it by tomorrow.
^D
```

Senior Manager

```
% zmail jim
I need a "Hello, world." program by this afternoon.
```

Chief Executive

```
% letter
letter: Command not found.
% mail
To: ^X ^F ^C
% help mail
help: Command not found.
% damn!
```

```
!: Event unrecognized
% logout
```

DOS Beer:
Requires you to use your own can opener, and requires you to read the directions carefully before opening the can. Originally only came in an 8-oz. can, but now comes in a 16-oz. can. However, the can is divided into 8 compartments of 2 oz. each, which have to be accessed separately. Soon to be discontinued, although a lot of people are going to keep drinking it after it's no longer available.

Mac Beer:
At first, came only a 16-oz. can, but now comes in a 32-oz. can. Considered by many to be a "light" beer. All the cans look identical. When you take one from the fridge, it opens itself. The ingredients list is not on the can. If you call to ask about the ingredients, you are told that "you don't need to know." A notice on the side reminds you to drag your empties to the trashcan.

Windows 3.1 Beer:
The world's most popular. Comes in a 16-oz. can that looks a lot like Mac Beer's. Requires that you already own a DOS Beer. Claims that it allows you to drink several DOS Beers simultaneously, but in reality you can only drink a few of them, very slowly, especially slowly if you are drinking the Windows Beer at the same time. Sometimes, for apparently no reason, a can of Windows Beer will explode when you open it.

OS/2 Beer:
Comes in a 32-oz can. Does allow you to drink several DOS Beers simultaneously. Allows you to drink Windows 3.1 Beer simultaneously too, but somewhat slower. Advertises that its cans won't explode when you open them, even if you shake them up. You never really see anyone drinking OS/2 Beer, but the manufacturer (International Beer Manufacturing) claims that 9 million six-packs have been sold.

Windows 95 Beer:
You can't buy it yet, but a lot of people have taste-tested it and claim it's wonderful. The can looks a lot like Mac Beer's can, but tastes more like Windows 3.1 Beer. It comes in 32-oz. cans, but when you look inside, the cans only have 16 oz. of beer in them. Most people will probably keep drinking Windows 3.1 Beer until their friends try Windows 95 Beer and say they like it. The ingredients list, when you look at the small print, has some of the same ingredients that come in DOS beer, even though the manufacturer claims that this is an entirely new brew.

Windows NT Beer:
Comes in 32-oz. cans, but you can only buy it by the truckload. This causes most people to have to go out and buy bigger refrigerators. The can looks just like Windows 3.1 Beer's, but the company promises to change the can to look just like Windows 95 Beer's - after Windows 95 beer starts shipping. Touted as an "industrial strength" beer, and suggested only for use in bars.

Unix Beer:
Comes in several different brands, in cans ranging from 8 oz. to 64 oz. Drinkers of

Unix Beer display fierce brand loyalty, even though they claim that all the different brands taste almost identical. Sometimes the pop-tops break off when you try to open them, so you have to have your own can opener around for those occasions, in which case you either need a complete set of instructions, or a friend who has been drinking Unix Beer for several years.

AmigaDOS Beer:
The company has gone out of business, but their recipe has been picked up by some weird German company, so now this beer will be an import. This beer never really sold very well because the original manufacturer didn't understand marketing. Like Unix Beer, AmigaDOS Beer fans are an extremely loyal and loud group. It originally came in a 16-oz. can, but now comes in 32-oz. cans too. When this can was originally introduced, it appeared flashy and colorful, but the design hasn't changed much over the years, so it appears dated now. Critics of this beer claim that it is only meant for watching TV anyway.

VMS Beer:
Requires minimal user interaction, except for popping the top and sipping. However cans have been known on occasion to explode, or contain extremely un-beer-like contents. Best drunk in high pressure development environments. When you call the manufacturer for the list of ingredients, you're told that is proprietary and referred to an unknown listing in the manuals published by the FDA. Rumors are that this was once listed in the Physicians' Desk Reference as a tranquilizer, but no one can claim to have actually seen it.

TBitmap

How do I load a bitmap from a resource without losing the pallette?
How can I erase the picture on image1.canvas?
How do I size a form to fit a bitmap?
How do I fill a graphic field from a BMP file?

Q: How do I load a bitmap from a resource without losing the pallette?

A:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  Bmp: TBitmap;
  HResInfo: THandle;
  BMF: TBitmapFileHeader;
  MemHandle: THandle;
  Stream: TMemoryStream;
  ResPtr: PByte;
  ResSize: Longint;
begin
  BMF.bfType := $4D42;
  HResInfo := FindResource(HInstance, 'BITMAP_2', RT_Bitmap);
  ResSize := SizeofResource(HInstance, HResInfo);
  MemHandle := LoadResource(HInstance, HResInfo);
  ResPtr := LockResource(MemHandle);
  Stream := TMemoryStream.Create;
  Stream.SetSize(ResSize + SizeOf(BMF));
  Stream.Write(BMF, SizeOf(BMF));
  Stream.Write(ResPtr^, ResSize);
  FreeResource(MemHandle);
  Stream.Seek(0, 0);
  Bmp := TBitmap.Create;
  Bmp.LoadFromStream(Stream);
  Canvas.Draw(0, 0, Bmp);
  Bmp.Free;
  Stream.Free;
end;
```

THE ENGINEERS SONG
(SUNG TO THE TUNE OF THE BEVERLY HILLBILLIES)

Come and listen to a story 'bout a man named Jed,
A poor College Kid barely kept his family fed,
But then one day he was talking to a recruiter,
He said "They'll pay ya big bucks if ya work on a computer",
VAX that is ... CRT's ... Workstations;

Well the first thing ya know ol' Jed's an Engineer,
The kinfolk said "Jed move away from here",
They said "Arizona is the place ya oughta be",
So he bought some jelly rolls, and he moved to Ahwatukee,
Motorola that is ... dry heat ... no amusement parks;

On his first day at work they stuck him in a cube,
Fed him more donuts and sat him at a tube,
They said "Your project's late but we know just what to do,
Instead of 40 hours, we'll work you fifty-two!"
OT that is ... Unpaid ... Mandatory

The weeks rolled by and things were looking bad,
Some schedules had slipped and some managers were mad,
They called another meeting and decided on a fix,
They answer was simple, "We'll work him sixty-six"
Tired that is ... Stressed out ... No social life

Months turned into years and his hair was turning gray,
Jed worked hard while his life slipped away,
Waiting to retire when he turned sixty-four,
Instead he got a call and they walked him out the door,
Laid-off that is ... Debriefed ... Unemployed ...

A math/engineering convention was being held. On the train to the convention, there were a bunch of math majors and a bunch of engineering majors. Each of the math majors had his/her train ticket. The group of engineers had only ONE ticket for all of them. The math majors started laughing and snickering.

Then, one of the engineers said "here comes the conductor" and then all of the engineers went into the bathroom. The math majors were puzzled. The conductor came aboard and said "tickets please" and got tickets from all the math majors. He then went to the bathroom and knocked on the door and said "ticket please" and the engineers stuck the ticket under the door. The conductor took it and then the engineers came out of the bathroom a few minutes later. The math majors felt really stupid.

So, on the way back from the convention, the group of math majors had one ticket for the group. They started snickering at the engineers, for the whole group had no tickets amongst them. Then, the engineer lookout said "Conductor coming!". All the engineers went to one bathroom. All the math majors went to another bathroom. Then, before the conductor came on board, one of the engineers left the bathroom, knocked on the other bathroom, and said "ticket please."

From: hashemi@physics.harvard.edu
Newsgroups: comp.lang.pascal.misc
Subject: What Right-Minded Adult Would Use C?
Date: 28 Sep 95 20:14:45 -0500
Organization: Harvard University Physics Department
Message-ID: <1995Sep28.201445.1@physics.harvard.edu>

What Right-Minded Adult Would Use C?

Most C programmers are NARGs (from "Not A Real Gentleman"),which chiefly means that they have no charm, and here's why.

Whereas a gracious adult seeks to make others feel glad about the extent of their knowledge (he is charming), a NARG will take every opportunity to make others feel dismayed about the limits of their knowledge (he is not charming). Computer-NARGs, not satisfied with the power of science to make non-scientists feel stupid, have developed elaborate secret languages and cyphers, the main effect of which is to confuse the layman. The mentality behind cryptic communication is no mystery to most of us. I recall from my childhood the fascination I had with secret codes. My friends and I would pass coded notes to one another in class. Someone finding such a note would, we hoped, wonder, "What fascinating message is contained herein?" and, being unable to read it, would never be disillusioned (unless, of course, she went and begged one of us CODE-WRITERS for a
translation). We got such a thrill out of writing in code that there was no need for us to worry about the content of our messages. It was using the cypher that was fun. In fact, the secret code made everything we wrote seem clever, and all because: OTHER PEOPLE DIDN'T KNOW WHAT WE WERE WRITING ABOUT. Not at all charming, and certainly not the way to win new friends, let alone lovers, in the adult world.

Of course, we hope kids will learn the value of charm in school. But the school system is not perfect, so many kids still grow up to be NARGs. Computer programming, as a young science, is ideal territory for a NARG. The NARG programmer can invent as many cyphers as he likes with which to mask the underlying simplicity of his profession. Non-programmers, of which there are many, are easily dazzled by such cyphers, mistaking the difficulty they present for a difficulty presented by the principles of computer programming. Thus, by writing in cyphers, NARGs promote themselves as masters of an arcane art, when in fact they are merely childlike practitioners of a simple craft. Instead of working to produce masterfully clear and unambiguous programs, they strive to encode as much information as possible into each cryptic symbol, and claim to be striving for
"elegance".

Consider UNIX, the NARG's favorite operating system. It is"powerful" (in NARG-speak) because its syntax is loose. It is "elegant" because instructions can be concatenated on a single line. Its commands are cryptic:

UNIX command	Function
man	help
mv	rename
kill	stop
ls	list

And then there is "C", the NARG's favorite programming language. Wheras a simple Pascal looks like,

```
program example (output);
begin
  write('example');
end.
```

the same program in "C" comes out as,

```
#include <stdio.h>

void main(void)
{
  printf("example");
}
```

C's cryptisism goes far beyond its reserved words. Whilephilosophers and linguists have built up a potent understanding of the syntactic constructs that lend themselves to clear and unambiguous expression, the designers of C have seen fit to throw all such constructs to the wind. For example, in C one finds that an assignment statement such as "let x be 3" can be treated not only as an assignment statement, but also as a number, and, moreover, AT THE SAME TIME. Thus we can say "let y be 'let x be 3'" and mean that 'let
x be 3' should be treated first as a command to do something, and then as a number to be assigned to the variable y. C allows the same object to be classified at the same time into two types that the merely college-educated laymen has been taught are mutually exclusive. Our conviction that something cannot be both a number and a statement is so ingrained that we cannot hope to make sense of the above C statement without help from a (you guessed it) C-PROGRAMMER.

NARGs defend C by saying it is "powerful" because its syntaxis loose ("you can play wonderful tricks with pointers") and "elegant" because it requires less typing ("i++ is

much more elegant than i:=i+1"). C programmers also claim you can do important things in C that you cannot do in an easy-to-read language like Pascal. Invariably in such cases, the C programmer either does not know Pascal ("you can't do objects in Pascal," or, "You can't do hash tables in Pascal"), or is suggesting that one do something uneccesarily and
unspeakably ugly that Pascal deliberately forbids ("I want to pass a function to a procedure by passing a pointer variable rather than a function variable"), or is referring to properties of the compiler ("what about compiler macros?"). C programmers are also fond of expressing fantasies about C being faster than Pascal, but I have yet to see anyone prove this, despite having been subjected to several attempts at a demonstration.

A practical problem with C and UNIX is that programmers, beinghuman, become distracted from their jobs by the joys of encryption and abbreviation. Witness the following entry in a C newsgroup:

|> I think you can use %[^\n]\ns to include spaces
|>
Make that %[^\n]%*c. \n is not a command to read a newline, it reads all whitespace following, and is often inconvenient in this context. %*c reads and discards one character: and as it follows %[^\n], this character, if it exists, is a newline.

What's more, with the rise of cryptisism, the original tenets of goodprogrammming are now openly spurned:

> I don't even care if it's non-portable, or "illegal". As long as it
> does what we what it to do, in my opinion it's good code.

Fortunately, the prevelance of cryptisism is good news forthose of us who stick to a clear language like Pascal, and try to make our code as readable as possible. Clear programming is not an asthetic goal, but a practical one. It leads to reliable code, quicker development, and extended utility; and for this people are prepared to pay. The longer the majority of programmers are distracted by cryptisism, pointer-tricks, and programming fasions, the longer those of us who keep ourselves disciplined can expect to have the advantage. Not only that, we will reap the benifits of NOT BEING NARGS.

The people who really get thrown off course by C are laymenwho want to acquire a basic understanding of programming. It is hard for them to avoid being suckered into learning C as their first language. They spend so much time struggling with it that, once they have learned it, they get the impression that they have learned

something about computers. Sadly, all they have learned about is a stupid-looking cypher composed by NARGs. The fascinating secrets of computer programming remain a mystery to them. AND THAT'S THE WAY THE NARG LIKES IT.

But the reign of the NARG in the computer world will soon drawto a close. Even as we speak, the Internet is being annexed by people from all walks of life. In the alternative newsgroups, NARGs are putting up notices demanding that people, "repect the newsgroup etiquette," by which they mean, "don't make fun of us." Soon, even computer programming will be overwhelmed by non-NARGs, and when that time comes, people will look at C and say,

"What the %[^\n]%* is this?"

So until then, let the NARGs have their fun.
Kevan Hashemihashemi@huhepl.harvard.edu

Here is a favourite obfuscated C example (taken from the Hacker's Jargon file). It is an excellent example of NARG code that I got from the internet.

```
/*
 * Program to compute an approximation of pi
 *  by Brian Westley, 1988
 */

#define _  -F<00||--F-OO--;
int F=00,OO=00;
main(){F_OO();printf("%1.3f\n",4.*-F/OO/OO);}F_OO()
{
              _ _ _
            _ _ _ _
          _ _ _ _ _ _
        _ _ _ _ _ _ _ _
      _ _ _ _ _ _ _ _ _ _
    _ _ _ _ _ _ _ _ _ _ _ _
  _ _ _ _ _ _ _ _ _ _ _ _ _ _
  _ _ _ _ _ _ _ _ _ _ _ _ _ _
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
  _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
   _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
     _ _ _ _ _ _ _ _ _ _ _ _ _
       _ _ _ _ _ _ _ _ _ _ _
         _ _ _ _ _ _ _ _ _
           _ _ _ _ _ _ _
             _ _ _ _ _
}
```

```
{$A+,B-,D+,F-,G+,I+,K+,L+,N+,P+,Q-,R-,S-,T-,V+,W-,X+,Y+}
{$M 16384,8192}
{*****************************************************************************
*}
{
}
{ TPanelClock - a VCL component that is provides time-of-date, NUM, CAPS, and
}
{   Scroll Key Statuses. When you click on this component (at run-time), it
}
{   will switch to showing free GDI, System, and User Resources. Source code
}
{   documentation is rather limited, with the exception of the rather arcane
}
{   properties which as described below. This component (such as it is) is
}
{   hereby given to the public domain. Should you find it useful at some
}
{   point in your programming career, please feel obligated to donate one of
}
{   your own equally useful components to the public domain. If you have any
}
{   suggestions for improvements, or if you find any bugs, please notify the
}
{   author (but please be gentle - this is my first component). Thank-you.
}
{
}
{  Author: Cameron D. Peters
}
{          Suite 311, 908 - 17th Avenue S.W.
}
{          Calgary, Alberta CANADA
}
{          CIS: 72561,3146
}
{          Phone: 403-228-9991
}
{          Fax: 403-228-0202
}
{
}
{  Revision History:
}
{    1.00  CDP  950525  Created
}
{
}
{  Installation
}
{    Use Tools|Install Components to add this to your VCL. TPanelClock will
}
{    be added to the additional page of your component palette.
}
{
}
{  Properties
```

```
}
{    I haven't created an on-line help file for this component, because I
}
{    don't really have the time, or possibly because I am just lazy. Perhaps
}
{    I'll create one if enough people download this file as it is! Anyways,
}
{    here are my notes on the properties which were not inherited (in no
}
{    particular order):
}
{
}
{    PanelMode - can be pmClock or pmResources. When it's pmClock, the
}
{      component shows the time-of-day, and the status of NUM, CAPS, and
}
{      SCRL. When it's pmResources, it will show the percentage of free
}
{      GDI, USER and System Resources.
}
{    AllowClick - when this is true, the user can click on the component
}
{      to switch back and forth between the clock and the resource monitor.
}
{    AlertLevel - if any of the resources fall below this level, they will
}
{      be shown using the AlertFont.
}
{    AlertFont - font used to display resources which have fallen below the
}
{      AlertLevel.
}
{    AlertMatchFont - when this is true, the AlertFont will be made to match
}
{      the Font, with the exception that the color of the AlertFont will be
}
{      set to clRed.
}
{    Spaces - the number of pixels of space between sections of the panel.
}
{    ClockWidth - the width of the clock in pixels.
}
{
}
{*************************************************************************
*}

unit PanClock;

interface

uses
  SysUtils,
  WinTypes,
  WinProcs,
  Messages,
```

```
  Classes,
  Graphics,
  Controls,
  Forms,
  Dialogs,
  ExtCtrls;

const
  {Key statuses}
  ksNumberOfKeyStatuses = 3;
  ksNumLock = 1;
  ksCapsLock = 2;
  ksScrollLock = 4;

  {Resource Monitors}
  rmNumberOfMonitors = 3;
  rmGDIResources = 1;
  rmSystemResources = 2;
  rmUserResources = 3;

type
  TResourceMonitor = array[rmGDIResources..rmUserResources] of integer;
  TPanelMode = (pmClock,pmResources);
  TPanelClock = class(TCustomControl)
  private
    { Private declarations }
    FAlertFont: TFont;
    FAlertLevel: Integer;
    FAlertMatchFont: Boolean;
    FAllowClick: Boolean;
    FBevel: TPanelBevel;
    FBevelWidth: Integer;
    FClockWidth: Integer;
    FHint2: String;
    FKeyState: Integer;
    FLastPaint: String[20];
    FPanelMode: TPanelMode;
    FSpace: Integer;
    FResources: TResourceMonitor;
  protected
    { Protected declarations }
    procedure Click; override;
    procedure Paint; override;
    procedure SetAlertFont(Value: TFont);
    procedure SetAlertLevel(Value: Integer);
    procedure SetAlertMatchFont(Value: Boolean);
    procedure SetBevel(Value: TPanelBevel);
    procedure SetBevelWidth(Value: Integer);
    procedure SetBounds(ALeft, ATop, AWidth, AHeight: Integer); override;
    procedure SetClockWidth(Value: Integer);
    procedure SetPanelMode(Value: TPanelMode);
    procedure SetSpace(Value: Integer);
    procedure WMDestroy(var Msg: TMsg); message WM_Destroy;
    procedure WMCreate(var Msg: TMsg); message WM_Create;
    procedure WMTimer(var Msg: TMsg); message WM_Timer;
  public
    { Public declarations }
```

```
    constructor Create(AOwner: TComponent); override;
  published
    { Published declarations }
    property AlertFont: TFont read FAlertFont write SetAlertFont;
    property AlertLevel: Integer read FAlertLevel write SetAlertLevel default
20;
    property AlertMatchFont: Boolean read FAlertMatchFont write
SetAlertMatchFont default TRUE;
    property Align;
    property AllowClick: Boolean read FAllowClick write FAllowClick default
TRUE;
    property Bevel: TPanelBevel read FBevel write SetBevel default bvLowered;
    property BevelWidth: Integer read FBevelWidth write SetBevelWidth default
1;
    property ClockWidth: Integer read FClockWidth write SetClockWidth default
96;
    property Color;
    property Enabled;
    property Font;
    property Height default 16;
    property Hint;
    property Hint2: String read FHint2 write FHint2;
    property PanelMode: TPanelMode read FPanelMode write SetPanelMode default
pmClock;
    property ParentColor;
    property ParentFont;
    property ParentShowHint;
    property ShowHint;
    property Space: Integer read FSpace write SetSpace default 1;
    property Width default 219;
  end;

procedure Register;

implementation

function IntFindMin(X,Y: Integer): Integer;
begin
  if (X < Y) then Result := X
  else Result := Y;
end;

function IntFindMax(X,Y: Integer): Integer;
begin
  if (X > Y) then Result := X
  else Result := Y;
end;

procedure Register;
begin
  RegisterComponents('Additional', [TPanelClock]);
end;


constructor TPanelClock.Create(AOwner: TComponent);

begin
```

```
  inherited Create(AOwner);
  SetBounds(0,0,219,16);
  Hint := 'Click to see system resources';
  Hint2 := 'Click to see clock';
  FAlertFont := TFont.Create;
  FAlertLevel := 20;
  FAlertMatchFont := TRUE;
  FAllowClick := TRUE;
  FBevel := bvLowered;
  FBevelWidth := 1;
  FClockWidth := 96;
  FSpace := 1;
end;


procedure TPanelClock.Click;

begin
  if (AllowClick) then begin
    if (PanelMode = pmClock) then PanelMode := pmResources
    else PanelMode := pmClock;
  end;
  inherited Click;
end;

procedure TPanelClock.Paint;
var
  ClientRect: TRect;
  StatusRect: TRect;
  TextMetric: TTextMetric;
  TopColor, BottomColor: TColorRef;
  OldColor, SaveFontColor: TColorRef;
  X: Integer;
  RWidth: Integer;

const
  KeyStates: array[1..ksNumberOfKeyStatuses] of String[4] =
('NUM','CAPS','SCRL');
  ResMonitors: array[1..rmNumberOfMonitors] of String[4] =
('GDI:','SYS:','USR:');

{This procedure is inside of the TPanelClock.Paint procedure.}
procedure PaintRect(ARect: TRect; S: String);
var
  X,Y: Integer;
  W,H: Integer;
  FRect: TRect;
begin
  FRect := ARect;
  if (Bevel <> bvNone) then
Frame3D(Canvas,ARect,TopColor,BottomColor,BevelWidth);
  W := Canvas.TextWidth(S);
  WinProcs.GetTextMetrics(Canvas.Handle,TextMetric);
  H := TextMetric.tmHeight;
  X := ARect.Left + IntFindMax((ARect.Right - ARect.Left - W) div 2,1);
  Y := ARect.Top + IntFindMax((ARect.Bottom - ARect.Top - H) div 2,1);
  Canvas.TextRect(ARect,X,Y,S);
```

```
  {Fill up the spacer}
  if (Space > 0) and (FRect.Right + Space <= ClientRect.Right) then begin
    FRect.Left := FRect.Right;
    FRect.Right := FRect.Left + Space;
    Canvas.Brush.Color := Self.Color;
    Canvas.FillRect(FRect);
  end;
end;

begin {TPanelClock.Paint}
  inherited Paint;
  ClientRect := GetClientRect;
  if (Bevel = bvLowered) then begin
    TopColor := clBtnShadow;
    BottomColor := clBtnHighlight;
  end
  else begin
    TopColor := clBtnHighlight;
    BottomColor := clBtnShadow;
  end;

  Canvas.Font := Self.Font;
  FLastPaint := TimeToStr(Time);
  OldColor := SetBkColor(Canvas.Handle,ColorToRGB(Color));
  StatusRect := ClientRect;
  if (PanelMode = pmClock) then begin
    StatusRect.Right := IntFindMin(StatusRect.Right,ClockWidth);
    PaintRect(StatusRect,FLastPaint);
    Inc(StatusRect.Left,ClockWidth+Space);
    RWidth := (ClientRect.Right - StatusRect.Left - (Space *
ksNumberOfKeyStatuses)) div ksNumberOfKeyStatuses;
    for x := 1 to ksNumberOfKeyStatuses do begin
      if (x = ksNumberOfKeyStatuses) then RWidth := ClientRect.Right;
      StatusRect.Right := IntFindMin(StatusRect.Left +
RWidth,ClientRect.Right-Space);
      if (StatusRect.Right - StatusRect.Left > (2*BevelWidth)) then begin
        if (((1 shl Pred(x)) and FKeyState) <> 0) then
PaintRect(StatusRect,KeyStates[x])
        else PaintRect(StatusRect,'');
      end;
      StatusRect.Left := StatusRect.Right + Space;
    end;
  end
  else begin
    if (FAlertMatchFont) then begin
      FAlertFont.Assign(Font);
      FAlertFont.Color := clRed;
    end;

    RWidth := (ClientRect.Right - ClientRect.Left - (Space *
rmNumberOfMonitors)) div rmNumberOfMonitors;
    for x := 1 to rmNumberOfMonitors do begin
      if (x = rmNumberOfMonitors) then RWidth := ClientRect.Right;
      StatusRect.Right := IntFindMin(StatusRect.Left +
RWidth,ClientRect.Right-Space);
      if (FResources[x] < AlertLevel) and (AlertFont <> NIL) then Canvas.Font
```

```
:= AlertFont
      else Canvas.Font := Self.Font;
      PaintRect(StatusRect,ResMonitors[x]+IntToStr(FResources[x])+'%');
      StatusRect.Left := StatusRect.Right + Space;
    end;
  end;
  SetBkColor(Canvas.Handle,OldColor);
end;

procedure TPanelClock.SetAlertFont(Value: TFont);
begin
  FAlertFont.Assign(Value);
  FAlertMatchFont := FALSE;
  Invalidate;
end;

procedure TPanelClock.SetAlertLevel(Value: Integer);
begin
  if (FAlertLevel <> Value) then begin
    FAlertLevel := IntFindMax(IntFindMin(Value,100),0);
    Invalidate;
  end;
end;

procedure TPanelClock.SetAlertMatchFont(Value: Boolean);
begin
  FAlertMatchFont := Value;
  if (Value) then begin
    FAlertFont.Assign(Font);
    FAlertFont.Color := clRed;
    Invalidate;
  end;
end;

procedure TPanelClock.SetBevel(Value: TPanelBevel);
begin
  FBevel := Value;
  Invalidate;
end;

procedure TPanelClock.SetBevelWidth(Value: Integer);
begin
  FBevelWidth := Value;
  Invalidate;
end;

procedure TPanelClock.SetBounds(ALeft, ATop, AWidth, AHeight: Integer);
begin
  inherited SetBounds(ALeft, ATop, IntFindMax(AWidth,ClockWidth), AHeight);
end;


procedure TPanelClock.SetClockWidth(Value: Integer);
begin
  FClockWidth := Value;
  Invalidate;
end;
```

```
procedure TPanelClock.SetPanelMode(Value: TPanelMode);
var
  Msg: TMsg;
  Temp: String;
begin
  FillChar(FResources,SizeOf(FResources),0);
  FLastPaint := '';
  if (FPanelMode <> Value) then begin
    FPanelMode := Value;
    WMTimer(Msg);
    Temp := Hint;
    Hint := Hint2;
    Hint2 := Temp;
  end;
end;

procedure TPanelClock.SetSpace(Value: Integer);
begin
  FSpace := Value;
  Invalidate;
end;

procedure TPanelClock.WMDestroy(var Msg: TMsg);
begin
  KillTimer(Handle,1);
  inherited
end;

procedure TPanelClock.WMCreate(var Msg: TMsg);
begin
  SetTimer(Handle,1,200,NIL);
  inherited;
end;

procedure TPanelClock.WMTimer(var Msg: TMsg);
var
  NewKeyState: Integer;
  NewResources: TResourceMonitor;
  X: Integer;
begin
  NewKeyState := 0;
  if (PanelMode = pmClock) then begin
    if (GetKeyState(VK_NUMLOCK) and $01) <> 0 then
Inc(NewKeyState,ksNumLock);
    if (GetKeyState(VK_CAPITAL) and $01) <> 0 then
Inc(NewKeyState,ksCapsLock);
    if (GetKeyState(VK_SCROLL) and $01) <> 0 then
Inc(NewKeyState,ksScrollLock);
    if (FLastPaint <> TimeToStr(Time)) or (FKeyState <> NewKeyState) then
begin
      FKeyState := NewKeyState;
      Paint;
    end;
  end
  else begin
    NewResources[rmGDIResources] :=
```

```
GetFreeSystemResources(GFSR_GDIResources);
    NewResources[rmSystemResources] :=
GetFreeSystemResources(GFSR_SystemResources);
    NewResources[rmUserResources] :=
GetFreeSystemResources(GFSR_UserResources);
    for x := 1 to rmNumberOfMonitors do
      if (NewResources[x] <> FResources[x]) then begin
        Move(NewResources,FResources,SizeOf(FResources));
        Paint;
        Break;
      end;
    end;
  inherited;
end;

end.
```

WRITE IN C

(sung to The Beatles "Let it Be")

When I find my code in tons of trouble,
Friends and colleagues come to me,
Speaking words of wisdom:
"Write in C."

As the deadline fast approaches,
And bugs are all that I can see,
Somewhere, someone whispers"
"Write in C."

Write in C, write in C,
Write in C, write in C.
LISP is dead and buried,
Write in C.

I used to write a lot of FORTRAN,
for science it worked flawlessly.
Try using it for graphics!
Write in C.

If you've just spent nearly 30 hours
Debugging some assembly,
Soon you will be glad to
Write in C.

Write in C, write in C,
Write In C, yeah, write in C.
Only wimps use BASIC.
Write in C.

Write in C, write in C,
Write in C, oh, write in C.
Pascal won't quite cut it.
Write in C.

<Guitar Solo>

Write in C, write in C,
Write in C, yeah, write in C.
Don't even mention COBOL.
Write in C.

And when the screen is fuzzy,
And the editor is bugging me.
I'm sick of ones and zeroes.
Write in C.

A thousand people swear that T.P.
Seven is the one for me.
I hate the word PROCEDURE,
Write in C.

Write in C, write in C,
Write in C, yeah, write in C.
PL1 is 80's,
Write in C.

Write in C, write in C,
Write in C, yeah, write in C.
The government loves ADA,
Write in C.

Here is a custom TEdit that will tab to the next control when the user hits the <ENTER>

```
unit Ems;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TEMS = class(TEdit)
  private
    { Private declarations }
  protected
    { Protected declarations }
  public
    { Public declarations }
     procedure KeyPress(var Key: Char); override;
  published
    { Published declarations }
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Samples', [TEMS]);
end;

procedure TEMS.KeyPress(var Key: Char);
begin
  if key = #13 then begin
    key := #0;
    PostMessage(self.parent.Handle, WM_NEXTDLGCTL, 0, 0);
  end
  else inherited Keypress(key);
end;

end.
```

Fifty Ways to Hose Your Code
Kind of by Paul Simon

The problem's all inside your code she said to me;
Recursion is easy if you take it logically.
I'm here to help you if you're struggling to learn C,
There must be fifty ways to hose your code.

She said it's really not my habit to #include,
And I hope my files won't be lost or misconstrued;
But I'll recompile at the risk of getting screwed,
There must be fifty ways to hose your code.

Just blow up the stack Jack,
Make a bad call Paul,
Just hit the wrong key Lee,
And set your pointers free.

Just mess up the bus Gus,
You don't need to recurse much,
You just listen to me.

She said it greives me to see you compile again.
I wish there were some hardware that wasn't such a pain.
I said I appreciate that and could you please explain,
About the fifty ways.

She said why don't we both just work on it tonight,
And I'm sure in the morning it'll be working just right.
Then she hosed me and I realized she probably was right,
There must be fifty ways to hose your code.

Just lose the address Les,
Clear the wrong Int Clint,
Traverse the wrong tree Lee,
And set your list free.

Just mess up the bus Gus,
You don't need to recurse much,
You just program in C.

'TWAS THE NIGHT BEFORE IMPLEMENTATION

'Twas the night before implementation,
And all through the house.
Not a program was working,
Not even a browse.
The programmers hung by their tubes in despair,
With hopes that a miracle soon would be there.

The users were nestled all snug in their beds.
While visions of inquiries danced in their heads.
When out of the scope there arose such a clatter,
I sprang from my desk to see what was the matter.
And what to my wondering eyes should appear,
But a super-programmer (with a six-pack of beer).

His resume glowed with experience so rare,
He turned out great code with a bit-pushers' flair.
More rapid than eagles, his programs they came.
And he whistled and shouted and called them by name:
On update! On add! On inquiry! On Delete!
On batch jobs! On closing! On functions complete!

His eyes were glazed over, fingers nimble and lean.
From weeends and nights in front of a screen.
A wink of his eye and a twist of his head,
Soon gave me to know I had nothing to dread.

He spoke not a word, but went straight to his work.
Turning specs into code; then turned with a jerk.
And laying his finger upon the <ENTER> key,
The system came up and worked perfectly.

The updates updated; the deletes, they deleted.
The inquiries inquired; and closing completed.
He tested each whistle, and tested each bell.
With nary an abend, all had gone well.

The system was finished, the tests were concluded.
The users last changes were even included.
And the user exclaimed with a snarl and a taunt,
"It's just what I asked for, but not what I want!"

1991 Unix support contact headlines

I work at the support hotline for a fairly large Unix vendor. Customer calls are intercepted by a group of receptionists, who determine the general nature of each caller's problem or question and then place it on an electronic queue. The receptionists attach a "headline" to each call, so that the support analysts can decide whether a particular call is in their area of expertise. Unfortunately, the receptionists are not generally familiar with Unix.

Spelling errors can happen.

"The cron log file has exceeded 250 mega bite"
"Air message on consol"

Sometimes there is strange imagery involved. Picture this:

"Cannot get into the library"
"Runaway process boards"
"Terminals need to be brightened up"
...you can ignore this problem until they're suicidal.
"Question about braking when dialing in from a modem"
...calling from your car phone?
"Does not see the boot"
...check the end of your foot.
"Terminal has no cusor and making a high pitch wine"
...mmmm, just LOVE that high pitch wine!
"Cannot get into Telnet"
...yeah, telnet is pretty boring.
"Constant memory vaults"
...you're using too many JUMP instructions.
"X's and O's on terminal"
...how cute, it's just telling you it loves you.
"Terminal density is gone - cannot see screen"
...someone call a physicist -- their system is losing its mass!
"Bust fault and reset of system"
...can the hardware guy install a bra?

There is some hardware we just don't support.

"Install wife terminal"
"Has a PC that knocks down all terminals"
"Foot disk needs to be reformatted"
...contact your chiropractor.
"Actuary on printer is out"
...are they at an insurance company?

This is clearly NOT a software problem.

"Trouble with electrical smell on system"

This one came up a few weeks after Gorbachev had his trouble:

"When logging on, getting overthrow signal"

Similarly:

"Warning regent table overthrow"

Here's a stumper.

"EGA controller error grade andy controller, bell doesn't work"

Users may get a little fed up.

"Is it possible to communicate with a Unix machine?"
"Too much paper during printing"

Sometimes, you just have to wonder...

"Getting a parody error"
"If terminal is off, can't get prompt back"
"Having ahard disfailure"
"Question about configuration of Woodperfect"
"Set off a background process accidentally and wants to kill"
...I, too, would kill after making such a mistake.
"Questions on fox based software"
...those animals really do understand relational databases!
"Problem logging onto root, gets Chinese characters"
...oh, your console is upside-down.
"Each time he accesses a dose you have to reset the terminal"
...wow, man, the screen is breathing...
"Kill process logs users off system"
...it does tend to do that.
"Question on repetitioning the disc"
...we have here a signed statement: you should increase swap.
"Q how to do PCP over x dot 25"
...please, don't network under the influence.
"UPS DOWN"
...and down is up, right, sir?

From the 5th-season episode "Schisms" of Star Trek: The Next Generation

Ode to Spot
by Lt. Commander Data

Felis catus--is your taxonomic nomenclature
An endothermic quadruped, carnivorous in nature?
Your visual, olfactory, and auditory senses
Contribute to your hunting skills and natural defenses.
I find myself intrigued by your subvocal oscillations--
A singular development of cat communications
That obviates your basic hedonistic predilection
For a rhythmic stroking of your fur to demonstrate affection.
A tail is quite essential for your acrobatic talents;
You would not be so agile if you lacked its counterbalance.
And when not being utilized to aid in locomotion
It often serves to illustrate the state of your emotion.
Oh, Spot--the complex levels of behavior you display
Connote a fairly well developed cognitive array.
And though you are not sentient, Spot, and do not comprehend--
I, none the less, consider you a true and valued friend.

The Spellchecker Song

I have a spelling checker.
It came with my PC.
It plane lee marks four my revue
Miss steaks aye can knot see.

Eye ran this poem threw it.
Your sure real glad two no.
Its very polished in its weigh,
My checker tolled me sew.

A checker is a blessing.
It freeze yew lodes of thyme.
It helps me right awl stiles two reed,
And aides me when aye rime.

Each frays comes posed up on my screen
Eye trussed too bee a joule.
The checker pours o'er every word
To cheque sum spelling rule.

Bee fore a veiling checkers
Hour spelling mite decline,
And if we're laks oar have a laps,
We wood bee maid too wine.

Butt now bee cause my spelling
Is checked with such grate flare,
There are know faults with in my cite,
Of nun eye am a wear.

Now spelling does not phase me,
It does knot bring a tier.
My pay purrs awl due glad den
With wrapped words fare as hear.

To rite with care is quite a feet
Of witch won should be proud,
And wee mussed dew the best wee can,
Sew flaws are knot aloud.

Sow ewe can sea why aye dew prays
Such soft wear four pea seas,
And why eye brake in two averse
Buy righting want too please.

SIGNIFICANT OTHERS SPEAK OUT !!!

So you've found yourself attracted to a computer nerd. (Sorry techies; that IS what those of us that exist in the real world call you.) Spousal units and significant others (collectively referred to herein as "SO's") who have long endured the idiosyncrasies of their techie mates have banded together to provide the unsuspecting "future significant other" a peek at existence with: THE TECHIE. But first, a couple of disclaimers: All persons and events portrayed in this article are real and any resemblance to actual people or incidents is entirely intentional. Techies portrayed herein are of the male variety but male SO's have confirmed that they experience the same phenomenon in relation to their female techies.

To properly co-exist with a techie, you must first understand three basic premises on which his view of the world is based:

1. There is a proper order in the universe. Computers come first; significant others somewhere thereafter.

2. Programmers, while reluctantly admitting (subsequent to intense pressure) that they are not God, are however, equal to God.

3. Computer illiterate people are complete morons.

These three premises result in techies having a drastically different way of thinking as compared to the average person. This unique approach to life will be exhibited on a daily basis in many subtle ways:

TOPIC	WHAT YOU'RE THINKING	WHAT HE'S THINKING
Ideal Vacation	Tahiti	Las Vegas -- during Comdex
Shopping Trip	New wardrobe	Computer bookstore
Eating Out	Chez Romantic	Vending machine at the office
Fun Weekend	Picnic in the mountains	Non-stop programming
6 A.M.	Romantic sunrise	Late night of programming
People over for Dinner	Friends, Conversation	Victims to view latest software developments
Tax Time	Call an accountant	Order a tax package for SO
Looking at Stereo Equipment	Casual browsing	Select model, Close deal
Share Housework	50/50	Refrain from complaining that Pepsi isn't restocked

Spending more time with Children	Interactive Learning	Set up Barbies next to computer
Reason to cash out Investments	Child's Education	This years BMW's look good

It is true that techies' rarely subscribe to GQ magazine but, in all fairness, let's dispense with the slide-rule, taped glasses, white, button-down shirt stereotype. They no longer wear slide-rules; laptops are in. Taped glasses - well, ok, sometimes. White shirts have been replaced by t-shirts and flowered Hawaiian atrocities. "Dressing up" for a special occasion entails putting on jeans and a wrinkled shirt with a collar. If you happen to be domestically inclined, don't bother ironing shirts (or if you're not, feeling guilty about NOT ironing them) because pressed shirts are simply not a priority in a Techie's life and neither he nor any of his contemporaries will notice that the shirt he's wearing looks like it's been trapped between his mattress and box springs for a year.33

Material possessions are of vital importance to the techie. Of paramount importance is: THE CAR. The cost of this is directly proportional to the size of: THE EGO. There are two types of vehicles owned by techies: 1966 Station wagons with deteriorating wood on the sides OR the most expensive vehicle income will allow. (Neither category would be caught dead, however, driving a car with a Mary Kay bumper sticker attached.) Single techies can be identified by their dumpy apartments, frayed clothing and impeccably maintained Ferraris.

Techie's with vehicles in the second category assemble their machines for the annual Testosterone 500. Grown men gather at an area race track, spend 90 percent of the day walking around bragging about their car to anyone who will listen and devote the balance of the time tearing around on a track hoping they won't kill themselves. What we are witnessing is NOT simply a car race, but rather a battle of the egos. This same group of techies has also mastered the art of maneuvering discussion of THE CAR into every conversation.

The home computer system is another source of competition. Our family of four (techie, SO, 8 and 2 year old) is the proud owner of six computers, seven monitors, three laser printers, two dot matrix printers, two scanners, two optical disk drives, a CD ROM drive, and four boxes of cables that "might come in handy someday". Most appalling of all is that the 2 year old is limited to a 286 with an EGA! HORRORS! Special effort is made to explain to visiting techies that we are in the process of upgrading her system.

Other elaborate electronic devices run a close second to the "home computer competition". Techies must always have the latest and the best of any electronic device on the market and they MUST be the first in their group to own one. We have established true superiority with our home PBX phone system with the capacity to

handle 10 incoming lines, conference calls, 45 auto-dial numbers and, best of all, music on hold. Oh, and our answering machine has voice mail capabilities, can receive fax transmissions and makes dinner.

As you've probably already noticed, dating a techie has special challenges and rewards. Although your social hours are restricted to 11:00 p.m. - 3:00 a.m., you do have the opportunity to meet other SO's who, like you, are hanging around the office waiting for "just one more compile". A techie's estimate of "15 more minutes" generally means they will appear an hour or two later having absolutely no clue that more than 15 minutes has passed.

If you do manage to convince your techie to take a vacation, plan on his inspecting the computer system at every hotel, gas station, restaurant, car rental agency and airline. Expect him to make suggestions for improvements to busboys, valets, maids and waiters, none of whom have the remotest interest in their establishment's computer system, much less any influence in this arena. Keep in mind also that no matter where you go, techies will find each other. The first trip I, my sweetie and his portable computer took together was to Europe. I was one of the lucky few to be dating a man who owned one of the first portable computers manufactured, which of course automatically entitled us to first class service everywhere. He no sooner had placed the computer on the airline tray table than six fellow techies leaped to his side to discuss the merits of the computer. Personal conversation with my traveling companion totalled ten minutes out of a six hour flight.

Lunching with a group of techies is comparable to being dropped into a remote village in central Albania, with one major difference: Sign language is completely useless. They are speaking a foreign language and they are completely oblivious to this fact. My suggestion: Don't bother going. No one will notice that you were there anyway, including your techie.

Parties dominated by techies are truly exciting experiences. Techies have never developed the art of smalltalk (their computers don't require this attribute) so don't expect to see a techie talking to a non-techie. If a techie was forced to bring his SO, he will feel obligated, however, to forego technical discussions for at least the first ten minutes.

If you are unfortunate enough to be an SO with a "real job", you will encounter additional difficulties. The techie cannot fathom anyone going to work earlier than 10:00 a.m. He will tell you to simply inform your boss that you won't be starting until then.

Techies are very well read. They devour books and articles on such exciting topics as memory management, VXD's and debugging but give them a book on relationships and watch the panic spread across their faces. Mention a couples workshop you think both of you should attend and watch those deadlines move up.

At some point in their relationship, the SO must reveal to the

techie that a romantic holiday does not entail bringing along a portable computer, stacks of computer magazines and a trunkload of listings. They will be expected to spend an entire weekend without their computer! If you make it through this traumatic experience, a marriage or move-in-together proposal may be in the air. Expect any proposal to be very practical. Important issues such as what kind of dog you will get, how much money will be allotted to ego-related purchases, and how much space will be allocated for the special, hands-off place for his computers in your future home must be settled before a techie will even consider a permanent relationship. (Critical tip: This allotted space will double in size within six months, often spewing out into other areas of your home if you have not planned ahead.) Your wedding date will be arranged around development conferences, COMDEX and technical crises.

If, at some point in your relationship, you decide to have children, you will have to fit baby-making in between compiles. If you do manage to conceive, take a few photographs of your techie to tape over the baby's crib so your child will recognize your techie's face as well as his back.

On a personal level, the techie is very supportive of his significant other. When I decided to diet, my techie stood by me and agreed to diet with me; as long as he didn't have to give up Pepsi and Twinkies. When I determined that I needed a new look, he promised not to laugh when I came back with a new hairdo and agreed to unlimited funding for purchases made at lingerie shops.

The techie is also an accomplished gift-giver. Just last month, for my birthday, my techie gave me a Bug Zapper. (You know, one of those things that vaporizes the bugs flying around on your patio.) It seems he "heard me mention that we should get one." Guess he missed the references to the diamond necklace and pearl earrings. Last Christmas I was the proud recipient of a portable toolkit -- it's a beaut.

Well, I'd better close now. I'm due for my 10:43 appointment to review the 1991 COMDEX floor plan with you-know-who. Never a dull moment.....

/***/

Biography: The author is married to a techie who denies exhibiting any of the aforementioned behavior and feigned ignorance when asked if he noticed these characteristics in any of his fellow techies.

Disclaimer:

This file is a personal collection of (mostly) my own code and some from compuserve and one or two from the internet web sites. Some of the code that I did not write has not been fully tested. I kept it because I thought that it was too valuable to let it disappear. If you come across something that doesn't work quite right, use the code to get the idea. There is a lot of good stuff here!

Now to protect the company: Borland has nothing to do with this help file. If anyone writes a program that causes body parts to fall off, or war to break out (again), don't blame Borland! If you don't like the humor section because you don't think that programming and humor go together (and I know that you are out there), don't blame Borland! I put it there because I like it. So there. (nyahh!) If you want proof that programming and humor does go together, look in the mirror! <G>

To all that use this help file, have a ball with this stuff.

Delphi is the greatest!!!
Keep on hackin'!!!

50 Ways to Scare People In the Computer Room

1. Log on, wait a sec, then get a frightened look on your face and scream "Oh my God! They've found me!" and bolt.
2. Laugh uncontrollably for about 3 minutes & then suddenly stop and look suspiciously at everyone who looks at you.
3. When your computer is turned off, complain to the monitor on duty that you can't get the damn thing to work. After he/she's turned it on, wait 5 minutes,turn it off again, & repeat the process for a good half hour.
4. Type frantically, often stopping to look at the person next to you evilly.
5. Before anyone else is in the lab, connect each computer to different screen than the one it's set up with.
6. Write a program that plays the "Smurfs" theme song and play it at the highest volume possible over & over again.
7. Work normally for a while. Suddenly look amazingly startled by something on the screen and crawl underneath the desk.
8. Ask the person next to you if they know how to tap into top-secret Pentagon files.
9. Use Interactive Send to make passes at people you don't know.
10. Make a small ritual sacrifice to the computer before you turn it on.
11. Bring a chainsaw, but don't use it. If anyone asks why you have it, say "Just in case..." mysteriously.
12. Type on VAX for a while. Suddenly start cursing for 3 minutes at everything bad about your life. Then stop and continue typing.
13. Enter the lab, undress, and start staring at other people as if they're crazy while typing.
14. Light candles in a pentagram around your terminal before starting.
15. Ask around for a spare disk. Offer $2. Keep asking until someone agrees. Then, pull a disk out of your fly and say "Oops, I forgot."
16. Every time you press Return and there is processing time required, pray "Ohpleaseohpleaseohpleaseohplease," and scream "YES!" when it finishes.
17. "DISK FIGHT!!!"
18. Start making out with the person at the terminal next to you (It helps if you know them, but this is also a great way to make new friends).
19. Put a straw in your mouth and put your hands in your pockets. Type by hitting the keys with the straw.
20. If you're sitting in a swivel chair, spin around singing "The Lion Sleeps Tonight" whenever there is processing time required.
21. Draw a picture of a woman (or man) on a piece of paper, tape it to your monitor. Try to seduce it. Act like it hates you and then complain loudly that women (men) are worthless.
22. Try to stick a Ninetendo cartridge into the 3 disc drive, when it doesn't work, get the supervisor.
23. When you are on an IBM, and when you turn it on, ask loudly where the smiling Apple face is when you turn on one of those.
24. Print out the complete works of Shakespeare, then when its all done (two days later)

say that all you wanted was one line.
25. Sit and stare at the screen, biting your nails noisely. After doing this for a while, spit them out at the feet of the person next to you.
26. Stare at the screen, grind your teeth, stop, look at the person next to grinding. Repeat procedure, making sure you never provoke the person enough to let them blow up, as this releases tension, and it is far more effective to let them linger.
27. If you have long hair, take a typing break, look for split ends, cut them and deposit them on your neighbor's keyboard as you leave.
28. Put a large, gold-framed portrait of the British Royal Family on your desk and loudly proclaim that it inspires you.
29. Come to the lab wearing several layers of socks. Remove shoes and place them of top of the monitor. Remove socks layer by layer and drape them around the monitor. Exclaim sudden haiku about the aesthetic beauty of cotton on plastic.
30. Take the keyboard and sit under the computer. Type up your paper like this. Then go to the lab supervisor and complain about the bad working conditions.
31. Laugh hysterically, shout "You will all perish in flames!!!" and continue working.
32. Bring some dry ice & make it look like your computer is smoking.
33. Assign a musical note to every key (ie. the Delete key is A Flat, the B key is F sharp, etc.). Whenever you hit a key, humits note loudly. Write an entire paper this way.
34. Attempt to eat your computer's mouse.
35. Borrow someone else's keyboard by reaching over, saying "Excuse me, mind if I borrow this for a sec?", unplugging the keyboard & taking it.
36. Bring in a bunch of magnets and have fun.
37. When doing calculations, pull out an abacus and say that sometimes the old ways are best.
38. Play Pong for hours on the most powerful computer in the lab.
39. Make a loud noise of hitting the same key over and over again until you see that your neighbor is noticing (You can hit the space bar so your fill isn't affected). Then look at your neighbor's keyboard. Hit his/her delete key several times, erasing an entire word. While you do this, ask: "Does *your* delete key work?" Shake your head, and resume hitting the space bar on your keyboard. Keep doing this until you've deleted about a page of your neighbor's document. Then, suddenly exclaim: "Well, whaddya know? I've been hitting the space bar this whole time. No wonder it wasn't deleting! Ha!" Print out your document and leave.
40. Remove your disk from the drive and hide it. Go to the lab monitor and complain that your computer ate your disk. (For special effects, put some Elmer's Glue on or around the disk drive. Claim that the computer is drooling.)
41. Stare at the person's next to your's screen, look really puzzled, burst out laughing, and say "You did that?" loudly. Keep laughing, grab your stuff and leave, howling as you go.
42. Point at the screen. Chant in a made up language while making elaborate hand gestures for a minute or two. Press return or the mouse, then leap back and yell "COVEEEEERRRRRR!" peek up from under the table, walk back to the computer and say. "Oh, good. It worked this time," and calmly start to type again.
43. Keep looking at invisible bugs and trying to swat them.
44. See who's online. Send a total stranger a talk request. Talk to them like you've

known them all your lives. Hangup before they geta chance to figure out you're a total stranger.
45. Bring an small tape player with a tape of really absurd sound effects. Pretend it's the computer and look really lost.
46. Pull out a pencil. Start writing on the screen. Complain that the lead doesn't work.
47. Come into the computer lab wearing several endangered species of flowers in your hair. Smile incessantly. Type a sentence, then laugh happily, exclaim "You''re such a marvel!!", and kis the screen. Repeat this after every sentence. As your ecstasy mounts, also hug the keyboard. Finally, hug your neighbor, then the computer assistant, and walk out.
48. Run into the computer lab, shout "Armageddon is here!!!!!", then calmly sit down and begin to type.
49. Quietly walk into the computer lab with a Black and Decker chainsaw, rev that baby up, and then walk up to the nearest person and say "Give me that computer or you'll be feeding my pet crocodile for the next week".
50. Two words: Tesla Coil.

```
/* Best Layout winner of
        "The Seventh International Obfuscated C Code Contest"
    by Merlyn LeRoy <...uunet!rosevax!jhereg!quest!digibd!merlyn> */

/*----------------------- westley.c ----------------------------*/

char*lie;

        double time, me= !0XFACE,

        not; int rested,     get, out;

        main(ly, die) char ly, **die ;{

            signed char lotte,


dear; (char)lotte--;

        for(get= !me;; not){

        1 -   out & out ;lie;{

        char lotte, my= dear,

        **let= !!me *!not+ ++die;

            (char*)(lie=


"The gloves are OFF this time, I detest you, snot\n\0sed GEEK!");

        do {not= *lie++ & 0xF00L* !me;

        #define love (char*)lie -

        love 1s *!(not= atoi(let

        [get -me?

            (char)lotte-


(char)lotte: my- *love -

        'I'   -   *love -   'U' -
```

```
            'l'   -   (long)   - 4 - 'U' ])- !!
            (time   =out=   'a'));} while( my - dear
            && 'l'-1l   -get-   'a'); break;}}
                 (char)*lie++;

(char)*lie++, (char)*lie++; hell:0, (char)*lie;
            get *out* (short)ly     -0-'R'-   get- 'a'^rested;
            do {auto*eroticism,
            that; puts(*( out
                 - 'c'
-('P'-'S') +die+ -2 ));}while(!"you're at it");

for (*((char*)&lotte)^=
            (char)lotte; (love ly) [(char)++lotte+
            !!0xBABE];){ if ('l' -lie[ 2 +(char)lotte]){ 'l'-1l ***die; }
            else{ if ('l' * get *out* ('l'-1l **die[ 2 ])) *((char*)&lotte) -=
            '4' - ('l'-1l); not; for(get=!

get; !out; (char)*lie   &   0xD0- !not) return!!
            (char)lotte;}

(char)lotte;
            do{ not* putchar(lie [out
            *!not* !!me +(char)lotte]);
            not; for(;!'a';);}while(
```

```
        love (char*)lie);{

register this; switch( (char)lie

        [(char)lotte] -1s *!out) {

        char*les, get= 0xFF, my; case' ':

        *((char*)&lotte) += 15; !not +(char)*lie*'s';

        this +1s+ not; default: 0xF +(char*)lie;}}}

        get - !out;

        if (not--)

        goto hell;

            exit( (char)lotte);}


/*------------------------ westley.hint ------------------------*/

/*
Best Layout: <...uunet!rosevax!jhereg!quest!digibd!merlyn> Merlyn LeRoy

        Brian Westley (Merlyn LeRoy on usenet)
        DigiBoard, Inc.
        1026 Blair Ave.
        St. Paul, MN   55104
        USA

Judges' comments:

    usage: westley

    If you would rather "Daisy" someone other than Westley, rename
    the program as needed.   :-)

    Read each block of code as if it were a piece of correspondence.
    For example, the first block of code would read:

        charlie,
```

doubletime me, OXFACE!
not interested, get out
mainly die, charly, *die*
signed charlotte

The original source had control-L's after each code block. To make it easier on news readers, we converted each control-L to a blank line.

Some ANSI compilers will not accept '1s' as a short integer - for these compilers replace the '1s' with '1'.

Selected notes from the author:

This is a "Picking the Daisy" simulation. Now, instead of mangling a daisy, simply run this program with the number of petals desired as the argument.

This is a good counter-example to peoples' complaints that C doesn't have an "English-like" syntax.

Lint complains about everything - null effect, xxx may be used before set, statement not reached, return(e) and return. Lint dumps core on some systems. My personal favorite lint complaint is

"warning: eroticism unused in function main".

Also obviously, (char)lotte and (char*)lie are incompatible types...
*/

Last night I dreamed that the Real World had adopted the "Unix Philosophy."

I went to a fast-food place for lunch. When I arrived, I found that the menu had been taken down, and all the employees were standing in a line behind the counter waiting for my orders. Each of them was smaller than I remembered, there were more of them than I'd ever seen before, and they had very strange names on their nametags.

I tried to give my order to the first employee, but he just said something about a "syntax error." I tried another employee with no more luck. He just said "Eh?" no matter what I told him. I had
similar experiences with several other employees. (One employee named "ed" didn't even say "Eh?," he just looked at me quizzically.) Disgusted, I sought out the manager (at least it said "man" on his nametag) and asked him for help. He told me that he didn't know anything about "help," and to try somebody else with a strange name for more information.

The fellow with the strange name didn't know anything about "help" either, but when I told him I just wanted to order he directed me to a girl named "oe," who handled order entry. (He also told me about several other employees I couldn't care less about, but at least I got the information I needed.)

I went to "oe" and when I got to the front of the queue she just smiled at me. I smiled back. She just smiled some more. Eventually I realized that I shouldn't expect a prompt. I asked for a hamburger. She didn't respond, but since she didn't say "Eh?" I knew I'd done something right. We smiled at each other a little while longer, then I told her I was finished with my order. She directed me to the cashier, where I paid and received my order.

The hamburger was fine, but it was completely bare... not even a bun. I went back to "oe" to complain, but she just said "Eh?" a lot. I went to the manager and asked him about "oe." The manager explained to me that "oe" had thousands of options, but if I wanted any of them I'd have to know in advance what they were and exactly how to ask for them.

He also told me about "vi," who would write down my order and let me correct it before it was done, and how to hand the written order to "oe." "vi" had a nasty habit of not writing down my corrections unless I told her that I was about to make a correction, but it was still easier than dealing directly with "oe."

By this time I was really hungry, but I didn't have enough money to order again, so I figured out how to redirect somebody else's order to my plate. Security was pretty lax at that place.

As I was walking out the door, I was snagged by a giant Net. I screamed and woke up.

"Windows Error-codes"

Recently the following undocumented error-codes were found. MicroSoft forgot to explain them in the manuals, so here they are:

WinErr: 001 Windows loaded - System in danger
WinErr: 002 No Error - Yet
WinErr: 003 Dynamic linking error - Your mistake is now in every file
WinErr: 004 Erronious error - Nothing is wrong
WinErr: 005 Multitasking attempted - System confused
WinErr: 006 Malicious error - Desqview found on drive
WinErr: 007 System price error - Inadeqaute money spent on hardware
WinErr: 008 Broken window - Watch out for glass fragments
WinErr: 009 Horrible bug encountered - God knows what has happened
WinErr: 00A Promotional literature overflow - Mailbox full
WinErr: 00B Inadeqaute disk space - Free at least 50MB
WinErr: 00C Memory hog error - More Rame needed. More! More! More!
WinErr: 00D Window closed - Do not look outside
WinErr: 00E Window open - Do not look inside
WinErr: 00F Unexplained error - Please tell us how this happened
WinErr: 010 Reserved for future mistakes by our developers
WinErr: 011 Window open - Do not look outside
WinErr: 012 Window closed - Do not look inside
WinErr: 013 Unexpected error - Huh ?
WinErr: 014 Keyboard locked - Try anything you can think of.
WinErr: 018 Unrecoverable error - System has been destroyed. Buy a new one. Old windows licence is not valid anymore.
WinErr: 019 User error - Not our fault. Is Not! Is Not!
WinErr: 01A Operating system overwritten - Please reinstall all your software. We are terribly sorry.
WinErr: 01B Illegal error - You are not allowed to get this error. Next time you will get a penalty for that.
WinErr: 01C Uncertainty error - Uncertainty may be inadeqaute.
WinErr: 01D System crash - We are unable to figure out our own code.
WinErr: 01E Timing error - Please wait. And wait. And wait. And wait.
WinErr: 01F Reserved for future mistakes of our developers.
WinErr: 020 Error recording error codes - Remaining errors will be lost.
WinErr: 042 Virus error - A virus has been activated in a dos-box. The virus, however, requires Windows. All tasks will automaticly be closed and the virus will be activated again.
WinErr: 079 Mouse not found - A mouse driver has not been installed. Please click the left mouse button to continue.
WinErr: 103 Error buffer overflow - Too many erros encountered. Next errors will not be displayed or recorded.
WinErr: 678 This will end your Windows session. Do you want to play another game?
WinErr: 683 Time out error - Operator fell asleep while waiting for the system to

complete boot procedure.
WinErr: 815 Insufficient Memory - Only 50.312.583 Bytes available

humor: Microsoft

If Microsoft made cars
Bill Gates meets Dr. Seuss

humor: Dr. Seuss

Bill Gates meets Dr. Seuss
computers and Dr. Seuss
Technical writing and Dr. Seuss

Engineering

engineering phrases
Engineer's song
Engineers / Mathematicians
SIGNIFICANT OTHERS SPEAK OUT!!!
geek relationships
Just another day at the office
Mouse Balls (real memo from IBM)
the spelling checker
Scientific phrases explained

humor: programming

operating systems
"C"

Automotive tech support
programming languages
Evolution of a programmer
Fifty Ways to Hose Your Code
'twas the night before implementation
The twelve bugs of Xmas
Edgar Allan Poe at the computer
50 Ways to Scare People In the Computer Room
If architects had to work like programmers...
Troutman's Laws of Computer Programming
Real programmers don't...
The Devil and Daniel Webster
Assembler OP codes
Computer virus list
Computer message list
Quotes from famous people about computers in particular and progress in general
Computers in the movies

humor: Misc.

Star Trek
Oxymorons
Why did the chicken cross the road?
Murphy's Laws
One Liners
"Ode to Spot" by Lt. Commander Data
How shit happens
"Things that make you go 'hmmmm'."
Medical definitions

humor: operating systems

new OS
Operating systems as beer
Undocumented Win95 error messages
1991 Unix support contact headlines
UNIX fast food
Undocumented DOS commands

File Handling

How do I read and write binary files?
Here are some general file management routines.
How do I copy a file?
How do I iterate through all the directories and sub-directories of my hard disk.
How can I search through subdirectories?
Here is a unit to read Lotus 123 files.
How do I close a file that was opened in a DLL (Delphi made) and called from VB?
How can I readln() from a file when the lines are longer than 255 bytes?
How do I get a file's date and time stamp?
How do I write to a printer port?

{How to read a Lotus 123 file}

```
UNIT U123;  {Soure PC MAG. DECEMBER 13 1988... and others}
            { YES !  I did it in TP seven years Ago !!!}

INTERFACE

{
This routines ARE simple to use as 123.. :-)
1)  Open the file
2)  Add what you want.. where you want
3)  Close the File
}

PROCEDURE Open123(n:string);
PROCEDURE Close123;
PROCEDURE ColW123(c:integer; a:byte);
PROCEDURE Add123Int(c,f:integer; v:integer);
PROCEDURE Add123Rea(c,f:integer; v:double);
PROCEDURE Add123TXC(c,f:integer; v:string);
PROCEDURE Add123TXL(c,f:integer; v:string);
PROCEDURE Add123TXR(c,f:integer; v:string);
PROCEDURE Add123FML(c,f:integer; s:string);

{
  Open123(n:string);
  n = File Name WITHOUT EXTENSION it ALways add WK1
  It didn't check for a valid File Name or Existing, is
  YOUR responsability to do that


  Close123;
  Close the Open File .. Always DO THIS !

  In the rest of PROCEDURES c=Column and f=Row
  c and F begins with 0 (cero)
  if you want to Add in cell A1, use c=0 f=0
  if you want to Add in cell B2, use c=1 f=1
  etc.


  Add123Int(c,f:integer; v:integer);
  Add a Integer value (v) in Col=c  Row=f

  Add123Rea(c,f:integer; v:double);
  Add a Double value (v) in Col=c  Row=f

  Add123TXC(c,f:integer; v:string);
  Add a Label (v) in Col=C  Row=f
  - Label CENTER -

  Add123TXR(c,f:integer; v:string);
  Add a Label (v) in Col=C  Row=f
  - Label at RIGHT -

  Add123TXL(c,f:integer; v:string);
```

```
  Add a Label (v) in Col=C  Row=f
  - Label at LEFT -

  ColW123(c:integer; a:byte);
  Change width of Col=c to size=a

  Add123FML(c,f:integer; s:string);
  Add Formula (s) at Col=c  Row=f

  Examples:
           Add123FML(0,0,'A5+B2+A3*C5');
           Add123FML(0,1,'@Sum(B1..B8)');

  ========================================
  THE ONLY VALID @ function is SUM   !!!!
  Sorry :-(
  ========================================

}


{ The rest of Comments are in SPANISH ... Sorry again }


IMPLEMENTATION
CONST
     C00 = $00;
     CFF = $FF;

VAR
   ALotus : File;

PROCEDURE Open123(n:string);

Type
    Abre = record
                  Cod  : integer;
                  Lon  : integer;
                  Vlr  : integer;
             end;

Var
   Formato  : array[1..6] of byte;
   Registro : Abre absolute Formato;


Begin
     Assign(ALotus,n+'.WK1');
     Rewrite(ALotus,1);
     with Registro do
     begin
          Cod:=0;
          Lon:=2;
          Vlr:=1030;
     end;
     BlockWrite(ALotus,Formato[1],6);
End;
```

```
PROCEDURE Close123;

Type
    Cierra = record
                  Cod  : integer;
                  Lon  : integer;
             end;

Var
   Formato  : array[1..4] of byte;
   Registro : Cierra absolute Formato;


Begin
     with Registro do
     begin
          Cod:=1;
          Lon:=0;
     end;
     BlockWrite(ALotus,Formato[1],4);
     Close(ALotus);
End;

PROCEDURE ColW123(c:integer; a:byte);

Type
    Ancho = record
                  Cod  : integer;
                  Lon  : integer;
                  Col  : integer;
                  Anc  : byte;
             end;

Var
   Formato  : array[1..7] of byte;
   Registro : Ancho absolute Formato;


Begin
     with Registro do
     begin
          Cod:=8;
          Lon:=3;
          Col:=c;
          Anc:=a;
     end;
     BlockWrite(ALotus,Formato[1],7);
End;


PROCEDURE Add123Int(c,f,v:integer);

Type
    Entero = record
                  Cod  : integer;
                  Lon  : integer;
```

```
                  Frm  : byte;
                  Col  : integer;
                  Fil  : integer;
                  Vlr  : integer;
             end;

Var
   Formato  : array[1..11] of byte;
   Registro : Entero absolute Formato;

Begin
     with Registro do
     begin
          Cod:=13;
          Lon:=7;
          Frm:=255;
          Fil:=f;
          Col:=c;
          Vlr:=v;
     end;

     Blockwrite(ALotus,Formato[1],11);
End;

PROCEDURE Add123Rea(c,f:integer; v:double);
Type
    Entero = record
                  Cod  : integer;
                  Lon  : integer;
                  Frm  : byte;
                  Col  : integer;
                  Fil  : integer;
                  Vlr  : double;
             end;
Var
   Formato  : array[1..17] of byte;
   Registro : Entero absolute Formato;
Begin
     with Registro do
     begin
          Cod:=14;
          Lon:=13;
          Frm:=2 or 128;
          Fil:=f;
          Col:=c;
          Vlr:=v;
     end;

     Blockwrite(ALotus,Formato[1],17);
End;


PROCEDURE GrabaTXT(c,f:integer; v:string; t:char);
Type
    Entero = record
                  Cod  : integer;
                  Lon  : integer;
```

```
                  Frm  : byte;
                  Col  : integer;
                  Fil  : integer;
                  Vlr  : array[1..100] of char;
             end;
Var
   Formato  : array[1..109] of byte;
   Registro : Entero absolute Formato;
   i        : word;
Begin
     with Registro do
     begin
          Cod:=15;
          Lon:=length(v)+7;
          Frm:=255;
          Fil:=f;
          Col:=c;
          Vlr[1]:=t;
          for i:=1 to Length(v) do Vlr[i+1]:=v[i];
          Vlr[i+2]:=chr(0);
     end;
     Blockwrite(ALotus,Formato[1],length(v)+11);
End;

PROCEDURE Add123TXL(c,f:integer; v:string);
begin
     GrabaTXT(c,f,v,'''');
end;
PROCEDURE Add123TXC(c,f:integer; v:string);
begin
     GrabaTXT(c,f,v,'^');
end;
PROCEDURE Add123TXR(c,f:integer; v:string);
begin
     GrabaTXT(c,f,v,'"');
end;




PROCEDURE Add123FML(c,f:integer; s:string);

Type
    Formula = record
                  Cod : integer;               {codigo}
                  Lon : integer;               {longitud}
                  Frm : byte;                  {formato}
                  Col : integer;               {columna}
                  Fil : integer;               {fila}
                  Res : Double;               {resultado de formula}
                  Tma : integer;               {tamanio de formula en
bytes}
                  Fml : array[1..2048] of byte; {formula}
              end;
    symbol = (cel,num,arr,mas,men,por,dvs,pot,pa1,pa2);
```

```
    consym = set of symbol;

Var
   Formato   : array[1..2067] of byte;
   Registro  : Formula absolute Formato;
   fabs      : boolean;                {flag que indica si ffml es absoluta}
   v,                                  {v    = string 's' sin blancos}
   nro       : string;                 {nro  = numero de ffml}
   cfml,                               {cfml = valor de columna en formula}
   ffml      : word;                   {ffml =   "    " fila     "    "   }
   nfml,                               {nfml =   "    " constante "   "   }
   i,                                  {i    = indice de 'v' (formula) }
   ii,                                 {ii   =    "    " 's'     "     }
   index,                              {index=    "    " Fml}
   j,ret,                              {usados para convertir a numeros}
   len,                                {len  = longitud de 'v'}
   lens      : integer;                {lens =     "     " 's'}
   sym       : symbol;                 {sym  = ultimo simbolo leido}
   symsig,                             {usados para analizar formula para }
   syminifac : consym;                 {grabarla con notacion posfija     }
   z         : byte;                   {indice para inicializar array}


   Procedure CalculaDir(var Reg : Formula);

   var
      veces : integer;

      (*   Primero, se decide si cfml es absoluta o relativa. Si es absoluta
           calcula el valor real. Si es relativa primero chequea si cfml<col.
           Si cfml<col le resta cfml a 49152 (C000); este numero es usado por
           Lotus para calcular la direccion de una celda a la izquierda de
           donde esta parado. Si col<=cfml le suma cfml a 32768 para encender
           el MSB que indica que es relativa (la C tambien lo prende).

           Segundo, se procede de la misma manera con ffml para determinar si
           es absoluta o relativa, y despues se calcula la direccion en base
           a eso y a la relacion de ffml con fil.
      *)

   begin
        with Reg do
        begin
             if v[i]='$' then             {calcula la columna (cfml)}
             begin
                  inc(i);
                  cfml:=ord(v[i])-ord('A');
                  inc(i);
                  while (v[i] in ['A'..'Z']) and (len>=i) do
                  begin
                       cfml:=(cfml+1)*26+ord(v[i])-ord('A');
                       inc(i);
                  end;
             end
             else
             begin
                  if (ord(v[i])-ord('A') < col) then
```

```
                    begin
                         cfml:=49152-col+(ord(v[i])-ord('A'));
                         inc(i);
                         veces:=1;
                         while (v[i] in ['A'..'Z']) and (len>=i) do
                         begin
                              cfml:=49152-col+(26*veces)+(ord(v[i])-ord('A'));
                              cfml:=cfml+((ord(v[i-1])-ord('A'))*26);
                              inc(i);
                              inc(veces);
                         end;
                    end
                    else
                    begin
                         cfml:=ord(v[i])-ord('A');
                         inc(i);
                         while (v[i] in ['A'..'Z']) and  (len>=i) do
                         begin
                              cfml:=(cfml+1)*26+ord(v[i])-ord('A');
                              inc(i);
                         end;
                         cfml:=cfml+32768-col;
                    end;
               end;

               Fml[index]:=Lo(cfml);        {graba cfml}
               inc(index);                  {que posee }
               Fml[index]:=Hi(cfml);        {dos bytes }
               inc(index);

               if v[i]='$' then             {calcula la fila (ffml)}
               begin
                    inc(i);
                    fabs:=true;
               end
               else
                   fabs:=false;
               j:=i;
               while (v[i] in ['0'..'9']) and (len>=i) do
               begin
                    inc(i);
               end;
               nro:=copy(v,j,i-j);
               val(nro,ffml,ret);

               if fabs then                 {siempre se resta 1 por estar en
base 0}
               begin
                    if ffml>0 then ffml:=ffml-1;
               end
               else
               begin
                    if fil<ffml then
                    begin
                         ffml:=32768+abs(ffml-fil)-1;
                    end
                    else
```

```
                begin
                     ffml:=49152-abs(ffml-fil)-1;
                end;
           end;

           Fml[index]:=Lo(ffml);        {graba ffml}
           inc(index);                  {que posee }
           Fml[index]:=Hi(ffml);        {dos bytes }
           inc(index);
      end;
   end;

   Procedure CalculaNum(var Reg : Formula);

   var
      VDoble  : array[1..8] of byte;
      dfml    : Double absolute VDoble;
      d       : real;
      esreal  : boolean;
      k       : byte;
      numero  : longint;
      codigo  : integer;

   begin
        with Reg do
        begin
             esreal:=false;
             j:=i;
             while (v[i] in ['0'..'9','.']) and (len>=i) do
             begin
                  if v[i]='.' then esreal:=true;
                  inc(i);
             end;
             nro:=copy(v,j,i-j);
             {R-}
                 val(nro,numero,codigo);
             {R+}
                 if (codigo=0) and (numero>=-32768) and (numero<=32767) then
                    esreal:=false
                 else
                     esreal:=true;

             if esreal then
             begin
                  val(nro,d,ret);             {convierte en real doble}
                  dfml:=d;
                  {ConvRD(d,dfml);}

                  Fml[index]:=0;              {0 = indica que sigue una
constante}
                  inc(index);                 {    real doble precision (8
bytes)}
                  for k:=1 to 8 do
                  begin
                       Fml[index]:=VDoble[k];   {graba dfml}
                       inc(index);             {son ocho bytes}
                  end;
```

```
            end
            else
            begin
                 val(nro,nfml,ret);          {convierte en entero}

                 Fml[index]:=5;              {5 = indica que sigue una
constante }
                 inc(index);                 {    entera con signo (2 bytes)
}
                 Fml[index]:=Lo(nfml);       {graba nfml}
                 inc(index);                 {son dos bytes}
                 Fml[index]:=Hi(nfml);
                 inc(index);
            end;
            dec(i);
       end;
  end;

  Procedure CalculaRan(var Reg : Formula);

  begin
       with Reg do
       begin
            Fml[index]:=2;              {2 = codigo de rango; le sigue 8
bytes}
            inc(index);                 {    que son (col1fil1..col2fil2)
}

            CalculaDir(Reg);            {calcula col1fil1}
            i:=i+2;                     {salta los 2 ..  }
            CalculaDir(Reg);            {calcula col2fil2}
       end;
  end;

  Procedure CalculaArr(var Reg : Formula);

  {** SOLO CODIFICA @TRUE,@FALSE,@SUM(COL1FIL1..COL2FIIL2) **}

  var
     func,dir : string;                 {func  = string del @}
                                        {dir   = del rango}
     N_arg,nc : byte;                   {N_arg = cantidad de argumentos}
                                        {nc    = numero de codigo (T,F,S)}

  begin
       with Reg do
       begin
            inc(i);
            case v[i] of
                        'F' : nc:=51;
                        'T' : nc:=52;
                        'S' : nc:=80;
            end;

            while (v[i] in ['A'..'Z']) and (len>=i) do inc(i);
            inc(i);
            if nc=80 then
```

```
            begin
                 CalculaRan(Reg);        {calcula el rango
(col1fil1..col2fil2}
                 N_arg:=1;               {hay un solo argumento}
            end;

            Fml[index]:=nc;
            inc(index);
            if nc=80 then
            begin
                 Fml[index]:=N_arg;      {graba numero de argumentos}
                 inc(index);
            end;
       end;
  end;

  Procedure TraerChar;

  begin
       inc(i);                           {carga el simbolo para }
       if len>=i then                    {la recursividad       }
       begin
            case v[i] of
                        'A'..'Z','$' : sym:=cel;
                        '0'..'9','.' : sym:=num;
                        '@'          : sym:=arr;
                        '+'          : sym:=mas;
                        '-'          : sym:=men;
                        '*'          : sym:=por;
                        '/'          : sym:=dvs;
                        '^'          : sym:=pot;
                        '('          : sym:=pa1;
                        ')'          : sym:=pa2;
            end;
       end;
  end;


  Procedure Expresion(symsig : consym; var Reg : Formula);
  var
     opsuma:symbol;

  Procedure Termino(symsig : consym; var Reg : Formula);
  var
     opmul:symbol;

  Procedure Factor(symsig : consym; var Reg : Formula);
  var
     opexp:symbol;

  Procedure Exponente(symsig : consym; var Reg : Formula);

  begin{Exponente}
       while (sym in syminifac) and (len>=i) do
       begin
            case sym of
                       num : begin
```

```
                              CalculaNum(Registro);
                              TraerChar;
                         end;
                    cel : begin
                              Reg.Fml[index]:=1;
                              inc(index);
                              CalculaDir(Registro);
                              dec(i);
                              TraerChar;
                         end;
                    arr : begin
                              CalculaArr(Registro);
                              TraerChar;
                         end;
          else
              begin
                   if sym=pa1 then
                   begin
                        TraerChar;
                        Expresion([pa2]+symsig,Registro);
                        if sym=pa2 then
                        begin
                             Reg.Fml[index]:=4;       {4 = simbolo '(' }
                             inc(index);
                             TraerChar;
                        end;
                   end;
              end;
          end;
     end;
end;{Exponente}

begin{Factor}
     Exponente(symsig+[pot],Registro);
     while (sym=pot) and (len>=i) do
     begin
          opexp:=sym;
          TraerChar;
          Exponente(symsig+[pot],Registro);
          if opexp=pot then
          begin
               Reg.Fml[index]:=13;                  {13 = simbolo '^' }
               inc(index);
          end;
     end;
end;{Factor}

begin{Termino}
     Factor(symsig+[por,dvs],Registro);
     while (sym in [por,dvs]) and (len>=i) do
     begin
          opmul:=sym;
          TraerChar;
          Factor(symsig+[por,dvs],Registro);
          if (opmul=por) or (opmul=dvs) then
          begin
               if opmul=por then Reg.Fml[index]:=11   {11 = simbolo '*' }
```

```
                else
                    Reg.Fml[index]:=12;                {12 = simbolo '/' }
                inc(index);
            end;
        end;
   end;{Termino}

   begin{Expresion}

      (*   Este es el primero de cuatro procedimientos recursivos (Expresion,
           Termino, Factor y Exponente) que se usan para transformar la
formula
           en una expresion en notacion posfija, tal como se debe grabar. La
           tecnica consiste en retrasar la transmision del operador
aritmetico.
           Ejemplo:  a+(b*c)^d  ==>  abc*(d^+  .

           Expresion analiza si es suma o resta. Luego llama a Termino. Al
           volver trae el proximo dato y llama otra vez a Termino. Al volver
           genera el codigo de suma o resta si hubo.

           Termino llama a Factor. Al volver trae el proximo dato y llama
otra
           vez a Factor. Al volver genera el codigo de multiplicacion o
division
           si hubo.

           Factor llama a Exponente. Al volver trae el proximo dato y llama
           otra vez a Exponente. Cuando vuele genera el codigo de
exponenciacion
           si hubo.

           Exponente analiza si el valor es un numero, una celda, un arroba o
           un parentesis. Si es un parentesis, vuelve a llamar a Expresion
para
           calcular el contenido este; sino genera el codigo correspondiente.

      *)

        if sym in [mas,men] then
        begin
             opsuma:=sym;
             TraerChar;
             Termino(symsig+[mas,men],Registro);
             if opsuma=men then
             begin
                  Reg.Fml[index]:=8;                   {8 = simbolo '-'
unario}
                  inc(index);
             end;
        end
        else
            Termino(symsig+[mas,men],Registro);
        while (sym in [mas,men]) and (len>=i) do
        begin
             opsuma:=sym;
             TraerChar;
```

```
            Termino(symsig+[mas,men],Registro);
            if (opsuma=mas) or (opsuma=men) then
            begin
                 if opsuma=mas then Reg.Fml[index]:=9   { 9 = simbolo '+' }
                 else
                     Reg.Fml[index]:=10;                {10 = simbolo '-' }
                 inc(index);
            end;
        end;
   end;{Expresion}


Begin
     with Registro do
     begin
          Cod:=16;                      {16= formula}
          Col:=c;
          Fil:=f;

          Frm:=0;                       {Comienzo con 0}
(*
          if p=true then Frm:=Frm+128; {Si se protege se prende el MSB}

          ch:=UpCase(ch);               {Veo que formato se quiere y prendo }
                                        {los bits respectivos               }
          case ch of
                   'F' : Frm:=Frm+  0; {'F' ==> decimales fijos    }
                   'S' : Frm:=Frm+ 16; {'S' ==> notacion cientifica}
                   'C' : Frm:=Frm+ 32; {'C' ==> moneda corriente   }
                   'P' : Frm:=Frm+ 48; {'P' ==> porcentaje         }
                   'M' : Frm:=Frm+ 64; {',' ==> miles con comas    }
                   'O' : Frm:=Frm+112; {'O' ==> otros              }
          end;

          Frm:=Frm+d;                   {Si ch<>'O' ==> d= cant. de decimales}
                                        {Si ch= 'O' ==> d= 1 --> general     }
                                        {                  2 --> DD/MMM/AA   }
                                        {                  3 --> DD/MMM      }
                                        {                  4 --> MM/AA       }
                                        {                  5 --> texto       }
                                        {                  6 --> hidden      }
                                        {                  7 --> date; HH-MM-
SS}
                                        {                  8 --> date; HH-MM }
                                        {                  9 --> date; int'l 1
}
                                        {                 10 --> date; int'l 2
}
                                        {                 11 --> time; int'l 1
}
                                        {                 12 --> time; int'l 2
}
                                        {               13-14 --> no utilizado}
                                        {                  15 --> default     }

  *)
           Res:=C00;
```

```
{          for z:=1 to 8 do Res[z]:=C00;} {se modifica automaticamente cuando
se recalcula y regraba}

          lens:=length(s);             {convierto todo a mayusculas}
          for ii:=1 to lens do s[ii]:=UpCase(s[ii]);
          i:=1;
          v:='';
          for ii:=1 to lens do         {paso el string 's' al string 'v' }
          begin                        {eliminando los espacios en blanco}
               if s[ii]<>' ' then
               begin
                    v:=v+s[ii];
                    inc(i);
               end;
          end;

          len:=i-1;
          i:=0;
          index:=1;

          syminifac:=[cel,num,arr,pa1];
          symsig:=syminifac;

          TraerChar;                   {toma el primer caracter de formula}
          Expresion(symsig,Registro);  {analiza y graba toda la formula}

          Fml[index]:=3;               {3 = fin de formula}
          Tma:=index;                  {tamanio de Fml}
          Lon:=15+Tma;                 {longitud de dato}
          BlockWrite(ALotus,Formato[1],19+index);
     end;
End;


END.
```

Why do you need a driver's license to buy liquor when you can't drink and drive?

Why isn't phonetic spelled the way it sounds?
Why are there interstate highways in Hawaii?
Why are there flotation devices under plane seats instead of parachutes?
Why is there a -"permanent press" setting on irons?

Why are cigarettes sold in gas stations when smoking is prohibited there?
Do you need a silencer if you are going to shoot a mime?
Have you ever imagined a world with no hypothetical situations?
How does the guy who drives the snowplow get to work in the mornings?
If 7-11 is open 24 hours a day, 365 days a year, why are there locks on the doors?

If a cow laughed, would milk come out her nose?
If nothing ever sticks to TEFLON, how do they make TEFLON stick to the pan?

If you tied buttered toast to the back of a cat and dropped it from a height, what would happen?

If you're in a vehicle going the speed of light, what happens when you turn on the headlights?

You know how most packages say "Open here". What is the protocol if the package says, "Open somewhere else"?

Why do they put Braille dots on the keypad of the drive-up ATM?
Why do we drive on parkways and park on driveways?
Why is it that when you transport something by car, it's called a shipment, but when you transport something by ship, it's called cargo?
You know that little indestructible black box that is used on planes, why can't they make the whole plane out of the same substance?

Why is it that when you're driving and looking for an address, you turn down the volume on the radio?

Did you know who in 1923 was:

1. President of the largest steel company?
2. President of the largest gas company?
3. President of the New York Stock Exchange?
4. Greatest wheat speculator?
5. President of the Bank of International Settlement?
6. Great Bear of Wall Street?

These men should have been considered some of the world's most successful men. At least they found the secret of making money. Now more than 55 years later, do you know what has become of these men?

1. The President of the largest steel company, Charles Schwab, died a pauper.
2. The President of the largest gas company, Edward Hopson, is insane.
3. The President of the N.Y.S.E., Richard Whitney, was released from prison to die at home.
4. The greatest wheat speculator, Arthur Cooger, died abroad, penniless.
5. The President of the Bank of International Settlement shot himself.
6. The Great Bear of Wall Street, Cosabee Rivermore, died of suicide.

The same year, 1923, the winner of the most important golf championship, Gene Sarazan, won the U.S. Open and PGA Tournaments. Today he is still playing golf and is solvent.

custom components

How do I make Borland style buttons (checkbox and radio button)?
Slider
How do I use my own bitmap on the toolbar?
TPanelClock component (comes with Caps/Num/scroll lock indication)
Component to allow for paradox style QBE
Here is a custom TEdit that will tab to the next control when the user hits the <ENTER>.
Here is a custom TStringGrid that will allow for inserting a whole row at a time.
How do I emulate TCollection in Delphi?
How do I decide whether or not to hightlight a day in a calendar?
How do I justify the text in a TEdit?

```
{Here is a component that will work with paradox style QBE:}

Unit UTqbe; { Small DBdataset derived class and TREMOTEQBE }

{ David Berneda
  100115,1155@compuserve.com }

{
   NEW !!! TRemoteQBE to execute qbe queries remotely
  (needs REMOQUE.EXE on the remote machine)

   NEW !!! Function GetUserName:String (returns IDAPI user name or Paradox)
}

{  NEW !!! AnswerType property allows to PARADOX,DBASE and ASCII answer table
types }
{  NEW !!! Query params can be defined (see demo.pas) }

interface

Uses Classes,DBTables,DB,SysUtils,DBConsts,LibConst,dbiProcs,dbiTypes,
     DsgnIntf;

{ WARNING: READ CAREFULLY AND GOOD LUCK USING TQBE }
(*
  Answer table name can be specified with or without alias.
  Eg: :dbdemos:void.db

  Assumed answer table Driver: PARADOX
  The new property: AnswerType allows to PARADOX,DBASE and ASCII answer table
types

  If the answer table exists, an atempt to delete it is made before copying
  the result cursor.
  The phisical answer table must be opened in exclusive mode and all related
  family files are erased together with the table.

  Other functions in this unit:

**  Function GetAliasPath(Const Alias:String):String;
      Returns the path for the "alias" or a empty string if not found.
      Eg: ('dbdemos') returns 'c:\delphi\demos\data'

**  Function GetDBTablePath(Const TableName:String):String;
      Returns the TableName with path instead of alias (if it has an alias).
      Eg: (':dbdemos:customer.db') returns 'c:\delphi\demos\data\customer.db'

*)

Const MaxParam   = 5;  { max number of query parameters }
      MaxParamLen=30; { max length of a substituted param }

Type TQBE=Class(TDBDataSet)
     private
       FAnswerTable:String;
       FAnswerType:TTableType;
       FBlankasZero,
```

```
      FAuxTables,
      FRequestLive:Boolean;
    protected
      function CreateHandle: HDBICur; override;
      procedure GenerateAnswer(Var p:HDbiCur);
      Function CreateSubstituted:TStrings;
    public
      FQBE: TStrings;
      NumParam:Integer;
      Param,Subst:Array[0..MaxParam] of String[MaxParamLen];
      procedure SetQBE(QBE: TStrings);
      Constructor Create(AOwner:TComponent); override;
      destructor Destroy; override;
      Procedure AddParam(Const tmpParam,tmpSubst:String);
      Function ReplaceString(s:String):String;
      Procedure ClearParams;
    published
      property QBE: TStrings read FQBE write SetQBE;
      property AnswerTable: String read FAnswerTable write FAnswerTable;
      property BlankasZero: Boolean read FBlankasZero write FBlankAsZero;
      property AuxTables: Boolean read FAuxTables write FAuxTables;
      property AnswerType:TTableType read FAnswerType write FAnswerType;
      property RequestLive: Boolean read FRequestLive write FRequestLive;
    End;

    TRemoteStatus=( rsStart,
                    rsWaitingBegin,
                    rsWaitingExecution,
                    rsOK,
                    rsError,
                    rsTimeout,
                    rsIdle,
                    rsCancel
                    );

    TNotifyRemote=procedure(Sender:TObject; what:TRemoteStatus) of object;

    TRemoteQBE=Class(TQBE)
    Private
      FCancel:Boolean;
      FOnWaiting:TNotifyRemote;
      FSecondsTimeout:Longint;
      FDoRemote:Boolean;
      Procedure Notify(what:TRemoteStatus);
    protected
      function CreateHandle: HDBICur; override;
    public
      Constructor Create(AOwner:TComponent); override;
    published
      property OnWaiting:TNotifyRemote read FOnWaiting write FOnWaiting;
      property SecondsTimeout:Longint read FSecondsTimeout write
FSecondsTimeout default 60;
      property DoRemote:Boolean read FDoRemote write FDoRemote default TRUE;
      property Cancel:Boolean read FCancel write FCancel;
    End;

Function GetAliasPath(Const Alias:String):String;
```

```
Function GetDBTablePath(Const TableName:String):String;
Procedure Register;

implementation

Uses WinProcs,IniFiles,Dialogs,dbiErrs,Forms;

Constructor TQBE.Create(AOwner:TComponent);
Begin
  inherited Create(AOwner);
  FQBE := TStringList.Create;
  NumParam:=0;
  FAnswerType:=ttParadox; { by default, Paradox answer tables }
end;

destructor TQBE.Destroy;
Begin
  FQBE.Free;
  inherited Destroy;
End;

Procedure TQBE.ClearParams;
Begin
  NumParam:=0; { reset params to zero (no params) }
End;

Procedure TQBE.AddParam(Const tmpParam,tmpSubst:String);
Begin
  if (tmpParam<>'') and (tmpParam<>tmpSubst) then
  Begin
    if NumParam<MaxParam then
    Begin
      Inc(NumParam);
      Param[NumParam]:=tmpParam;
      Subst[NumParam]:=tmpSubst;
    End
    Else Raise Exception.Create('Max number of query parameters achieved');
  End;
End;

Function TQBE.ReplaceString(s:String):String;
Var t,i:Integer;
Begin
  for t:=1 to NumParam do
  Repeat
    i:=Pos(Param[t],s);
    if i>0 then s:=Copy(s,1,i-1)+Subst[t]+Copy(s,i+Length(Param[t]),255);
  Until i=0;
  result:=s;
End;

Function TQBE.CreateSubstituted:TStrings;
Var NewQBE:TStrings;
    t:Integer;
Begin
  NewQBE:=TStringList.Create;
  if Assigned(NewQBE) then
```

```
  Begin
    With FQBE do
    for t:=0 to Count-1 do NewQBE.Add(ReplaceString(Strings[t]));
  End;
  result:=NewQBE;
End;

function TQBE.CreateHandle: HDBICur;
Var p:HDbiCur;
    Stmt:hDBIStmt;
    NewQBE:TStrings;
Begin
  NewQBE:=CreateSubstituted;
  try
    Check(dbiQPrepare(DBHandle,qryLangQBE,NewQBE.GetText,Stmt));
    if FRequestLive then
Check(dbiSetProp(hDBIObj(Stmt),stmtLIVENESS,Longint(wantLive)))
                    Else
Check(dbiSetProp(hDBIObj(Stmt),stmtLIVENESS,Longint(wantDefault)));
    if FBlankAsZero then Check(dbiSetProp(hDBIObj(Stmt),stmtBLANKS,1));
    if FAuxTables then Check(dbiSetProp(hDBIObj(Stmt),stmtAUXTBLS,1));
    Check(dbiQExec(Stmt,@p));
    Check(dbiQFree(Stmt));
    GenerateAnswer(p);
  finally
    NewQBE.Free;
  end;
  Result:=p;
End;

procedure TQBE.GenerateAnswer(Var p:HDbiCur);
Var aBatTblDesc:BATTblDesc;
    tmpType:String;
    r:Longint;
    dbiErr:DBIRESULT;
Begin
  if (FAnswerTable<>'') And Assigned(p) then
  try
    Check(DbiSetToBegin(p));
    With aBatTblDesc do
    Begin
      hDB:=DBHandle;
      StrPCopy(szTblName,GetDBTablePath(FAnswerTable));
      Case FAnswerType of
        ttParadox: tmpType:=szParadox;
        ttDbase  : tmpType:=szDbase;
        ttAscii  : tmpType:=szAscii;
      end;
      StrPCopy(szTblType,tmpType);
      szUsername[0]:=#0;
      szPassword[0]:=#0;
    End;
    r:=0;

dbiErr:=dbiDeleteTable(DBHandle,aBatTblDesc.szTblName,aBatTblDesc.szTblType);
    if dbiErr<>DBIERR_NOSUCHTABLE then Check(dbiErr);
    Check(DbiBatchMove(nil,p,@aBatTblDesc,nil,batchCOPY,0,
```

```
                            nil, nil, nil, 0, nil, nil,
                            nil, nil, nil, nil, TRUE, TRUE,
                            r, TRUE));
    Check(dbiCloseCursor(p));
    Check(DbiOpenTable(DBHandle, aBatTblDesc.szTblName,
aBatTblDesc.szTblType,
                       nil, nil, 0,
                       dbiReadWrite,
                       dbiOpenShared,
                       xltField,
                       False,
                       nil,
                       p));
  finally
  End;
end;

procedure TQBE.SetQBE(QBE: TStrings);
begin
  FQBE.Assign(QBE);
end;

Function HasAlias(Const TableName:String):Boolean;
Begin
  Result:=Pos(':',TableName)>0;
End;

Function GetAliasPath(Const Alias:String):String;
Var AliasList:TStringList;
    i:Longint;
    DBPath:String;
Begin
  Result:='';
  AliasList:=TStringList.Create;
  try
    Session.GetAliasNames(AliasList);
    i:=AliasList.IndexOf(Alias);
    if i<0 then raise EDatabaseError.Create('Alias '+Alias+' doesnt exist')
    else
    Begin
      Session.GetAliasParams(Alias,AliasList);
      DBPath := AliasList.Values['PATH'];
      if DBPath='' then raise EDatabaseError.Create('Alias path from
'+Alias+' invalid')
                   else Result:=DBPath;
    end;
  finally
    AliasList.Free;
  end;
End;

Procedure SplitTableName(Const TableName:String; Var Alias,Name:String);
Var p1,p2:Integer;
Begin
  Name:=TableName;
  Alias:='';
  p1:=Pos(':',TableName);
```

```
  if p1>0 then
  Begin
    p2:=Pos(':',Copy(TableName,p1+1,255));
    if p2>0 then
    Begin
      Alias:=Copy(TableName,p1+1,p2-1);
      Name:=Copy(TableName,p1+p2+1,255);
    End;
  End;
End;

Function GetDBTablePath(Const TableName:String):String;
Var Alias,Name:String;
Begin
  if not HasAlias(TableName) then Result:=TableName
  else
  Begin
    SplitTableName(TableName,Alias,Name);
    if Alias<>'' then Result:=GetAliasPath(Alias)+'\'+Name
                 else Result:=TableName;
  End;
End;

Type TString=Array[0..255] of char;

Function GetUserName:String;
Var St:TString;
Begin
  result:='Unknown';
  if DbiGetNetUserName(St)=DBIERR_NONE then
     result:=StrPas(St)
  else
  With TIniFile.Create('win.ini') do
  try
    result:=ReadString('PDOXWIN','USERNAME','UNKNOWN');
  finally
    Free;
  End;
end;

{ TREMOTE QBE }

Constructor TRemoteQBE.Create(AOwner:TComponent);
Begin
  inherited Create(AOwner);
  FSecondsTimeout:=60;
  FDoRemote:=TRUE;
  FCancel:=FALSE;
End;

Procedure TRemoteQBE.Notify(what:TRemoteStatus);
Begin
  if Assigned(FOnWaiting) then
  try
    FOnWaiting(Self,what);
  except
   on Exception do ;
```

```
  end;
End;

function TRemoteQBE.CreateHandle: HDBICur;

    Function WaitServer(Table:TTable; NumQuery:Longint; Const
WaitFor:String):String;
    Var Timeout:Longint;
    Begin
      result:='';
      Timeout:=GetTickCount+SecondsTimeout*1000;
      Repeat
        Notify(rsIdle);
        Table.Refresh;
        Application.ProcessMessages;
        if Table.FindKey([NumQuery]) then
           result:=Table.FieldByName('Estado').AsString
        else
           result:='';
      Until (Pos(result,WaitFor)>0) or
            (GetTickCount>Timeout) or
            FCancel or
            Application.Terminated;
      if FCancel then result:='C'; { canceled }
    End;

Var Status:String;
    SAnswer:TString;
    t:TTable;
    NewQBE:TStrings;
    NumQuery:Longint;
Begin
  FCancel:=False;
  if not FDoRemote then
  Begin
    result:=Inherited CreateHandle;
    exit;
  end;
  result:=nil;
  Notify(rsStart);
  t:=TTable.Create(Self);
  with t do
  try
    DatabaseName:='REMOQUE';
    TableName:='remoque.db';
    IndexFieldNames:='NumeroConsulta';
    Open;
    Append;
    FieldByName('Fecha').AsDateTime:=SysUtils.Date;
    FieldByName('Hora').AsDateTime:=Now;
    FieldByName('Cliente').AsString:='DELPHI';
    FieldByName('Cliente').AsString:='DELPHI';
    FieldByName('Usuario').AsString:=GetUserName;
    FieldByName('TipoConsulta').AsString:='Q';
    NewQBE:=CreateSubstituted;
    try
      FieldByName('Consulta').Assign(NewQBE);
```

```
    finally
      NewQBE.Free;
    end;
    FieldByName('Estado').AsString:='P';
    FieldByName('Error').AsString:='';
    Post;
    NumQuery:=FieldByName('NumeroConsulta').AsInteger;
    Notify(rsWaitingBegin);
    Status:=WaitServer(t,NumQuery,'EOM');
    if Status='E' then
    Begin
      Notify(rsWaitingExecution);
      Status:=WaitServer(t,NumQuery,'OM');
    end;
    if Status='O' then
    Begin
      Notify(rsOK);
      StrPCopy(SAnswer,GetAliasPath('REMOQUE')
+'\PERSONAL\'+FieldByName('NumeroConsulta').AsString);
      Check(DbiOpenTable(DBHandle, SAnswer, szParadox,
                                      nil, nil, 0,
                                      dbiReadWrite,
                                      dbiOpenShared,
                                      xltField,
                                      False,
                                      nil,
                                      Result));
      GenerateAnswer(result);
    end
    else
    if Status='M' then
    Begin
      Notify(rsError);
      raise Exception.Create(FieldByName('Error').AsString);
    end
    else
    if Status='P' then
    Begin
      Notify(rsTimeout);
      if FindKey([NumQuery]) then
      try
        Edit;
        FieldByName('Estado').AsString:='T';
        FieldByName('Error').AsString:='Timeout';
        Post;
      finally
        raise Exception.Create('Timeout while waiting RemoQUE Server');
      end;
    end
    else
    if Status='C' then { cancelled query }
    try
      Notify(rsCancel);
      if FindKey([NumQuery]) then
      try
        Edit;
        FieldByName('Estado').AsString:='C';
```

```
        FieldByName('Error').AsString:='Canceled';
        Post;
      finally
      end;
    finally
      SysUtils.Abort;
    End;
  finally
    Free;
  end;
End;

Procedure Register;
Begin
  RegisterComponents(LoadStr(srDAccess),[TQBE,TRemoteQBE]);
End;

end.
```

The Twelve Bugs of Christmas

For the first bug of Christmas, my manager said to me
    See if they can do it again.

For the second bug of Christmas, my manager said to me
    Ask them how they did it and
    See if they can do it again.

For the third bug of Christmas, my manager said to me
    Try to reproduce it
    Ask them how they did it and
    See if they can do it again.

For the fourth bug of Christmas, my manager said to me
    Run with the debugger
    Try to reproduce it
    Ask them how they did it and
    See if they can do it again.

For the fifth bug of Christmas, my manager said to me
    Ask for a dump
    Run with the debugger
    Try to reproduce it
    Ask them how they did it and
    See if they can do it again.

For the sixth bug of Christmas, my manager said to me
    Reinstall the software
    Ask for a dump
    Run with the debugger
    Try to reproduce it
    Ask them how they did it and
    See if they can do it again.

For the seventh bug of Christmas, my manager said to me
    Say they need an upgrade
    Reinstall the software
    Ask for a dump
    Run with the debugger
    Try to reproduce it
    Ask them how they did it and
    See if they can do it again.

For the eighth bug of Christmas, my manager said to me

Find a way around it
Say they need an upgrade
Reinstall the software
Ask for a dump
Run with the debugger
Try to reproduce it
Ask them how they did it and
See if they can do it again.

For the ninth bug of Christmas, my manager said to me
Blame it on the hardware
Find a way around it
Say they need an upgrade
Reinstall the software
Ask for a dump
Run with the debugger
Try to reproduce it
Ask them how they did it and
See if they can do it again.

For the tenth bug of Christmas, my manager said to me
Change the documentation
Blame it on the hardware
Find a way around it
Say they need an upgrade
Reinstall the software
Ask for a dump
Run with the debugger
Try to reproduce it
Ask them how they did it and
See if they can do it again.

For the eleventh bug of Christmas, my manager said to me
Say it's not supported
Change the documentation
Blame it on the hardware
Find a way around it
Say they need an upgrade
Reinstall the software
Ask for a dump
Run with the debugger
Try to reproduce it
Ask them how they did it and
See if they can do it again.

For the twelfth bug of Christmas, my manager said to me

Tell them it's a feature
Say it's not supported
Change the documentation
Blame it on the hardware
Find a way around it
Say they need an upgrade
Reinstall the software
Ask for a dump
Run with the debugger
Try to reproduce it
Ask them how they did it and
See if they can do it again.

humor: "C"

obfuscated "C"

"C" vs Pascal
Write in "C" (song)

MNEMONIC	INSTRUCTION
-A-	
AAC	Alter All Commands
AAD	Alter All Data
AAO	Add And Overflow
AAR	Alter at Random
AB	Add Backwards
ABC	AlphaBetize Code
ABR	Add Beyond Range
ACC	Advance CPU clock
ACDC	Allow Controller to die peacefully
ACQT	Advance Clock to Quitting Time
ADB	Another Damn Bug [UNIX]
AEE	Absolve engineering errors
AFF	Add Fudge Factor
AFHB	Align Fullword on Halfword Boundary
AFP	Abnormalize Floating Point
AFR	Abort Funny Routine
AFVC	Add Finagle's Variable Constant
AGB	Add GarBage
AI	Add Improper(ly)
AIB	Attack Innocent Bystander
AMM	Answer My Mail
AMM	Add Mayo and Mustard
AMS	Add Memory to System
ANFSCD	And Now For Something Completely Different
AOI	Annoy Operator Immediate
AR	Advance Rudely
AR	Alter Reality
ARN	Add and Reset to Nonzero
ARZ	Add and Reset to Zero
AS	Add Sideways
AT	Accumulate Trivia
AWP	Argue With Programmer
AWTT	Assemble with Tinker Toys
-B-	
BA	Branch Anywhere
BAC	Branch to Alpha Centauri
BAF	Blow all Fuses
BAFL	Branch and Flush
BAH	Branch and Hang
BALC	Branch and Link Cheeseburger
BAP	Branch and Punt
BAW	Bells and Whistles
BB	Branch on bug
BBBB	Byte Baudy Bit and Branch
BBD	Branch on Bastille Day
BBIL	Branch on Burned-Out Indicator Light
BBLB	Branch on Blinking Light Bulb
BBT	Branch on Binary Tree
BBW	Branch Both Ways
BCB	Burp and Clear Bytes

BCF	Branch and Catch Fire
BCF	Branch on Chip box Full
BCIL	Branch Creating Infinite Loop
BCR	Backspace Card Reader
BCU	Be Cruel and Unusual
BD	Backspace Disk
BD	Branch to Data
BDC	Break Down and Cry
BDI	Branch to Data, Indirect
BDM	Branch and Disconnect Memory
BDT	Burn Data Tree
BE	Branch Everywhere
BEW	Branch Either Way
BF	Belch Fire
BFF	Branch and Form Feed
BFM	Be Fruitful and Multiply
BH	Branch and Hang
BIR	Branch Inside Ranch
BIRM	Branch on index register missing
BLC	Branch and Loop Continuous
BLI	Branch and Loop Infinite
BLM	Branch, Like, Maybe
BLMWM	Branch, Like, Maybe, Wow, Man
BLP	Boot from Line Printer
BLR	Branch and Lose Return
BLSH	Buy Low, Sell High
BM	Branch Maybe
BMI	Branch on Missing Index
BMI	Branch to Muncee, Immediate
BMP	Branch and Make Popcorn
BMR	Branch Multiple Registers
BNA	Branch to Nonexistant Address
BNCB	Branch and Never Come Back
BNR	Branch for No Reason
BOB	Branch on Bug
BOD	Beat on the Disk
BOD	Branch on Operator Desperate
BOH	Branch on Humidity
BOHP	Bribe operator for higher priority
BOI	Byte Operator Immediately
BOP	Boot OPerator
BOT	Branch On Tree
BPB	Branch on Program Bug
BPDI	Be Polite, Don't Interrupt
BPIM	Bury Programmer in Manuals
BPL	Branch PLease
BPO	Branch on Power Off
BPP	Branch & Pull Plug
BR	Byte and Run
BRA	Branch to Random Address
BRI	Branch Indefiniteley
BRO	BRanch to Oblivion
BRSS	Branch on Sunspot
BS	Behave Strangely
BSC	Branch on Second Coming
BSI	Backup Sewer Immediately
BSM	Branch and Scramble Memory

BSO	Branch on sleepy operator
BSP	Backspace Punch
BSR	Branch and Stomp Registers
BSST	BackSpace and Stretch Tape
BST	Backspace and Stretch Tape
BTD	Byte The Dust
BTD	Branch on Time of Day
BTJ	Branch and Turn Japanese
BTO	Branch To Oblivion
BTW	Branch on Third Wednesday
BU	Branch Unexpectedly
BVS	Branch & Veer South
BW	Branch on Whim
BWABL	Bells, Whistles, and Blinking Lights
BWC	Branch When Convenient
BWF	Busy - Wait Forever
BWOP	BeWilder OPerator
BYDS	Beware Your Dark Side
BYTE	BYte TEst

-C-

CAC	Calling All Cars...
CAC	Cash And Carry
CAF	Convert ASCII to Farsii
CAI	Corrupt Accounting Information
CAIL	Crash After I Leave
CAR	Cancel Accounts Receivable
CAT	Confused And Tired [UNIX]
CB	Consult Bozo
CBA	Compare and Branch Anyway
CBBR	Crash & Blow Boot ROM
CBNC	Close, but no Cigar
CBS	Clobber BootStrap
CC	Call Calvery
CC	Compliment Core
CCB	Chocolate Chip Byte-mode
CCB	Consult Crystal Ball
CCC	Crash if Carry Clear
CCCP	Conditionally Corrupt Current Process
CCD	Clear Core and Dump
CCD	Choke Cough and Die
CCR	Change Channels at Random
CCS	Chinese Character Set
CCWR	Change Color of Write Ring
CD	Complement Disk
CDC	Close Disk Cover
CDC	Clear Disk and Crash
CDIOOAZ	Calm Down, It's Only Ones and Zeroes
CDS	Change Disk Speed
CEMU	Close Eyes and Monkey With User Space
CEX	Call EXterminator
CF	Come From (replaces GOTO)
CFE	Call Field Engineer
CFP	Change and Forget Password
CFS	Corrupt File Structure
CG	Convert to Garbage

CH	Create Havoc
CHAPMR	Chase Pointers Around Machine Room
CHCJ	Compare Haig to Christine Jorgensen
CHSE	Compare Half-words and Swap if Equal
CIB	Change Important Byte
CIC	Cash In Chips
CID	Compare and Ignore Data
CIMM	Create Imaginary Memory Map
CIZ	Clear If Zero
CLBR	Clobber Register
CLBRI	Clobber Register Immediately
CM	Circulate Memory
CMD	Compare Meaningless Data
CMD	CPU Melt Down
CMI	Clobber Monitor Immediately
CML	Compute Meaning of Life
CMP	Create Memory Prosthesis
CMS	Click MicroSwitch
CN	Compare Nonsensically
CNB	Cause Nervous Breakdown
CNS	Call Nonexistent Subroutine
COD	Crash On Demand
COLB	Crash for Operator's Lunch Break
COCS	Copy Object Code to Source
COM	Clear Operator's Mind
COMF	COMe From
CON	Call Operator Now
COS	Copy Object code to Source file
COWYHU	Come Out With Your Hands Up
CP	Compliment Programmer
CP%FKM	CPU - FlaKeout Mode
CP%WM	CPU - Weird Mode
CPB	Create Program Bug
CPN	Call Programmer Names
CPPR	Crumple Printer Paper and Rip
CRASH	Continue Running After Stop or Halt
CRB	Crash and Burn
CRD	Confirm Rumor by Denial
CRM	Clear Random Memory
CRM	CReate Memory
CRN	Convert to Roman Numerals
CRN	Compare with Random Number
CRYPT	reCuRsive encrYPt Tape mneumonic [UNIX]
CS	Crash System
CSL	Curse and Swear Loudly
CSN	Call Supervisor Names
CSNIO	Crash System on Next I/O
CSS	Crash Subsidiary Systems
CSU	Call Self Unconditional
CTDMR	Change Tape Density, Mid Record
CTT	Call Time & Temperature
CU	Convert to Unary
CUC	Cheat Until Caught
CVFL	ConVert Floating to Logical
CVFP	ConVert FORTRAN to PASCAL
CVG	Convert to Garbage
CWAH	Create Woman and Hold

CWB	Carry With Borrow
CWDC	Cut Wires and Drop Core
CWG	Chase Wild Goose
CWGK	Compare Watt to Genghis Khan
CWIT	Compare Watt to Ivan the Terrible
CWM	Compare Watt to Mussolini
CWOM	Complement Write-only Memory
CZZC	Convert Zone to Zip Code

-D-

DA	Develop Amnesia
DAB	Delete All Bugs
DAO	Divide And Overflow
DAP	De-select Active Peripheral
DAUF	Delete All Useless Files
DB	Drop Bits
DBL	Desegregate Bus Lines
DBR	Debase Register
DBTP	Drop Back Ten and Punt
DBZ	Divide by Zero
DC	Divide and Conquer
DC	Degauss Core
DCAD	Dump Core And Die
DCD	Drop Cards Double
DCGC	Dump Confusing Garbage to Console
DCI	Disk Crash Immediate
DCON	Disable CONsle
DCR	Double precision CRash
DCT	Drop Cards Triple
DCWPDGD	Drink Coffee, Write Program, Debug, Get Drunk
DD	Destroy Disk
DD	Drop Disk
DDC	Dally During Calculations
DDOA	Drop Dead On Answer
DDS	Delaminate Disk Surface
DDWB	Deposit Directly in Wastepaper Basket
DE	Destroy Peripherals
DEB	Disk Eject Both
DEC	Decompile Executable Code
DEI	Disk Eject Immediate
DEM	Disk Eject Memory
DGT	Dispense Gin & Tonic
DHTPL	Disk Head Three Point Landing
DIA	Develop Ineffective Address
DIE	DIsable Everything
DIIL	Disable Interrupts and enter Infinite Loop
DIRFO	Do It Right For Once
DISC	DISmount Cpu
DK	Destroy Klingons
DK%WMM	Disk Unit - Washing Machine Mode
DKP	Disavow Knowledge of Programmer
DLN	Don't Look Now...
DLP	Drain Literal Pool
DMAG	Do MAGic
DMNS	Do What I Mean, Not What I Say
DMPE	Decide to Major in Phys. Ed.

DMPK	Destroy Memory Protect Key
DMZ	Divide Memory by Zero
DNPG	Do Not Pass Go
DO	Divide and Overflow
DOC	Drive Operator Crazy
DPCS	Decrement Program Counter Secretly
DPMI	Declare Programmer Mentally Incompetent
DPN	Double Precision No-op
DPR	Destroy Program
DPS	Disable Power Supply
DR	Detach Root
DRAF	DRAw Flowchart
DRAM	Decrement RAM
DRD	DRop Dead
DRI	Disable Random Interrupt
DROM	Destroy ROM
DRT	Disconnect Random Terminal
DS	Deadlock System
DSI	Do Something Interesting
DSO	Disable System Operator
DSP	Degrade System Performance
DSR	Detonate Status Register
DSTD	Do Something Totally Different
DSUIT	Do Something Utterly, Indescribably Terrible
DT%FFP	DecTape - unload and Flappa FlaP
DT%SHO	DecTape - Spin Hubs Opposite
DTB	Destructively Test Bit
DTC	Destroy This Command
DTE	Decrement Telephone Extension
DTI	Do The Impossible
DTRT	Do The Right Thing
DTVFL	Destroy Third Variable From Left
DU	Dump User
DUD	Do Until Dead
DW	Destroy World
DWIM	Do What I Mean
DWIT	Do What I'm Thinking

-E-

EA	Enable Anything
EAC	Emulate Acoustic Coupler
EAL	Enable AC to Logic rack
EAO	Enable AC to Operator
EBB	Edit and Blank Buffer
EBB	Empty Bit Bucket
EBR	Erase Before Reading
EBRS	Emit Burnt Resistor Smell
EC	Eat card
ECL	Early Care Lace
ECO	Electrocute Computer Operator
ECP	Erase Card Punch
ED	Eject Disk
ED	Execute Data (verrrry useful)
EDD	Eat Disk and Die
EDIT	Erase Data and Increment Time
EDP	Emulate Debugged Program

EDR	Execute Destructive Read
EDS	Execute Data Segment
EEOIFNO	Execute Every Other Instruction From Now On
EEP	Erase Entire Program
EFB	Emulate Five-volt Battery
EFD	Emulate Frisbee Using Disk Pack
EFD	Eject Floppy Disk
EFE	Emulate Fatal Error
EHC	Emulate Headless Chicken
EIAO	Execute In Any Order
EIO	Erase I/O page
EIOC	Execute Invalid Op-code
EIP	Execute Programmer Immeditely
EJD%V	EJect Disk with initial velocity V
ELP	Enter Loop Permenantly
EM	Emulate 407
EM	Evacuate Memory
EMM	Emulate More Memory
EMPC	Emulate Pocket Calculator
EMSE	Edit and Mark Something Else
EMSL	Entire Memory Shift Left
EMT	Electrocute Maintenance Technician
EMW	Emulate Matag washer
ENA	ENable Anything
ENF	Emit Noxious Fumes
ENO	Emulate No-Op
EO	Electrocute Operator
EOB	Execute Operator and Branch
EOI	Explode On Interrupt
EOS	Erase Operating System
EP	Execute Programmer
EPI	Execute Programmer Immediately
EPITS	Execute Previous Instruction Then Skip
EPL	Emulate Phone Line
EPP	Eject Printer Paper
EPS	Electrostatic Print and Smear
EPS	Execute Program Sideways
EPSW	Execute Program Status Word
EPT	Erase Process Table
EPT	Erase Punched Tape
ERIC	Eject Random Integrated Circuit
ERM	Erase Reserved Memory
EROM	Erase Read Only Memory
EROS	Erase Read-only Storage
ESB	Eject Selectric Ball
ESC	Emulate System Crash
ESD	Eject Spinning Dish
ESD	Eat Shit & Die
ESL	Exceed Speed of Light
ESP	Enable SPrinkler system
ETI	Execute This Instruction
ETM	Emulate Trinary Machine
EVC	Execute Verbal Commands
EWD	Enter Warp Drive
EWM	Enter Whimsy Mode
EXI	Execute Invalid Operation
EXOP	Execute Operator

EXPP	Execute Political Prisoner

-F-

FAY	Fetch Amulet of Yendor
FB	Find Bugs
FC	Fry Console
FCJ	Feed Cards and Jam
FD	Forget Data
FDR	Feed Disk Randomly
FERA	Forms Eject and Run Away
FFF	Form Feed Forever
FLD	FLing Disk
FLI	Flash Lights Impressively
FM	Forget Memory
FMP	Finish My Program
FOPC	False Out-of-Paper Condition
FPC	Feed Paper Continuously
FPT	Fire Photon Torpedoes
FRG	Fill with Random Garbage
FS	Feign Sleep
FSM	Fold, Spindle and Mutilate
FSRA	Forms Skip and Run Away

-G-

GBB	Go to Back of Bus
GCAR	Get Correct Answer Regardless
GDP	Grin Defiantly at Programmer
GDR	Grab Degree and Run
GENT	GENerate Thesis
GEW{JO}	Go to the End of the World {Jump Off}
GFD	Go Forth and Divide
GFM	Go Forth and Multiply
GIE	Generate Irreversible Error
GLC	Generate Lewd Comment
GMC	Generate Machine Check
GMCC	Generate Machine Check and Cash
GND	Guess at Next Digit
GORS	GO Real Slow
GREM	Generate Random Error Message
GREP	Global Ruin, Expiration, and Purgation [UNIX]
GRMC	Generate Rubber Machine Check
GS	Get Strange
GSB	Gulp and Store Bytes
GSI	Generate Spurious Interrupts
GSU	Geometric Shift Up
GTJ	Go To Jail

-H-

HACF	Halt And Catch Fire
HAH	Halt And Hang
HBW	Hang Bus & Wait
HCP	Hide Central Processor
HCRS	Hang in CRitical Section
HDO	Halt and Disable Operator

HDH	Hi Dee Ho
HDRW	Halt and Display Random Word
HELP	Type "No Help Available"
HF	Hide File
HGD	Halt, Get Drunk
HHB	Halt and Hang Bus
HIS	Halt in Imposible State
HOO	Hide Operator's Output
HRPR	Hang up and Ruin Printer Ribbon
HSC	Halt on System Crash
HSJ	Halt, Skip and Jump
HTC	Halt & Toss Cookies
HTS	Halt & Throw Sparks
HUAL	Halt Until After Lunch
HUP	Hang Up Phone
HWP	Halt Without Provocation

-I-

IAND	Illogical AND
IAE	Ignore All Exceptions
IAI	Inquire and ignore
IBM	Increment and Branch to Muncee
IBP	Insert Bug and Proceed
IBR	Insert BUgs at Random
ICB	Interrupt, Crash and Burn
ICM	Immerse Central Memory
ICMD	Initiate Core Melt Down
ICSP	Invert CRT Screen Picture
IDC	Initiate Destruct Command
IDI	Invoke Divine Intervention
IDNOP	InDirect No-OP
IDPS	Ignore Disk Protect Switch
IEOF	Ignore End Of File
IF	Invoke Force
IGI	Increment Grade Immediately
IGIT	Increment Grade Immediately Twice
IHC	Initiate Head Crash
II	Inquire and Ignore
IIB	Ignore Inquiry and Branch
IIC	Insert Invisible Characters
IIL	Irreversable Infinite Loop
IM	Imagine Memory
IMBP	Insert Mistake and Blame Programmer
IMP	Imitate Monty Python
IMPG	IMPress Girlfriend
IMV	IMpress Visitors
INCAM	INCrement Arbitrary Memory
INI	Ignore Next Instruction
INOP	Indirect No-OP
INR	INstigate Rumor
INW	INvalidate Warranty
IOI	Ignore Operator's Instruction
IOR	Illogical OR
IP	Increment and Pray
IPI	Ignore Previous Instruction
IPM	Ignore Programmer's Mistakes

IPOP	Interrupt Processor, Order Pizza
IPS	Incinerate Power Supply
IPS	Increment Power Supply
IPT	Ignite Paper Tape
IRB	Invert Record and Branch
IRBI	Insert Random Bits Indexed
IRC	Insert Random Commands
IRE	Insert Random Errors
IRI	Ignore Rude Interrupts
IRPF	Infinite Recursive Page Fault
ISC	Ignore System Crash
ISC	Insert Sarcastic Comments
ISC	Ignore Supervisor Calls
ISI	Increment and Skip on Inifinity
ISP	Increment and Skip on Pi
ISTK	Invert STacK
ITML	Initiate Termites into Macro Library
IU	Ignore User(s)
IZ	Ignore Zeroes

-J-

JAA	Jump Almost Always
JBS	Jump and Blow Stack
JCI	Jump to Current Instruction
JFM	Jump on Full Moon
JHRB	Jump to H&R Block
JLP	Jump and Lose Pointer
JMAT	JuMp on Alternate Thursdays
JN	Jump to Nowhere
JNL	Jump when programmer is Not Looking
JOM	Jump Over Moon
JOP	Jump OPerator
JPA	Jump when Pizza Arrives
JRAN	Jump RANdom
JRCF	Jump Relative and Catch Fire
JRGA	Jump Relative and Get Arrested
JRL	Jump to Random Location
JRSR	Jump to Random Subroutine
JSC	Jump on System Crash
JSU	Jump Self Unconditional
JT	Jump if Tuesday
JTT	Jump and Tangle Tape
JTZ	Jump to Twilight Zone
JWN	Jump When Necessary

-K-

KCE	Kill Consultant on Error
KEPITU	Kill Every Process In The Universe
KP	Krunch Paper
KSR	Keyboard Shift Right
KUD	Kill User's Data

-L-

LAC	Lose All Communication

LAGW	Load And Go Wrong
LAP	Laugh At Program(mer)
LCC	Load and Clear Core
LCD	Launch Cartridge Disk
LCK	Lock Console Keyswitch
LEB	Link Edit Backwards
LIA	Load Ineffective Address
LMB	Lose Message and Branch
LMO	Load and Mug Operator
LMYB	Logical MaYBe
LN	Lose inode Number [UNIX]
LNP	Load N digits of Pi
LOSM	Log Off System Manager
LP%PAS	Line Printer - Print And Smear
LP%RDD	Line Printer - Reverse Drum Direction
LP%TCR	Line Printer - Tangle and Chew Ribbon
LPA	Lead Programmer Astray
LPRTC	Load Program counter from Real Time Clock
LR	Load Revolver
LRA	Load RetroActively
LRD	Load Random Data
LSPSW	Load and Scramble PSW
LTS	Link To Sputnik
LUM	LUbricate Memory
LWE	Load WhatEver
LWM	Load Write-only Memory

-M-

MAB	Melt Address Bus
MAN	Make Animal Noises
MAZ	Multiply Answer by Zero
MBC	Make Batch Confetti
MBH	Memory Bank Hold-up
MBR	Multiply and be Fruitful
MBTD	Mount Beatles on Tape Drive
MBTOL	Move Bug To Operator's Lunch
MC	Move Continuous
MD	Move Devious
MDB	Move and Drop Bits
MDC	Make Disk Crash
MDDHAF	Make Disk Drive Hop Across Floor
MFO	Mount Female Operator
MLB	Memory Left shift and Branch
MLP	Make Lousy Program
MLP	Multiply and Lose Precision
MLR	Move and Lose Record
MMLG	Make Me Look Good
MNI	Misread Next Instruction
MOG	Make Operator Growl
MOP	Modify Operator's Personality
MOU	MOunt User [causes computer to screw you]
MPLP	Make Pretty Light Pattern
MRZ	Make Random Zap
MSGD	Make Screen Go Dim
MSP	Mistake Sign for Parity
MSPI	Make Sure Plugged In

MSR	Melt Special Register
MST	Mount Scotch Tape
MT%HRDV	Mag Tape - High speed Rewind and Drop Vaccuum
MTI	Make Tape Invalid
MW	Malfunction Whenever
MW	Multiply Work
MWAG	Make Wild-Assed Guess
MWC	Move and Wrap Core
MWT	Malfunction Without Telling

-N-

NBC	Negate By Clearing
NMI	Negate Most Integers
NOP	Needlessly Omit Pointer
NPC	Normalize Program Counter
NTGH	Not Tonight, i've Got a Headache

-O-

OCF	Open Circular File
OMC	Obscene Message to Console
OML	Obey Murphy's Laws
OPP	Order Pizza for Programmer
OSI	Overflow Stack Indefinitely
OTL	Out To Lunch

-P-

P$*!	Punch Obscenity
PA	Punch in ASCII
PAS	Print And Smear
PAUD	PAUse Dramatically
PAZ	Pack Alpha Zone
PBC	Print and Break Chain
PBD	Print and Break Drum
PBM	Pop Bubble Memory
PBPBPBP	Place Backup in Plain Brown Paper Bag, Please
PBST	Play Batch mode Star Trek
PCI	Pleat Cards Immediate
PCR	Print and Cut Ribbon
PD	Play Dead
PD	Punch Disk
PDLD	Power Down and Lock Door (to computer room)
PDSK	Punch DiSK
PEHC	Punch Extra Holes in Cards
PEP	Print on Edge of Paper
PFD	Punt on Fourth Down
PFE	Print Floating Eye [rogue]
PFML	Print Four Million Lines
PI	Punch Invalid
PIBM	Pretend to be an IBM
PIC	Print Illegible Characters
PIC	Punch Invalid Character
PLSC	Perform Light Show on Console
PNIH	Place Needle in Haystack
PNRP	Print Nasty Replies to Programmer

PO	Punch Operator
POCL	Punch Out Console Lights
POG	Print Only Greek
POPI	Punch OPerator Immediately
POPN	Punch OPerator's Nose
PPA	Print Paper Airplanes
PPL	Perform Perpetual Loop
PPP	Print Programmer's Picture
PPR	Play Punk Rock
PPSW	Pack Program Status Word
PSP	Print and Shred Paper
PSR	Print and Shred Ribbon
PTP	Produce Toilet Paper
PVLC	Punch Variable Length Card
PWP	Print Without Paper
PWS	create PoWer Surge
PYS	Program Yourself

-Q-

QWYA	Quit While Your Ahead

-R-

RA	Randomize Answer
RAM	Read Ambiguous Memory
RAN	Random Opcode [similar to 16-bit what gate]
RASC	Read And Shred Card
RAST	Read And Shred Tape
RAU	Ridicule All Users
RBAO	Ring Bell and Annoy Operator
RBG	Read Blank Tape
RBLY	Restore Back-up from Last Year
RBT	Rewind and Break Tape
RC	Rewind Core
RCAJ	Read Card And Jam
RCB	Read Command Backwards
RCB	Run Clock Backwards
RCC	Read Card and Chew
RCCP	Randomly Corrupt Current Process
RCF	Rewind Cabinet Fans
RCKG	Read Count Key and Garbage
RCL	Rotate Carry Left
RCR	Rewind Card Reader
RCRV	Randomly Convert to Reverse Video
RCSD	Read Card, Scramble Data
RD	Reverse Directions
RD	Randomize Data
RDA	Refuse to Disclose Answer
RDB	Run Disk Backwards
RDB	Replace Database with Blanks
RDD	Reverse Disk Drive
RDDBF	Rock Disk Drive Back and Forth
RDEB	Read and Drop Even number of Bits
RDF	Randomize Directory Filenames
RDI	Reverse Drum Immediate
RDR	Reverse Disk Rotation

RDS	Read SiDeways
RENVR	RENname Variables Randomly
RET	Read and Erase Tape
RF	Read Fingerprints
RG	Record Garbage
RHO	Randomize and Halt if not = to 0
RIC	Rotate Illogical thru Carry
RID	Read Invalid Data
RIOP	Rotate I/O Ports
RIR	Read Invalid Record
RIRG	Read Inter-record Gap
RJE	Return Jump and Explode
RLC	Relocate and Lose Core
RLC	Reread Last Card
RLC	Rotate Left with Carolyn
RLI	Rotate Left Indefinitely
RLP	Rewind Line Printer
RLP	Refill Light Pen
RM	Ruin My files
RMI	Randomize Memory Immediate
RMT	Remove Trap
RMV	Remove Memory Virtues
RN	Read Noise
RNBS	Reflect Next Bus Signal
ROC	Randomize Op Codes
ROC	Rotate Outward from Center
ROD	ROtate Diagonally
ROM	Read Operator's Mind
ROO	Rub Out Operator
ROOP	Run Out Of Paper
ROPF	Read Other People's Files
ROS	Reject Operating System
ROS	Return On Shield
RP	Read Printer
RPB	Read Print and Blush
RPB	Raise Parity Bits
RPBR	Reverse Parity and BRanch
RPC	Rotate Program Counter
RPM	Read Programmer's Mind
RPU	Read character and Print Upsidedown
RRC	Rotate Random thru Carry
RRR	Read Record and Run away
RRR	Randomly Rotate Register
RRRL	Random Rotate Register Left
RRRR	Random Rotate Register Right
RRSGWSSNK	Round and Round She Goes, Where She Stops, Nobody Knows
RRT	Record and Rip Tape
RS	Random Slew
RSD	On Read Error Self-Destruct
RST	Rewind and Stretch Tape
RSTOM	Read From Store-only Memory
RT	Reduce Throughput
RTS	Return To Sender
RVAC	Return from VACation
RWCR	ReWind Card Reader
RWD	ReWind Disk
RWF	Read Wrong File

-S-

SA	Store Anywhere
SAD	Search(seek) and Destroy
SAI	Skip All Instructions
SAS	Sit And Spin
SAS	Show Appendix Scar
SBE	Swap Bits Erratically
SC	Scramble Channels
SC	Shred Cards
SCB	Spindle Card and Belch
SCCA	Short Circuit on Correct Answer
SCD	Shuffle and Cut DEC
SCH	Slit Cards Horizontal
SCI	Shred Cards Immediate
SCM	Set for Crash Mode
SCOM	Set Cobol-Only Mode
SCRRC	SCRamble Register Contents
SCST	Switch Channel to Star Trek
SCTR	Stick Card To Reader
SD	Scramble Directory
SD	Slip Disk
SDC	Spool Disk to Console
SDD	Seek and Destroy Data
SDDB	Snap Disk Drive Belt
SDE	Solve Differential Equations
SDI	Self Destruct Immediately
SDM	Search and Destroy Memory
SDR	Slam Down Rondo [worst soda ever made]
SEB	Stop Eating and Burp
SEOB	Set Every Other Bit
SEX	Set EXecution register [real on the RCA 1802]
SEX	Sign EXtend
SFH	Set Flags to Half-mast
SFP	Send for Pizza
SFR	Send for Reinforcements
SFT	Stall For Time
SFTT	Strip Form Tractor Teeth
SHAB	Shift a Bit
SHABM	Shift a Bit More
SHB	Stop and Hang Bus
SHCD	SHuffle Card Deck
SHIT	Stop Here If Thursday
SHON	Simulate HONeywell CPU [permanent NO-OP]
SHRC	SHRed Card
SHRT	SHRed Tape
SID	Switch to Infinite Density
SIP	Store Indefinite Precision
SJV	Scramble Jump Vectors
SLP	Sharpen Light Pen
SMC	Scramble Memory Contents
SMD	Spontaneous Memory Dump [classfied data only]
SMR	Skip on Meaningless Result
SMS	Shred Mylar Surface
SNARF	System Normalize and Reset Flags
SNM	Show No Mercy

SNO	Send Nukes on Overflow
SOAWP	SOlve All the World'd Problems
SOB	Stew On Brew [a real PDP-11 instruction]
SOD	Surrender Or Die !
SOI	Screw O'Coin Intentionally (personal one)
SOP	Stop and Order Pizza
SOS	Sign off, Stupid
SOT	Sit on a Tack
SP	Scatter Print
SPA	Sliding Point Arithmetic
SPD	Spin Dry Disc
SPB	Simulate Peanut Butter
SPS	Set Panel Switches
SPSW	Scramble Program Status Word
SQPWYC	Sit Quietly and Play With Your Crayons
SRBO	Set Random Bits to Ones
SRBZ	Set Random Bits to Zeroes
SRC	Select Random Channel
SRCC	Select Reader and Chew Cards
SRD	Switch to Random Density
SRDR	Shift to Right Double Ridiculous
SRO	Sort with Random Ordering
SROS	Store in Read Only Storage
SRR	Shift Registers Random
SRSD	Seek Record and Scratch Disk
SRSZ	Seek Record and Scar Disk
SRTC	Stop Real-Time Clock
SRU	Signoff Random User
SRZ	Subtract and Reset to Zero
SRDR	Shift Right Double Ridiculous
SRSD	Seek Record and Scar Disk
SRZ	Subtract and Reset to Zero
SSB	Scramble Status Byte
SSJ	Select Stacker and Jam
SSJP	Select Stacker and JumP
SSM	Solve by Supernatural Means
SSP	Seek SPindle
SSP	Smoke and SPark
SST	Seek and Stretch Tape
ST	Set and Test
STA	STore Anywhere
STC	Slow To a Crawl
STD	Stop, Take Drugs
STM	STretch Magtape
STM	Skip on Third Monday
STO	Strangle Tape Operator
STROM	Store in Read-only Memory
STPR	SToP Rain
STROM	STore in Read-Only Memory
STTHB	Set Terminal to Three Hundred Baud
SUIQ	Subtract User's IQ
SUME	SUprise ME
SUP	Solve Unsolvable Problem
SUR	Screw Up Royally
SUS	Stop Until Spring
SUS	Subtract Until Senseless
SWAT	SWAp Terminals

SWN	SWap Nibbles
SWOS	Store in Write Only Storage
SWS	Sort to Wrong Slots
SWT	Select Wrong Terminal
SWU	Select Wrong Unit
SWZN	Skip Whether Zero or Not
SZD	Switch to Zero Density

-T-

TAH	Take A Hike
TAI	Trap Absurd Inputs
TARC	Take Arithmetic Review Course
TBFTG	Two Burgers and Fries To Go
TC	Transmit Colors (but avoid red)
TDB	Transfer and Drop Bits
TDRB	Test and Destroy Random Bits
TDS	Trash Data Segment
TLNF	Teach me a Lesson i'll Never Forget
TLO	Turn Indicator Lights Off
TLW	Transfer and Lose Way
TN	Take a Nap
TOAC	Turn Off Air Conditioner
TOG	Time Out, Graduate
TOG	Take Out Garbage
TOH	Take Operator Hostage
TOO	Turn On/Off Operator
TOP	Trap OPerator
TOS	Trash Operating System
TPD	Triple Pack Decimal
TPDH	Tell Programmer to Do it Him/Herself
TPF	Turn Power Off
TPN	Turn Power On
TPR	TeaR Paper
TR	Turn into Rubbish [UNIX]
TRA	Te Rdls Arvs [Type Ridiculous Abbreviations]
TSH	Trap Secretary and Halt
TSM	Trap Secretary and Mount
TST	Trash System Tracks
TT%CN	TeleType - Clunk Noise
TT%EKB	TeleType - Electrify KeyBoard
TTA	Try, Try Again
TTIHLIC	Try To Imagine How Little I Care
TTITT	Turn 2400 foot Tape Into Two 1200 foot Tapes
TTL	Tap Trunk Line
TTL	Time To Logoff
TYF	Trust Your Feelings

-U-

UA	Unload Accumulator
UAI	Use Alternate Instrucction set
UAPA(AM)	Use all Power Available (And More)
UCB	Uncouple CPU and Branch
UCK	Unlock Console Keyswitch
UCPUB	Uncouple CPU's and Branch
UDR	Update and Delete Record

UER	Update and Erase Record
UFO	Unidentified Flag Operation
ULDA	UnLoaD Accumulator
UMR	Unlock Machine Room
UNPD	UNPlug and Dump
UOP	Useless OPeration
UP	Understand Program(mer)
UPA	Use all Power Available
UPC	Understand Program(mer)'s Comments
UPCI	Update Card In Place
UPI	Undo Previous Instruction (only in EMACS)
URB	Update, Resume and Branch
UTF	Unwind Tape onto Floor
UTF	Use The Force
UUBR	Use Undefined Base Register

-V-

VAX	Violate All eXecutions
VFE	Violate Field Engineer
VFO	Violate Female Operator
VMA	Violate Maintenance Agreement
VNO	Violate Noise Ordinance
VPA	Vanishing Point Arithmetic
VVM	Vaporize Virtual Memory

-W-

WAD	Walk Away in Disgust
WAT	WAste Time
WBB	Write to the Bit Bucket
WBT	Water Binary Tree
WC	Waste Core [UNIX]
WCR	Write to Card Reader
WDR	Warp disk DRive
WED	Write and Erase Data
WEMG	Write Eighteen Minute Gap
WF	Wait Forever
WGPB	Write Garbage in Process-control Block
WHFO	Wait Until Hell Freezes Over
WHP	Wave Hands over Program
WI	Write Illegibly
WI	Why Immediate
WID	Write Invalid Data
WNHR	Write New Hit Record
WNR	Write Noise Record
WPET	Write Past End of Tape
WPM	Write Programmer's Mind
WSE	Write Stack Everywhere
WSWW	Work in Strange and Wonderous Ways
WUPO	Wad Up Printer Output
WWLR	Write Wrong-Length Record
WWR	Write Wrong Record
WSWW	Work in Strange and Wondrous Ways

-X-

XIO	eXecute Invalid Opcode
XKF	eXecute Kermit the Frog
XMB	eXclusive MayBe
XOH	eXecute no-Op and Hang
XOR	eXecute OpeRator
XOS	eXchange Operator's Sex
XPR	eXecute Programmer
XPSW	eXecute Program Status Word
XSP	eXecute Systems Programmer
XVF	eXchange Virtue for Fun

-Y-

YAB	Yet Another Bug
YASE	Yet Another Stupid Error

-Z-

ZAP	Zero and Add Packed
ZAR	Zero Any Register
ZD	Zap Directory
ZEOW	Zero Every Other Word
ZPI	ZaP Immediate

Sperry (Unisys) 1100/90 Opcodes :

BBL	Branch on Burned out Light
BAH	Branch And Hang
BLI	Branch and Loop Infinite
BPB	Branch on Program Bug
BPO	Branch if Power Off
CPB	Create Program Bug
CRN	Convert to Roman Numerals
DAO	Divide And Overflow
ERS	Erase Read-only Storage
HCF	Halt and Catch Fire
IAD	Illigical And
IOR	Illogical Or
MDB	Move and Drop Bits
MWK	Multiply WorK
PAS	Print And Smear
RBT	Read and Break Tape
RPM	Read Programmer's Mind
RRT	Record and Rip Tape
RSD	Read and Scramble Data
RWD	ReWind Disk
TPR	Tear PapeR
WED	Write and Erase Data
WID	Write Invalid Data
XIO	Execute Invalid Opcode
XOR	Execute OperatoR
XPR	Execute ProgrammeR

ab	add bug

ac	accept compliment
ai	add improper
amm	add more money
arz	add and reset to zero
balo	branch and lose output
bbi	branch on blinking indicator
bcb	blow circuit breakers
bfcb	branch on full chip box
bah	branch and hang
bahu	branch and hang user
betr	backspace and eject trapped rodents
bo	byte operator
bpo	branch and power off
bscr	backspace card reader
bsd	backspace disk
bspr	backspace printer
bsst	backspace and stretch tape
bud	branch to unknown device
can	change account number
cc	change channel
cia	find head and execute
cll	compare logical later
cm	circulate memory
cmt	compare under masking tape
cob	change output to binary
coh	change out to hexadecimal
cp	circulate pages
css	crash and save system (eric special)
cvc	convert to chinese
cvm	crash vm
cvps	crash vps
cvrn	convert to roman numerals
da	delete account
dac	divide and conquer
db	disable buzzer
dbcwe	disable buzzer and close window early
dcl	drop cards and laugh
ddu	disconnect dial-up users
dibb	divide into bit bucket
do	divide and overflow
dup	decrease user priority
ec	erase card
ed	eject disk
edal	erase disk and laugh
eoo	erase old output
epqj	emergency pull and quit job
eard	erase and read disk
erom	erase read-only memory
exo	execute operator
fjpr	force john porter and run
fmqg	fill message que with garbage
fsra	form skip and run away
fuar	force users at random
fudt	force user and disable terminal
ghcs	priveledged instruction only (not for users)
gtp	get the point
hcf	halt and catch fire

hcu	help cute users
hsdpl	halt system during peak load
ibpc	ignore buzzer and play cards
iibr	ignore inquiry and branch
imf	ignore message and force
ink	i'm not kidding
ipc	ignore previous command
irb	insert random bug
isc	identify strange character
itt	increase turnaround time
jcp	jam card punch
jcr	jam card reader
jpr	jam printer
kt	knot tape
loc	lock operator's console
mdb	move and drop bits
mbta	never return
mlp	multiply and lose precision
mlr	move and lose record
mnfa	moved no forwarding address
mti	make tape invalid
mwc	move and warp core
mwwt	mount and write wrong tape
mudd	mount user on disk drive
obs	pad storage with obscenities
ofp	override file protect
ot	overwrite tape
owau	open window and attack users
pcac	pick card any card
pfd	play frisbee with disk
pmt	punch magnetic tape
ps	print and smear
pxh	punch extra holes
qfp	query file and purge
rax	huh?
rcsd	read card and scramble data
rdc	read and drop cards
rip	read and interchange parity
rirg	read inter-record gap
riv	read invalid
rjp	read job and purge
rnr	read noise record
rpr	read printer
rprb	read printer and blush
rrp	remove ribbon and print
rsc	read and shred card
rtb	read tape backwards
rwcr	rewind card reader
rwj	run wrong job
rwtb	rewind and break tape
sbta	store bus terminal address
sc	shuffle cards
sd	slip disk
shl	search high and low
sic	as is
soc	space out characters
spss	what a joke

```
srsd              seek record and scar disk ('on a clear disk you can seek
forever')
ss                save system
ssr               stop and slice ribbon
sss               steal system and sell
ssdb              save system and destroy backups
ssj               select stacker and jam
stf               store twenty-four
tdb               transfer and drop bits
trte              translate to english
twr               translate wrong record
ued               update and erase disk
uer               update and erase record
uet               update and erase tape
vbo               vary buzzer offline
vro               vary radio offline
vtvo              vary tv offline
vuo               vary users offline
wdo               wash and dry output
wia               write in ascii
wtb               write tape backwards
wwrl              write wrong record length
```

In the Beginning was the Plan
And then came the Assumptions
And the Assumptions were without form
And the Plan was completely without substance
and the darkness was upon the face of the workers
and they spoke among themselves, saying
"It is a crock of shit, and it stinketh."
And the workers went unto their Supervisors and sayeth,
"It is a pail of dung and none may abide the odor thereof."
And the Supervisors went unto their Managers and sayeth unto them,
"It is a container of excrement and it is very strong,
Such that none may abide by it."
And the Managers went unto their Directors and sayeth,
"It is a vessel of fertilizer, and none may abide it's strength."
And the Directors spoke amongst themselves, saying one to another,
"It contains that which aids plant growth, and it is very strong."
And the Directors went unto the Vice Presidents and sayeth unto them,
"It promotes growth and is very powerful,"
And the Vice Presidents went unto the President and sayeth unto him,
"This new plan will actively promote the growth and efficiency
of this Company, and in these Areas in particular."
And the President looked upon The Plan,
And saw that it was good, and The Plan became Policy.
This is How Shit Happens

Birthday Virus - Keeps advancing your clock by another year.

Mario Cuomo Virus - a very powerful virus, if it would ever run.

Politically Correct Virus - prefers to call itself an "electronic microorganism".

Oprah Winfrey Virus - first appears on system as a 120 KB file, later swells to 200 KB, then returns to its original size. Periodic printouts appear to keep you surprised.

AT&T Virus - constantly reminds you how it's giving you much better service than the other viruses.

Sprint Virus - Periodically runs sound file of a pin dropping.

MCI Virus - Encourages you to send it to your friends and family.

Ollie North Virus - Converts your printer into a paper shredder.

Nike Virus - Just does it.

Ross Perot Virus - runs for awhile, leaves the system, then re-appears, but with less effect.

Airport Virus - You're in (insert location), but your data are in (exotic foriegn destination).

Arnold Virus - stays resident after it terminates. It'll be back.

Right-to-life Virus - before allowing you to delete any file, it first asks you if you've considered the alternatives.

Gridlock Virus - Keeps shuffling information that it calls 'bills' between your CPU and BUS, sending messages like 'House Bill #xxxx is unacceptable to Senate'. Never gets any work done.

Right-Wing-Hardliner Virus - Won't allow any changes on your system, but keeps saying that things will get better as soon as it takes over the Whitehouse.

Left-Wing-Drivel Virus - Deletes all monetary files, but keeps smiling and sending messages about how the economy is going to get better.

Government Economist Virus - nothing works on your system, but all your diagnostic software says everything is just fine.

Dan Quayle Virus - prevents your processes from spawning any child without first joining into a binary network. Also said to cause speling erors in files stored on your disk.

Jack Kevorkian Virus - enables irreparably damaged files to delete themselves.

Bobbit virus - It turns a 7.5meg hardrive to a 3 1/2 inch floppy drive.

Al Gore UNIX Virus - Whenever you inquire about one of your environment variables, it shows you the current setting, but then tacks on an alarmist message concerning the future of the variable.

Tipper Gore Virus - When you attempt to play any sound file, it pops up a warning window stating that some lyrics may be unsuitable for children.

Ponzi Virus - It logs onto your bank's computer and transfers $1 into the accounts of the owners of the last 10 computers it was on. It then attaches itself to the next 10 items of mail you send.

Joke Virus - poses as a harmless list of funny computer virus names!

==========

Subject: Computer Messages

==========

< 1> Error 13: Illegal brain function. Process terminated.

< 2> REALITY.DAT not found. Atempting to restore Universe.....

..... REALITY.SYS Corrupted - Unable to recover Universe

Press Esc key to reboot Universe, or any other key to continue...

< 3> REALITY.SYS corrupted- reboot Universe (Y/N)?

< 4> USER ERROR: replace user and press any key to continue.

< 5> Volume in Drive C: TOO_LOUD!

< 6> Press [ESC] to detonate or any other key to explode.

< 7> BREAKFAST.COM halted... cereal port not responding!

< 8> Virus detected! P)our chicken soup on motherboard?

< 9> .signature not found! reformat hard drive? [Yn]

<10> Backup not found! A)bort, R)etry or P)anic?

<11> Spellchecker not found. Press -- to continue ...

<12> A)bort, R)etry or S)elf-destruct?

<13> A)bort, R)etry, I)gnore, V)alium?

<14> A)bort, R)etry, I)nfluence with large hammer.

<15> A)bort, R)etry, P)ee in drive door

<16> Backup not found: A)bort, R)etry, M)assive heart failure?

<17> Bad command or file name. Go stand in the corner.

<18> Close your eyes and press escape three times.

< 19> DYNAMIC LINKING ERROR: Your mistake is now everywhere.

< 20> Computer possessed? Try DEVICE=C:\EXOR.SYS

< 21> SENILE.COM found... Out Of Memory.

< 22> APATHY ERROR: Don't bother striking any key.

< 23> ZAP! Process discontinued. Enter any 12-digit prime number to resume.

Q: How do I rotate text in display?

A:

```
unit Rotate;
{*********************************************************
PROGRAM
  UNIT ROTATE.PAS

PURPOSE
  To contain the text rotation routines. All documentation
  for the routines are within the routines.

HISTORY
  6/18/1995 First created by Curtis Keisler

COPYRIGHT & DISCLAIMER
  See ANGLE.DPR for copyright and disclaimer notice.
*********************************************************}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls;

type
  TForm1 = class(TForm)
    PaintBox1: TPaintBox;
    procedure PaintBox1Paint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

{*********************************************************}
procedure CanvasSetTextAngle(c: TCanvas; d: Word);
{---------------------------------------------------------
  PURPOSE
    To change the current text output rotation angle. All
    subsequent output will be at the angle provided.

  INPUT PARAMETERS
    c - The canvas on which to output the text. The font for
      the canvas must be a scaleable font.
    d - The angle in tenths of degrees. For example 10 would
      be 1 degree. 450 would be 45 degrees. 1 would be
      1/10 of a degree. To reset the text back to normal,
```

```
      make d = 0.

  HISTORY
    6/18/1995 First version written by Curtis Keisler.
-------------------------------------------------------------}
var
  LogRec: TLOGFONT;     {* Storage area for font information *}

begin
  {* Get the current font information. We only want to modify the angle *}
  GetObject(c.Font.Handle,SizeOf(LogRec),Addr(LogRec));

  {* Modify the angle. "The angle, in tenths of a degrees, between the base
     line of a character and the x-axis." (Windows API Help file.)*}
  LogRec.lfEscapement := d;

  {* Delphi will handle the deallocation of the old font handle and *}
  c.Font.Handle := CreateFontIndirect(LogRec);
end; {* CanvasSetTextAngle *}

{*********************************************************}
procedure CanvasTextOutAngle(c: TCanvas; x,y: Integer; d: Word; s: string);
{-------------------------------------------------------------
  PURPOSE
    To output rotated text in the same font as the font on
    the supplied canvas. The font must also be a scaleable
    font.

  INPUT PARAMETERS
    c - The canvas on which to output the text. The font for
      the canvas must be a scaleable font.
    x,y - The x,y screen coordinates you would normally
      supply the TextOut procedure.
    d - The angle in tenths of degrees. For example 10 would
      be 1 degree. 450 would be 45 degrees. 1 would be
      1/10 of a degree.
    s - The text to be output to the canvas.
  HISTORY
    6/18/1995 First version written by Curtis Keisler.
-------------------------------------------------------------}
var
  LogRec: TLOGFONT;     {* Storage area for font information *}
  OldFontHandle,        {* The old font handle *}
  NewFontHandle: HFONT; {* Temporary font handle *}

begin
  {* Get the current font information. We only want to modify the angle *}
  GetObject(c.Font.Handle, SizeOf(LogRec), Addr(LogRec));

  {* Modify the angle. "The angle, in tenths of a degrees, between the base
     line of a character and the x-axis." (Windows API Help file.)*}
  LogRec.lfEscapement := d;

  {* Create a new font handle using the modified old font handle *}
  NewFontHandle := CreateFontIndirect(LogRec);

  {* Save the old font handle! We have to put it back when we are done! *}
```

```
  OldFontHandle := SelectObject(c.Handle,NewFontHandle);

  {* Finally. Output the text! *}
  c.TextOut(x,y,s);

  {* Put the font back the way we found it! *}
  NewFontHandle := SelectObject(c.Handle,OldFontHandle);

  {* Delete the temporary (NewFontHandle) that we created *}
  DeleteObject(NewFontHandle);

end; {* CanvasTextOutAngle *}

procedure TForm1.PaintBox1Paint(Sender: TObject);
var
  degree,i,           {* Iteration variables *}
  midX,midY: integer; {* The middle of the form *}
  deg2Rad: Real;      {* Used to convert the degrees to radians *}

begin
  {* Used to convert the degrees to radians *}
  deg2Rad := PI / 180;

  {* Choose a scalable font! *}
  PaintBox1.Font.Name := 'Arial';
  PaintBox1.Font.Size := 12;

  {* Compute the center of the screen *}
  midX := PaintBox1.Width div 2;
  midY := PaintBox1.Height div 2;

  {* Draw 16 different angles *}
  for i := 0 to 15 do begin
    {* Compute each angle. i * (360 / 16) *}
    degree := round(i * 22.5);

    {*
       Draw the from the edges of a circle with radius of 50 pixels.
       I use -y because y is the opposite direction of the normal
       cartesian coordinate system that the sin() function is based
       upon.
       Multiply degree by 10 because the function wants 10ths of a
       degree.
    *}
    CanvasTextOutAngle(PaintBox1.Canvas,
                               round(midX + 50 * cos(degree * deg2Rad)),
                               round(midY - 50 * sin(degree * deg2Rad)),
                               degree*10,
                               'abcd');
  end; {* Next angel (i) *}

  {* Set the subsequent angle to 45 degrees (or 45 * 10 tenths = 450 *}
  CanvasSetTextAngle(PaintBox1.Canvas,450); {* 45 degrees *}

  {* Output text *}
  PaintBox1.Canvas.TextOut(10,100,'45 degrees');
  PaintBox1.Canvas.TextOut(40,100,'To the right 30 pixels');
```

```
  {* Set it back *}
  CanvasSetTextAngle(PaintBox1.Canvas,0);

end;

end.
```

```
unit Tlink;

{ TLink unit: doubly linked lists 5/22/95}
{ by Jeff Atwood, JAtwood159@AOL.COM. }
{ }
{ This unit can be used for stacks, deques, and free lists too.  }
{ }
{ I couldn't find a doubly-linked list implemented as an object ANYWHERE }
{ so I wrote it myself, after much trial, error, and poring over         }
{ obscure programming reference books. Hey-- it's not brain surgery, but }
{ pointers can be so naughty. }
{ }
{ For simplicity's sake, and to keep this a one-day project, I am only    }
{ storing simple integers in the cells. You can easily, easily change     }
{ that to any data type supported by Delphi including records. I would    }
{ NOT recommend trying to store a whole object with methods in there...   }
{ I couldn't get that to work. But if you can, E-Mail me. I don't know if }
{ it's even possible. }
{ }
{ There is one main object, which uses the "CELL" record type for each    }
{ entry in the list. I don't know how to hide the CELL record type from   }
{ the user, but it should be internal to this unit. The main object is    }
{ the TLink, which keeps track of the size, first, last, and current      }
{ cell records. You can move around in the list by using the Move methods }
{ and find using the Seek method. It's all fairly straightforward, look   }
{ at the demo form for examples, there are also comments in the code.     }
{ }
{ If you're feeling ambitious, I recommend you modify the cell record to  }
{ store pointers instead of integers. Don't forget to make copies of the  }
{ data, because if you point to the actual location, you're screwed when  }
{ the user destroys that instance. You gotta copy it... How many times    }
{ did I get burned by THAT one?? Also, it would be cool to turn this into }
{ a VCL component, if anyone wants to do that. }
{ }
{ This code is freeware. Please E-Mail me any cool additions, bug fixes,  }
{ rants, raves, etc. at JAtwood159@AOL.COM! Thanks for trying my code, I  }
{ hope it helps someone... }

interface

type

  CellPtr = ^Cell;
  Cell = record
    data: Integer;
    next: CellPtr;
    prev: CellPtr;
  end;

  TList = class(TObject)
  private
    top: CellPtr;
    bottom: CellPtr;
    current: CellPtr;
    size: Longint;
```

```
  public
    constructor create;
    destructor destroy; override;
    function IsEmpty: Boolean;
    function GetSize: Longint;
    procedure InsertBottom(item: Integer);
    procedure InsertTop(item: Integer);
    function InsertCurrent(item: Integer): Boolean;
    function FindFirst(item: Integer; var absLoc: longint): Boolean;
    function Delete: Boolean;
    function MoveFirst: Boolean;
    function MoveLast: Boolean;
    function MoveNext: Boolean;
    function MovePrevious: Boolean;
    function Seek(absLoc: longint): Boolean;
    function GetData(var item: Integer): Boolean;
  end;

implementation

{ set up the TList object with default values }
constructor TList.create;
begin
  inherited create;
  top := nil;
  bottom := nil;
  current := nil;
  size := 0;
end;

{ destroy the entire list, cell by cell }
destructor TList.destroy;
var
  curCell: CellPtr;
  nextCell: CellPtr;
begin
  curCell := top;
  while not (curCell = nil) do begin
    nextCell := curCell^.next;
    freemem(curCell, SizeOf(Cell));
    curCell := nextCell;
  end;
  top := nil;
  bottom := nil;
  current := nil;
  inherited destroy;
end;

{ returns true if the list has no cells }
function TList.isEmpty: Boolean;
begin
  result := (size = 0);
end;

{ returns number of cells in list }
function TList.getSize: Longint;
begin
```

```
  result := size;
end;

{ insert cell at bottom of list }
procedure TList.InsertBottom(item: Integer);
var
  newCell: CellPtr;
begin
  GetMem(newCell, Sizeof(Cell));
  newCell^.data := item;
  newCell^.prev := bottom;
  newCell^.next := nil;
  { special case: this is first cell added }
  if bottom = nil then
    top := newCell
  else
    bottom^.next := newCell;
  bottom := newCell;
  size := size + 1;
end;

{ insert cell at top of list }
procedure TList.InsertTop(item: Integer);
var
  newCell: CellPtr;
begin
  GetMem(newCell, Sizeof(Cell));
  newCell^.data := item;
  newCell^.prev := nil;
  newCell^.next := top;
  { special case: this is first cell added }
  if top = nil then
    bottom := newCell
  else
    top^.prev := newCell;
  top := newCell;
  size := size + 1;
end;

{ insert cell after current item }
function TList.InsertCurrent(item: Integer): Boolean;
var
  newCell: CellPtr;
begin
  if (current = nil) then
    result := False
  else begin
    GetMem(newCell, Sizeof(Cell));
    newCell^.data := item;
    newCell^.prev := current;
    newCell^.next := current^.next;
    { special case: current cell is last cell }
    if current^.next = nil then
      bottom := newCell
    else
      current^.next^.prev := newCell;
    current^.next := newCell;
```

```
    size := size + 1;
    result := True;
  end;
end;

{ Look for item in data field. Starts at top of list }
{ and looks at every item until a match is found.    }
{ if found, makes matched cell current, and returns  }
{ absolute location of match where 1 = top.          }
function TList.FindFirst(item: Integer; var absLoc: longint): Boolean;
var
  curCell: CellPtr;
  cnt: longInt;
begin
  result := False;
  curCell := top;
  cnt := 0;
  absLoc := 0;
  while not (curCell = nil) do begin
    cnt := cnt + 1;
    if curCell^.Data = item then begin
      absLoc := cnt;
      current := curCell;
      result := True;
      exit;
    end;
    curCell := curCell^.next;
  end;
end;


{ delete the current cell }
function TList.Delete: Boolean;
label
  exitDelete;
begin
  { we can only delete the current record }
  if current = nil then
    result := False
  else begin
    { see if list has one item }
    if size = 1 then begin
      top := nil;
      bottom := nil;
      goto exitDelete;
    end;
    { see if we're at the top of list }
    if current^.prev = nil then begin
      top := current^.next;
      top^.prev := nil;
      goto exitDelete;
    end;
    { see if we're at the bottom of list }
    if current^.next = nil then begin
      bottom := current^.prev;
      bottom^.next := nil;
      goto exitDelete;
```

```
    end;
    { we must be in middle of list of size > 1 }
    current^.prev^.next := current^.next;
    current^.next^.prev := current^.prev;
    goto exitDelete;
  end;

  { arrgh-- a goto! but this is a textbook goto! }
  exitDelete: begin
                result := True;
                freemem(current, SizeOf(Cell));
                current := nil;
                size := size - 1;
                if size = 0 then begin
                  top := nil;
                  bottom := nil;
                end;
              end;

end;

{ make first value in list current }
function TList.MoveFirst: Boolean;
begin
  if top = nil then
    result := False
  else begin
    current := top;
    result := True;
  end;
end;

{ make last value in list current }
function TList.MoveLast: Boolean;
begin
  if bottom = nil then
    result := False
  else begin
    current := bottom;
    result := True;
  end;
end;

{ make next value in list current }
function TList.MoveNext: Boolean;
begin
  if (current = nil) or (current^.next = nil) then
    result := False
  else begin
    current := current^.next;
    result := True;
  end
end;

{ make previous value in list current }
function TList.MovePrevious: Boolean;
begin
```

```
  if (current = nil) or (current^.prev = nil) then
    result := False
  else begin
    current := current^.prev;
    result := True;
  end;
end;

{ return data item from current list position }
function TList.GetData(var item: Integer): Boolean;
begin
  if (current = nil) then
    result := False
  else begin
    item := current^.data;
    result := True;
  end;
end;

{ make current the absolute cell N in the list }
{ where top = 1 }
function TList.Seek(absloc: longint): Boolean;
var
  curCell: CellPtr;
  cnt: longint;
begin
  result := False;
  if absloc <= 0 then
    exit;
  curCell := top;
  while not (curCell = nil) do begin
    cnt := cnt + 1;
    if cnt = absloc then begin
      current := curCell;
      result := True;
      exit;
    end;
    curCell := curCell^.next;
  end;
end;

end.
```

Glossary of Blonde Medicine

Artery.....Study of paintings
Bacteria.....Back door to a cafeteria
Barium.....What doctors do when treatment fails
Bowel.....A letter like A, E, I, O, or U
Caesarean Section.....A district in Rome
Catarrh.....Stringed instrument
Cat Scan.....Searching for kitty
Colic.....A sheep dog
Coma.....A punctuation mark
D&C.....Where Washington is
Dilate.....To live long
Enema.....Not a friend
Fester.....Quicker
Fibula.....Small lie
Hangnail.....Coat hook
High Colonic.....Jewish religious holiday
Impotent.....Distinguished; well known
Labor Pain.....Getting hurt at work Medical
Staff.....Doctor's cane
Morbid.....Higher offer
Nitrate.....Cheaper than a day rate
Node.....Was aware of
Outpatient.....Person who has fainted
Papsmear.....Fatherhood test
Pelvis.....Cousin of Elvis
Postoperative.....Letter carrier
Prostate.....Flat on your back Recovery
Room.....Place to do upholestry
Rectum.....Dang near killed him
Rheumatic.....Amourous
Secretion.....Hiding something
Seizure.....Roman emperor
Tablet.....A small table
Terminal Illness.....Getting sick at the airport
Tumor.....More than one
Urine.....Opposite of "You're Out"
Varicose.....Near by
Vein.....Conceited

Q: How do I do click and drag in a TListbox?

A:

```
{coolbox.pas}
{A note from the author:

   I needed to do some spiffy things with the listboxes so I wrote this.  If
it already exists, then great, but I couldn't find it.  With this small
program, you can multi-select items from ListBox1 and drag them to ListBox2.
No big deal, except that you can INSERT a selected item into a specific spot
in the list.

   The really cool thing here is that you can select an item in ListBox2 and
move it into a another spot within the list by using the arrows or by
dragging and dropping.  Again, I couldn't find any code that already did
this.  I hope you find this code useful, and if you do any other cool things
with it, please let me know.  This code is only lightly documented, so if you
have any questions, jsut ask.  One reason for the length of this program is
that it is relatively (dare I say it?) bug-free.  I'll probably pay for that
claim.

Richard Howard 71553,2544
Mei Technology Corporation
26 August 1995}

unit CoolBox;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Buttons, Spin;

type
  TForm1 = class(TForm)
    ListBox1: TListBox;
    ListBox2: TListBox;
    SpinButton1: TSpinButton; {for moving items in listbox2.}
    procedure ListBox2DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure ListBox2DragDrop(Sender, Source: TObject; X, Y: Integer);
    procedure ListBox2MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
      State: TDragState; var Accept: Boolean);
    procedure FormCreate(Sender: TObject);
    procedure MoveUp(Sender: TObject);
    procedure MoveDown(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
```

```
  Form1: TForm1;
  MoveSelectedItem : Integer; {the item in ListBox2 being moved}
  DnListBox1 : Boolean; {indicates which listbox to work with}
  DnListBox2 : Boolean; {indicates which listbox to work with}

implementation

{$R *.DFM}

procedure TForm1.ListBox2DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  if (Source is TListBox) then Accept := True;
  {because this is such a small program, 'ACCEPT := True' would work.  But
  larger programs need a little more control.}
end;

procedure TForm1.ListBox2DragDrop(Sender, Source: TObject; X, Y: Integer);
var
  i : Integer; {serves two purposes: 1) a counting variable for ListBox1,
               and 2) the item that the SELECTED item is being dropped on to
               in ListBox 2}
begin {procedure}
  {instructions for moving items from ListBox1 to ListBox2}
  if DnListBox1 then
  begin {if 1}
    for i := 0 to ListBox1.Items.Count - 1 do {look at ALL items in ListBox1}
    begin {for}
      if ListBox1.Selected[i] then
        ListBox2.Items.Insert(ListBox2.ItemAtPos(Point(X,Y), True),
ListBox1.Items[i]);
        ListBox1.Selected[i] := False; {after copying to LB2, UNselect it}
    end; {for}
   DnListBox1 := False;
  end; {if 1}

  {instructions for moving an item WITHIN ListBox2}
  if DnListBox2 then
  begin {if 2}
    {i = the item UNDER the moving, selected item}
    i := ListBox2.ItemAtPos(Point(X, Y), True);
    ListBox2.Items.Move(MoveSelectedItem, i); {puts the moved item into
place}
    ListBox2.ItemIndex := i; {select (highlight) the item you moved}
    if i = -1 then ListBox2.ItemIndex := ListBox2.Items.Count-1;
    DnListBox2 := False;
  end; {if 2}
end; {procedure}

procedure TForm1.ListBox2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin {procedure}
  DnListBox1 := False;{tells the OnDragDrop procedure which instructions to
use}
  DnListBox2 := True;{tells the OnDragDrop procedure which instructions to
use}
  if Button = mbLeft then
```

```
    if ListBox2.ItemAtPos(Point(X, Y), True) >= 0 then
      MoveSelectedItem := ListBox2.ItemIndex;
end; {procedure}

procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y: Integer;
  State: TDragState; var Accept: Boolean);
begin
  DnListBox1 := True;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  {I just threw these in here to look nice.  They can be pretty handy.}
  SendMessage(ListBox1.Handle, LB_SetHorizontalExtent, 1000, LongInt(0));
  SendMessage(ListBox2.Handle, LB_SetHorizontalExtent, 1000, LongInt(0));
end;

procedure TForm1.MoveUp(Sender: TObject);
var
  i : Integer;
begin {procedure}
  if ListBox2.ItemIndex > 0 then
  begin {if}
    i := ListBox2.ItemIndex;
    ListBox2.Items.Move(i, i-1);
    ListBox2.ItemIndex := i-1;
  end; {if}
end; {procedure}

procedure TForm1.MoveDown(Sender: TObject);
var
  i : Integer;
begin {procedure}
  if (ListBox2.ItemIndex < ListBox2.Items.Count-1) and
     (ListBox2.ItemIndex <> -1) then
  begin {if}
    i := ListBox2.ItemIndex;
    ListBox2.Items.Move(i, i+1);
    ListBox2.ItemIndex := i+1;
  end; {if}
end; {procedure}

end.

{*********************}

{coolproj.dpr}
program Coolproj;

uses
  Forms,
  Coolbox in 'COOLBOX.PAS' {Form1};

{$R *.RES}

begin
  Application.CreateForm(TForm1, Form1);
```

```
  Application.Run;
end.

{*********************}

{Coolbox.dfm}

object Form1: TForm1
  Left = 245
  Top = 163
  Width = 349
  Height = 253
  Caption = 'Form1'
  Font.Color = clWindowText
  Font.Height = -13
  Font.Name = 'System'
  Font.Style = []
  PixelsPerInch = 96
  OnCreate = FormCreate
  TextHeight = 16
  object ListBox1: TListBox
    Left = 16
    Top = 24
    Width = 129
    Height = 177
    DragMode = dmAutomatic
    ItemHeight = 16
    Items.Strings = (
      'List 1'
      'List 2'
      'List 3'
      'List 4'
      'List 5'
      'List 6')
    MultiSelect = True
    TabOrder = 0
    OnDragOver = ListBox1DragOver
  end
  object ListBox2: TListBox
    Left = 168
    Top = 24
    Width = 129
    Height = 177
    DragMode = dmAutomatic
    ItemHeight = 16
    Items.Strings = (
      'Test 1'
      'Test 2'
      'Test 3'
      'Test 4')
    TabOrder = 1
    OnDragDrop = ListBox2DragDrop
    OnDragOver = ListBox2DragOver
    OnMouseDown = ListBox2MouseDown
  end
  object SpinButton1: TSpinButton
    Left = 308
```

```
Top = 76
Width = 20
Height = 53
DownGlyph.Data = {
  7E040000424D7E04000000000000360400002800000009000000060000000100
  0800000000004800000000000000000000000000000000000000000000000000
  80000080000000808000800000008000800080800000C0C0C00061898D00A5BF
  C200000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  000000000000000000000000000000000000D2E0E100A4A0A000808080000000
  FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00030303030303
  0303030000000303030300030303030000000303030000000303030000000303
  0000000000030300000000300000000000000300000003030303030303030300
  0000}
TabOrder = 2
UpGlyph.Data = {
  7E040000424D7E04000000000000360400002800000009000000060000000100
  0800000000004800000000000000000000000000000000000000000000000000
  80000080000000808000800000008000800080800000C0C0C00061898D00A5BF
  C200000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
  0000000000000000000000000000000000000000000000000000000000000000
```

```
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      0000000000000000000000000000000000000000000000000000000000000000
      000000000000000000000000000000000000D2E0E100A4A0A000808080000000
      FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00030303030303
      0303030000000300000000000000030000000303000000000003030000000303
      0300000003030300000003030303000303030300000003030303030303030300
      0000}
    OnDownClick = MoveDown
    OnUpClick = MoveUp
  end
end
```

TTable

How do I use a TList to hold TTable variables?
How do I setup the column list in code for a dynamically created TTable?

Q: How do I use a TList to hold TTable variables?

A:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, DBTables, StdCtrls, Grids, DBGrids;

type
  TForm1 = class(TForm)
    Button1: TButton;
    DataSource1: TDataSource;
    DBGrid1: TDBGrid;
    Table1: TTable;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  list: TList;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  list := TList.create;
  with list do begin
    add(TTable.create(self));
    with TTable(items[count - 1]) do begin
      DatabaseName := 'DBDEMOS';
      TableName := 'customer.db';
      name := 'foo';
      open;
    end;
    Datasource1.DataSet := TTable(items[count - 1]);
  end;
end;

end.
```

TList

How do I use a TList to hold TTable variables?
How can I use a TList to hold variables?

Notice: This unit makes extensive use of array types that exceed the maximum "safe" size of 65519 bytes. While the compiler "allows" the declaration without error, application program should not ordinarily try to allocate memory to such structures. Segment wraparound problems can otherwise occur. For instance, most of these routines will not work on an array that "straddles" a segment boundary. If you notice carefully in this unit, the large arrays are used only for typecasting purposes, and no memory is allocated to them.

{ General-purpose array manipulation routines by J. W. Rider }

```
unit asorts;                                {Last modified: 09APR91}
interface

{ $define MONITOR} { <--- remove space before "$" to enable
                          monitoring various sorting routines }
{$ifdef MONITOR}
var monitor : procedure; { for monitoring results of sort }
procedure nullmonitor; { to turn monitoring off }
{$endif}


                      { *** Type definitions *** }

{ "comparefunc" -- comparison function argument for "qsort", "bsearch"
                   "lfind" and "lsearch"

  "icomparefunc"-- comparison function argument for "virtual" routines

  "swapproc"    -- exchange procedure for "virtual" routines

  "testfunc"    -- test function argument for "scan" }

type comparefunc = function (var a,b):longint;
     icomparefunc= function (a,b:longint):longint;
     swapproc    = procedure(a,b:longint);
     testfunc    = function (var a):boolean;


                    { *** C compatibility routines *** }

{ "qsort", "bsearch", "lfind", "lsearch" and "swab" are analogous to
   standard C functions of the same names }

{ quicksort the elements of an array }
procedure qsort(var base; length_base, sizeof_element:word;
                f:comparefunc);

{ binary search a sorted array for an element}
function bsearch(var key,base; length_base, sizeof_element:word;
                 f:comparefunc):word;

{ linear search an array for an element }
function lfind(var key,base; length_base, sizeof_element:word;
                 f:comparefunc):word;
```

```
{ linear search an array for an element; append if not found }
function lsearch(var key,base; length_base, sizeof_element:word;
                 f:comparefunc):word;

{ move one array of words to another, swapping bytes }
procedure swab(var source, destination; numwords:word);


       { *** "riderized" (i.e, generally nonstandard) routines *** }

{ the remaining routines generally have no standard implementation in other
  languages }

{ binary search a sorted array for an element.  Return the index of
  its location, or the negative of the index where it should be inserted }
function bfind(var key,base; length_base, sizeof_element:word;
                 f:comparefunc):longint;

{ inserts an element into a sorted array. }
function binsert(var key,base; length_base, sizeof_element:word;
                 f:comparefunc):word;

{ fibonacci search a sorted array; marginally faster than "bsearch" }
function fibsearch(var key,base; length_base, sizeof_element:word;
                   f:comparefunc):word;

{ fill an array with an element }
procedure fill(var key,destination; count, sizeof_element:word);

{ order an array by the "heapsort" algorithm }
procedure heapsort(var base; length_base, sizeof_element:word;
                   f:comparefunc);

{ return the address of variable as a longint value }
function longaddr(var x):longint;

{ a not-so-quick sorting routine, compare with qsort }
procedure naivesort(var base; length_base, sizeof_element:word;
                   f:comparefunc);

{ scan a subarray for the first element that meets a specific criteria }
function scan(var source; count, sizeof_element:word; f:testfunc):word;

{ order an array by the "selection sort" algorithm }
procedure selsort(var base; length_base, sizeof_element:word;
                  f:comparefunc);

{ order an array by the "shell sort" algorithm }
procedure shellsort(var base; length_base, sizeof_element:word;
                   f:comparefunc);

{ randomly permute the elements of an array }
procedure shuffle(var base; length_base, sizeof_element:word);

{ fill a subarray with an element }
procedure subfill(var key,destination;
```

```
                  count, sizeof_key,sizeof_element:word);

{ move subarray to array or array to subarray }
procedure submove(var source,destination;
                  count, sizeof_source, sizeof_destination:word);

{ swap two elements or variables of the same size }
procedure swap(var var1,var2; sizeof_element:word);

{ sort a "virtual" array by the quicksort algorithm }
procedure vqsort(length_base:longint; f:icomparefunc; s:swapproc);

{ sort a "virtual" array by using a selection sort algorithm }
procedure vselsort(length_base:longint; f:icomparefunc; s:swapproc);

{ randomly permute a "virtual" array }
procedure vshuffle(length_base:longint; s:swapproc);

{ move subarray to subarray }
procedure xsubmove(var source,destination;
             count,sizeof_source,sizeof_destination,sizeof_move:word);

implementation

function bfind(var key,base; length_base, sizeof_element:word;
                 f:comparefunc):longint;
var b:array [0..$fffe] of byte absolute base; l,h,x,c:longint;
begin if length_base>0 then begin
         l:=0; h:=pred(length_base);
         repeat
            x:=(l+h) shr 1; c:=f(key,b[x*sizeof_element]);
            if      c<0 then h:=pred(x)
            else if c>0 then l:=succ(x)
            else{if c=0 then}begin bfind:=succ(x); exit; end;
         until l>h;
         bfind:=-l; end
      else bfind:=0; end;


function binsert(var key,base;length_base,sizeof_element:word;
                   f:comparefunc):word;
var b:array [0..$fffe] of byte absolute base; x:longint;
begin
   x:=bfind(key,base,length_base,sizeof_element,f);
   if x<=0 then x:=-x else dec(x);
   move(b[x*sizeof_element],b[succ(x)*sizeof_element],
        (length_base-x)*sizeof_element);
   move(key,b[x*sizeof_element],sizeof_element);
   binsert:=succ(x); end;


function bsearch(var key,base; length_base, sizeof_element:word;
                 f:comparefunc):word;
var c:longint;
begin
   c:=bfind(key,base,length_base,sizeof_element,f);
   if c>0 then bsearch:=c
```

```
   else bsearch:=0; end;


function fibsearch(var key,base; length_base, sizeof_element:word;
                   f:comparefunc):word;
var b:array [0..$fffe] of byte absolute base; i,p,q,imax:word; t:longint;
begin
  imax:=length_base*sizeof_element;
  q:=0; p:=sizeof_element; i:=p+q; { set up for fibonacci sequencing }
  while imax>(i+p) do begin q:=p; p:=i; inc(i,q); end;
  dec(i,sizeof_element); {zero-base adjustment}
  while true do begin
        if i<imax then t:=f(key,b[i])
        else           t:=-1; { simulate "too big" for "out of range" }
        if t=0 then begin fibsearch:=succ(i div sizeof_element); exit end
        else if t<0 then
             if q=0 then begin fibsearch:=0; exit end
             else begin dec(i,q); q:=p-q; dec(p,q) end
        else { if t>0 then }
             if p=sizeof_element then begin fibsearch:=0; exit end
             else begin inc(i,q); dec(p,q); dec(q,p) end end end;


procedure fill(var key,destination; count, sizeof_element:word);
var b:array [0..$fffe] of byte absolute destination;
    x,moved:word;
begin if count>0 then begin
         move(key,destination,sizeof_element);
         moved:=1; dec(count); x:=sizeof_element;
         while count>moved do begin
            move(destination,b[x],x);
            dec(count,moved); moved:=moved shl 1; x:=x shl 1; end;
         move(destination,b[x],count*sizeof_element); end; end;


procedure heapsort(var base; length_base, sizeof_element:word;
                   f:comparefunc);
var b: array[0..$fffe] of byte absolute base;
    p:pointer; nx:longint; k,kx:word;

    procedure aux1(kx:word);

       procedure aux2; var jx:word;
       begin
          while kx<=(nx shr 1) do begin
                jx:=kx shl 1;
                if (jx<nx) and (f(b[jx],b[jx+sizeof_element])<0) then
                   inc(jx,sizeof_element);
                if f(p^,b[jx])>=0 then exit;
                move(b[jx],b[kx],sizeof_element);
                {$ifdef MONITOR}
                if @monitor<>nil then monitor;
                {$endif}
                kx:=jx end end;

    begin {aux1}
       move(b[kx],p^,sizeof_element);
```

```
        {$ifdef MONITOR}
        if @monitor<>nil then monitor;
        {$endif}
        aux2;
        move(p^,b[kx],sizeof_element);
        {$ifdef MONITOR}
        if @monitor<>nil then monitor;
        {$endif}
        end;

begin {heapsort}
   getmem(p,sizeof_element);
   nx:=pred(length_base)*sizeof_element;
   for k:=(length_base shr 1) downto 1 do aux1(pred(k)*sizeof_element);
   repeat
      swap(b[0],b[nx],sizeof_element);
      {$ifdef MONITOR}
      if @monitor<>nil then begin monitor; monitor; monitor end;
      {$endif}
      dec(nx,sizeof_element);
      aux1(0);
      until nx<=0;
   freemem(p,sizeof_element) end;

function lfind(var key,base; length_base, sizeof_element:word;
                  f:comparefunc):word;
var b:array [0..$fffe] of byte absolute base; i,j:word;
begin
   j:=0;
   for i:=1 to length_base do begin
       if f(key,b[j])=0 then begin lfind:=i; exit end;
       inc(j,sizeof_element); end;
   lfind:=0; end;


function longaddr(var x):longint;
begin longaddr:=(longint(seg(x)) shl 4) + ofs(x); end;


function lsearch(var key,base; length_base, sizeof_element:word;
                  f:comparefunc):word;
var b:array [0..$fffe] of byte absolute base; i:word;
begin
   i:=lfind(key,base,length_base,sizeof_element,f);
   if i=0 then begin
      move(key,b[length_base*sizeof_element],sizeof_element);
      lsearch:=succ(length_base); end
   else lsearch:=i; end;

procedure naivesort(var base; length_base, sizeof_element:word;
                    f:comparefunc);
var b: array[0..$fffe] of byte absolute base;
    i,j,l,r:word;
begin
i:=0;
for l:=1 to pred(length_base) do begin
   j:=i+sizeof_element;
```

```
   for r:=succ(l) to length_base do begin
       if f(b[i],b[j])>0 then begin
          swap(b[i],b[j],sizeof_element);
          {$ifdef MONITOR}
          if @monitor<>nil then monitor;
          {$endif}
          end;
       inc(j,sizeof_element); end;
   inc(i,sizeof_element); end; end;

{$ifdef MONITOR}
{ dummy "monitor" }
procedure nullmonitor; begin pointer((@@monitor)^):=NIL end;
{$endif}

procedure qsort(var base; length_base, sizeof_element:word;
                    f:comparefunc);
var b: array[0..$fffe] of byte absolute base;
    j:longint; x:word; { not preserved during recursion }

    procedure sort(l,r: word);
    var i:longint;
    begin
      i:=l*sizeof_element;
      while l<r do begin
         j:=r*sizeof_element;
         x:=((longint(l)+r) SHR 1)*sizeof_element;
         while i<j do begin
           while f(b[i],b[x])<0 do inc(i,sizeof_element);
           while f(b[x],b[j])<0 do dec(j,sizeof_element);
           if i<j then begin
              swap(b[i],b[j],sizeof_element);
              if i=x then x:=j else if j=x then x:=i;
              {$ifdef MONITOR}
              if @monitor<>nil then monitor;
              {$endif}
              end;
           if i<=j then begin
              inc(i,sizeof_element); dec(j,sizeof_element) end; end;
         if (l*sizeof_element)<j then sort(l,j div sizeof_element);
         l:=i div sizeof_element; end; end;

begin sort(0,pred(length_base)); end; {procedure qsort}


function scan(var source; count, sizeof_element:word; f:testfunc):word;
var b:array[0..$fffe] of byte absolute source;
    i,j:word;
begin
   j:=0;
   for i:=1 to count do begin
       if f(b[j]) then begin scan:=i; exit; end;
       inc(j,sizeof_element); end;
   scan:=0; end;


procedure selsort(var base; length_base, sizeof_element:word;
```

```
                    f:comparefunc);
var b:array[0..$fffe] of byte absolute base;
    i,ix,j,jx,k,kx:word;
begin
ix:=0;
for i:=1 to pred(length_base) do begin
    kx:=ix; jx:=ix;
    for j:=succ(i) to length_base do begin
        inc(jx,sizeof_element);
        if f(b[jx],b[kx])<0 then kx:=jx end;
    if kx<>ix then begin
       swap(b[kx],b[ix],sizeof_element);
       {$ifdef MONITOR}
       if @monitor<>nil then monitor;
       {$endif}
       end; inc(ix,sizeof_element) end; end;


procedure shellsort(var base; length_base, sizeof_element:word;
                    f:comparefunc);
var b:array[0..$fffe] of byte absolute base;
    p:pointer; h,jx:longint; i,hx,ix:word;

    procedure aux; begin
        while f(b[jx-hx],p^)>0 do begin
            move(b[jx-hx],b[jx],length_base); dec(jx,hx);
            {$ifdef MONITOR}
            if @monitor<>nil then monitor;
            {$endif}
            if jx<hx then exit end end;

begin if length_base>0 then begin
   getmem(p,length_base);
   if p<>nil then begin
      h:=1; repeat h:=3*h+1 until h>length_base;
      repeat
         h:=h div 3; hx:=h*sizeof_element; ix:=hx;
         for i:=succ(h) to length_base do begin
             move(b[ix],p^,sizeof_element);
             {$ifdef MONITOR}
             if @monitor<>nil then monitor;
             {$endif}
             jx:=ix; aux;
             if jx<>ix then move(p^,b[jx],sizeof_element);
             {$ifdef MONITOR}
             if @monitor<>nil then monitor;
             {$endif}
             inc(ix,sizeof_element) end;
         until h=1;
      freemem(p,length_base) end end end;


procedure shuffle(var base; length_base, sizeof_element:word);
var b: array[0..$fffe] of byte absolute base;
    i,ix,j,jx:word;
begin if length_base>0 then
  for i:=pred(length_base) downto 1 do begin
```

```
      ix:=i*sizeof_element;
      j:=random(succ(i));
      if i<>j then begin
         jx:=j*sizeof_element;
         swap(b[ix],b[jx],sizeof_element); end; end; end;


procedure subfill(var key,destination;
                  count, sizeof_key,sizeof_element:word);
var b:array [0..$fffe] of byte absolute destination; i,j:word;
begin
  j:=0;
  for i:=1 to count do begin
      move(key,b[j],sizeof_key);
      inc(j,sizeof_element); end; end;


procedure submove(var source, destination;
                  count, sizeof_source,sizeof_destination:word);
var sm:word;
begin if sizeof_source=sizeof_destination then
         move(source,destination,count*sizeof_source)
      else begin
         if sizeof_source>sizeof_destination then sm:=sizeof_destination
         else                                     sm:=sizeof_source;
      xsubmove(source,destination,
               count,sizeof_source,sizeof_destination,sm); end; end;


procedure swab(var source, destination; numwords:word);
var a: array [1..$7fff] of word absolute source;
    b: array [1..$7fff] of word absolute destination;
    i:word;

begin if longaddr(source)>=longaddr(destination) then
         for i:=1 to numwords do b[i]:=system.swap(a[i])
      else
         for i:=numwords downto 1 do b[i]:=system.swap(a[i]) end;


procedure swap(var var1,var2; sizeof_element:word);
type chunk = array [0..$f] of byte;
var a:array [0..$fffe] of byte absolute var1;
    b:array [0..$fffe] of byte absolute var2;
    ac: array [1..$fff] of chunk absolute var1;
    bc: array [1..$fff] of chunk absolute var2;
    c:chunk; { swap buffer }
    k:byte; x:word;

    procedure swapchunk(var e,f:chunk);
    begin c:=e; e:=f; f:=c; end;

    procedure swapbytes(var e,f; len:byte);
    begin move(e,c,len); move(f,e,len); move(c,f,len); end;

begin
   for k:=1 to (sizeof_element shr 4) do swapchunk(ac[k],bc[k]);
```

```
   k:=(sizeof_element and $f);
   if k>0 then begin
      x:=(sizeof_element and $fff0); swapbytes(a[x],b[x],k); end; end;


procedure vqsort(length_base:longint; f:icomparefunc; s:swapproc);
var j,x:longint; { not preserved during recursion }

    procedure sort(l,r:longint);
    var i:longint;
    begin
      i:=l; j:=r;
      x:=(i+j) SHR 1;
      while i<j do begin
        while f(i,x)<0 do inc(i);
        while f(x,j)<0 do dec(j);
        if i<j then begin
           s(i,j);
           if i=x then x:=j else if j=x then x:=i; end;
        if i<=j then begin inc(i); dec(j) end; end;
      if l<j then sort(l,j);
      if i<r then sort(i,r); end;

begin sort(1,length_base); end; {procedure vqsort}


procedure vselsort(length_base:longint; f:icomparefunc; s:swapproc);
var i,j,k:longint;
begin for i:=1 to pred(length_base) do begin
    k:=i;
    for j:=succ(i) to length_base do if f(j,k)<0 then k:=j;
    if k<>i then s(k,i) end end;


procedure vshuffle(length_base:longint; s:swapproc);
var i,j:longint;
begin for i:=length_base downto 2 do begin
      j:=succ(random(i));
      if i<>j then begin s(i,j); end; end; end;


procedure xsubmove(var source,destination;
             count,sizeof_source,sizeof_destination,sizeof_move:word);
var a:array [0..$fffe] of byte absolute destination;
    b:array [0..$fffe] of byte absolute source;
    i,j,k:word; r:boolean;
begin
   r:=longaddr(source)>=longaddr(destination);
   if r then begin j:=0; k:=0; end
   else begin
      j:=pred(count)*sizeof_destination; k:=pred(count)*sizeof_source; end;
   for i:=1 to count do begin
       move(b[k],a[j],sizeof_move);
       if r then begin
          inc(j,sizeof_destination); inc(k,sizeof_source) end
       else begin
          dec(j,sizeof_destination); dec(k,sizeof_source) end; end; end;
```

```
{$ifdef MONITOR}
begin {initialization}
nullmonitor;
{$endif}

end.
```

Notes on using the above unit:

ASORTS

General-purpose routines for sorting, searching and moving arrays of arbitrary elements.

by J. W. Rider

ASORTS provides the Turbo Pascal programmer with type definitions, functions and procedures to handle a variety of array sorting and searching tasks. Several of these are analogous to functions of the same name in the C programming language:

qsort -- sort an array

bsearch -- do a binary search of a sorted array for an element that satisfies some key

lfind -- do a sequential ("linear") search of an unsorted (or sorted) array for an element that satisfies some key

lsearch -- do a sequential search of an unsorted (or sorted) array for an element that satisfies some key. If the element is not found, then it is appended to the end of the array.

(Non-C programmers: please note that "bsearch" and "lfind" -- not "lsearch"! -- do the equivalent task for sorted and unsorted arrays, respectively. It would make more sense to have "bfind" to search through a sorted array, and make "bsearch" insert a missing element into the array at the right location. However, these routines are provided for compatibility in converting C programs.)

swab -- move one portion of memory to another, swapping high and low order bytes in the process

There are some routines that are provided for utility above and beyond what is available in standard C libraries:

bfind -- do a binary search of a sorted array for an element that satisfies some key. (This differs from "bsearch" in that is "bfind" reports the "insertion index" if the key element is not found.)

binsert -- do a binary search of a sorted array for an element that satisfies some key. If the element is not found, then the key element is inserted in the proper location in the array to maintain the sorted order.

fibsearch-- searches an ordered array using a "fibonacci" search algorithm.

fill -- moves a single item into all of the elements of an array

heapsort -- order an array by the "heapsort" algorithm.

longaddr -- returns the address of a variable expressed as a longint value

naivesort-- a particularly inefficient sorting algorithm that the author has been known to use when "qsort" was not available. (The use of "naivesort" in applications is NOT recommended!)

scan -- returns the index of the first element in an array that meets a specific criterion.

selsort -- "selection" sorting, another sorting routine.

shellsort-- yet another sorting routine, different from the rest.

shuffle -- randomly permutes ("unsorts") the elements of an array

subfill -- moves a single item into all of the elements of a subarray of the base array

submove -- moves the elements of the source array into a subarray of the destination array (or, the elements of a subarray of the source to the destination array)

swap -- Exchanges two array elements or variables.

(NOTE! The "Asorts.Swap" procedure replaces the default "System.Swap" which simply exchanges the high and low-order bytes of a two byte expression.)

vqsort -- a quicksort algorithm for "virtual" arrays. "VQSort" does not presume any kind of structure inherent within the "thing" being sorted. Instead, "VQSort" provides the sorting logic to other routines that will perform the actual comparisons and exchanges.

vselsort -- a "virtual" selection sort.

vshuffle -- a "virtual" shuffler.

xsubmove -- moves the elements of a subarray of the source array into the elements of a subarray of the destination.

While these routines are provided to be of assistance to the programmer, the number of different searching and sorting algorithms does raise the issue of how to select any one algorithm for the programmer to employ.
Unfortunately, that is very application dependent. For general purpose array sorting, you would do well to compare the "qsort" and "shellsort" routines on your actual data. For sorting of typed files, I'd suggest a
variant of "VSelSort".

CONCEPTS

The ARRAYs to be manipulated are passed as "untyped vars" to these routines. (In the "Interface" section, these arrays are called "base", "source" or "destination".) These routines will treat the ARRAYs as if they were declared to be of type:

ArrayType = ARRAY [1..MaxArray] OF ElementType

WARNING! It is *very* important to avoid defining an array so that the last byte is in a memory segment different from the first byte. As long as you never declare an array larger than 65519, or $10000-15, bytes, it should not be a problem.

Each ELEMENT of the ARRAY is presumed to be fixed size, and this size must be passed to the routines. (In the "Interface" section, if an ELEMENT needs to passed directly to a routine as an argument, it is passed as an untyped var called "key".) Also, the number of elements in the ARRAY must also be passed. For
instance, to fill an array of real numbers with 0:

```
var RealArray : array [1..10] of Real;
    x         : real;

x:=0;
fill(RealArray,10,sizeof(real),x);
```

A SUBARRAY is a "byte-slice" of an array. For instance, if "ElementType" is an "array [1..8] of byte", then a "subelement" would be any contiguous collection of bytes within the element, like 3,4 and 5. The SUBARRAY would be the collection of all of the subelements stored in an ARRAY. If "ElementType" is a record of fields, then a "subelement" would be any contiguous group of fields.

For sorting and searching, a COMPARISON FUNCTION must be passed to the routines. COMPARISON FUNCTIONs take two untyped vars, return a longint value, and must be declared "far" at compilation. (DIFFERENT! In C, only an integer-sized value is returned.) For instance, to sort the array of real numbers declared earlier:

```
function RealCompare(var a,b):longint; far;
begin
   if real(a)>real(b) then realcompare:=1
   else if real(a)<real(b) then realcompare:=-1
   else realcompare:=0;
end;

qsort(realarray,10,sizeof(real),realcompare);
```

"Virtual" arrays are data structures whose elements can be accessed indirectly by an index. For instance, information that is physically stored in multiple arrays might be sorted by a key in just one of the arrays. ASORTS provides a few routines for handling such "virtual" arrays. "VQSort" and "VSelSort" will provide the sorting logic for ordering the arrays. "VShuffle" will similarly "unorder" the array. To sort an array of "DBRec" with respect to an array of integer priorities, declare:

```
var Array1 : array [1..MaxDBRec] of DBRec;
    Priority: array [1..MaxDBRec] of integer;

function ComparePriority(a,b:longint):longint; far;
begin ComparePriority:=longint(Priority[a])-Priority[b] end;

procedure SwapPriDBRec(a,b:longint); far;
begin asorts.swap(Priority[a],Priority[b],sizeof(integer));
      asorts.swap(Array1[a],Array1[b],sizeof(DBRec)); end;
```

and sort the two arrays with:

```
vqsort(MaxDBRec,ComparePriority,SwapPriDBRec);
```

```
INTERFACE

function bfind(var key{:element}, base{:array};
                  length_base, sizeof_element:word;
                  f:comparefunc):longint;
```

Searches a sorted array for a "key" element. Return the index of its location, or the negative of the index of the largest element in the array that is smaller than the key (i.e., the element that you want to insert the new element after).

```
function binsert(var key{:element}, base{:array};
                    length_base, sizeof_element:word;
```

```
                    f:comparefunc):longint;
```

WARNING: This routine overwrites memory if used incorrectly.

Inserts the key element into the correct position of an ordered array. Unlike "lsearch", which only adds the key if it's not already present, "binsert" ALWAYS inserts a new element into the array. "Binsert" returns the index where the element is inserted.

```
function bsearch(var key{:element}, base{:array};
                     length_base, sizeof_element:word;
                     f:comparefunc):word;
```

"Bsearch" attempts to locate the "key" element within the previously SORTED array "base". If successful, the index of the found element is returned; otherwise, 0 is returned to indicate that the element is not present.

```
type comparefunc = function (var a,b{:element}):longint; {far;}
```

Declares the type of the comparison function to be passed to sorting and searching routines. CompareFunc's are user-defined functions that takes two arguments a and b. A and B must be typeless in the declaration, but otherwise are of the same type as the elements of the "base" array. For "qsort" and "bsearch", the function needs to return a negative integer if "A<B"; a positive integer if "A>B"; and 0 if "A=B". For "lfind" and "lsearch", the function needs to return 0 if "A=B", and some non-zero integer if "A<>B".

```
function fibsearch(var key{:element}, base{:array};
                       length_base, sizeof_element:word;
                       f:comparefunc):word;
```

"Fibsearch" attempts to locate the "key" element within the previously SORTED array "base". If successful, the index of the found element is returned; otherwise, 0 is returned to indicate that the element is not present. (This procedure is included because a user asked about the algorithm on one of Borland's CompuServe Forums. It looks like an interesting alternative to "bsearch", but I have not run extensive comparisons.)

```
procedure fill(var key{:element}, destination{:array};
                   count, sizeof_element:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Moves the "key" element to the first "count" indices in the "destination" array.

```
procedure heapsort(var base {:array};
                   length_base, sizeof_element:word;
                   f:comparefunc);
```

WARNING: This routine overwrites memory if used incorrectly.

"HeapSort" reorders the elements of the "base" array using a heapsort algorithm (like, you expected something else?). The function "f" is used to compare two elements of the array, and must return a negative number if the first argument is "less than" the second, a postive number if the first argument is "greater than" the second, and zero if the two arguments are "equal".

```
type icomparefunc = function (a,b:longint):longint;
```

Declares the type of the comparison function to be passed as an argument for sorting "virtual" arrays. Instead of passing the elements to be compared as is done for "comparefunc", ICompareFunc's use the location of the elements.

```
function lfind(var key{:element}, base{:array};
                   length_base, sizeof_element:word;
                   f:comparefunc):word;
```

"Lfind" attempts to locate the "key" element within the array "base". If successful, the index of the found element is returned; otherwise, 0 is returnd to indicate the element is not present.

```
function longaddr (var x): longint;
```

Returns the address of a variable expressed as a longint value so that address can be used for comparisons, etc.

```
function lsearch(var key{:element}, base{:array};
                     length_base, sizeof_element:word;
                     f:comparefunc):word;
```

WARNING: This routine overwrites memory if used incorrectly.

Does a linear search of the "base" array, looking for the "key" element. If the key element is found, "lsearch" returns the index of the array. *** NOTE! *** Otherwise, the key element is appended to the end of the array. It is the programmer's responsibility to ensure that "sizeof_element" bytes are available at the end of the array and that the count of contained elements is adjusted. To avoid the append, use "lfind" instead.

```
var monitor : procedure;
procedure nullmonitor;
```

"Monitor" and "NullMonitor" were debugging devices developed in the process of putting together the ASORTS unit. They can be optionally declared by defining a compilation variable "MONITOR". Every time that the "Qsort" algorithm swaps a pair of array elements, the "monitor" procedure is called. This will allow the user to watch the progress of the sort. This is of marginal practical value, and by default, these two identifiers are not defined. Calling "NullMonitor" will turn off monitoring even if the "monitor" procedure variable is defined.

```
procedure naivesort(var base {:array};
                    length_base, sizeof_element:word;
                    f:comparefunc);
```

WARNING: This routine slowly overwrites memory if used incorrectly.

"NaiveSort" also reorders the elements or an array. However, it does it more slowly than "QSort". The use of the procedure is not recommended. It is provided for comparison only. (i.e., don't waste your time trying to improve upon it. It's only here for comic relief.)

```
procedure qsort(var base {:array};
                    length_base, sizeof_element:word;
                    f:comparefunc);
```

WARNING: This routine overwrites memory if used incorrectly.

"Qsort" reorders the elements of the "base" array using a quicksort algorithm. The function "f" is used to compare two elements of the array, and must return a negative number if the first argument is "less than" the second, a postive number if the first argument is "greater than" the second, and zero if the two arguments are "equal".

```
function scan(var source; count, sizeof_element:word;
                  f:testfunc):word;
```

"Scan" does a linear search of the "source" array and returns the index of the first element that satisfies the test function "f". If no element is found, "scan" returns 0.

```
procedure selsort(var base{:array};
                  length_base, sizeof_element:word;
                  f:comparefunc);
```

WARNING: This routine overwrites memory if used incorrectly.

"SelSort" reorders the elements of the "base" array using a selection sort algorithm. This algorithm minimizes the number of times that each element is moved, but will maximize the of comparisons. For most applications, the comparisons take more time than the exchanges, so you are not likely to want to use this algorithm for ordinary array sorting. See also, "VSelSort".

```
procedure shellsort(var base{:array};
                    length_base, sizeof_element:word;
                    f:comparefunc);
```

WARNING: This routine overwrites memory if used incorrectly.

"ShellSort" reorders the elements of the "base" array using a sort routine first defined by an individual whose last name was Shell. In concept, the algorithm is an insertion-type sort (as opposed to exchange-type sort (of which "qsort" and "selsort" are examples). This turns out to be a very efficient sorting routine for in-memory sorting.

```
procedure shuffle(var base{:array}; length_base, sizeof_element:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Randomly permutes ("unsorts") the elements of an array. The SYSTEM "Randomize" procedure should be called at least once in a program that shuffles an array.

```
procedure subfill(var key{:subelement}, destination{:subarray};
                  count, sizeof_key, sizeof_element:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Partially fills an array with the "key". The address of "Destination" should be the address of the first byte in the
subarray. The portion of the array outside of the subarray is left unchanged.

```
procedure submove(var source{:[sub]array},
                      destination{:[sub]array};
                      count,
                      sizeof_source_element,
                      sizeof_destination_element:word);
```

WARNING: This routine overwrites memory if used incorrectly.

If the size of the source elements are smaller than that of the destination elements,

moves the first "count" elements of the source into a subarray of the same size in destination. If larger, only moves that portion of the source that will fill the first "count" elements of the destination. If equal in size, does a simple "move" of the bytes.

```
procedure swab(var source, destination; numwords:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Moves the contents of source to the destination, swapping high and low order bytes in the process. (Note: while this is provided for C compatibility, the third argument is used differently. In C, the third argument is the number of bytes to move and is expected to be even. Here, the third argument is the number of byte-pairs to move.)

```
procedure swap(var var1,var2; sizeof_element:word);
```

WARNING: This routine overwrites memory if used incorrectly.

"Swap" exchanges the contents of the variable "var1" with the contents of the variable "var2". (Note: this is a redefinition of the "swap" function defined in the System unit.)

```
type swapproc = procedure (a,b:longint); {far;}
```

Declares the type of the exchange procedure that is passed as an argument to "selsort" and other virtual exchange sort algorithms.

```
type testfunc = function (var a):boolean; {far;}
```

Declares the type of the criterion function to be passed to the "scan" function. The actual TestFunc should expect an array element to be passed through "a" and return true if the element satisfies the criteria. Otherwise, it should return false.

```
procedure vqsort(length_base:longint; f:icomparefunc; s:swapproc);
```

"VQSort" provides the logic to do a quicksort of an indexed entity (a "virtual" array), but depends upon the user-defined routines "f" and "s" to do the actual work of accessing specific elements in the array, comparing and exchanging them. This makes VQSort useful for sorting elements when they are stored in something other than a single contiguous array.

```
procedure vselsort(length_base:longint; f:icomparefunc; s:swapproc);
```

"VSelSort" provides the logic to do a selection sort of an indexed entity (a "virtual" array), but depends upon the user-defined routines "f" and "s" to do the actual work of accessing specific elements in the array, comparing and exchanging them. As mentioned in the description for "SelSort", the algorithm minimizes the number of exchanges of elements required to put something into sorted order, but makes a large number of comparisons to do so. This would make "VSelSort" useful for something like sorting an external file of larger records based upon integer keys stored in an array in memory.

```
procedure vshuffle(length_base:longint; s:swapproc);
```

Of course, what fun is there in being able to order virtual arrays if we can't mix all the elements up again?

```
procedure xsubmove(var source{:subarray}, destination{:subarray};
                       count, sizeof_source_element,
                       sizeof_destination_element,
                       sizeof_subelement:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Moves a subarray from the source to the destination. The size of the subelements is presumed to be the same in both subarrays. "XsubMove" does not check to make sure that the sizes are consistent.

REFERENCES

Knuth, The Art of Computer Programming, Sorting and Searching.

Press, et al., Numerical Recipes.

Sedgewick, Algorithms.

ASORTS

General-purpose routines for sorting, searching and moving arrays of arbitrary elements.

ASORTS provides the Turbo Pascal programmer with type definitions, functions and procedures to handle a variety of array sorting and searching tasks. Several of these are analogous to functions of the same name in the C programming language:

qsort -- sort an array

bsearch -- do a binary search of a sorted array for an element that satisfies some key

lfind -- do a sequential ("linear") search of an unsorted (or sorted) array for an element that satisfies some key

lsearch -- do a sequential search of an unsorted (or sorted) array for an element that satisfies some key. If the element is not found, then it is appended to the end of the array.

(Non-C programmers: please note that "bsearch" and "lfind" -- not "lsearch"! -- do the equivalent task for sorted and unsorted arrays, respectively. It would make more sense to have "bfind" to search through a sorted array, and make "bsearch" insert a missing element into the array at the right location. However, these routines are provided for compatibility in converting C programs.)

swab -- move one portion of memory to another, swapping high and low order bytes in the process

There are some routines that are provided for utility above and beyond what is available in standard C libraries:

bfind -- do a binary search of a sorted array for an element that satisfies some key. (This differs from "bsearch" in that is "bfind" reports the "insertion index" if the key element is not found.)

binsert -- do a binary search of a sorted array for an element that satisfies some key. If the element is not found, then the key element is inserted in the proper location in the array to maintain the sorted order.

fibsearch-- searches an ordered array using a "fibonacci" search algorithm.

fill -- moves a single item into all of the elements of an array

heapsort -- order an array by the "heapsort" algorithm.

longaddr -- returns the address of a variable expressed as a longint value

naivesort-- a particularly inefficient sorting algorithm that the author has been known to use when "qsort" was not available. (The use of "naivesort" in applications is NOT recommended!)

scan -- returns the index of the first element in an array that meets a specific criterion.

selsort -- "selection" sorting, another sorting routine.

shellsort-- yet another sorting routine, different from the rest.

shuffle -- randomly permutes ("unsorts") the elements of an array

subfill -- moves a single item into all of the elements of a subarray of the base array

submove -- moves the elements of the source array into a subarray of the destination array (or, the elements of a subarray of the source to the destination array)

swap -- Exchanges two array elements or variables.

(NOTE! The "Asorts.Swap" procedure replaces the default "System.Swap" which simply exchanges the high and low-order bytes of a two byte expression.)

vqsort -- a quicksort algorithm for "virtual" arrays. "VQSort" does not presume any kind of structure inherent within the "thing" being sorted. Instead, "VQSort" provides the sorting logic to other routines that will perform the actual comparisons and exchanges.

vselsort -- a "virtual" selection sort.

vshuffle -- a "virtual" shuffler.

xsubmove -- moves the elements of a subarray of the source array into the elements of a subarray of the destination.

While these routines are provided to be of assistance to the programmer, the number of

different searching and sorting algorithms does raise the issue of how to select any one algorithm for the programmer to employ. Unfortunately, that is very application dependent. For general purpose array sorting, you would do well to compare the "qsort" and "shellsort" routines on your actual data. For sorting of typed files, I'd suggest a
variant of "VSelSort".

CONCEPTS

The ARRAYs to be manipulated are passed as "untyped vars" to these routines. (In the "Interface" section, these arrays are called "base", "source" or "destination".) These routines will treat the ARRAYs as if they were declared to be of type:

ArrayType = ARRAY [1..MaxArray] OF ElementType

WARNING! It is *very* important to avoid defining an array so that the last byte is in a memory segment different from the first byte. As long as you never declare an array larger than 65519, or $10000-15, bytes, it should not be a problem.

Each ELEMENT of the ARRAY is presumed to be fixed size, and this size must be passed to the routines. (In the "Interface" section, if an ELEMENT needs to passed directly to a routine as an argument, it is passed as an untyped var called "key".) Also, the number of elements in the ARRAY must also be passed. For instance, to fill an array of real numbers with 0:

```
var
  RealArray : array [1..10] of Real;
  x         : real;

    x:=0;
    fill(RealArray,10,sizeof(real),x);
```

A SUBARRAY is a "byte-slice" of an array. For instance, if "ElementType" is an "array [1..8] of byte", then a "subelement" would be any contiguous collection of bytes within the element, like 3,4 and 5. The SUBARRAY would be the collection of all of the subelements stored in an ARRAY. If "ElementType" is a record of fields, then a "subelement" would be any contigous group of fields.

For sorting and searching, a COMPARISON FUNCTION must be passed to the routines. COMPARISON FUNCTIONs take two untyped vars, return a longint value, and must be declared "far" at compilation. (DIFFERENT! In C, only an integer-sized value is returned.) For instance, to sort the array of real numbers declared earlier:

```
    function RealCompare(var a,b):longint; far;
    begin
       if real(a)>real(b) then realcompare:=1
       else if real(a)<real(b) then realcompare:=-1
```

```
      else realcompare:=0;
    end;

    qsort(realarray,10,sizeof(real),realcompare);
```

"Virtual" arrays are data structures whose elements can be accessed indirectly by an index. For instance, information that is physically stored in multiple arrays might be sorted by a key in just one of the arrays. ASORTS provides a few routines for handling such "virtual" arrays. "VQSort" and "VSelSort" will provide the sorting logic for ordering the arrays. "VShuffle" will similarly "unorder" the array. To sort an array of "DBRec" with respect to an array of integer priorities, declare:

```
   var Array1 : array [1..MaxDBRec] of DBRec;
       Priority: array [1..MaxDBRec] of integer;

   function ComparePriority(a,b:longint):longint; far;
   begin ComparePriority:=longint(Priority[a])-Priority[b] end;

   procedure SwapPriDBRec(a,b:longint); far;
   begin asorts.swap(Priority[a],Priority[b],sizeof(integer));
         asorts.swap(Array1[a],Array1[b],sizeof(DBRec)); end;
```

and sort the two arrays with:

```
   vqsort(MaxDBRec,ComparePriority,SwapPriDBRec);
```

```
INTERFACE

function bfind(var key{:element}, base{:array};
                  length_base, sizeof_element:word;
                  f:comparefunc):longint;
```

Searches a sorted array for a "key" element. Return the index of its location, or the negative of the index of the largest element in the array that is smaller than the key (i.e., the element that you want to insert the new element after).

```
function binsert(var key{:element}, base{:array};
                    length_base, sizeof_element:word;
                    f:comparefunc):longint;
```

WARNING: This routine overwrites memory if used incorrectly.

Inserts the key element into the correct position of an ordered array. Unlike "lsearch", which only adds the key if it's not already present, "binsert" ALWAYS inserts a new element into the array. "Binsert" returns the index where the element is inserted.

```
function bsearch(var key{:element}, base{:array};
```

```
                    length_base, sizeof_element:word;
                    f:comparefunc):word;
```

"Bsearch" attempts to locate the "key" element within the previously SORTED array "base". If successful, the index of the found element is returned; otherwise, 0 is returned to indicate that the element is not present.

```
type comparefunc = function (var a,b{:element}):longint; {far;}
```

Declares the type of the comparison function to be passed to sorting and searching routines. CompareFunc's are user-defined functions that takes two arguments a and b. A and B must be typeless in the declaration, but otherwise are of the same type as the elements of the "base" array. For "qsort" and "bsearch", the function needs to return a negative integer if "A<B"; a positive integer if "A>B"; and 0 if "A=B". For "lfind" and "lsearch", the function needs to return 0 if "A=B", and some non-zero integer if "A<>B".

```
function fibsearch(var key{:element}, base{:array};
                       length_base, sizeof_element:word;
                       f:comparefunc):word;
```

"Fibsearch" attempts to locate the "key" element within the previously SORTED array "base". If successful, the index of the found element is returned; otherwise, 0 is returned to indicate that the element is not present. (This procedure is included because a user asked about the algorithm on one of Borland's CompuServe Forums. It looks like an interesting alternative to "bsearch", but I have not run extensive comparisons.)

```
procedure fill(var key{:element}, destination{:array};
                    count, sizeof_element:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Moves the "key" element to the first "count" indices in the "destination" array.

```
procedure heapsort(var base {:array};
                    length_base, sizeof_element:word;
                    f:comparefunc);
```

WARNING: This routine overwrites memory if used incorrectly.

"HeapSort" reorders the elements of the "base" array using a heapsort algorithm (like, you expected something else?). The function "f" is used to compare two elements of the array, and must return a negative number if the first argument is "less than" the second, a postive number if the first argument is "greater than" the second,

and zero if the two arguments are "equal".

```
type icomparefunc = function (a,b:longint):longint;
```

Declares the type of the comparison function to be passed as an argument for sorting "virtual" arrays. Instead of passing the elements to be compared as is done for "comparefunc", ICompareFunc's use the location of the elements.

```
function lfind(var key{:element}, base{:array};
                  length_base, sizeof_element:word;
                  f:comparefunc):word;
```

"Lfind" attempts to locate the "key" element within the array "base". If successful, the index of the found element is returned; otherwise, 0 is returnd to indicate the element is not present.

```
function longaddr (var x): longint;
```

Returns the address of a variable expressed as a longint value so that address can be used for comparisons, etc.

```
function lsearch(var key{:element}, base{:array};
                    length_base, sizeof_element:word;
                    f:comparefunc):word;
```

WARNING: This routine overwrites memory if used incorrectly.

Does a linear search of the "base" array, looking for the "key" element. If the key element is found, "lsearch" returns the index of the array. *** NOTE! *** Otherwise, the key element is appended to the end of the array. It is the programmer's responsibility to ensure that "sizeof_element" bytes are available at the end of the array and that the count of contained elements is adjusted. To avoid the append, use "lfind" instead.

```
var monitor : procedure;
procedure nullmonitor;
```

"Monitor" and "NullMonitor" were debugging devices developed in the process of putting together the ASORTS unit. They can be optionally declared by defining a compilation variable "MONITOR". Every time that the "Qsort" algorithm swaps a pair of array elements, the "monitor" procedure is called. This will allow the user to watch the progress of the sort. This is of marginal practical value, and by default, these two

identifiers are not defined. Calling "NullMonitor" will turn off monitoring even if the "monitor" procedure variable is defined.

```
procedure naivesort(var base {:array};
                    length_base, sizeof_element:word;
                    f:comparefunc);
```

WARNING: This routine slowly overwrites memory if used incorrectly.

"NaiveSort" also reorders the elements or an array. However, it does it more slowly than "QSort". The use of the procedure is not recommended. It is provided for comparison only. (i.e., don't waste your time trying to improve upon it. It's only here for comic relief.)

```
procedure qsort(var base {:array};
                    length_base, sizeof_element:word;
                    f:comparefunc);
```

WARNING: This routine overwrites memory if used incorrectly.

"Qsort" reorders the elements of the "base" array using a quicksort algorithm. The function "f" is used to compare two elements of the array, and must return a negative number if the first argument is "less than" the second, a postive number if the first argument is "greater than" the second, and zero if the two arguments are "equal".

```
function scan(var source; count, sizeof_element:word;
                  f:testfunc):word;
```

"Scan" does a linear search of the "source" array and returns the index of the first element that satisfies the
test function "f". If no element is found, "scan" returns 0. procedure

```
selsort(var base{:array}; length_base, sizeof_element:word; f:comparefunc);
```

WARNING: This routine overwrites memory if used incorrectly.

"SelSort" reorders the elements of the "base" array using a selection sort algorithm. This algorithm minimizes the number of times that each element is moved, but will maximize the of comparisons. For most applications, the comparisons take more time than the exchanges, so you are not likely to want to use this algorithm for ordinary array sorting. See also, "VSelSort".

```
procedure shellsort(var base{:array};
                    length_base, sizeof_element:word;
                    f:comparefunc);
```

WARNING: This routine overwrites memory if used incorrectly.

"ShellSort" reorders the elements of the "base" array using a sort routine first defined by an individual whose last name was Shell. In concept, the algorithm is an insertion-type sort (as opposed to exchange-type sort (of which "qsort" and "selsort" are examples). This turns out to be a very efficient sorting routine for in-memory sorting.

```
procedure shuffle(var base{:array};
                      length_base, sizeof_element:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Randomly permutes ("unsorts") the elements of an array. The SYSTEM "Randomize" procedure should be called at least once in a program that shuffles an array.

```
procedure subfill(var key{:subelement}, destination{:subarray};
                  count, sizeof_key, sizeof_element:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Partially fills an array with the "key". The address of "Destination" should be the address of the first byte in the subarray. The portion of the array outside of the subarray is left unchanged.

```
procedure submove(var source{:[sub]array},
                      destination{:[sub]array};
                      count,
                      sizeof_source_element,
                      sizeof_destination_element:word);
```

WARNING: This routine overwrites memory if used incorrectly.

If the size of the source elements are smaller than that of the destination elements, moves the first "count" elements of the source into a subarray of the same size in destination. If larger, only moves that portion of the source that will fill the first "count" elements of the destination. If equal in size, does a simple "move" of the bytes.

```
procedure swab(var source, destination; numwords:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Moves the contents of source to the destination, swapping high and low order bytes

in the process. (Note: while this is provided for C compatibility, the third argument is used differently. In C, the third argument is the number of bytes to move and is expected to be even. Here, the third argument is the number of byte-pairs to move.)

procedure swap(var var1,var2; sizeof_element:word);

WARNING: This routine overwrites memory if used incorrectly.

"Swap" exchanges the contents of the variable "var1" with the contents of the variable "var2". (Note: this is a redefinition of the "swap" function defined in the System unit.)

```
type swapproc = procedure (a,b:longint); {far;}
```

Declares the type of the exchange procedure that is passed as an argument to "selsort" and other virtual exchange sort algorithms.

```
type testfunc = function (var a):boolean; {far;}
```

Declares the type of the criterion function to be passed to the "scan" function. The actual TestFunc should expect an array element to be passed through "a" and return true if the element satisfies the criteria. Otherwise, it should return false.

```
procedure vqsort(length_base:longint; f:icomparefunc; s:swapproc);
```

"VQSort" provides the logic to do a quicksort of an indexed entity (a "virtual" array), but depends upon the user-defined routines "f" and "s" to do the actual work of accessing specific elements in the array, comparing and exchanging them. This makes VQSort useful for sorting elements when they are stored in something other than a single contiguous array.

```
procedure vselsort(length_base:longint; f:icomparefunc; s:swapproc);
```

"VSelSort" provides the logic to do a selection sort of an indexed entity (a "virtual" array), but depends upon the user-defined routines "f" and "s" to do the actual work of accessing specific elements in the array, comparing and exchanging them. As mentioned in the description for "SelSort", the algorithm minimizes the number of exchanges of elements required to put something into sorted order, but makes a large number of comparisons to do so. This would make "VSelSort" useful for something like sorting an external file of larger records based upon integer keys stored in an array in memory.

```
procedure vshuffle(length_base:longint; s:swapproc);
```

Of course, what fun is there in being able to order virtual arrays if we can't mix all the elements up again?

```
procedure xsubmove(var source{:subarray}, destination{:subarray};
                       count, sizeof_source_element,
                       sizeof_destination_element,
                       sizeof_subelement:word);
```

WARNING: This routine overwrites memory if used incorrectly.

Moves a subarray from the source to the destination. The size of the subelements is presumed to be the same in both subarrays. "XsubMove" does not check to make sure that the sizes are consistent.

REFERENCES

Knuth, The Art of Computer Programming, Sorting and Searching.

Press, et al., Numerical Recipes.

Sedgewick, Algorithms.

```
program cmpsrt;
{ Compares various sorting algorithms on random arrays }

{   -- as distributed, 100 random arrays of 100 integers are
       generated and sorted against 6 different sorting methods.
       When completed, the program reports the total number
       of comparisons and exchanges that each method used.  }

{ assumes "MONITOR" defined in ASORTS.TPU }

{ NOTE: This program will not compile correctly with ASORTS.PAS
  as you find it in the distribution package. ASORTS.PAS must be
  compiled with the symbol "MONITOR" defined at compilation time.
  There are several ways to do this:

  1. Edit ASORTS.PAS, removing the space between the left brace
and the dollar sign in the "$define MONITOR" compiler directive.
Then recompile the unit to a TPU.

  2. Load the ASORTS.PAS in the IDE. Under Options/Compiler,
define the Conditional symbol "MONITOR". Then Compile/Build the
TPU.

  3. With the command line compiler, TPC, include a "/DMONITOR"
option on the command line.

While the MONITOR symbol is essential to compiling this
demonstration program, you probably will not wish to incur the
additional overhead when using ASORTS for your production
programs. The original ASORTS.PAS, without the MONITOR symbol
defined, is already configured to sort without external
monitoring. }

uses asorts;

const
  max = 100; { <-- change this to compare effect of array length }

type
  list = array[1..max] of longint;

var
  data,data2: ^list;
  i,j: word;

const
  numsorts = 6;
  sortnames: array [1..numsorts] of string[9] = (
      'HeapSort','QSort','SelSort','ShellSort',
      'VQSort','VSelSort');


var
  currentsort:word;
  compcount:array [1..numsorts] of longint;
  swapcount:array [1..numsorts] of longint;
```

```
function longintcomp(var a,b):longint; far;
var int1: longint absolute a;
    int2: longint absolute b;
begin
  inc(compcount[currentsort]);
  if int1<int2 then longintcomp:=-1
  else if int1=int2 then longintcomp:=0
  else longintcomp:=1;
end;

procedure swapcounter; far;
begin
  inc(swapcount[currentsort])
end;

function icompdata(a,b:longint):longint; far;
begin
   inc(compcount[currentsort]);
   if data^[a]<data^[b] then icompdata:=-1
   else if data^[a]=data^[b] then icompdata:=0
   else icompdata:=1;
end;

procedure iswapdata(a,b:longint); far;
var c:longint;
begin
   inc(swapcount[currentsort]);
   c:=data^[a]; data^[a]:=data^[b]; data^[b]:=c
end;

procedure checksort;
var i:word;
begin
  for i:=max downto 2 do
      if data^[i]<data^[i-1] then begin
         writeln;
         writeln('Sort algorithm ',sortnames[currentsort],' failed!');
         exit;
      end;
end;


begin {tstsrt2}
  asorts.monitor:=swapcounter; { "MONITOR" must be defined in ASORTS.PAS }
  new(data);
  new(data2);
  for i:=1 to numsorts do begin
      compcount[i]:=0; swapcount[i]:=0;
  end;
  Randomize;

  writeln;
  for j:= 1 to 100 do begin
    write(#13,j,'':15);

  { this could be changed to compare the effect of different
    "pre-orderings" on the sorting algorithms.  I'm not sure
```

```
    what substitute you could make -- "almost sorted" arrays
    are harder to generate than "random" arrays. }

  for i:=1 to max do begin
      data2^[i]:=longint(random($7fff))-$3fff;
   end;

  for currentsort:=1 to numsorts do begin
      write(#13,j,sortnames[currentsort]:10,'':9);
      data^:=data2^;
      case currentsort of
           1: heapsort(data^,max,sizeof(longint),longintcomp);
           2: qsort(data^,max,sizeof(longint),longintcomp);
           3: selsort(data^,max,sizeof(longint),longintcomp);
           4: shellsort(data^,max,sizeof(longint),longintcomp);
           5: vqsort(max,icompdata,iswapdata);
           6: vselsort(max,icompdata,iswapdata);
       end;
       checksort;
    end;
  end;

writeln(#13,'':25);
writeln('Sort Method':15,'Comparisons':15,'Exchanges':15);
for currentsort:=1 to numsorts do begin
    write(sortnames[currentsort]:15,
          compcount[currentsort]:15);
    if currentsort in [1,4] then
       { for heap and shell, count three moves as a swap }
       writeln((swapcount[currentsort] div 3):15)
    else
       writeln(swapcount[currentsort]:15); end; end.
```

```
program tstsrt;
{ Exercises most of the facilities of the ASORTS unit }

uses asorts;

{ $define MONITOR} {<-- MONITOR needs to be defined in ASORTS.PAS
                        also }

const
  max = 19; { must be byte-sized }

type
  list = array[1..max] of integer;

var
  data,data2: list;
  i: integer;
  b:integer;
  sortcount,qsc:integer;

const
  bs : set of byte = [];
  cmax:word=0;

function intcomp(var a,b):longint; far;
var int1: integer absolute a;
    int2: integer absolute b;
begin
  if int1<int2 then intcomp:=-1
  else if int1=int2 then intcomp:=0
  else intcomp:=1;
end;

procedure datamon; far; var i:byte; begin
  inc(sortcount); for i:=1 to cmax do write(data[i]:4); writeln; end;

begin {tstsrt}
  Writeln('Now generating up to ',max,' random numbers...');
  Randomize;
  for i:=1 to max do begin
      b:=random(256);

      { If "b" has already been generated, "lsearch" should find it;
        otherwise "lsearch" should add it to the end. }

      if b in bs then
         if lsearch(b,data,cmax,sizeof(integer),intcomp)>cmax then
            writeln('Error in "lsearch": element not found ',b)
         else
      else if lsearch(b,data,cmax,sizeof(integer),intcomp)<=cmax then
         writeln('Error in "lsearch": invalid element inserted ',b)
      else begin bs:=bs + [b]; inc(cmax) end; end;
  datamon; write(' (Press return)'); readln;

  Writeln('Now sorting ',cmax,' random numbers...');

{$ifdef MONITOR}  { This will let us keep track of the how the sort is
```

```
                  progressing }
{ !!! MONITOR must be defined in ASORTS for this to work }
asorts.monitor:=datamon;
data2:=data; {for subsequent comparison}
sortcount:=0;
{$endif}

  qsort(data,cmax,sizeof(integer),intcomp);

{$ifdef MONITOR}
qsc:=sortcount; sortcount:=0;
writeln('Now let''s see how the NaiveSort compares to the QuickSort that');
writeln('   we just finished');
  write(' (Press return)'); readln;

data:=data2;
naivesort(data,cmax,sizeof(integer),intcomp);

writeln('And the score is: QSort:',qsc,', vs NaiveSort:',sortcount, 'swaps');

{ This is not important for this program, but if you call "qsort" from
  multiple locations, what the procedure does might not always make sense.
  So, we turn the monitor off. }

asorts.nullmonitor;

{$else}
  datamon;
{$endif}
  write(' (Press return)'); readln;
  writeln('Now searching for ',cmax,' sorted numbers...');
  for i:=0 to 255 do begin
      { All byte values will be sought.  It would be an error for
        "bsearch" to find a value that was not inserted into the
        array.  Also, to fail to find a value that was inserted
        into the array }
      if bsearch(i,data,cmax,sizeof(integer),intcomp)=0 then
         if i in bs then
            Writeln('Error in "bsearch": element not found ',i)
         else
      else if not (i in bs) then
         writeln('Error in "bsearch": invalid element found ',i)
      else if i<>data[bsearch(i,data,cmax,sizeof(integer),intcomp)] then
         writeln('Error in "bsearch": wrong index returned');

      if fibsearch(i,data,cmax,sizeof(integer),intcomp)=0 then
         if i in bs then
            Writeln('Error in "fibsearch": element not found ',i)
         else
      else if not (i in bs) then
         writeln('Error in "fibsearch": invalid element found ',i)
      else if i<>data[fibsearch(i,data,cmax,sizeof(integer),intcomp)] then
         writeln('Error in "fibsearch": wrong index returned');
      end;
  writeln('....Search complete.');

  { We are now going to exercise the submove and xsubmove procedures
```

```
    in ASORTS.  For the simple submove, the first five elements of "data"
    are going to be moved to "pseudo" array that starts at data[9].  The
    target array is presumed to consist of elements that are two integers
    in size.  So, the moved values will wind up in every other integer
    displayed.}
  writeln('Now doing a simple array submove ... (1->9,2->11,...5->17)');
  submove(data[1],data[9],5,2,4);
  datamon; write(' (Press return)'); readln;


  { For the more general "xsubmove", we are going to presume that the
    source array is also two integers per element, but we only want to move
    the first element.  (The source and target are overlayed in this example
    so that what is seen are pairs of numbers appear in "data".) }
  writeln('Now doing a complex array submove ...(1->2,3->4,...9->10)');
  xsubmove(data[1],data[2],5,4,4,2);
  datamon; write(' (Press return)'); readln;

  { Now put 255 into the even slots }
  writeln('Now interlacing "255" into the array');
  b:=255;
  subfill(b,data[2],9,2,4);
  datamon; write(' (Press return)'); readln;

  { Now put 0 everywhere }
  writeln('Now filling array with 0''s...');
  b:=0;
  fill(b,data,19,sizeof(integer));
  datamon; write(' (Press return)'); readln;

  { Now let's tryout the binary insertion procedure }
  writeln('Now creating a new, sorted random array ... ');
  cmax:=0; bs:=[];
  for i:=1 to max do begin
      b:=random(256);
      b:=binsert(b,data,cmax,sizeof(integer),intcomp);
      inc(cmax); end;
  datamon; write(' (Press return)'); readln;

  { Now, let's check out the swab procedure }
  writeln('Now swabbing the array...results should be the same');
  swab(data,data,19); datamon;
  swab(data,data,19); datamon;
  write(' (Press return)'); readln;

  { That only leaves "shuffle" to be exercised, so let's mess up everything
    before we exit. }
  writeln('Now shuffling ',cmax,' numbers...');
  shuffle(data,cmax,sizeof(integer));
  datamon; write(' (Press return)'); readln;

  writeln('Done.');
end.
```

Gray Paper - Delphi Property Editors for Beginners

Updated with corrections

CIS: 74131,2175
September 10, 1995

Warning!

The information found in this document is an account of the author's research into property editors in Delphi. I am not directly associated with Borland, nor have I any covert knowledge beyond the content of the manuals and the extent of my own experimentation. Wow, what a disclaimer, eh? Probably just an attention grabber.

Introduction

One of the more interesting aspects of Borland's Delphi®1 is the relative ease with which one can create and customize components in the visual environment. Whether or not the component is actually visible at run time, many of the component's properties and methods can be interacted with during development. It is this interaction (via the Object Inspector) that makes the process of development so much more friendly.

When a developer decides that a component is the best vehicle for a program requirement, (s)he must also consider how the component is presented within the development environment. Delphi provides many standard property editors for the bulk of data types available in Object Pascal. However, there is no generic way for Delphi to know how to edit custom data types or objects. Borland has provided a way for developers (through the DsgnInfc unit) to register property editors customized for programmer-defined data types.

Although Borland is usually fairly good at providing documentation on all aspects of their compilers, the rush to release Delphi (thanks, by the way) precluded their production of a complete and user-friendly guide to creating custom Property Editors. They have, of course, offered to sell you the source code for the Visual Component Library, which contains many examples. It may seem strange, but I have not opted to obtain the source code. I will, of course, but the price is as much as the whole desktop package. Some of you may not be able to afford it either, so with a little prompting from my friends, I have decided to pass on to you what I have learned. The scope of this paper has been limited to property editors using custom dialog windows and programmer defined data types.

Questions, Questions ... but Few Answers.

How do Property Editors work? Well, that was my first question, anyway. Obviously, when you select the [...] button next to a property of type **`TString`**, you see a dialog appear containing an editor. You type in lines of text and ... *viola* ... the property is set. So what goes on behind the scenes? How did Delphi know to display the editor dialog? How did the text entered into the editor get into the property? And how does the text get stored with the form so it is reloaded when the form is loaded?

It is important to answer these questions or you will not be successful at creating custom

property editors for yourself. Unfortunately, the questions aren't answered very well in the documentation. So let me have a crack at how I have answered these questions for myself. Just one caveat: I am guessing. Well, a bit, anyway.

First, consider the following assumptions one can make about property editors:

1. The Delphi design-mode must know that the editor exists, and what data type it is designed to handle.
2. The property editor must exist, and must be designed to know how to edit the data type.
3. The property editor must have access to the property that is being edited.
4. The property editor probably has a number of special attributes that determine how it is invoked.
5. There are probably certain things about property editors that no one is willing to tell us. (... at least not without forking over some cash.)

Let's assume that we have a custom data type we want to edit. The declaration for this data type may be as follows:

```
Type    TMyDataType = Record
              Name : String;
              Amount : Real;
              Items : Integer;
              Paid : Boolean;
        End;
```

In our component object, a property exists of this data type. Say ...

```
MyComponent = Class(TComponent)
Private
   FRecord : TMyDataType;
Published
   Property TheRecord : TMyDataType Read FRecord Write FRecord;
End;
```

Normally, of course, if the data type was one that Delphi had a property editor for it would have no trouble handling the property. However, in this case, the data type TMyDataType is unknown. There is no editor available to edit the property. What is even worse, is that Delphi will not allow you to publish the property. Record data types are not supportable. You must create an object from your data type. Here is how you might implement your Record as a Property:

```
Type
   TMyDataType = Record
         Name : String;
         Amount : Real;
         Items : Integer;
         Paid : Boolean;
   End;

   TMyDataObject = Class(TObject)
   Private
         MyData : TMyDataType;
```

```
  Public
        Constructor Create( Name : String );
        Destructor Free;
        Procedure SetRecord( var Rec : TMyDataType );
        Procedure GetRecord( var Rec : TMyDataType );
  End;
```

Now the data type is an object. After you have written the Constructor, Destructor and the access methods SetRecord and GetRecord, you can move on to the next step. Delphi will now accept the following component (simplified, of course):

```
MyComponent = Class(TComponent)
  Private
        FRecord : TMyDataObject;
  Published
        Property TheRecord : TMyDataType Read FRecord Write FRecord;
End;
```

Now you might think that you could just replace the "`Write FRecord`" clause of the Property statement with something like SetFRecord. Well, you certainly could but this would not represent a property editor. No matter how hard you try to load a form with your special editor you will be unsuccessful. This is not how property editors work. First you must create a dialog form capable of editing and/or displaying your new data type. Then you need to create the property editor object that will open the dialog form and interact with the data.

Creating the Property Editor Dialog

There are two major steps:

1. Create a Dialog Form that can create and edit your data type.
2. Create a descendant of `TPropertyEditor` that can interact with your Dialog Form.

So, for step 1 you just create a new form and program the actions required for the form to handle the fields in your record. It can be simple with labels and edit boxes; or it can be complex with all types of data validation, etc. Make it pretty or make it ugly ... but you gotta make it. Let assume that you have created a form and called it TMyDataForm. Lets also assume that the (partial) declaration for this form is:

```
Type
  TMyDataForm = Class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    strName: TEdit;
    strAmount: TEdit;
    strItems: TEdit;
    chkPaid: TCheckBox;
     btnOK: TButton;
```

```
    btnCancel: TButton;
    Procedure FormCreate(Sender: TObject);
  Private
    { Private declarations }
  Public
    { Public declarations }
    Function GetName : String;
    Function GetAmount : Real;
    Function GetItems : Integer;
    Function GetPaid : Boolean;
  End;
```

The idea here is that the form will accept the Name, Amount, Items and Paid field data, then accept the values entered after pressing the btnOK button. The public functions GetName, GetAmount, GetItems and GetPaid return the appropriate data type for each field by translating (where necessary) the string version of the data.

Now assume you have the editor dialog form ready for action. How do you turn it into a Property Editor? You need to create a descendant of **`TPropertyEditor`** that knows how to display the form and obtain the data.

Creating the Property Editor Object

In the same file as your form, create an entry for the property editor object:

```
TMyDataEditor = Class(TPropertyEditor)
  Function  GetAttributes : TPropertyAttributes;  Override;
  Function  GetValue      : String;               Override;
  Procedure Edit;                                 Override;
end;
```

The reasoning here is to create a descendant of **`TPropertyEditor`** that overrides the default methods you must customize for your editor. The method GetAttributes is required to return a set of property attributes for this property editor. Let's just settle for the standard attributes for a dialog:

```
Function  TMyDataEditor.GetAttributes : TPropertyAttributes;
begin
  Result := inherited GetAttributes +
     [paDialog] -[paSubProperties];
end;
```

Note the use of inherited. It is used to obtain the attributes inherited from **`TPropertyEditor`**. Furthermore, I have added the paDialog attribute because this editor is a Dialog. There are a number of attributes to consider, and you can find a list of attributes later in this text.

When the property is displayed in the Object Inspector, it calls the GetValue method to obtain a string representation of the contents of the data. You can put anything that seems reasonable here, but it is likely with records you will only want to show the data type. To instruct Delphi to display the data type (TStrings data types are displayed as (TStrings)), try the following code in

GetValue:

```
Function  TMyDataEditor.GetValue : String;
begin
   Result := '(TMyDataObject)';
end;
```

Of course, you can also use a more generic method that looks up the property name by dereferencing the GetPropType record pointer. **GetPropType** is a pointer to a record of information about the data type being edited. Try the following code, since it will always display the correct type name:

```
Function  TMyDataEditor.GetValue : String;
begin
  FmtStr(Result, '(%s)', [ GetPropType^.Name ]);
end;
```

Yet again, you might decide that you would rather see the Name field as a representative of the record. If so, you will need to locate the contents of the Name field in your data structure object. To do this, you need to get a reference to the **`TMyDataObject`** object being edited. It would be rather stupid if you could not do this, so Delphi defines the GetOrdValue identifier. You can use this to locate the data, and access it. You will need to typecast this reference, since GetOrdValue is a generic pointer. In this case you might try:

```
Function  TMyDataEditor.GetValue : String;
begin
  Result := TMyDataObject(GetOrdValue).GetName;
end;
```

Now the contents of the Name field will be displayed in the Object Inspector.

The Property Editor's Edit Method

Now you must program the Edit method. This is where Delphi will go when it needs to edit any property of type **`TMyDataObject`**. The Edit method is a procedure. Since it does not return anything, your code must handle it internally. Remember that the GetValue method is used to return a string that represents the contents of the property. Delphi will call it whenever it needs to display the property in the Object Inspector. Now, I will give you an example of how you might implement an editor for the **`TMyDataObject`** data type. Please not that the code has been simplified for clarity. Normally you would want to trap errors.

```
Procedure TMyDataEditor.Edit;
var
  MyData : TMyDataObject;  { reference to object being edited }
  MyRecord : TMyDataType;  { temp holding record }
Begin
  { create the form }
  with TMyDataForm.Create( Application ) do begin
    { if the OK button was pressed }
    if ShowModal = mrOK then begin
```

```
        { fill the temporary record }
        with MyRecord do begin
          Name := GetName;
          Amount := GetAmount;
          Items := GetItems;
          Paid := GetPaid;
        end;
        { set the record in the object being edited }
        MyData := TMyDataObject(GetOrdValue);
        MyData.SetRecord( MyRecord );
      end;
      { free the form }
      Free;
    end;
End;
```

So, the edit method creates the Dialog Form, shows it modal, fills the temporary record with the edited fields then uses the SetRecord method of the **`TMyDataObject`** being edited to set the object's record. That's it. The property is edited.

Registering the Editor

That was a fair amount of work, wasn't it. Yet it's not so bad once you get through it. The last very important step is to register the newly created property editor. It is probably best to include the registration code in the Register procedure provided by the component you are writing. Assuming that this is what you want to do, place the following code into the Register procedure of your component:

```
RegisterPropertyEditor( TypeInfo(TMyDataObject), Nil,'',
  TMyDataEditor );
```

This instructs Delphi to register our editor TMyDataEditor as the editor for properties of type TMyDataObject when the component is loaded. Assuming all went well .... That's it.

Some Details

Of course, there are always details. Consider the following list of things to remember:

1. Ensure that your dialog form returns **`mrOK`** when the OK button is pressed.
2. Remember to include **`DsgnIntf`** in the Uses statement of your editor dialog form source file.
3. Keep the Editor form and the TPropertyEditor code in the same file.
4. Put the RegisterPropertyEditor statement in the component code file's Register procedure.
5. Include plenty of **`Try..Finally`** statements in your code. Errors can be hard to find.

You have seen a few specific items used to access information by your property editor. They are:

GetPropType

This method returns a pointer to a record of type information available at run time. The following PTypeInfo definition is defined:

```
Type
  PTypeInfo = ^TTypeInfo;
  TTypeInfo = record
    Kind: TTypeKind;
    Name: string;
    TypeData: TTypeData;
  End;
```

In our example, I only used the Name field to retrieve the type's proper name.

GetOrdValue

Actually, this is one of four methods provided for use in the GetValue method. Since our data type is ordinal, we will use GetOrdValue. However, GetFloatValue, GetStringValue and GetMethodValue are provided (see Component Writer's Help).

Inherited GetAttributes

I used this inherited method to obtain the default attributes of the TPropertyEditor object. As you saw, I used paDialog, paReadOnly and paSubProperties, as this indicates that the property can use a dialog to edit the entire property, it's read-only in the Object Inspector and there are no Sub-Properties to edit.. That's true, in our case. The complete list of options is:

paAutoUpdate
: Calls the SetValue method after each change in the value, rather than after the entire value is approved.

paDialog
: The editor can display a dialog box for editing the entire property. The Edit method opens the dialog box.

paReadOnly
: The property is read-only in the Object Inspector. This is used for sets and fonts, which the user cannot type directly.

paMultiSelect
: The property should appear when the user selects multiple components. Most property editors allow multiple selection. The notable exception is the editor for the Name property.

paSortList
: The Object Inspector should sort the list of values alphabetically.

paSubProperties
: The property is an object that has subproperties that can display. The GetProperties method handles the subproperty lists.

paValueList
: The editor can give a list of enumerated values. The GetValues method builds the list.

Oh, and read the Component Writer's Guide[2] and Help as though it really means something. Read the manual first, then look at the help topics in the Help file. They can be confusing, but what isn't. Try the code in this document. No, it is not complete. There is, however, enough for you to get started. *There is no cerebral thrill greater than the thrill of a programmer who has just solved a confusing problem.*[3]

In Closing

This paper has in no way exhausted the subject of Property Editors. In fact, it has only scraped the surface of what you can do. In the case of About dialog boxes, you can create property editors that do nothing but display dialogs. Great for registration dialogs, too. Well, maybe.

I originally wrote this paper as a guide for myself as I uncovered more about property editors. Since then, on the behest of some colleges, I have rewritten it to be a bit more general and possibly instructive. I can not vouch for the accuracy of anything written here, except to say ... It Works for Me.

About the Author

You'll probably want to skip this.

The author is the Chief Architect, 'Keeper of the Model' and Principal Developer for the Data Warehouse project at BC Tel (what a mouthful). He lives in beautiful British Columbia, Canada (Hollywood of the North) with his Wife Julia and Dog Buster. Fortunately, both children are grown and living on their own.

References and Notes

---

[1] Delphi is Copyright © 1995 by Borland International

[2] I think the word ‘guide’ as used here should be taken with a grain of salt. Perhaps the word ‘overview’ is more appropriate.

[3] ‘Programming and the Eureka Experience’ by Greg Truesdell

Q: How do I put components into a TDBGrid?

A: I saw this on compuserve and it is great!

HOW TO PUT COMPONENTS INTO A GRID

This article and the accompanying code shows how to put just about any component into a cell on a grid. By component I mean anything from a simple combobox to a more complicated dialog box. The techniques described below to anything that is termed a visual component. If you can put it into a form you can probably put it into a grid.

There are no new ideas here, in fact, the basic technique simply mimics what the DBGrid does internally. The idea is to float a control over the grid. Inside DBGrid is a TDBEdit that moves around the grid. It's that TDBEdit that you key you data into. The rest of the unfocused cells are really just pictures. What you will learn here, is how to float any type of visual control/component around the grid.

COMPONENT #1 - TDBLOOKUPCOMBO

You need a form with a DBGrid in it. So start an new project and drop a DBGrid into the main form.

Next drop in a TTable and set it's Alias to DBDEMOS, TableName to GRIDDATA.DB and set the Active property to True. Drop in a DataSource and set it's DataSet property to point to Table1. Go back to the grid and point it's DataSource property to DataSource1. The data from GRIDDATA.DB should appear in your grid..

The first control we are going to put into the grid is a TDBLookupCombo so we need a second table for the lookup. Drop a second TTable into the form. Set it's Alias also to DBDEMOS, TableName to CUSTOMER.DB and Active to True. Drop in a second data source and set its DataSet to Table2.

Now go get a TDBLookupCombo from the Data Controls pallet and drop it any where on the form, it doesn't matter where since it will usually be invisible or floating over the grid. Set the LookuoCombo's properties as follows.

DataSource	DataSource1
DataField	CustNo
LookupSource	DataSource2
LookupField	CustNo
LookupDisplay	CustNo {you can change it to Company later but keep it custno for now)

So far it's been nothing but boring point and click. Now let's do some coding.

The first thing you need to do is make sure that DBLookupCombo you put into the form is invisible when you run the app. So select Form1 into Object Inspector goto the Events tab and double click on the onCreate event. You should now have the shell for the onCreate event displayed on your screen.

```
procedure TForm1.FormCreate(Sender: TObject);
begin

end;
```

Set the LookupCombo's visible property to False as follows.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DBLookupCombo1.Visible := False;
end;
```

Those of you who are paying attention are probably asking why I didn't just set this in the Object Inspector for the component. Actually, you could have. Personally, I like to initialize properties that change at run time in the code. I set static properties that don't change as the program runs in the object inspector. I think it makes the code easier to read.

Now we to be able to move this control around the grid. Specifically we want it to automatically appear as you either cursor or click into the column labeled DBLookupCombo. This involves defining two events for the grid, OnDrawDataCell and OnColExit. First lets do OnDrawDataCell. Double click on the grid's OnDrawDataCell event in the Object Inspector and fill in the code as follows.

```
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
  Field: TField; State: TGridDrawState);
begin
  if (gdFocused in State) then
  begin
     if (Field.FieldName = DBLookupCombo1.DataField) then
     begin
       DBLookupCombo1.Left := Rect.Left + DBGrid1.Left;
       DBLookupCombo1.Top := Rect.Top + DBGrid1.top;
       DBLookupCombo1.Width := Rect.Right - Rect.Left;
      { DBLookupCombo1.Height := Rect.Bottom - Rect.Top; }
       DBLookupCombo1.Visible := True;
     end;
  end;
end;
```

The reasons for the excessive use begin/end will become clear later in the demo. The code is saying that if the State parameter is gdFocused then this particular cell is the one highlighted in the grid. Further if it's the highlighted cell and the cell has the same field name as the lookup combo's datafield then we need to move the LookupCombo over that cell and make it visible. Notice that the position is determined relative to the form not to just the grid. So, for example, the left side of LookupCombo needs to be the offset of the grid (DBGrid1.Left) into the form plus the offset of the cell into the grid (Rect.Left).

Also notice that the Height of the LookupCombo has been commented out above. The reason is that the LookupCombo has a minimum height. You just can't make it any smaller. That minimum height is larger than the height of the cell. If you un-commented the height line above. Your code would change it and then Delphi would immediately change it right back. It causes an annoying screen flash so don't fight it. Let the LookupCombo be a little larger than the cell. It looks a little funny but it works.

Now just for fun run the program. Correct all you missing semi-colons etc. Once its running try moving the cursor around the grid. Pretty cool, hu? Not! We're only part of the way there. We need to hide the LookupCombo when we leave the column. So define the grid's onColExit. It should look like this;

```
procedure TForm1.DBGrid1ColExit(Sender: TObject);
begin
  If DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField then
    DBLookupCombo1.Visible := false;
end;
```

This uses the TDBGrids SelectedField property to match up the FieldName associated with the cell with that of the LookupCombo. The code says, "If the cell you are leaving was in the DBLookupCombo column

then make it invisible".

Now run it again. Was that worth the effort or what?

Now things look right but we're still missing one thing. Try typing a new customer number into one of the LookupCombo. The problem is that the keystrokes are going to the grid, not to the LookupCombo. To fix this we need to define a onKeyPress event for the grid. It goes like this;

```
procedure TForm1.DBGrid1KeyPress(Sender: TObject; var Key: Char);
begin
  if (key <> chr(9)) then
  begin
    if (DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField) then
    begin
      DBLookupCombo1.SetFocus;
      SendMessage(DBLookupCombo1.Handle, WM_Char, word(Key), 0);
    end;
  end;
end;
```

This code is saying that if the key pressed is not a tab key (Chr(9)) and the current field in the grid is the LookupCombo then set the focus to the LookupCombo and then pass the keystroke over to the LookupCombo. OK so I had to use a WIN API function. You don't really need to know how it works just that it works.

But let me explain a bit anyway. To make Window's SendMessage function work you must give it the handle of the component you want to send the message to. Use the component's Handle property. Next it wants to know what the message is. In this case it's Window's message WM_CHAR which says I'm sending the LookupCombo a character. Finally, you need to tell it which character, so word(Key). That's a typecast to type word of the events Key parameter. Clear as mud, right? All you really need to know is to replace the DBLookupCombo1 in the call to the name of the component your putting into the grid. If you want more info on SendMessage do a search in Delphi's on-line help.

Now run it again and try typing. It works! Play with it a bit and see how the tab key gets you out of "edit mode" back into "move the cell cursor around mode".

Now go back to the Object Inspector for the DBLookupCombo component and change the LookupDIsplay property to Company. Run it. Imagine the possibilities.

COMPONENT #2 - TDBCOMBO

I'm not going to discuss installing the second component, a DBCombo, because I don't really have anything new to say. It's really the same as #1. Here's the incrementally developed code for your review.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DBLookupCombo1.Visible := False;
  DBComboBox1.Visible := False;
end;

procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
  Field: TField; State: TGridDrawState);
begin
  if (gdFocused in State) then
  begin
     if (Field.FieldName = DBLookupCombo1.DataField) then
     begin
```

```
      DBLookupCombo1.Left := Rect.Left + DBGrid1.Left;
      DBLookupCombo1.Top := Rect.Top + DBGrid1.top;
      DBLookupCombo1.Width := Rect.Right - Rect.Left;
      DBLookupCombo1.Visible := True;
    end
    else if (Field.FieldName = DBComboBox1.DataField) then
    begin
      DBComboBox1.Left := Rect.Left + DBGrid1.Left;
      DBComboBox1.Top := Rect.Top + DBGrid1.top;
      DBComboBox1.Width := Rect.Right - Rect.Left;
      DBComboBox1.Visible := True;
    end
  end;
end;

procedure TForm1.DBGrid1ColExit(Sender: TObject);
begin
  If DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField then
    DBLookupCombo1.Visible := false
  else If DBGrid1.SelectedField.FieldName = DBComboBox1.DataField then
    DBComboBox1.Visible := false;
end;

procedure TForm1.DBGrid1KeyPress(Sender: TObject; var Key: Char);
begin
  if (key <> chr(9)) then
  begin
    if (DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField) then
    begin
      DBLookupCombo1.SetFocus;
      SendMessage(DBLookupCombo1.Handle, WM_Char, word(Key), 0);
    end
    else if (DBGrid1.SelectedField.FieldName = DBComboBox1.DataField) then
    begin
      DBComboBox1.SetFocus;
      SendMessage(DBComboBox1.Handle, WM_Char, word(Key), 0);
    end;
  end;
end;
```

COMPONENT #3 - TDBCHECKBOX

The DBCheckBox gets even more interesting. In this case it seems appropriate to leave something in the non-focused checkbox cells to indicate that there's a check box there. You can either draw the "stay behind" image of the checkbox or you can blast in a picture of the checkbox. I chose to do the latter. I created two BMP files one that's a picture of the box checked (TRUE.BMP) and one that's a picture of the box unchecked (FALSE.BMP). Put two TImage components on the from called ImageTrue and ImageFalse and attach the BMP files to there respective Picture properties. Oh yes you also need to put a DBCheckbox component on the form. Wire it to the CheckBox field in DataSource1 and set the Color property to clWindow. First edit the onCreate so it reads as follows;

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  DBLookupCombo1.Visible := False;
  DBCheckBox1.Visible := False;
  DBComboBox1.Visible := False;
```

```
  ImageTrue.Visible := False;
  ImageFalse.Visible := False;
end;
```

Now we need to modify the onDrawDataCell to do something with cells that do not have the focus. Here comes the code.

```
procedure TForm1.DBGrid1DrawDataCell(Sender: TObject; const Rect: TRect;
  Field: TField; State: TGridDrawState);
begin
  if (gdFocused in State) then
  begin
     if (Field.FieldName = DBLookupCombo1.DataField) then
     begin
      ...SEE ABOVE
     end
     else if (Field.FieldName = DBCheckBox1.DataField) then
     begin
       DBCheckBox1.Left := Rect.Left + DBGrid1.Left + 1;
       DBCheckBox1.Top := Rect.Top + DBGrid1.top + 1;
       DBCheckBox1.Width := Rect.Right - Rect.Left{ - 1};
       DBCheckBox1.Height := Rect.Bottom - Rect.Top{ - 1};
       DBCheckBox1.Visible := True;
     end
     else if (Field.FieldName = DBComboBox1.DataField) then
     begin
      ...SEE ABOVE
     end
  end
  else {in this else area draw any stay behind bit maps}
  begin
    if (Field.FieldName = DBCheckBox1.DataField) then
    begin
     if TableGridDataCheckBox.AsBoolean then
       DBGrid1.Canvas.Draw(Rect.Left,Rect.Top,ImageTrue.Picture.Bitmap)
     else
       DBGrid1.Canvas.Draw(Rect.Left,Rect.Top,ImageFalse.Picture.Bitmap)
    end
  end;
```

It's the very last part we're most interested in. If the state is not gdFocused and the column in CheckBox then this last bit executes. All it does is check the value of the data in the field and if it's true it shows the TRUE.BMP otherwise it shows the FALSE.BMP. I created the bit maps so they are indented so you can tell the difference between a focused and unfocused cell. Make onColExit look like this;

```
procedure TForm1.DBGrid1ColExit(Sender: TObject);
begin
  If DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField then
    DBLookupCombo1.Visible := false
  else If DBGrid1.SelectedField.FieldName = DBCheckBox1.DataField then
    DBCheckBox1.Visible := false
  else If DBGrid1.SelectedField.FieldName = DBComboBox1.DataField then
    DBComboBox1.Visible := false;
end;
```

Edit onKeyPress to;

```
procedure TForm1.DBGrid1KeyPress(Sender: TObject; var Key: Char);
begin
  if (key <> chr(9)) then
  begin
    if (DBGrid1.SelectedField.FieldName = DBLookupCombo1.DataField) then
    begin
      DBLookupCombo1.SetFocus;
      SendMessage(DBLookupCombo1.Handle, WM_Char, word(Key), 0);
    end
    else if (DBGrid1.SelectedField.FieldName = DBCheckBox1.DataField) then
    begin
      DBCheckBox1.SetFocus;
      SendMessage(DBCheckBox1.Handle, WM_Char, word(Key), 0);
    end
    else if (DBGrid1.SelectedField.FieldName = DBComboBox1.DataField) then
    begin
      DBComboBox1.SetFocus;
      SendMessage(DBComboBox1.Handle, WM_Char, word(Key), 0);
    end;
  end;
end;
```

Finally, here's the last trick. The caption of the checkbox needs to change as the user checks or unchecks the box. My first thought was to do this in the TDBCheckBox's onChange event, the only problem is that it doesn't have one. So I had to go back to the Windows API and send another message. "SendMessage(DBCheckBox1.Handle, BM_GetCheck, 0, 0)" which returns a 0 if the box is unchecked, otherwise it's checked.

```
procedure TForm1.DBCheckBox1Click(Sender: TObject);
begin
  if SendMessage(DBCheckBox1.Handle, BM_GetCheck, 0, 0) = 0 then
     DBCheckBox1.Caption := '  ' + 'False'
  else
     DBCheckBox1.Caption := '  ' + 'True'
end;
```

That's it. Hopefully you learned something. I've tried this technique with dialog boxes. It works and it's simple. Have fun with it. You don't really need to completely understand it as long as you know how to edit the code and replace the above component names with with the name of the component you want to drop into the grid.

---------- REVISED - 7/11/95 ----------

Fred Dalgleish was nice enough to point out 2 stichy points about the Original grid demo. First, once a component in the grid has the focus it takes 2 Tab presses to move to the next grid cell. The other has to do with adding new records.

Problem # 1 - Two Tab Presses Required.

A component installed in the grid is actually floating over the top of the grid and not part of the grid it self. So when that component has the focus it takes two tab presses to move to the next cell. The first tab moves from the floating component to the Grid cell underneath and the second to move to the next grid cell. If this behavior bugs you heres how to fix it.

First in the form that contains grid add private variable called WasInFloater of type boolean, like so.

```
type
  TForm1 = class(TForm)
```

```
  ...
  ...
 private
   { Private declarations }
    WasInFloater : Boolean;
  ...
  ...
  end;
```

Next create an onEnter event for the LookupCombo where WasInFloater is set to true. Then point the onEnter event for each component that goes into the grid at this same single onEnter event.

```
procedure TForm1.DBLookupCombo1Enter(Sender: TObject);
begin
  WasInFloater := True;
end;
```

Finally, and here's the tricky part, define the following onKeyUp event for the grid.

```
procedure TForm1.DBGrid1KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (Key in [VK_TAB]) and WasInFloater then
  begin
    SendMessage(DBGrid1.Handle, WM_KeyDown, Key, 0);
    WasInFloater := False;
  end;
end;
```

What's happening here is that the grids onKeyUp is sending it self a KeyDown when the focus just switched from one of the floating controls. This solution handles both tab and shift-tab.

Problems #2 - New record disappears when component gets focus
The second problem is that if you press add record on the navigator in the demo a new record is added but then when you click on one of the components installed in the grid the new record disappears. The reason for this is that there is a strange grid option called dgCancelOnExit which is True by default. Set it to False and the above problem goes away.

In my opinion Borland should have had this default set to False to begin with. I find it getting in the way all the time and based on forum messages I'm not alone. The option is basically saying that if the grid looses focus then cancel and current edit's! Anyway I've got it turned off in just about every grid I've ever installed.

Note: This was written by Alec Bergamini, at 75664,1224. His company is Out & About Productions.

What is new? (The newest is listed first.)

CreateFont
How can I tell the default display size of an AVI file?
How do I size an AVI display to fit a panel?
How do I justify the text in a TEdit?
How do I decide whether or not to hightlight a day in a calendar?
How do I fill a graphic field from a BMP file?
How do I write to a printer port?
Here are some functions to use with the COMP type.
How do I do a ShowMessage on the OnExit() of a TEdit and still get a cursor on the next component?
How can I tell which row and column is currently selected in a TDBGrid?
How efficient is inc(i)?
How do I manage disk volume labels in Delphi?
How can I verify if a string is a valid date?
How do I know which line number I am currently on in a TMemo?
How do I broadcast changes to WIN.INI?
How do I detect the pentium bug?
How do I disable the screensaver?
How do I size a form to fit a bitmap?
How do I place the mouse anywhere on the form that I want?
How do I create a form with the equivelent of the visual basic 'FIXED DOUBLE' border with NO caption bar?
How do I eject paper from the printer?
ChartFX
How can I get information about window's tasks?
How do I use arrays in a Delphi DLL with VB?
How do I flush the data buffer to the table? (i.e. hard write to disk)
How do I paint with a cross-hatched brush?
How do I print from WinWord with DDE?
Here are some extra math functions.
How do I pass arrays into a procedure?

How can I trap for my own hotkeys?
How do I right justify a column of numeric data in a TStringGrid?
How do I print to different printer resolutions?
What are the values for the virtual keys?
How do I iterate through tabbed notebook pages to see each object? (update)
How do I use a 32 bit DLL with 16 bit Delphi?
How do I make it so that only the form I select comes to the top? (i.e. *without* the main form)
How do I get a single returned from a C++ DLL (MS)?
How do I call a dll function at runtime if the DLL is not known about at compile time?
How do I get a file's date and time stamp?
How do I fill a TMemo from a PChar?

How do I draw on the pages of a TTabbedNotebook?
How do I fake TTabbedNotebook with multiple forms?
How do I emulate TCollection in Delphi?
How can I make one window the child of another without using MDI?
Ole automation spec
How can I create an empty table that has the structure of another table?
How can I have an animated icon (when the form is minimized)?
How do I use FmtStr()?
How can I find out what all of the available fonts are?
When would I use an array[0..0]?
How can I readln() from a file when the lines are longer than 255 bytes?
How do I use "array of const"?
How do I do a TextOut() so that I don't get the white space around the text?
How do I paint the background of my form with a bitmap?
Is there a way to associate a string with each component?
TWriter/TReader
How do I format the text of a TEdit so that a 1 becomes 001, and 10 becomes 010, etc?
THelpEvent
How do I close a file that was opened in a DLL (Delphi made) and called from VB?
How can I create a windows meta-file from ellipse(), textOut(), etc. commands?
How can I erase the picture on image1.canvas?
How can I force the user to select only a mono-space font in a TFontDialog box?
How do I get RTTI (run time type information) from components?
How do I setup the column list in code for a dynamically created TTable?
How can I emulate the "C" function: StrTok()?
How do I reduce the amount of memory taken from the data segment? (or How do I allocate memory dynamically?)
How do I put components into a TDBGrid? (i.e. checkbox, combobox, etc.)
Here is a unit to read Lotus 123 files.
Here is a double linked list as an object.
How do I get the julian date?
How do I use a TList to hold TTable variables?
How do I do click and drag in a TListbox?
How do I make a virtual table (in memory)?
Here is a custom TStringGrid that will allow for inserting a whole row at a time.
How do I iterate through tabbed notebook pages to see each object?
How do I calculate the width of a TDBGrid so that it will display all the fields?
Here is a custom TEdit that will tab to the next control when the user hits the <ENTER>.
TPanelClock component (comes with Caps/Num/scroll lock indication)
Component to allow for paradox style QBE
Gray Paper - Delphi Property Editors for Beginners
How do I rotate text in display?

Q: How do I reduce the amount of memory taken from the data segment? (or How do I allocate memory dynamically?)

A:

Let's say your data structure looks like this:

```
type
  TMyStructure = record
    Name: String[40];
    Data: array[0..4095] of Integer;
    end;
```

That's too large to be allocated globally, so instead of declaring a global variable,

```
var
  MyData: TMyStructure;
```

you declare a pointer type,

```
type
  PMyStructure = ^TMyStructure;
```

and a variable of that type,

```
var
  MyDataPtr: PMyStructure;
```

Such a pointer consumes only four bytes of the global data segment.

Before you can use the data structure, you have to allocate it on the heap:

```
New(MyDataPtr);
```

and now you can access it just like you would global data. The only difference is that you have to use the caret operator to dereference the pointer:

```
MyDataPtr^.Name := 'Tony Chang';
MyDataPtr^.Data[0] := 12345;
```

Finally, after you're done using the memory, you deallocate it:

```
Dispose(MyDataPtr);
```

Here is some information that shows how to use this dynamic allocation for matrix code writing in a general way:

```
type
  PRow = ^TRow;
  TRow = array[1..65520 div SizeOf(TSomeType)] of TSomeType;
```

```
  { where TSomeType is the type of the elements of the array }
  PMatrix = ^TMatrix;
  TMatrix = array[1..16380] of PRow;

procedure InitMatrix(Mat: PMatrix; Rows, Cols: Integer);
var
  R: Integer;
begin
  GetMem(Mat, Rows * SizeOf(PRow));
  for R := 1 to Rows do
    GetMem(Mat^[R], Cols * SizeOf(TSomeType));
end;
```

Once the matrix has been initialized, you reference the individual elements as you would an ordinary two-dimensional array, except that you have to dereference the pointers:

```
type
  TSomeType = Integer;
var
  MyMat: PMatrix;
begin
InitMatrix(MyMat, 37, 42);
MyMat^[14]^[10] := 99;
MyMat^[5]^[19] := 12345;
{ etc. }
end;
```

See Also pointers and assignments

Q: How do I setup the column list in code for a dynamically created TTable?

A: The following examples should get you started. They are based on an article (I believe by Ray Konopka) that appeared an issue or 2 ago in PC Techniques.

```
function TBizObj.CreateField(Dataset: TDataset; FieldName: string): TField;
begin
 with Dataset do begin
   Result := FindField( FieldName ); { First, see if it exists }
   if Result = nil then begin { If so, no need to create it. }
      { Have the FieldDef object create its own Field Object }
     if Owner <> nil then
       Result := FieldDefs.Find(FieldName).CreateField(Owner)
     else
       Result := FieldDefs.Find(FieldName).CreateField(Dataset);
     { Give the new Field Object a generic Name so that it appears in the
Object Inspector}
     Result.Name := Name + FieldName;
   end;
 end;
end;

function TBizObj.CreateCalcField( Dataset: TDataset; FieldName : string;
 FieldClass : TFieldClass; Size : Word ) : TField;
begin
 Result := Dataset.FindField( FieldName );
 if Result = nil then begin
   if FieldClass = nil then
     DBErrorFmt( SUnknownFieldType, [ FieldName ] );
   Result := FieldClass.Create( Owner );
   try
     Result.FieldName := FieldName;
     Result.Size := Size;
     Result.Calculated := True;
     Result.Dataset := Dataset;
     Result.Name := Dataset.Name + FieldName;
   except
     Result.Free;
     raise;
   end;
 end;
end; { TBizObj.CreateCalcField }
```

Q: How can I emulate the "C" function: StrTok()? Note: StrTok() searches one string (P) for tokens which are separated by delimiters defined in the second string (Divider). The first call to StrTok() returns a pointer to the first character of the first token in P and writes a null character into P immediately following the returned token. Subsequent calls with null for the first arguement will work through the string P in this way until no tokens remain.

A:

```
function  NextToken(P: PChar; Divider: PChar): PChar;
const next: PChar = nil;
begin
  if P = nil then P := next;
  if P <> nil then begin
    next := StrPos(P, Divider);
    if next <> nil then begin
      next^ := #0 ;
      next := @next[StrLen(Divider)];
    end;
  end;
  result := P;
end;
```

Q: How do I get RTTI (run time type information) from components?

A:

The screen resolution topic shows one way. Here is another example:

```
Uses TypInfo;

Function AssignFontProperty( anObj: TObject; Const fname: String ):
  Boolean;
Var
  PInfo: PPropInfo;
  aFont: TFont;
  FontGet: Function( anObj: TObject ): TFont;
Begin
  (* try to get a pointer to the property information for a property with the
     passed name. TObject.ClassInfo returns a pointer to the RTTI table,
which
     we need to pass to GetPropInfo *)
  PInfo := GetPropInfo( anObj.ClassInfo, 'font' );
  Result := PInfo <> Nil;
  If result Then
    (* found a property with this name, check if it has the correct type *)
    If (PInfo^.Proptype^.Kind = tkClass) and
        GetTypeData(PInfo^.Proptype)^.ClassType.InheritsFrom(TFont) Then
Begin
      { try to get the read proc for this property }
      @FontGet:= PInfo^.GetProc;
      If Assigned( FontGet ) and (PtrRec(PInfo^.GetProc).Seg <> $FFFF) Then
        { hiword of $FFFF seems to signify a property wich reads directly
          from a field instead of via a method! }
      Begin
        aFont := FontGet( anObj );
        If aFont Is TFont Then
          aFont.Name := fname
        Else
          ShowMessage('FontGet barfed');
      End
      Else Begin
        { no read method for the property, get field directly }
        aFont := TFont(GetOrdProp( anObj, PInfo ));
        If aFont Is TFont Then
          aFont.Name := fname
        Else
          ShowMessage('GetOrdProp barfed');
      End;
    End
    Else Begin
      (* nope, wrong type, complain *)
      Result := False;
      ShowMessage( 'Property Font is not of type TFont!');
    End;
End;

procedure TForm1.BtnTestClick(Sender: TObject);
```

```
Var
  i: Integer;
begin
  For i:= 0 To ComponentCount-1 Do
    AssignFontProperty( Components[i], 'Symbol' );
end;
```

Put a few controls on your form, assign the BtnTestClick handler to a buttons OnClick property, run the app, click on the button and all will be greek to you <g>.

Here is a tip or two on using this information:

You already know which properties that one has in most cases. So it would only be a useful method in a base class of a class hierarchy (e.g. in TObject, which you cannot modify). A function that only looks for a property by name would require only one line of code:

```
Function HasProperty( anObj: TObject; Const name: String ): Boolean;
Begin
  Result := GetPropInfo( anObj.ClassInfo, name ) <> Nil;
End;
```

But that is not very useful, it still does not solve the problem of how to _use_ the property!

```
Var
  aComponent: TComponent;
  i: Integer;
begin
  For i:= 0 To ComponentCount-1 Do Begin
    aComponent := Components[i];
    If HasProperty( aComponent, 'Font' ) Then
      aComponent.Font.Name := ....    <== will not compile
```

You still would need to cast aComponent to a class that has Font as public or published property and of course it better be one that is in the ancestry of aComponent. Using the run-time type info for the access solves this problem but is a bit awkward as you saw.

general component information

What limits are known of the standard Delphi controls?
How do I do screen updates all at once (without a ripple effect).
I want to know how I can make a variable that is a "pointer" to Canvas.Font.
How can I tell the size of a given object at runtime?
How can my component tell if I'm running via the IDE or the EXE?
Is it possible to access components by their name property (i.e. 'SpeedButton' + IntToStr(i))?
How do I rotate text in display?
How do I get RTTI (run time type information) from components?
Is there a way to associate a string with each component?

TFontDialog

How can I force the user to select only a mono-space font?

```
FontDialog1.Options := [fdFixedPitchOnly];
```

Note: For other options see TFontDialogOptions.

TImage

How can I erase the picture on image1.canvas?
How do I paint with a cross-hatched brush?
How do I size a form to fit a bitmap?

Q: How can I erase the picture on image1.canvas?

A:

If you want to eliminate the bitmap then try `image1.picture := nil;`

If not, then try this:

```
with image1.picture.bitmap.canvas do FillRect(ClipRect);
```

This will fill the canvas with the current brush color.

Q: How can I create a windows meta-file from ellipse(), textOut(), etc. commands?

A: Here's the latest version of my (Neil Rubenking) metafile-creation example. I've marked the area where you insert your own drawing code. Note that not *every* GDI function can be used in a metafile DC. I've seen a list of those that cannot; alas, it apparently wasn't in the Windows API Help for the metafile functions.

PS - if you're the puzzling type, try to puzzle out just what this program draws before you run it. Or at least, figure what the PolyLine method call draws...

```
CONST W = 1000; H = 500; R2 = 353;

procedure TForm1.Button1Click(Sender: TObject);
VAR CDC : hDC;

  procedure spiral(CV: TCanvas; Colr: TColor; AngInc: Double;
    RInc: Integer; Sgn1, Sgn2: Integer);
  VAR Theta: Double;
  begin
    Sgn1 := Sgn1 DIV Abs(Sgn1);
    Sgn2 := Sgn2 DIV Abs(Sgn2);
    CV.pen.Color := Colr;
    Theta := 0;
    CV.MoveTo(H,H);
    WHILE Theta < pred(H DIV RInc) DO
      BEGIN
        Theta := Theta + AngInc;
        CV.LineTo(H+Sgn1*Round(RInc*Theta*Cos(Sgn2*Theta)),
                  H+Sgn1*Round(RInc*Theta*Sin(Sgn2*Theta)));
      END;
  end;

begin
  DeleteFile('EXAMPLE.WMF');
  WITH TBitmap.Create DO
    try
      CDC := CreateMetafile(NIL);
      Canvas.Handle := CDC;
      SetMapMode(Canvas.Handle, MM_ANISOTROPIC);
      SetWindowExt(Canvas.Handle, W, W);
      WITH Canvas DO
        BEGIN
          {==== replace with your own image-creation code ===}
          Brush.Color := clWhite;
          pen.Width := 10;
          pen.Style := psInsideFrame;
          pen.Color := clBlack;
          Ellipse(0,0,W,W);
          Spiral(Canvas, clRed, pi/4, 18, 1, 1);
          Spiral(Canvas, clBlue, pi/4, 18, -1, 1);
          Pen.Color := clBlack;
          PolyLine([Point(H,0), Point(H,H), Point(H-R2, H+R2),
            Point(H,H), Point(H,W), Point(H,H),
```

```
            Point(H+R2,H+R2)]);
          {====== end image-creation code =====}
        END;
      WITH TMetafile.Create DO
        try
          Handle := CloseMetafile(CDC);
          Inch   := W;
          Height := 128;
          Width  := 128;
          SaveToFile('EXAMPLE.WMF');
        finally
          Free;
        end;
    finally
      Free;
    end;
end;
```

Q: How do I close a file that was opened in a DLL (Delphi made) and called from VB?

A: This is a known problem. It comes from the fact that VB closes the 5 DOS standard handles (0..4) at startup. So the open file routine will reuse one of these handles to open the first disk file. That is not a problem in using the file, but the Pascal Close routine has a build-in safety feature: it refuses to close a file that has one of the standard handles! That is a Good Thing under DOS but screws up the works in your situation since the file opened by the DLL is never closed, not even when the DLL goes down! VC++ is obviously less restricted and will close a standard handle.

You can fix this problem yourself. Instead of using the Pascal Close/CloseFile routine to close the file in the DLL use this one:

```
Procedure ReallyCloseFile(Var F); Assembler;
Asm
  les bx, F
  mov bx, es:[bx]
  mov ah, $3E
  call Dos3Call
End;
```

THelpEvent

The parameters for THelpEvent are documented incorrectly in the help file - even in the latest update.

As documented:

```
THelpEvent = function (Command: Word; Data: Longint): Boolean of object;
```

ACTUAL parameters:

```
THelpEvent = function (Command: Word; Data: Longint; var CallHelp: Boolean):
Boolean of object;
```

```
var s: string;
begin
  FmtStr(s, '%.3d', [StrToInt(edit1.text)]);
  edit1.text := s;
end;
```

Here's what you'd need to do in order to use TWriter/TReader to write a string to a stream. I'm doing it with a TMemoryStream still to keep it simple. The key is the calls to Read/WriteListBegin and Read/WriteListEnd. Without this you get a exception.

```
procedure TForm1.Button1Click(Sender: TObject);
var sWrite,sRead : string[25];
    MyStream : TMemoryStream;
    MyWriter : TWriter;
    MyReader : TReader;
begin
    MyStream := TMemoryStream.Create;
    MyStream.SetSize(4096);
    MyWriter := TWriter.Create(MyStream,4096);
    sWrite := 'sWriteContents';

    MyWriter.WriteListBegin;
    MyWriter.WriteString(sWrite);
    MyWriter.WriteListEnd;
    MyWriter.free;

    MyStream.Seek(0,0);

    MyReader := TReader.Create(MyStream,4096);
    MyReader.ReadListBegin;
    sRead := MyReader.ReadString;
    MyReader.ReadListEnd;
    MyReader.free;

    Label1.Caption := sRead;
    MyStream.free;
end;
```

Q: Is there a way to associate a string with each component?

A: Since the Tag property is a longint, you can type cast it as a Pointer or PChar. So, you can basically store a pointer to a record by using the Tag property.

Note: You're not going to be able to store the string, or pointer rather, at design time. This is something you'll have to do at run time. Take a look at this example:

```
var
 i: integer;
begin
  for i := 0 to ComponentCount - 1 do
    if Components[i] is TEdit then
      Components[i].Tag := LongInt(NewStr('Hello '+IntToStr(i)));
end;
```

Here, I loop through the components on the form. If the component is a TEdit, I assign a pointer to a string to its Tag property. The NewStr function returns a PString (pointer to a string). A pointer is basically the same as a longint or better, occupies the same number of bytes in memory. Therefore, you can type cast the return value of NewStr as a LongInt and store it in the Tag property of the TEdit component. Keep in mind that this could have been a pointer to an entire record. Now I'll use that value:

```
var
 i: integer;
begin
  for i := 0 to ComponentCount - 1 do
    if Components[i] is TEdit then begin
      TEdit(Components[i]).Text := PString(Components[i].Tag)^;
      DisposeStr(PString(Components[i].Tag));
    end;
end;
```

Here, again I loop through the components and work on only the TEdits. This time, I extract the value of the component's Tag property by typecasting it as a PString (Pointer to a string) and assigning that value to the TEdit's Text property. Of course, I must dereference it with the caret (^) symbol. Once I do that, I dispose of the string stored in the edit component. Important note: if you store anything in the TEdit's Tag property as a pointer, you are responsible for disposing of it also.

FYI, Since Delphi objects are really pointers to class instances, you can also store objects in the Tag property. As long as you remember to Free them.

Three methods spring to mind to use Tags to access strings that persist from app to app.

1. If your strings stay the same forever, create a string resource in Resource Workshop

(or equiv) and use the Tags as indexes into your string resource.

2. Use TIniFile and create a section for your strings, and give each string a name with number so that your ini file has a section like this:

[strings]
string1=Aristotle
string2=Plato
string3=Well this is Delphi, afterall

Then you can fetch them back out this way:

```
var s1: string;
...
s1 := IniFile1.ReadString('strings', 'string'+IntToStr(Tag), '');
```

3. Put your strings into a file, with each followed by a carriage return. Read them into a TStringList. Then your Tags become an index into this stringlist:

```
StringList1.LoadFromFile('slist.txt');
...
s1 := StringList1[Tag];
```

Given the way Delphi is set up, I think the inifile method is easiest.

Q: How do I paint the background of my form with a bitmap?

A:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
    procedure FormPaint(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  Bitmap: TBitmap;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
  Bitmap := TBitmap.Create;
  Bitmap.LoadFromFile('C:\WINDOWS\cars.BMP');
end;

procedure TForm1.FormPaint(Sender: TObject);
var
  X, Y, W, H: LongInt;
begin
  with Bitmap do begin
    W := Width;
    H := Height;
  end;
  Y := 0;
  while Y < Height do begin
    X := 0;
    while X < Width do begin
      Canvas.Draw(X, Y, Bitmap);
      Inc(X, W);
    end;
    Inc(Y, H);
  end;
end;

end.
```

Q: How do I do a TextOut() so that I don't get the white space around the text?

A:

```
with canvas do begin
  brush.style := bsClear;
  font.color := clWhite;
  TextOut(25, 25, 'I am the greatest!');
end;
```

obfuscated "C"

Best Formatted "C" code

romantic "C"
pallandrome "C"

Most cryptic, but working "C" code

Plays Othello (well!) 1988
Best One-Liner 1988
Best small program

A Guide to Effective Scientific Communication

Phrase	Translation
It has long been known	I haven't bothered to look up the reference
It is believed	I think
It is generally believed	A couple of other guys think so too
It is not unreasonable to assume	If you believe this, you'll believe anything
Of great theoretical importance	I find it kind of interesting
Of great practical importance	I can get some mileage out of it
Typical results are shown	The best results are shown
3 samples were chosen for further study	The others didn't make sense, so we ignored them
The 4 hour sample was not studied	I dropped it on the floor
The 4 hour determination may not be significant	I dropped it on the floor, but scooped most of it up
The significance of these results is unclear	Look at the pretty artifact
It has not been possible to provide definitive answers	The experiment was negative, but at least I can publish the data somewhere
Correct within an order of magnitude	Wrong
It might be argued that	I have such a good answer for this objection that I shall now raise it
Much additional work will be required	This paper is not very good, but neither are all the others in this miserable field
These investigations proved highly rewarding	My grant is going to be renewed
I thank X for assistance with the experiments and Y for useful discussions on the interperetation of the data	X did the experiment and Y explained it to me

This Plays the game of reversi (Othello)! Compile and run. It then asks for a playing level. Enter 0-10 (easy-hard). It then asks for your move. A move is a number within 11-88, or a 99 to pass. Illegal moves (except for an illegal pass) are rejected. Then the computer does its move (or a 0 to pass), until the board is full. It plays rather well, for such a small program! Lievaart had to leave out the board printing routine, so you'll have to take a real game board to play it. ... Also due to space-limitations (the rules for 1987 had a limit of 1024 byes), Lievaart took out the passing-handler, which makes its ending-game rather poor. But further it knows all the rules, uses alpha-beta pruning, and it plays f.i. on mobility(!). Most important: it can play a pretty good game of Reversi!

The Author was kind enough to supply the fully functional version of the program. The file lievaart2.c contains what the program would have been without the size restriction. This version has the full end game logic and displays the board after each move!

Short version:

```
#D Y return
#D R for
#D e while
#D I printf
#D l int
#D C y=v+111;H(x,v)*y++= *x
#D H(a,b)R(a=b+11;a<b+89;a++)
#D s(a)t=scanf("%d",&a)
l V[1100],u,r[]={-1,-11,-10,-
9,1,11,10,9},h[]={11,18,81,88},ih[]={22,27,72,77},
bz,lv=60,*x,*y,m,t;S(d,v,f,a,b)l*v;{l c=0,*n=v+100,bw=d<u-1?
a:-9000,w,z,i,zb,q=
3-f;if(d>u){R(w=i=0;i<4;i++)w+=(m=v[h[i]])==f?300:m==q?-300:
(t=v[ih[i]])==f?-50
:t==q?50:0;return w;}H(z,0){if(GZ(v,z,f,100)){c++;w= -S(d+1,n,q,-b,-
bw);if(w>bw
){zb=z;bw=w;if(w>=b||w>=8003)Y w;}}}if(!c){bz=0;C;Y-S(d+1,n,q,-b,-bw);}
bz=zb;Y
d>=u-1?bw+(c<<3):bw;}main(){R(;t<1100;t+=100)R(m=0;m<100;m++)V[t+m]=m<11||
m>88
||(m+1)%10<2?3:0;V[44]=V[55]=1;V[45]=V[54]=2;I("Level:");s(u);e(lv>0)
{do{I("Yo\
u:");s(m);}e(!GZ(V,m,2,0)&&m!=99);if(m!
=99)lv--;if(lv<15&&u<10)u+=2;I("Wait\n")
;I("Value:%d\n",S(0,V,1,-9000,9000));I("move: %d\n",(lv-=GZ(V,bz,1,0),bz));}}
GZ
(v,z,f,o)l*v;{l*j,q=3-f,g=0,i,h,*k=v+z;if(*k==0)R(i=7;i>=0;i--){j=k+
(h=r[i]);e(
*j==q)j+=h;if(*j==f&&j-h!=k){if(!g){g=1;C;}e(j!=k)*((j-=h)+o)=f;}}Y g;}
```

Long version

```
#D Y return
```

```
#D R for
#D e while
#D I printf
#D l int
#D W if
#D C y=v+111;H(x,v)*y++= *x
#D H(a,b)R(a=b+11;a<b+89;a++)
#D s(a)t=scanf("%d",&a)
#D U Z I
#D Z I("123\
45678\n");H(x,V){putchar(".XO"[*x]);W((x-V)%10==8){x+=2;I("%d\n",(x-V)/10-
1);}}
l V[1600],u,r[]={-1,-11,-10,-
9,1,11,10,9},h[]={11,18,81,88},ih[]={22,27,72,77},
bz,lv=60,*x,*y,m,t;S(d,v,f,_,a,b)l*v;{l c=0,*n=v+100,j=d<u-1?
a:-9000,w,z,i,g,q=
3-f;W(d>u){R(w=i=0;i<4;i++)w+=(m=v[h[i]])==f?300:m==q?-300:
(t=v[ih[i]])==f?-50:
t==q?50:0;Y w;}H(z,0){W(E(v,z,f,100)){c++;w= -S(d+1,n,q,0,-b,-j);W(w>j)
{g=bz=z;
j=w;W(w>=b||w>=8003)Y w;}}}W(!c){g=0;W(_){H(x,v)c+= *x==f?1:*x==3-f?-1:0;Y
c>0?
8000+c:c-8000;}C;j= -S(d+1,n,q,1,-b,-j);}bz=g;Y d>=u-1?j+(c<<3):j;}main()
{R(;t<
1600;t+=100)R(m=0;m<100;m++)V[t+m]=m<11||m>88||(m+1)%10<2?
3:0;I("Level:");V[44]
=V[55]=1;V[45]=V[54]=2;s(u);e(lv>0){Z do{I("You:");s(m);}e(!E(V,m,2,0)&&m!
=99);
W(m!=99)lv--;W(lv<15&&u<10)u+=2;U("Wait\n");I("Value:%d\n",S(0,V,1,0,-
9000,9000
));I("move: %d\n",(lv-=E(V,bz,1,0),bz));}}E(v,z,f,o)l*v;{l*j,q=3-
f,g=0,i,w,*k=v
+z;W(*k==0)R(i=7;i>=0;i--){j=k+(w=r[i]);e(*j==q)j+=w;W(*j==f&&j-w!=k){W(!g)
{g=1
;C;}e(j!=k)*((j-=w)+o)=f;}}Y g;}
```

This one won for best layout. Putchar must exist in the C library and not just as a macro. If it fails to compile, add the line: #include <stdio.h> at the top of the program.

Line by line symmetry performed better than any C beautifier. Think of if it as a C ink blot. :-)

```
                                rahctup,putchar
                                      ( )
                               ,LACEDx0 = 0xDECAL,
                                    rof ; for
                                  (;(int) (tni);)
                                    (int) (tni)
                                = reviled ; deliver =
                                    redivider
                                        ;
for ((int)(tni)++,++reviled;reviled* *deliver;deliver++,++(int)(tni)) rof
                                        =
                                   (int) -1- (tni)
                                ;reviled--;--deliver;
                                   (tni)  =  (int)
                                - 0xDECAL + LACEDx0 -
                                    rof ; for
       (reviled--,(int)--(tni);(int) (tni);(int)--(tni),--deliver)
                                 rahctup = putchar
                                (reviled* *deliver)
                                        ;
                                 rahctup * putchar
                                 ((char) * (rahc))
                                        ;
                                       /*\
                                      {\*/}
```

Best One Liner:David Korn

The Judges believe that this is the best one line entry ever received. Compile on a UN*X system, or at least using a C implementation that fakes it. Very few people are able to determine what this program does by visual inspection. I suggest that you stop reading this section right now and see if you are one of the few people who can.

Several points are important to understand in this program:

1) What is the symbol `unix' and what is its value in the program? Clearly `unix' is not a function, and since `unix' is not declared to be a data type (such as int, char, struct foo, enum, ...) what must `unix' be?

2) What is the value of the symbol "have"? (hint: the value is NOT 4 characters, or 'h', or a string) Consider the fact that:

```
char *x;
```

defines a pointer to a character (i.e. an address), and that the `=' assigns things is compatible types. Since:

```
x = "have";
```

is legal C, what type of value is "have"?

3) Note that the following expressions yield the same value:

x[3] *(x+3) *(3+x)

since addition is communitive. What can be said about the value:

3[x]

David Korn's /bin/ksh provides us with a greatly improved version of the /bin/sh. The source for v7's /bin/sh greatly inspired this contest.

```
main() { printf(&unix["\021%six\012\0"],(unix)["have"]+"fun"-0x60);}
```

Q: How can I create an empty table that has the structure of another table?

A:

```
uses DB, DBTables, StdCtrls;

procedure TForm1.Button1Click(Sender: TObject);
var
  tSource, TDest: TTable;
begin
  TSource := TTable.create(self);
  with TSource do begin
    DatabaseName := 'dbdemos';
    TableName := 'customer.db';
    open;
  end;
  TDest := TTable.create(self);
  with TDest do begin
    DatabaseName := 'dbdemos';
    TableName := 'MyNewTbl.db';
    FieldDefs.Assign(TSource.FieldDefs);
    IndexDefs.Assign(TSource.IndexDefs);
    CreateTable;
  end;
  TSource.close;
end;
```

Best small program: <cs.vu.nl!maart> Maarten Litmaath

Try: litmaath eschew obfuscation

Note the unusual structure:

```
         while (<condition>)
                ;
```

Did you notice that the body is empty?

The best one can do to understand how the program works is to give it two small strings as arguments, and follow the program closely. One could make the body of the 'while' loop an 'fprintf' with interesting variables like:

```
    fprintf(stderr,
             "argv=%lo *argv=%lo **argv=%c argv[1]=%lo *argv[1]=%c
argc=%d\n",
             (long) argv, (long) *argv, *argv && **argv ? **argv : '@',
             (long) argv[1], argv[1] && *argv[1] ? *argv[1] : '@', argc);
```

Furthermore, it's interesting to note that only two variables are used to achieve everything.

```
main(argc, argv)
int     argc;
char    **argv;
{
  while (*argv != argv[1] && (*argv = argv[1]) && (argc = 0) || (*++argv
         && (**argv && ((++argc)[*argv] && (**argv <= argc[*argv] ||
         (**argv += argc[*argv] -= **argv = argc[*argv] - **argv)) &&
         --argv || putchar(**argv) && ++*argv--) || putchar(10))));
}
```

Note: The program sorts the characters of each arguement on a separate line. WOW!

Q: How do I use "array of const"?

A: An array of const is in fact an open array of TVarRec (a predeclared Delphi type you can look up in the online help). So the following is the general battle plan:

```
procedure AddStuff( Const A: Array of Const );
Var i: Integer;
Begin
  For i:= Low(A) to High(A) Do
  With A[i] Do
    Case VType of
    vtExtended: Begin
       { add real number, all real formats are converted to extended
automatically }
      End;
    vtInteger: Begin
       { add integer number, all integer formats are converted to LongInt
automatically }
      End;
    vtObject: Begin
        If VObject Is DArray Then
          With DArray( VObject ) Do Begin
            { add array of doubles }
          End
        Else If VObject Is IArray Then
          With IArray( VObject ) Do Begin
            { add array of integers }
          End;
      End;
    End; { Case }
End; { AddStuff }
```

For further information see "open arrays" in the on-line help.

Here is a bit more from Steve:

An "array of const" is an array of values passed as const. Internally, these are represented by TVarRec structures. The brackets are simply to delimit the array. Arrays of const give you the ability to send a variable number of parameters to a routine in a type-safe manner. Here's an example:

```
 type
   TVarRec = record
     Data: record case Integer of
       0: (L: LongInt);
       1: (B: Boolean);
       2: (C: Char);
       3: (E: ^Extended);
       4: (S: ^String);
       5: (P: Pointer);
       6: (X: PChar);
       7: (O: TObject);
```

```
      end;
    Tag: Byte;
    Stuff: array[0..2] of Byte;
    end;

function PtrToStr(P: Pointer): String;
const
  HexChar: array[0..15] of Char = '0123456789ABCDEF';

  function HexByte(B: Byte): String;
  begin
  Result := HexChar[B shr 4] + HexChar[B and 15];
  end;

  function HexWord(W: Word): String;
  begin
  Result := HexByte(Hi(W)) + HexByte(Lo(W));
  end;

begin
Result := HexWord(HiWord(LongInt(P))) + ':' + HexWord(LoWord(LongInt(P)));
end;

procedure Display(X: array of const);
var
  I: Integer;
begin
for I := 0 to High(X) do with TVarRec(X[I]), Data do
  begin
  case Tag of
    0: ShowMessage('Integer: ' + IntToStr(L));
    1: if B then ShowMessage('Boolean: True')
      else ShowMessage('Boolean: False');
    2: ShowMessage('Char: ' + C);
    3: ShowMessage('Float: ' + FloatToStr(E^));
    4: ShowMessage('String: ' + S^);
    5: ShowMessage('Pointer: ' + PtrToStr(P));
    6: ShowMessage('PChar: ' + StrPas(X));
    7: ShowMessage('Object: ' + O.ClassName);
    end;
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  P: array[0..5] of Char;

begin
P := 'Hello'#0;
Display([-12345678, True, 'A', 1.2345, 'ABC', Ptr($1234, $5678), P,
  Form1]);
end;
```

Q: How can I readln() from a file when the lines are longer than 255 bytes?

A: ReadLn will accept an array [0..something] of Char as buffer to put the read characters in and it will make a proper zero-terminated char out of them. The only limitation is this: the compiler needs to be able to figure out the size of the buffer *at compile time*, which makes the use of a variable declared as PChar and allocated at run-time impossible.

Workaround:

```
Type
  TLine = Array [0..1024] of Char; {use longest line you may encounter here}
  PLine = ^TLine;

Var
  pBuf: PLine;
...
  New( pBuf );

...
  ReadLn( F, pBuf^ );
```

To pass pBuf to functions that take a parameter of type Pchar, use a typecast like PChar(pBuf).

Note: you can use a variable declared as of type TLine or an equivalent array of char directly, of course, but I tend to allocate anything larger than 4 bytes on the heap...

Q: When would I use an array[0..0]?

A: Declaring a variable of type : array [0..0] allows you to dynamically allocate the array on the heap and then access the elements of the array with a subscript. Consider the following:

```
 {$R-} {range checking has to be off or the compiler will complain}
 type
   MyRecordType = record
     Field1 : integer;
     Field2 : word;
   end;
   StructArr = array [0..0] of MyRecordType;
   MyStructPtr = ^StructArr;
 var
   myPtr : MyStructPtr;
   iNumberOfElements : word;
   TheNthElement : integer;
 begin
   {Set the number of elements you want}
   iNumberOfElements := SomeArbitraryValueDeterminedAtRuntime;
   {Allocate the RAM}
   GetMem(myPtr, iNumberOfElements * SizeOf(MyRecordType));
   {Now you have a dynamic array of StructArr[0..iNumberOfElements-1].
    That is, you didn't have to declare the number of elements until
    runtime.}
   try
     {The Nth element is...}
     TheNthElement := myPtr^[TheNthElementIndex].Field1;
     {                      ^^^^^^^^^^^^^^^^^^^       }
     {That array[0..0] declaration makes ^this^ possible. }
     {...and it *must* be a variable.  Constants will cause a range error.}
   finally
     FreeMem(myPtr, SizeOf(MyRecordType) *
SomeArbitraryValueDeterminedAtRuntime);
   end;
 end;
```

Very useful when you want an array, but don't know how many elements you ar going to put in it.

General programming

Optimizing code
How do I link to existing OBJ files?
Which memory model does Delphi use?
How do I use a resource (.RES) file?

Dynamic components

How can VCL components be created on the fly at run-time?
How do I move one image across a background image?
How do I use one of the cursor files in the c:\delphi\images\cursors?
How to dynamically add an object to a form.
How do I make a new component that has a string editor?
How do I assign a method to the event of a dynamically created object?
How do I populate a popup menu on the fly?
How do I save an object to a disk file?
How do I drag and move components in runtime?

Q: How do I use FmtStr()?

A: The Format routine formats the argument list given by the Args parameter using the format string given by the Format parameter.

Format strings contain two types of objects--plain characters and format specifiers. Plain characters are copied verbatim to the resulting string. Format specifiers fetch arguments from the argument list and apply
formatting to them.

Format specifiers have the following form:

"%" [index ":"] ["-"] [width] ["." prec] type

A format specifier begins with a % character. After the % come the following, in this order:

- an optional argument index specifier, [index ":"]
- an optional left-justification indicator, ["-"]
- an optional width specifier, [width]
- an optional precision specifier, ["." prec]
- the conversion type character, type

The following conversion characters are supported:

d Decimal. The argument must be an integer value. The value is converted to a string of decimal digits. If the format string contains a precision specifier, it indicates that the resulting string must contain at least the specified number of digits; if the value has less digits, the resulting string is left-padded with zeros.

e Scientific. The argument must be a floating-point value. The value is converted to a string of the form "-d.ddd...E+ddd". The resulting string starts with a minus sign if the number is negative, and one digit always precedes the decimal point. The total number of digits in the resulting string (including the one before the decimal point) is given by the precision specifer in the format string--a default precision of 15 is assumed if no precision specifer is present. The "E" exponent character in the resulting string is always followed by a plus or minus sign and at least three digits.

f Fixed. The argument must be a floating-point value. The value is converted to a string of the form "-ddd.ddd...". The resulting string starts with a minus sign if the number is negative. The number of digits after the decimal point is given by the precision specifier in the format string--a default of 2 decimal digits is assumed if no precision specifier is present.

g General. The argument must be a floating-point value. The value is converted to the shortest possible decimal string using fixed or scientific format. The number of

significant digits in the resulting string is given by the precision specifier in the format string--a default precision of 15 is assumed if no precision specifier is present. Trailing zeros are removed from the resulting string, and a decimal point appears only if necessary. The resulting string uses fixed point format if the number of digits to the left of the decimal point in the value is less than or equal to the specified precision, and if the value is greater than or equal to 0.00001. Otherwise the resulting string uses scientific format.

n Number. The argument must be a floating-point value. The value is converted to a string of the form "-d,ddd,ddd.ddd...". The "n" format corresponds to the "f" format, except that the resulting string contains thousand separators.

m Money. The argument must be a floating-point value. The value is converted to a string that represents a currency amount. The conversion is controlled by the CurrencyString, CurrencyFormat, NegCurrFormat, ThousandSeparator, DecimalSeparator, and CurrencyDecimals global variables, all of which are initialized from the Currency Format in the International section of the Windows Control Panel. If the format string contains a precision specifier, it overrides the value given by the CurrencyDecimals global variable.

p Pointer. The argument must be a pointer value. The value is converted to a string of the form "XXXX:YYYY" where XXXX and YYYY are the segment and offset parts of the pointer expressed as four hexadecimal digits.

s String. The argument must be a character, a string, or a PChar value. The string or character is inserted in place of the format specifier. The precision specifier, if present in the format string, specifies the maximum length of the resulting string. If the argument is a string that is longer than this maximum, the string is truncated.

x Hexadecimal. The argument must be an integer value. The value is converted to a string of hexadecimal digits. If the format string contains a precision specifier, it indicates that the resulting string must contain at least the specified number of digits; if the value has less digits, the resulting string is left-padded with zeros.

Conversion characters may be specified in upper case as well as in lower case--both produce the same results.

For all floating-point formats, the actual characters used as decimal and thousand separators are obtained from the DecimalSeparator and ThousandSeparator global variables.

Index, width, and precision specifiers can be specified directly using decimal digit string (for example "%10d"), or indirectly using an asterisk charcater (for example "%*.*f"). When using an asterisk, the next argument in the argument list (which must be an integer value) becomes the value that is actually used. For example "Format('%*.*f', [8, 2, 123.456])" is the same as "Format('%8.2f', [123.456])".

A width specifier sets the minimum field width for a conversion. If the resulting string is shorter than the minimum field width, it is padded with blanks to increase the field width. The default is to right-justify the result by adding blanks in front of the value, but if the format specifier contains a left-justification indicator (a "-" character preceding the width specifier), the result is left-justified by adding blanks after the value.

An index specifier sets the current argument list index to the specified value. The index of the first argument in the argument list is 0. Using index specifiers, it is possible to format the same argument multiple times. For example "Format('%d %d %0:d %d', [10, 20])" produces the string '10 20 10 20'.

The Format function can be combined with other formatting functions. For example

```
S := Format('Your total was %s on %s', [
  FormatFloat('$#,##0.00;;zero', Total),
  FormatDateTime('mm/dd/yy', Date)]);
```

which uses the FormatFloat and FormatDateTime functions to customize the format beyond what is possible with Format.

misc infomation

General Q&A on constructors.
Why can't my program find any of the resources that I put in a .RES file if that .RES file is the same name as my form's unit name?
What is the Object Pascal equivalent of C's "union" reserved word?
Does anyone know about a syntax problem with the ordering of the USES statement?
What are the caveats with callbacks and variable scope resolution?
Why aren't Delphi EXE files as small as BP files?
What is a Callback function, and how do I create one?
How do I put an ampersand in a caption without it being an underline?
How do I write programs using function pointers?
How do I store dates beyond the year 2000?
How do I write a simple windows program in BP?
How do I make sound the way that it worked in BP7?
How do I write a method that responds to a windows message?
How do I dial a phone?
How would I blank the screen in Delphi?
How do I execute a program and have my code wait until it is finished?
How do I change an icon to bitmap?
How do I read and write to a com-port?
How do I do bit-wise manipulation?
How do I get a DOS environment variable string?
Is there a way to get the integer value of a month given only the long form of a month name?
How do I get the julian date?
Here is a double linked list as an object.
How do I use "array of const"?
How do I use FmtStr()?
How do I eject paper from the printer?
How do I disable the screensaver?
How do I detect the pentium bug?
How can I verify if a string is a valid date?
How efficient is inc(i)?

memory management and arrays

How do I access hardware memory directly?
How can I parse a PChar?
Is there any way to dynamically redimension an array?
ARRAY: searching and sorting routines.
How do I emulate a VB control array?
How can I use huge arrays? (i.e. > 64K)
How do I do pointer arithmetic in Delphi?
How do I reduce the amount of memory taken from the data segment? (or How do I allocate memory dynamically?)
When would I use an array[0..0]?
How do I pass arrays into a procedure?
How do I use arrays in a Delphi DLL with VB?

Windows system housekeeping

How can I tell if share is loaded from Delphi?
How can I get the windows or dos versions?
Simulating multi-tasking (for long CPU intensive methods).
How can I tell which CPU is being used?
How do I detect for a co-processor?
Sendkeys
How do I terminate all running tasks?
How can I detect if there is already another copy of my app running and exit if so?
How can I detect the presense of a DLL that may or may not be loaded?
How can I check to see if there is a disk in the "A" drive?
How do I detect whether a drive exists or not?
How do I disable and enable the keyboard?
How do I override the default message handler for my Application?
How do I set the WindowState to wsMinimized when I minimize a form?
How can I get messages before my application's window procedure is called?
How do I handle TEdit text with windows messages only?
How can I create a windows meta-file from ellipse(), textOut(), etc. commands?
How can I make one window the child of another without using MDI?
How can I get information about window's tasks?
How do I disable the screensaver?
How do I broadcast changes to WIN.INI?
How do I manage disk volume labels in Delphi?

math

Why can't we have unsigned longInts?
Getting the High/Low order byte from a word.
I am migrating from VB. Where is the exponent function?
Here is a unit for huge numbers (64 and 128 bits).
Here are some extra math functions.
Here is a function to duplicate the payment() method in dBASE.
Here are some functions to use with the COMP type.

general

Doing Date Math On Calculated fields
IDAPI specifications
How do I fill a listbox from a table?
How do I fill a listbox from a memo field in a table?
How do I link tables with queries?
How do I set focus on a specific field on a TDBGrid?
Here is how to fill an outline component from a table.
How do I fill a listbox from a table?
How do I set a format for a data field?
FindNearest only show the record found when I enter a complete value.
I use the help file example for the ADD method and I get an error message "comma expected".
What is the syntax for the SQL SubString() method?
How do I access ACCESS tables?
How do I do a locate on a non-indexed field?
How can I tell the name of a field if I just know its number?
How can I determine the current record number for a dataset?
How can I determine when the current record in a dataset has changed?
How do I link tables with queries?
How do I make tables in a loop?
How do I make a virtual table (in memory)?
How can I create an empty table that has the structure of another table?
What are the values for the virtual keys?
How do I setup the column list in code for a dynamically created TTable?
How do I flush the data buffer to the table? (i.e. hard write to disk)

DLL

How do I manipulate a TStringList in a DLL?
How Do I pass a struct (**by reference**) from VB to a DLL created in Delphi?
DLL template
How do I call a "C" DLL from Delphi?
How can I detect the presense of a DLL that may or may not be loaded?
How do I call a dll function at runtime if the DLL is not known about at compile time?
How do I get a single returned from a C++ DLL (MS)?
How do I use a 32 bit DLL with 16 bit Delphi?

Specific components

TBitmap
TBlobStream
TCalendar
TCanvas
TCompatibleStream
TDBGrid
TDBNavigator
TEdit
TField
TFont
TFontDialog
THelpEvent
TImage
TList
TListBox
TMediaPlayer
TMemo
TMenu
TOutline
TReader
TReport
TSpeedButton
TStoredProc
TStringGrid
TTabbed Notebook
TTable
TWriter

ChartFX

using components

How do I make it so that when the user hits <enter>, it goes to the next object as though he had hit the tab key?
How can a right clicked component be determined while in an event handler of a popup menuitem?
How do I trap for right mouse clicks on my VBX and have a popup menu display?
How do I use a case statement to determine which object calls the procedure?

TFont

How can I find out what all of the available fonts are?
How can I determine the length in pixels of a string after a specific font has been applied to it?

paradox

How do I make it so that the form comes up without prompting the user for the password?
How do I transfer the text in a TMemo component on a form to a TMemofield in a Paradox table?
How do I make an ASCII text table from a paradox table?
How do I pack a Paradox table?
How to I create a Paradox table with an Auto Increment type field programatically?
What is the Delphi equivalent for the Paradox for Windows function 'cMax'?
How can I get my indexes to be listed in the drop down of the tTable IndexName property?
How do I fill a graphic field from a BMP file?

dBASE

How do I pack a dBASE table?
How do I create complex indices on dBASE tables in Delphi?
I can't open my DBF file because the MDX is missing.
How do I do a search on an expression index?
How can I view dBASE records marked for deletion?

Q: How can I have an animated icon (when the form is minimized)?

A: Note: this code does not work with Win95. It uses four TImage components to hold the icons to be used. The icons are changed from a timer.

```
{CurrentState is a byte that knows which icon is currently displayed.  It is
initialized in the form's OnCreate() method.}

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  if IsIconic(Application.Handle) then
    Begin
     case CurrentState Of
       1: Application.Icon := Image1.Picture.Icon;
       2: Application.Icon := Image2.Picture.Icon;
       3: Application.Icon := Image3.Picture.Icon;
       4: Application.Icon := Image4.Picture.Icon;
     end;{case}
     InvalidateRect(Application.Handle, nil, True); {This is critical!}
     inc(CurrentState);
     if CurrentState > 4 then
       CurrentState := 1;
    End;
end;
```

OLE

Ole automation spec

Ole automation spec

The following declarations are used in the definition of the new automation features:

```
  const

  { Parameter types }

    apVoid       = $00;
    apSmallint = $02;
    apInteger    = $03;
    apSingle     = $04;
    apDouble     = $05;
    apCurrency = $06;
    apDateTime = $07;
    apString     = $08;
    apWordBool = $0B;
    apVariant    = $0C;
    apTypeMask = $7F;
    apByRef      = $80;

  { Automation entry flags }

    afMethod    = $00000001;
    afPropGet = $00000002;
    afPropSet = $00000004;
    afVirtual = $00000008;

  type

  { Automation entry parameter list }

    PParamList = ^TParamList;
    TParamList = record
      ResType: Byte;
      ParamCount: Byte;
      ParamTypes: array[*] of Byte;
    end;

  { Automation table entry }

    TAutoEntry = record
      DispID: Integer;
      Name: PShortString;
      Flags: Integer;
      Params: PParamList;
```

```
    Address: Pointer;
  end;

{ Automation table layout }

  TAutoTable = record
    EntryCount: Integer;
    Entries: array[*] of TAutoEntry;
  end;
```

The compiler will support a new "automated" section which allows automated properties and methods to be declared.

The VMT is extended with a new automation table entry. The entry is NIL if the class has no "automated" section. Otherwise it points to a TAutoTable as defined above.

Only properties and methods can be declared in an "automated" section. Field declarations are not allowed. The visibility of an identifier declared in an "automated" section is the same as a "public" identifier.

All property types, parameter types, and function result types used in property and method declarations in an "automated" section must belong to the following set of types: Smallint, Integer, Single, Double, Currency, TDateTime, String, WordBool, and Variant.

Property declarations in an automated section can only include access specifiers ("read" and "write"). No other specifiers ("index", "stored", "default", "nodefault") are allowed. Access specifiers must list a method identifier (field identifiers are not allowed), and access methods must use register calling conventions. Array properties are supported, but property overrides (i.e. property declarations that don't include the property type) are not allowed.

Method declarations in an "automated" section must use register calling conventions. Methods can be virtual, but not dynamic. Method overrides are allowed in an "automated" section.

A property or method declaration may optionally specify a "dispid" directive. The "dispid" keyword must be followed by an integer constant expression which gives the dispatch ID of the property or method. If a "dispid" clause is not present, the compiler automatically picks a number that is one larger than the largest dispatch ID used by any property or method in the class and its ancestors. Specifying an already used dispatch ID in a "dispid" clause is an error.

For each non-override method declared in an "automated" section, the compiler generates one entry in the automation table. The afMethod flag is set in the Flags field to indicate that the entry represents a method.

For each property declared in an "automated" section, the compiler generates one or two entries in the automation table. The first entry (if present) describes the "read" method and has the afPropGet flag set in the Flags field. The second entry (if present) describes the "write" method and has the afPropPut flag set in the Flags field.

If an automation table entry describes a static method, the Address field contains the address of the method's entry point. If an automation table entry describes a virtual method, the afVirtual flag is set in the Flags field, and the Address field contains the v-table offset of the method.

The Name field of an automation table entry points to a ShortString that contains the identifier of the property or method.

The Params field of an automation table entry points to a TParamList record that describes the function result and parameter types of the method. For a procedure method, the ResType field contains apVoid. For a function
method, the ResType field contains the function result type. The ParamCount field contains the number of parameters, and the ParamTypes array contains the types of the parameters. The apByRef flag is set to indicate a "var" parameter.

Q: How can I make one window the child of another without using MDI?

A:

```
unit Unit2;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs;

type
  TForm2 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
    procedure CreateParams(var Params: TCreateParams); override;
  end;

var
  Form2: TForm2;

implementation

{$R *.DFM}

procedure TForm2.CreateParams(var Params: TCreateParams);
begin
  inherited CreateParams(Params);  { call the inherited first }
  with Params do begin
    Style := Style or WS_CHILD;  { add a style flag }
    WndParent := Application.MainForm.Handle;
  end;
end;

end.
```

```
StringVar := Format('%x', [IntegerVar]);
```

or

```
StringVar := IntToHex(IntegerVar, NumberOfDigits);
```

```
IntegerVar := StrToInt('$' + StringVar);
```

Threads

Here is an example of threads.

example of threads

```
{ thsorts.dpr }

program ThSorts;

uses
  Forms,
  main in 'main.pas' {Form1},
  secform in 'secform.pas' {Form2},
  thform in 'thform.pas' {Form3};

{$R *.RES}

begin
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.Run;
end.

{ main.pas }

unit main;

interface

uses
  SysUtils, WinTypes, WinProcs,
  Messages, Classes, Graphics,
  Controls, Forms, Dialogs,
  StdCtrls, ComCtrls, Buttons;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Label2: TLabel;
    Label1: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    BitBtn1: TBitBtn;
    BubbleTrackBar: TTrackBar;
    QuickTrackBar: TTrackBar;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
    T1 : THandle;
    T2 : THandle;
  public
    { Public declarations }
  end;

  TForm2 = class(TForm)
  private
    { Private declarations }
```

```
  public
    { Public declarations }
  end;

  TForm3 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

const
  aMax = 300;

var
  Form1: TForm1;
  Form2: TForm2;
  Form3: TForm3;
  a,b : array[0..aMax-1] of integer;
  numItems : integer;

implementation

{$R *.DFM}

procedure BubbleSort(var ia:array of integer; items: integer);
var
  i,j,t : integer;
begin
  for i := items downto 0 do
  begin
    for j := 0 to items-1 do
      if ia[j] < ia[j+1] then
      begin
        t := ia[j];
        form2.canvas.pixels[ia[j+1]+5,j+1+5] := clBlue;
        form2.canvas.pixels[ia[j]+5,j+5] := clBlue;
        ia[j] := ia[j+1];
        ia[j+1] := t;
        form2.canvas.pixels[ia[j+1]+5,j+1+5] := clYellow;
        form2.canvas.pixels[ia[j]+5,j+5] := clYellow;
      end;
   end;
end;

procedure QuickSort(var ia:array of integer; iLo,iHi : integer);
var
  Lo,Hi,Mid,T : integer;
begin
  Lo := iLo;
  Hi := iHi;
  mid := ia[(Lo+hi) div 2];
  repeat
    while ia[Lo] < mid do Inc(Lo);
    while ia[Hi] > mid do Dec(Hi);
    if Lo <= Hi then
    begin
```

```
      T := ia[Lo];
      form3.canvas.pixels[ia[Lo]+5,Lo+5] := clBlue;
      form3.canvas.pixels[ia[Hi]+5,Hi+5] := clBlue;
      ia[Lo] := ia[Hi];
      ia[Hi] := T;
      form3.canvas.pixels[ia[Lo]+5,Lo+5] := clLime;
      form3.canvas.pixels[ia[Hi]+5,Hi+5] := clLime;
      inc(Lo);
      dec(Hi);
    end;
  until Lo > Hi;
  if Hi > iLo then QuickSort(ia,iLo,Hi);
  if Lo < iHi then QuickSort(ia,Lo,iHi);
end;

function BubbleThread(parms:pointer) : LongInt; far;
begin
  BubbleSort(a,numItems-1);
end;

function QuickThread(parms:pointer) : LongInt; far;
begin
  QuickSort(b,0,numItems-1);
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  i : integer;
  ThreadID : dWord;
begin
  numItems := strToInt(Edit1.Text);
  if numItems <= aMax then
  begin
    form2.free;
    form2 := TForm2.Create(self);
    form2.top := 140;
    form2.left := 2;
    form2.clientheight := numItems+10;
    form2.clientwidth := numItems+10;
    form2.color := clBlue;
    form2.caption := 'Bubble Sort';
    form2.show;

    form3.free;
    form3 := TForm3.Create(self);
    form3.top := 140;
    form3.left := 320;
    form3.clientheight := numItems+10;
    form3.clientwidth := numItems+10;
    form3.color := clBlue;
    form3.caption := 'Quick Sort';
    form3.show;

    Randomize;
    for i := 0 to numItems-1 do
    begin
      a[i] := random(numItems);
```

```
      b[i] := a[i];
      form2.canvas.pixels[a[i]+5,i+5] := clYellow;
      form3.canvas.pixels[b[i]+5,i+5] := clLime;
    end;
    T1 := createThread(nil,0,@BubbleThread,nil,0,threadID);
    setThreadPriority(T1,BubbleTrackBar.Position);
    T2 := createThread(nil,0,@QuickThread,nil,0,threadID);
    setThreadPriority(T2,QuickTrackBar.Position);
  {  BubbleSort(a,j-1);
    QuickSort(b,0,j-1); }
  end
  else
    Form1.Caption := 'Too Large!';
end;

end.

{ main.dfm }

object Form1: TForm1
  Left = 192
  Top = 107
  Width = 309
  Height = 186
  Caption = 'Threaded Sorts'
  Font.Color = clWindowText
  Font.Height = -13
  Font.Name = 'System'
  Font.Style = []
  PixelsPerInch = 96
  TextHeight = 16
  object Label2: TLabel
    Left = 8
    Top = 0
    Width = 36
    Height = 19
    Caption = 'Items'
  end
  object Label1: TLabel
    Left = 0
    Top = 51
    Width = 76
    Height = 16
    Caption = 'Bubble Sort'
  end
  object Label3: TLabel
    Left = 10
    Top = 83
    Width = 66
    Height = 16
    Caption = 'Quick Sort'
  end
  object Label4: TLabel
    Left = 32
    Top = 32
    Width = 46
    Height = 16
```

```
  Caption = 'Priority'
  Color = clSilver
  Font.Color = clNavy
  Font.Height = -13
  Font.Name = 'Arial'
  Font.Style = [fsBold]
  ParentColor = False
  ParentFont = False
end
object Label5: TLabel
  Left = 87
  Top = 32
  Width = 140
  Height = 16
  Caption = '-2             0             +2'
  Color = clSilver
  Font.Color = clNavy
  Font.Height = -13
  Font.Name = 'System'
  Font.Style = [fsBold]
  ParentColor = False
  ParentFont = False
end
object Edit1: TEdit
  Left = 52
  Top = 0
  Width = 53
  Height = 24
  TabOrder = 0
  Text = '300'
end
object BitBtn1: TBitBtn
  Left = 120
  Top = 0
  Width = 113
  Height = 27
  Caption = 'Start Sorting'
  TabOrder = 1
  OnClick = Button1Click
  Glyph.Data = {
    36050000424D360500000000000036040000280000000F00000100000000100
    08000000000000010000CE0E0000C40E000000000000000000000000000000000
    80000080000000808000800000008000800080800000C0C0C000C0DCC000F0CA
    A6003F3F5F003F3F7F003F3F9F003F3FBF003F3FDF003F3FFF003F5F3F003F5F
    5F003F5F7F003F5F9F003F5FBF003F5FDF003F5FFF003F7F3F003F7F5F003F7F
    7F003F7F9F003F7FBF003F7FDF003F7FFF003F9F3F003F9F5F003F9F7F003F9F
    9F003F9FBF003F9FDF003F9FFF003FBF3F003FBF5F003FBF7F003FBF9F003FBF
    BF003FBFDF003FBFFF003FDF3F003FDF5F003FDF7F003FDF9F003FDFBF003FDF
    DF003FDFFF003FFF3F003FFF5F003FFF7F003FFF9F003FFFBF003FFFDF003FFF
    FF005F3F3F005F3F5F005F3F7F005F3F9F005F3FBF005F3FDF005F3FFF005F5F
    3F005F5F5F005F5F7F005F5F9F005F5FBF005F5FDF005F5FFF005F7F3F005F7F
    5F005F7F7F005F7F9F005F7FBF005F7FDF005F7FFF005F9F3F005F9F5F005F9F
    7F005F9F9F005F9FBF005F9FDF005F9FFF005FBF3F005FBF5F005FBF7F005FBF
    9F005FBFBF005FBFDF005FBFFF005FDF3F005FDF5F005FDF7F005FDF9F005FDF
    BF005FDFDF005FDFFF005FFF3F005FFF5F005FFF7F005FFF9F005FFFBF005FFF
    DF005FFFFF007F3F3F007F3F5F007F3F7F007F3F9F007F3FBF007F3FDF007F3F
    FF007F5F3F007F5F5F007F5F7F007F5F9F007F5FBF007F5FDF007F5FFF007F7F
```

```
      3F007F7F5F007F7F7F007F7F9F007F7FBF007F7FDF007F7FFF007F9F3F007F9F
      5F007F9F7F007F9F9F007F9FBF007F9FDF007F9FFF007FBF3F007FBF5F007FBF
      7F007FBF9F007FBFBF007FBFDF007FBFFF007FDF3F007FDF5F007FDF7F007FDF
      9F007FDFBF007FDFDF007FDFFF007FFF3F007FFF5F007FFF7F007FFF9F007FFF
      BF007FFFDF007FFFFF009F3F3F009F3F5F009F3F7F009F3F9F009F3FBF009F3F
      DF009F3FFF009F5F3F009F5F5F009F5F7F009F5F9F009F5FBF009F5FDF009F5F
      FF009F7F3F009F7F5F009F7F7F009F7F9F009F7FBF009F7FDF009F7FFF009F9F
      3F009F9F5F009F9F7F009F9F9F009F9FBF009F9FDF009F9FFF009FBF3F009FBF
      5F009FBF7F009FBF9F009FBFBF009FBFDF009FBFFF009FDF3F009FDF5F009FDF
      7F009FDF9F009FDFBF009FDFDF009FDFFF009FFF3F009FFF5F009FFF7F009FFF
      9F009FFFBF009FFFDF009FFFFF00BF3F3F00BF3F5F00BF3F7F00BF3F9F00BF3F
      BF00BF3FDF00BF3FFF00BF5F3F00BF5F5F00BF5F7F00BF5F9F00BF5FBF00BF5F
      DF00BF5FFF00BF7F3F00BF7F5F00BF7F7F00BF7F9F00BF7FBF00BF7FDF00BF7F
      FF00BF9F3F00BF9F5F00BF9F7F00BF9F9F00BF9FBF00BF9FDF00BF9FFF00BFBF
      3F00BFBF5F00BFBF7F00BFBF9F00BFBFBF00BFBFDF00BFBFFF00BFDF3F00BFDF
      5F00BFDF7F00BFDF9F00BFDFBF00BFDFDF00F0FBFF00A4A0A000808080000000
      FF0000FF000000FFFF00FF000000FF00FF00FFFF0000FFFFFF00070707070707
      0707070707070707070007070707F807070707070707070700070707070700
      0007070707070707070007070707070FF00070707070707070700070707070700
      FBFF00070707070707000707070F80000FF0000F807070707070007070707070FF
      FB00070707070707070007070707070FFFB00070707070707070007070707070700
      FBFFFB00070707070700070F8000000FF000000F807070707000707070FBFF
      FBFF00070707070707000707070700FBFFFBFF00070707070700070707070FF
      FBFFFBFF000707070700070707070700000000000F807070700070707070707
      0707070707070707070007070707070707070707070707070700}
  end
  object BubbleTrackBar: TTrackBar
    Left = 88
    Top = 56
    Width = 145
    Height = 17
    TickStyle = tsAuto
    TickMarks = tmBottomRight
    Orientation = tbHorizontal
    Position = 0
    SelStart = 0
    SelEnd = 0
    Min = -2
    Max = 2
  end
  object QuickTrackBar: TTrackBar
    Left = 88
    Top = 80
    Width = 142
    Height = 17
    TickStyle = tsAuto
    TickMarks = tmBottomRight
    Orientation = tbHorizontal
    Position = 0
    SelStart = 0
    SelEnd = 0
    Min = -2
    Max = 2
  end
end

{ secform.pas }
```

```
unit secform;

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs;

type
  TForm2 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.DFM}

end.

{ secform.dfm }

object Form2: TForm2
  Left = 200
  Top = 104
  Width = 435
  Height = 300
  Caption = 'Form2'
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  PixelsPerInch = 96
  TextHeight = 13
end

{ thform.pas }

unit thform;

interface

uses
  SysUtils, Windows, Messages, Classes, Graphics, Controls, Forms, Dialogs;

type
  TForm3 = class(TForm)
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  Form3: TForm3;

implementation

{$R *.DFM}

end.

{ thform.dfm }

object Form3: TForm3
  Left = 200
  Top = 104
  Width = 435
  Height = 300
  Caption = 'Form3'
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'MS Sans Serif'
  Font.Style = []
  PixelsPerInch = 96
  TextHeight = 13
end
```

Quotes from famous people about computers in particular and progress in general:

"Computers in the future may weigh no more than 1.5 tons."
--Popular Mechanics, forecasting the relentless march of science, 1949

"I think there is a world market for maybe five computers."
--Thomas Watson, chairman of IBM, 1943

"I have traveled the length and breadth of this country and talked with the best people, and I can assure you that data processing is a fad that won't last out the year."
--The editor in charge of business books for Prentice Hall, 1957

"But what ... is it good for?"
--Engineer at the Advanced Computing Systems Division of IBM,1968, commenting on the microchip.

"There is no reason anyone would want a computer in their home."
--Ken Olson, president, chairman and founder of Digital Equipment Corp., 1977

"This 'telephone' has too many shortcomings to be seriously considered as a means of communication. The device is inherently of no value to us."
--Western Union internal memo, 1876.

"The wireless music box has no imaginable commercial value. Who would pay for a message sent to nobody in particular?"
--David Sarnoff's associates in response to his urgings for investment in the radio in the 1920s.

"The concept is interesting and well-formed, but in order to earn better than a 'C,' the idea must be feasible."
--A Yale University management professor in response to Fred Smith's paper proposing reliable overnight delivery service. (Smith went on to found Federal Express Corp.)

"Who the hell wants to hear actors talk?"
--H.M. Warner, Warner Brothers, 1927.

"I'm just glad it'll be Clark Gable who's falling on his face and not Gary Cooper."
--Gary Cooper on his decision not to take the leading role in "Gone With The Wind."

"A cookie store is a bad idea. Besides, the market research reports say America likes crispy cookies, not soft and chewy cookies like you make."
--Response to Debbi Fields' idea of starting Mrs. Fields' Cookies.

"We don't like their sound, and guitar music is on the way out."

--Decca Recording Co. rejecting the Beatles, 1962.

"Heavier-than-air flying machines are impossible."
--Lord Kelvin, president, Royal Society, 1895.

"If I had thought about it, I wouldn't have done the experiment. The literature was full of examples that said you can't do this."
--Spencer Silver on the work that led to the unique adhesives for 3-M "Post-It" Notepads.

"So we went to Atari and said, 'Hey, we've got this amazing thing, even built with some of your parts, and what do you think about funding us? Or we' ll give it to you. We just want to do it. Pay our salary, we'll come work for you.' And they said, 'No.' So then we went to Hewlett-Packard, and they said, 'Hey, we don't need you. You haven't got through college
yet.'"
--Apple Computer Inc. founder Steve Jobs on attempts to get Atari and H-P interested in his and Steve Wozniak's personal computer.

"Professor Goddard does not know the relation between action and reaction and the need to have something better than a vacuum against which to react. He seems to lack the basic knowledge ladled out daily in high schools."
--1921 New York Times editorial about Robert Goddard's revolutionary rocket work.

"You want to have consistent and uniform muscle development across all of your muscles? It can't be done. It's just a fact of life. You just have to accept inconsistent muscle development as an unalterable condition of weight training."
--Response to Arthur Jones, who solved the "unsolvable" problem by inventing Nautilus.

"Drill for oil? You mean drill into the ground to try and find oil? You're crazy."
--Drillers who Edwin L. Drake tried to enlist to his project to drill for oil in 1859.

"Stocks have reached what looks like a permanently high plateau."
--Irving Fisher, Professor of Economics, Yale University, 1929.

"Airplanes are interesting toys but of no military value."
--Marechal Ferdinand Foch, Professor of Strategy, Ecol Superieure de Guerre.

"Everything that can be invented has been invented."
--Charles H. Duell, Commissioner, U.S. Office of Patents, 1899.

"Louis Pasteur's theory of germs is ridiculous fiction".
--Pierre Pachet, Professor of Physiology at Toulouse, 1872

"The abdomen, the chest, and the brain will forever be shut from the intrusion of the wise and humane surgeon".

--Sir John Eric Ericksen, British surgeon, appointed Surgeon-Extraordinary to Queen Victoria 1873.

"640K ought to be enough for anybody."

-- Bill Gates, 1981

Q: How do I emulate TCollection in Delphi?

A: Borland Pascal 7.0 has a very useful object called a TCollection that allows you to sort items by any key you wish and that disposes of any objects it owns when the collection itself is freed. Inexplicably, these very useful behaviors were omitted from Delphi's TStringList. But now... <trumpet fanfare>

This unit contains a class called "TSortableList" derived from TStringList that supports the ability for you to define descendent classes that sort the list any way you wish and has the option to "own" the objects that are added to it, so that they are disposed with the list.

How to use a TSortableList that owns its objects:

1. Create a new TSortableList and tell it that it should own any objects in it.

```
var
  my_list : TSortableList;
begin
my_list := TSortableList.Create(True);     { "True" = Owns objects }
my_list.Add('Aaa', TObject.Create);        { In a TStringList,
this... }
my_list.Add('Bbb', TObject.Create);        { ...would be... }
my_list.Add('Ccc', TObject.Create);        { ...very bad! }
my_list.Free;                              { All objects freed }
end;
```

How to sort on an arbitrary key:

Suppose you wanted a list of strings sorted by all but the first character (i.e. this order -> "ZAble", "YBaker", "XCharlie").

1. Declare a descendent of TSortableList and override the Compare method. The Compare method should return an integer such that the result is:

-1 if the item at index i1 is "less" than the item at i2
0 if the item at index i1 is "equal" than the item at i2
1 if the item at index i1 is "more" than the item at i2

```
TExList = class(TSortableList)
  function Compare(i1, i2 : Integer) : Integer; override;
  end;
```

2. Define the new compare method

```
function Compare(i1, i2 : Integer) : Integer;
  begin
  case Key of
    1 :
      Result := AnsiCompareText(Copy(Strings[i1], 2, 254),
                                Copy(Strings[i2], 2, 254));
```

```
    else
      Result := inherited Compare(i1, i2);
    end;
  end;
```

3. Specify the key you just defined.

```
var
  my_list : TExList;
begin
my_list := TExList.Create;
my_list.Key := 1;               { <<<<< New key is made active }
DoSomeStuff;
my_list.Free;
end;
```

There you go! I *strongly* suggest that any Key that you define Compare methods for have a value of at least 1 and that all unhandled Key values be passed to the inherited method, as above. A Key of 0 is defined to be the default alphabetical sort.

Note that you can define a Key based on the objects in a list, like so:

```
function Compare(i1, i2 : Integer) : Integer;
  begin
  case Key of
    1 :
      Result := AnsiCompareText(TSomeObject(Objects[i1]).Text,
                                TSomeObject(Objects[i2].Text);
    else
      Result := inherited Compare(i1, i2);
    end;
  end;
```

Of course, it is your responsibility to be sure that the objects in the list are the type that your Compare method assumes them to be.

Important

If you do have a list that is a) sorted, and b) determines its order via values derived from the objects that are stored in the list, watch out for changing objects in such a way as to change their sort order; the TSortableList will not know that the list is now out of order and calls to routines that depend on knowing this (such as Add) may fail to work. Your best bet in this case is to set Sorted to False, make whatever changes to the objects you wish, and then set Sorted back to True. This will resort the list.

I also took the liberty of "protecting" the Find method against the possibility that someone would call it when the list was not sorted -- in that case, it now calls IndexOf (which, somewhat recursively, would call Find if the list *was* sorted). This is exactly what happened in the example code for Find! If you look at the method list for TStringList, you won't find Find -- but you can do a topic search for it. The example

code for Find works, but only by chance -- add another string to either end of the list, and "Flowers" won't be found. What's wrong with the example code is that the list's Sorted property is not set to True; the reason it (accidently) works is because the item that was being found ("Flowers") happened to be the middle item in the list, which is where the search algorithm looks first.

```
unit TSortLst;

interface

Uses
  Classes;

type
  TSortableList = class(TStringList)
    private
    FOwnsObjects : Boolean;
    FKey : Integer;
    FAscending : Boolean;
    procedure QuickSort(left, right: Integer);
    function CallCompare(i1, i2 : Integer) : Integer;
    procedure SetAscending(value : Boolean);
    procedure SetKey(value : Integer);
    protected
    procedure PutObject(index : Integer; AObject : TObject); override;
    function Compare(i1, i2 : Integer) : Integer; virtual;
    public
    constructor Create(owns_objects : Boolean);
    procedure Clear; override;
    procedure Delete(index : Integer); override;
    function Find(const s : string; var index : Integer): Boolean; override;
    procedure Sort; override;
    property Ascending : Boolean read FAscending write SetAscending;
    property Key : Integer read FKey write SetKey;
    property OwnsObjects : Boolean read FOwnsObjects;
    end;

implementation

Uses
  SysUtils;

{ Private Methods }

procedure TSortableList.QuickSort(left, right: Integer);
  var
    i, j, pivot : Integer;
    s : String;
  begin
  i := left;
  j := right;

  { Rather than store the pivot value (which was assumed to be a string),
    store the pivot index }
```

```
  pivot := (left + right) shr 1;

  repeat
    while CallCompare(i, pivot) < 0 do
      Inc(i);
    while CallCompare(j, pivot) > 0 do
      Dec(j);
    if i <= j then
      begin
      Exchange(i, j);

      { If we just moved the pivot item, reset the pivot index }
      if pivot = i then
        pivot := j
      else if pivot = j then
        pivot := i;
      Inc(i);
      Dec(j);
      end;
    until i > j;
  if left < j then
    QuickSort(left, j);
  if i < right then
    QuickSort(i, right);
  end;

function TSortableList.CallCompare(i1, i2 : Integer) : Integer;
  begin
  Result := Compare(i1, i2);
  if not FAscending then
    Result := -Result;
  end;

procedure TSortableList.SetAscending(value : Boolean);
  begin
  if value <> FAscending then
    begin
    FAscending := value;
    if Sorted then
      begin
      Sorted := False;
      Sorted := True;
      end
    end;
  end;

procedure TSortableList.SetKey(value : Integer);
  begin
  if value <> FKey then
    begin
    FKey := value;
    if Sorted then
      begin
      Sorted := False;
      Sorted := True;
      end
    end;
```

```
  end;

{ Protected Methods }

function TSortableList.Compare(i1, i2 : Integer) : Integer;
  begin
  Result := AnsiCompareText(Strings[i1], Strings[i2]);
  end;

{ Public Methods }

constructor TSortableList.Create(owns_objects : Boolean);
  begin
  inherited Create;
  FOwnsObjects := owns_objects;
  FKey := 0;
  FAscending := True;
  end;

procedure TSortableList.Clear;
  var
    index : Integer;
  begin
  Changing;
  if FOwnsObjects then
    for index := 0 to Count - 1 do
      GetObject(index).Free;
  inherited Clear;
  Changed;
  end;

procedure TSortableList.Delete(index: Integer);
  begin
  Changing;
  if FOwnsObjects then
    GetObject(index).Free;
  inherited Delete(index);
  Changed;
  end;

function TSortableList.Find(const s : string; var index : Integer): Boolean;
  begin
  if not Sorted then
    begin
    index := IndexOf(s);
    Result := (index <> -1);
    end
  else
    Result := inherited Find(s, index);
  end;

procedure TSortableList.PutObject(index: Integer; AObject: TObject);
  begin
  Changing;
  if FOwnsObjects then
    GetObject(index).Free;
  inherited PutObject(index, AObject);
```

```
  Changed;
  end;

procedure TSortableList.Sort;
  begin
  if not Sorted and (Count > 1) then
    begin
    Changing;
    QuickSort(0, Count - 1);
    Changed;
    end;
  end;

end.
```

If you have any comments, suggestions or even criticisms <g> for TSortableList, hey, tough! No, really, send me some mail at 71744,422. I am particularly interested in bug reports. Version 1.0 5/12/95 - Mike Stortz

Q: How do I fake TTabbedNotebook with multiple forms?

A: This uses a TabSet to do the "page choosing". First, you must make an array that will hold the form pointers that looks something like this:

```
FakePage: array[0..2] of TForm;
```

Then you need these methods:

```
procedure TForm1.FormShow(Sender: TObject);
var i: integer;
begin
  FakePage[0] := TForm2.create(application);
  FakePage[1] := TForm3.create(application);
  FakePage[2] := TForm4.create(application);
  for i := 0 to 2 do begin
    FakePage[i].parent := panel1;
    FakePage[i].TabOrder := panel1.TabOrder;
    with panel1 do
      FakePage[i].SetBounds(left + 2, top + 2, width - 4, height - 4);
  end;
  FakePage[TabSet1.TabIndex].show;
end;

procedure TForm1.TabSet1Change(Sender: TObject; NewTab: Integer;
  var AllowChange: Boolean);
begin
  FakePage[TabSet1.TabIndex].hide;
  FakePage[NewTab].show;
end;
```

This is the code that we place on the forms :

First, the prototype in the form's declaration:

```
procedure CreateParams(var params: TCreateParams); override;
```

Then the custom code:

```
procedure TForm2.CreateParams(var params: TCreateParams);
begin
  inherited CreateParams(params);
  with params do begin
    WndParent := application.MainForm.handle;
    style := ws_child or ws_ClipSiblings;
    x := 0;
    y := 0;
  end;
end;
```

Here is an example of how to manipulate the form itself:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  case Tabset1.TabIndex of
    0: (FakePage[0] as TForm2).edit1.text := 'This is for form 2.';
    1: (FakePage[1] as TForm3).edit1.text := 'This is for form 3.';
    2: (FakePage[2] as TForm4).edit1.text := 'This is for form 4.';
  end;
end;
```

Q: How do I draw on the pages of a TTabbedNotebook?

A:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  dc: hdc;
  s: PChar;
begin
  s := StrNew('Yo, mama!');
  dc := GetDC((TabbedNotebook1.pages.objects[0] as TTabPage).handle);
  TextOut(dc, 10, 10, s, StrLen(s));
  ReleaseDC((TabbedNotebook1.pages.objects[0] as TTabPage).handle, dc);
  StrDispose(s);
end;
```

Q: How do I make it so that only the form I select comes to the top? (i.e. *without* the main form)

A: Try this in any secondary window that you DON'T want dragging the program along:

```
  ...
  private {This goes in the for's type declaration.}
    { Private declarations }
    procedure CreateParams(VAR Params: TCreateParams); override;
  ...
procedure TForm2.CreateParams(VAR Params: TCreateParams);
begin
  Inherited CreateParams(Params);
  Params.WndParent := GetDesktopWindow;
end;
```

By setting the form's parent window handle to the desktop, you remove the link that would normally force the whole application to come to the top when this form comes to the top.

Q: How do I use a 32 bit DLL with 16 bit Delphi?

A: Here is something that I got from compuserve that looks good:

CALL32nt.pas: Library for Delphi/TPW/BPW to call 32 bit functions in Windows NT or Windows 95

Adapted to Pascal by Christian Ghisler from CALL32.DLL, a DLL for Visual Basic written and placed in the Public Domain by Peter Golde

This unit is placed in the public domain. Please feel free to redistribute as you wish. No guarantees are made as to its suitability or usefulness, and no support can be provided.

To call a function in a 32-bit DLL, follow these steps:

1. Declare the function you wish to call. Declare it in the ordinary fashion, with the following exceptions:

> Declare it as a function variable > Add an additional argument at the end, of type Longint:

For example, if you are calling the function: (C code)

GetWindowText(HWND hwnd, LPSTR lpsz, int cch)

declare it as follows (remember that ints and all handles are 32 bits, so use a Longint):

var
GetWindowText:function(hwnd:Longint;lpsz:PChar;cch:longint;id:Longint):Longint;

2. Each function needs an identifier to distinguish the function from other called functions. Declare this identifier in a var block.

For the above example:

var id_GetWindowText:longint;

3. In the initialization section of your application, set the address of the called function to the address of Call32:

@GetWindowtext:=@Call32;

4. Also in the initialization section of your application, declare the actual library and name of the function you

want to call with the Declare32 function. Pass it the name of the function (CASE SENSITIVE!!!), the library name, and a string describing the argument types.

Each letter in the string declares the type of one argument, and should be either "i" for a 32-bit integer or handle
type, "p" for any pointer type, or "w" for an HWND parameter to which you want to pass a 16-bit HWND and have it be automatically converted to a 32-bit HWND. Save the return value of Declare32 in a global variable to pass as the last parameter to the function you declared earlier. So, in continuing the example, you would call:

```
id_GetWindowText:=Declare32('GetWindowText','user32','wpi');
```

(As a side note, this more properly would be declared as 'GetWindowTextA', since this is the real exported name. However, Declare32 will automatically add an 'A' to the end of a function name if necessary.)

To call the function, you would call:

```
cbCopy:=GetWindowText(hwnd, sz, cb, id_GetWindowText);
```

It is important to use the correct data types when calling DLL functions. There are two important points to pay
attention to when using CALL32NT.PAS.

First, only 32-bit integers can be passed to a DLL procedure. Since virtually all 32-bit functions take int, UINT, LONG, DWORD, or HANDLE parameters, which are all 32 bits, this is not a major restriction. However, you must remember to always declare function arguments as Longint, not Integer.

Second, 16-bit handles and 32-bit handles are not interchangeable. For example, a 16-bit bitmap handle that you get from calling a 16-bit DLL or from the Delphi/TPW environment cannot be passed to a 32-bit function expecting a bitmap handle. Similarly, a 32-bit handle obtained from a 32-bit function cannot be passed to a 16- bit DLL. The only exception is window handles (HWND). If you declare a function parameter with the "w" letter in the argument description string passed to Declare32, the corresponding parameter will be automatically converted from a 16-bit HWND to a 32-bit HWND when the call is made. You must still declare the argument as a LONG. This is convenient, for example, when passing the value returned by the "handle" property of a form/control to a 32-bit DLL function. Only windows created by your application can be translated.

The following is a summary of data types:

C data type	Type specified in Declare	Character for Declare32
int, UINT	Longint	i
LONG, DWORD	Longint	i
HANDLE	Longint	i
WORD, short	not supported	
HWND	Longint	w (i for no 16->32

```
translation)
   LPSTR               PChar                    p
   LPLONG, LPDWORD,
   LPUINT, int FAR *   VAR x:Longint            p
   LPWORD              VAR x:Word               p
```

Note on Declare32 function names: Declare32 will automatically try three different names for the function name you pass in. First, it uses the exact name you pass in. If it doesn't find that function name, it converts the name to the stdcall decorated name convention by adding an underscore at the beginning and adding "@nn" at the end, where "nn" is the number of bytes of arguments. If it doesn't find that name, it adds an "A" to the end of the original name to try the Win32(R) ANSI function calling convention.

If there occurs an error in Declare32, the returned id will be less than 0. Also, the variable Call32NTError will be set, so you only have to check one variable to check that all went well. You can use this variable to distinguish between Windows 3.1 and Windows NT/95: if Call32NTError is false, you can use the declared 32-bit functions, otherwise you must use 16-bit replacement functions. This allows you to write programs which work in both 16 and 32 bit environments.

If you have to pass a record containing a pointer, you must use the function GetVDMPointer32W to create a 0:32 pointer from your 16:16 pointer.

CALL32NT requires the Microsoft Windows NT operating system or Windows 95 Preview or later to perform its task. The program will also run in Win 3.1, but of course the functions will not work.

```
Unit Call32nt;
{Delphi/TPW/BPW Unit to call 32-bit functions from 16 bit programs}
{Written in Turbo Pascal for Windows 1.5 /Delphi}
{By Christian Ghisler, CIS: 100332,1175         }
{Released to the public domain on June 14,1995  }

{$W-}
{No Windows Stack frame!}
{$R-}
{No range checking!}

{
Translation by Christian Ghisler, from:
//-------------------------------------------------------
// CALL32.C
//
// This creates a DLL for 16-bit Visual Basic programs to
// call 32-bit DLLs on Windows NT 3.1.  It uses the
// Generic Thunks feature of the WOW subsystem on Windows
// NT to load and call 32 bit DLLs.  This file should
// be compile into a 16-bit DLL.
//
// Writted by Peter Golde.
//-------------------------------------------------------
```

```
}
interface

uses wintypes,
     winprocs,
     {$ifdef ver80}sysutils {$else} strings {$endif};

const Call32NTError:boolean=false;

type tPROC32ENTRY=record
    hinst:longint;      { 32-bit instance handle of
library                  }
    lpfunc:tfarproc;    { 32-bit function address of
function                }
    dwAddrXlat,         { bit mask of params: 1 indicates arg is
address     }
    dwHwndXlat,         { bit mask of params: 1 indicates arg is 16-bit
hwnd }
    nParams:longint;    { number of
parameters                              }
  end;
  pPROC32ENTRY=^tPROC32ENTRY;
  tPROC32LIST=array[0..0] of tPROC32ENTRY;
  pPROC32LIST=^tPROC32LIST;

{ rgProc32Entry points to an array of PROC32ENTRY functions, which
  is grown as needed.  The value returned by Declare32 is an
  index into this array.}
const
  cRegistered:integer=0;          { number of registered functions. }
  cAlloc:integer=0;               { number of alloced PROC32ENTRY structures.
}
  rgPROC32ENTRY:pPROC32LIST=nil;  { array of PROC32ENTRY structures. }
  CALLOCGROW=10;                  { number of entries to grow rgProc32Entry
by}
  rgProc32handle:thandle=0;       { Handle auf globalen Speicherbereich für
rgProc32Entry }

{ These are the addresses of the Generic Thunk functions in
  the WOW KERNEL.}
  fGotProcs:bool=FALSE;        { Did we successfully get the addresses? }

var
  Callproc32W:function (address:pointer;n,c:longint):longint;
  FreeLibrary32W:function(handle:longint):bool;
  GetProcAddress32W:function(module:longint;funcname:pchar):pointer;
  LoadLibraryEx32W:function(libname:pchar;a,b:longint):longint;
  lpvGetLastError:function:pchar;
  lpvGetCapture:pointer;

procedure Call32(iProc:longint);
function Declare32(lpstrName,lpstrLib,lpstrArg:pchar):longint;
function GetVDMPointer32W(name:pchar;Length:word):longint;    {Get 32-bit
pointer from 16-bit pointer and length}

implementation
```

```
{/-------------------------------------------------
// XlatHwnd
//   Translates a 16-bit HWND into a 32-bit HWND.
//   The HWND must be one in our 16-bit process.
//   NULL is translated to NULL and doesn't cause
//   and error.
//
//   Unfortunately, WOW does not export a function
//   for doing this, so our procedure is as follows:
//   We do 16-bit SetCapture call to the window
//   to set the capture, and then a 32-bit GetCapture
//   call to get the 32-bit equivalent handle.  The
//   capture is then restored to what it was beforehand.
//
//   May cause VB runtime error, and hence never return.
//-------------------------------------------------}
procedure XlatHwnd(var phwnd:longint);
var hwnd16,
    hwndCapturePrev:word;
    hwnd32,
    hinstUser:longint;

begin
  hwnd16:=LOWORD(phwnd);         { 16-bit hwnd }

  { Check for valid 16-bit handle. }
  if (phwnd<>word(hwnd16)) then exit;
  if (hwnd16<>0) and not IsWindow(hwnd16) then exit;

  { Get Address of 32-bit GetCapture }
  if (@lpvGetCapture=nil) then begin
    hinstUser:=LoadLibraryEx32W('user32', 0, 0);
    if (hinstUser<>0) then begin
      lpvGetCapture:=GetProcAddress32W(hinstUser, 'GetCapture');
      FreeLibrary32W(hinstUser);
    end;
    if (@lpvGetCapture=nil) then exit;
  end;


  {/ Set capture to window, get capture to get 32-bit handle.
  // Be sure to restore capture afterward.
  // NULL isn't translated }

  if (hwnd16<>0) then begin
    hwndCapturePrev:=SetCapture(hwnd16);
    hwnd32:=CallProc32W(lpvGetCapture,0,0);
    if (hwndCapturePrev<>0) then
      SetCapture(hwndCapturePrev)
    else
      ReleaseCapture;
    if (hwnd32=0) then exit;
  end;

  phwnd:=hwnd32;
```

```
end;

{//-----------------------------------------------------
// MungeArgs
//   Modify the args array so it can be passed to
//   to CallProc32W.  This uses the PROC32ENTRY structure
//   to set up the arg list correctly on the stack
//   so CallProc32W can be call.  HWND translation is
//   performed.  The frame is changed as follows:
//           In:                Out:
//            unused             number of params
//   dwArgs-> unused             address xlat mask
//            PROC32ENTRY index  32-bit function address.
//            argument           argument, possible HWND xlated
//            argument           argument, possible HWND xlated
//            ...                ...
//-----------------------------------------------------}
type plongint=^longint;
     pfarproc=^tfarproc;
procedure MungeArgs(dwArgs:longint);
var pentry:pPROC32ENTRY;
    iArg:integer;
    dwHwndXlat:longint;

begin
  pentry:=@rgProc32Entry^[plongint(dwArgs+4)^];
  iArg:=2;

  plongint(dwArgs-4)^:=pentry^.nParams;
  plongint(dwArgs)^:=pentry^.dwAddrXlat;
  pfarproc(dwArgs+4)^:=pentry^.lpfunc;
  dwHwndXlat:=pentry^.dwHwndXlat;
  while (dwHwndXlat<>0) do begin
    if (dwHwndXlat and 1)<>0 then
    XlatHwnd(plongint(dwArgs+4*iArg)^);
    inc(iArg);
    dwHwndXlat:=dwHwndXlat shr 1;
  end;
end;

{//-----------------------------------------------------
// Call32
//   This function is called by applications directly.
//   Arguments to the function are also on the stack
//   (iProc is the PROC32ENTRY index).  We correctly
//   set up the stack frame, then JUMP to CallProc32W,
//   which eventually returns to the user.
//-----------------------------------------------------}

var dest:tfarproc;          {Destination for jump back!}
var addit:word;             {value to add to sp to restore stack pointer}
var _sp,_bp:word;

procedure Call32(iProc:longint);
begin
  if iProc<0 then begin      {Procedure is invalid -> stop execution!}
    if messagebox(0,'Error calling 32 bit function, continue?','Call32',
```

```
      mb_yesno or mb_iconquestion)=idno then halt(1);
    addit:=(-iProc) shl 2;  {4 more for id!}
    asm
      mov sp,bp
      pop bp
      pop di
      mov word(dest),di
      pop di
      mov word(dest+2),di
      add sp,addit
      xor ax,ax             {return 0}
      xor dx,dx
      jmp dest
    end;
  end;

  asm                       { here comes the thunking call! }
    pop     bp              { restore BP }
    mov     bx, sp          { bx = sp on entry }
    sub     sp, 8           { 2 additional words }
    mov     ax, ss:[bx]     { ax = return address offst }
    mov     dx, ss:[bx+2]   { dx = return address segment }
    mov     ss:[bx-8], ax
    mov     ss:[bx-6], dx
    push    ds              { Save our DS }
    push    ss
    push    bx              { Push pointer to args }
    call    MungeArgs       { Munge the args }
    pop     es              { es is our DS }
    jmp    CallProc32W      { Jump to the call thunker }
  end;
end;

{/-----------------------------------------------------
// Declare32
//   This function is called directly from VB.
//   It allocates and fills in a PROC32ENTRY structure
//   so that we can call the 32 bit function.
//-----------------------------------------------------}
function Declare32(lpstrName,lpstrLib,lpstrArg:pchar):longint;
var
  hinst:longint;                  { 32-bit DLL instance handle }
  lpfunc:pointer;                 { 32-bit function pointer    }
  dwAddrXlat,                     { address xlat mask          }
  dwHwndXlat,                     { hwnd xlat mask             }
  nParams:longint;                { number of params           }
  szBuffer:array[0..127] of char; { scratch buffer             }
  hinstKernel:word;               { Instance handle of WOW KERNEL.DLL }
  hinstKernel32:longint;          { Instance handle of Win32 KERNEL32.DLL }
  rg:record
    lpstrName:pchar;
    nparams:longint;
  end;
  olderror:boolean;               { Was there an error before?}

begin
  {/ First time called, get the addresses of the Generic Thunk
```

```
  // functions.  Raise VB runtime error if can't (probably because
  // we're not running on NT). }
  olderror:=Call32NTError;
  Call32NTError:=true;
  Declare32:=-1-lstrlen(lpstrArg);
  if not fGotProcs then begin
    hinstKernel:=LoadLibrary('KERNEL');
    if (hinstKernel < 32) then exit;

    @CallProc32W:=GetProcAddress(hinstKernel, 'CALLPROC32W');
    @FreeLibrary32W:=GetProcAddress(hinstKernel, 'FREELIBRARY32W');
    @LoadLibraryEx32W:=GetProcAddress(hinstKernel, 'LOADLIBRARYEX32W');
    @GetProcAddress32W:=GetProcAddress(hinstKernel, 'GETPROCADDRESS32W');
    FreeLibrary(hinstKernel);

    if (@LoadLibraryEx32W<>nil) and (@GetProcAddress32W<>nil) and
(@FreeLibrary32W<>nil) then begin
      hinstKernel32:=LoadLibraryEx32W('kernel32', 0, 0);
      @lpvGetLastError:=GetProcAddress32W(hinstKernel32, 'GetLastError');
      FreeLibrary32W(hinstKernel);
    end;

    if (@CallProc32W=nil) or (@FreeLibrary32W=nil) or (@LoadLibraryEx32W=nil)
or
       (@GetProcAddress32W=nil) or (@lpvGetLastError=nil) then begin
      exit;
    end;
    fGotProcs:=TRUE;
  end;

  { If needed, allocate a PROC32ENTRY structure }
  if (cRegistered = cAlloc) then begin
    if (rgProc32Entry<>nil) then begin
      globalunlock(rgProc32handle);
      rgProc32handle:=GlobalReAlloc(rgProc32handle,
                      (cAlloc + CALLOCGROW) * sizeof(tPROC32ENTRY),
GMEM_MOVEABLE);
      rgProc32Entry:=Globallock(rgProc32handle);
    end else begin
      rgProc32handle:=GlobalAlloc(GMEM_MOVEABLE, CALLOCGROW *
sizeof(tPROC32ENTRY));
      rgProc32Entry:=Globallock(rgProc32handle);
    end;
    if (rgProc32Entry=nil) then exit;
    inc(cAlloc,CALLOCGROW);
  end;

  {/ Process the arg list descriptor string to
  // get the hwnd and addr translation masks, and the
  // number of args. }

  dwAddrXlat:=0;
  dwHwndXlat:=0;
  nParams:=lstrlen(lpstrArg);
  if (nParams > 32) then exit;  {Too many parameters}

  while (lpstrArg[0]<>#0) do begin
```

```
    dwAddrXlat:=dwAddrXlat shl 1;
    dwHwndXlat:=dwHwndXlat shl 1;
    case lpstrArg[0] of
      'p':dwAddrXlat:=dwAddrXlat or 1;
      'i': ;
      'w':dwHwndXlat:=dwHwndXlat or 1;
    else
      exit;
    end;
    inc(lpstrArg);
  end;

  {/ Load the 32-bit library. }
  hinst:=LoadLibraryEx32W(lpstrLib, 0, 0);
  if (hinst=0) then begin
    exit;
  end;

  {/ Get the 32-bit function address.  Try the following three
  // variations of the name (example: NAME):
  //    NAME
  //    _NAME@nn     (stdcall naming convention: nn is bytes of args)
  //    NAMEA        (Win32 ANSI function naming convention) }
  lpfunc:=GetProcAddress32W(hinst, lpstrName);
  if (lpfunc=nil) and (lstrlen(lpstrName) < 122) then begin
    { Change to stdcall naming convention. }
    rg.lpstrName:=lpstrName;
    rg.nparams:=nParams * 4;
    wvsprintf(szBuffer, '_%s@%d', rg);
    lpfunc:=GetProcAddress32W(hinst, szBuffer);
  end;
  if (lpfunc=nil) and (lstrlen(lpstrName) < 126) then begin
    { Add suffix "A" for ansi }
    strcopy(szBuffer, lpstrName);
    strcat(szBuffer, 'A');
    lpfunc:=GetProcAddress32W(hinst, szBuffer);
  end;
  if (lpfunc=nil) then begin
    FreeLibrary32W(hinst);
    exit;
  end;

  {/ Fill in PROC32ENTRY struct and return index. }
  rgProc32Entry^[cRegistered].hinst:=hinst;
  rgProc32Entry^[cRegistered].lpfunc:=lpfunc;
  rgProc32Entry^[cRegistered].dwAddrXlat:=dwAddrXlat;
  rgProc32Entry^[cRegistered].dwHwndXlat:=dwHwndXlat;
  rgProc32Entry^[cRegistered].nParams:=nParams;
  Declare32:=cRegistered;
  inc(cRegistered);
  Call32NTError:=olderror;  {If there was no error, set Call32NTErrorOccurred
to false}
end;

function GetVDMPointer32W(name:pchar;Length:word):longint;
var lpGetVDMPointer32W:function(name:pchar;UINT:word):longint;
begin
```

```
@lpGetVDMPointer32W:=getProcAddress(GetModuleHandle('kernel'),'GetVDMPointer3
2W');
  if @lpGetVDMPointer32W<>nil then
    GetVDMPointer32W:=lpGetVDMPointer32W(name,Length)
  else
    GetVDMPointer32W:=0;
end;

{//------------------------------------------------------
// WEP
//   Called when DLL is unloaded.  We free all the
//   32-bit DLLs we were using and clear the
//   PROC32ENTRY list.
//------------------------------------------------------}
var exitsave:tfarproc;

procedure cleanuplibs; far;
begin
  Exitproc:=Exitsave;
  dec(cRegistered);
  while (cRegistered >= 0) do begin
    FreeLibrary32W(rgProc32Entry^[cRegistered].hinst);
    dec(cregistered);
  end;
  if (rgProc32Entry<>nil) then begin
    globalunlock(rgProc32handle);
    GlobalFree(rgProc32handle);
  end;
  rgProc32Entry:=NIL;
  rgProc32handle:=0;
  cRegistered:=0;
  cAlloc:=0;
end;

begin
  @Callproc32W:=nil;
  @FreeLibrary32W:=nil;
  @GetProcAddress32W:=nil;
  @LoadLibraryEx32W:=nil;
  @lpvGetLastError:=nil;
  lpvGetCapture:=nil;
  exitsave:=exitproc;
  exitproc:=@cleanuplibs;
end.
```

Q: How do I pass arrays into a procedure?

A: Well,

```
procedure ThisAintIt(green: array[1..2] of char)
```

isn't it. Here is an example of how to do it:

```
type
  PArr = ^TArr;
  Tarr = array[1..10] of integer;

procedure foo(PassingItIn: PArr);
var i: integer;
begin
  for i := 1 to 10 do PassingItIn^[i] := i;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  i: integer;
  TheArr: PArr;
begin
  TheArr := AllocMem(sizeof(TArr));
  foo(TheArr);
  for i := 1 to 10 do Listbox1.items.add(IntToStr(TheArr^[i]));
  FreeMem(TheArr, sizeof(TArr));
end;
```

TField

How do I setup the column list in code for a dynamically created TTable?

Q: What are the values for the virtual keys?

A:

```
  vk_LButton   = $01;
  vk_RButton   = $02;
  vk_Cancel    = $03;
  vk_MButton   = $04;   { NOT contiguous with L & RBUTTON }
  vk_Back      = $08;
  vk_Tab       = $09;
  vk_Clear     = $0C;
  vk_Return    = $0D;
  vk_Shift     = $10;
  vk_Control   = $11;
  vk_Menu      = $12;
  vk_Pause     = $13;
  vk_Capital   = $14;
  vk_Escape    = $1B;
  vk_Space     = $20;
  vk_Prior     = $21;
  vk_Next      = $22;
  vk_End       = $23;
  vk_Home      = $24;
  vk_Left      = $25;
  vk_Up        = $26;
  vk_Right     = $27;
  vk_Down      = $28;
  vk_Select    = $29;
  vk_Print     = $2A;
  vk_Execute   = $2B;
  vk_SnapShot  = $2C;
{ vk_Copy      = $2C not used by keyboards }
  vk_Insert    = $2D;
  vk_Delete    = $2E;
  vk_Help      = $2F;
{ vk_A thru vk_Z are the same as their ASCII equivalents: 'A' thru 'Z' }
{ vk_0 thru vk_9 are the same as their ASCII equivalents: '0' thru '9' }
  vk_NumPad0   = $60;
  vk_NumPad1   = $61;
  vk_NumPad2   = $62;
  vk_NumPad3   = $63;
  vk_NumPad4   = $64;
  vk_NumPad5   = $65;
  vk_NumPad6   = $66;
  vk_NumPad7   = $67;
  vk_NumPad8   = $68;
  vk_NumPad9   = $69;
  vk_Multiply  = $6A;
  vk_Add       = $6B;
  vk_Separator = $6C;
  vk_Subtract  = $6D;
  vk_Decimal   = $6E;
  vk_Divide    = $6F;
  vk_F1        = $70;
  vk_F2        = $71;
  vk_F3        = $72;
```

```
vk_F4         = $73;
vk_F5         = $74;
vk_F6         = $75;
vk_F7         = $76;
vk_F8         = $77;
vk_F9         = $78;
vk_F10        = $79;
vk_F11        = $7A;
vk_F12        = $7B;
vk_F13        = $7C;
vk_F14        = $7D;
vk_F15        = $7E;
vk_F16        = $7F;
vk_F17        = $80;
vk_F18        = $81;
vk_F19        = $82;
vk_F20        = $83;
vk_F21        = $84;
vk_F22        = $85;
vk_F23        = $86;
vk_F24        = $87;
vk_NumLock    = $90;
vk_Scroll     = $91;
```

Printing

How do I print a form?
How do I print a bitmap to a specific size?
How do I print to different printer resolutions?
How do I print from WinWord with DDE?
Demo for formatting the output. (columns, justification, headers, footers, grayscale, page numbering)

Q: How do I print to different printer resolutions?

A: Despite what Delphi's help says you have to change the printer object's Font.PixelsPerInch. And for some reason, you have to set it AFTER calling an API routine that gets the printer's hDC. The following function does it. After calling this function, you can use Font.Size to set a font size independently of the printer's resolution.

{--
Sets the logical dots per inch for the printer and sets the printer axes to point RIGHT and DOWN. Thus (0,0) is at the top left corner of the page. Returns the page size in logical coordinates.

Note: Must be called AFTER Printer.BeginDoc.
--}

```
function SetPrinterScale(dpi : integer) : TPoint;
var
  DeviceDpiX, DeviceDpiY : integer;
begin
  with Printer do begin
    SetMapMode(Handle, MM_ISOTROPIC);
    SetWindowExt(Handle, dpi, dpi);
    DeviceDpiX := GetDeviceCaps(Handle, LOGPIXELSX);
    DeviceDpiY := GetDeviceCaps(Handle, LOGPIXELSY);
    SetViewPortExt(Handle, DeviceDpiX, DeviceDpiY);
    Result := Point(PageWidth, PageHeight);
    with Canvas do begin
      DPtoLP(Handle, Result, 1); { This API call is required... }
      Font.PixelsPerInch := DPI; { ...to make this work.  (Who knows why?)
}
    end;
  end;
 end;
```

Q: How do I right justify a column of numeric data in a TStringGrid?

A: You must respond to the OnDrawCell event and draw the particulaur column of cells yourself. YOu can leave DefaultDrawing set to True and just *overwrite* the default drawing of your particular column - this is easier than totally taking over drawing. To right-align a cell's text:

```
VAR vCol, vRow : LongInt;
begin
  vCol := Col; vRow := Row;
  WITH Sender AS TStringGrid, Canvas DO
    IF vCol = 2 THEN BEGIN
      SetTextAlign(Handle, TA_RIGHT);
      FillRect(Rect);
      TextRect(Rect, Rect.RIGHT-2, Rect.Top+2,
      Cells[vCol, vRow]);
    END;
end;
```

Q: How can I trap for my own hotkeys?

A: First: set the form's `KeyPreview := true;`

Then, you do something like this:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (ssCtrl in shift) and (chr(key) in ['A', 'a']) then
    ShowMessage('Ctrl-A');
end;
```

Problem: This is the problem where the application runs fine as long as it is running under Delphi or Turbo Debugger but if is runs from Program Manager, it either hangs the system or causes a stack fault.

Solution: I discovered that the real problem is that I had an old copy of CTL3DV2.DLL in the program directory (as well as in the \WINDOWS\SYSTEM directory). As soon as I deleted this DLL, everything worked fine.

Q: How do I print from WinWord with DDE?

A: This answer involves opening an existing Word file that has a bookmark already saved. This code will select and replace the text at the bookmark with our own text, and then it will print.

Note: The ExecuteMacro() commands are separated into different buttons because of a timing issue. If you want everything to work from the same procedure, change the TRUE to FALSE.

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DdeMan, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    DCC: TDdeClientConv;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
  P: PChar;
begin
  WinExec('e:\winapps\winword\winword.exe', sw_ShowNormal);
  with DCC do begin
    SetLink('winword', '');
    if not OpenLink then
      ShowMessage('Link not established')
    else
      ExecuteMacro('[FileOpen("c:\temp\foobar.doc")]', True);
  end;
```

```
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  DCC.ExecuteMacro('[EditGoTo("TheBookmarkName")]', True);
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  DCC.ExecuteMacro('[Insert("This is the new text that is inserted.")]',
True);
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
  DCC.ExecuteMacro('[FilePrint]', True);
end;

end.

{*************************************************}

object Form1: TForm1
  Left = 202
  Top = 102
  Width = 403
  Height = 89
  Caption = 'Form1'
  Font.Color = clWindowText
  Font.Height = -13
  Font.Name = 'System'
  Font.Style = []
  PixelsPerInch = 96
  TextHeight = 16
  object Button1: TButton
    Left = 8
    Top = 16
    Width = 89
    Height = 33
    Caption = 'Open'
    TabOrder = 0
    OnClick = Button1Click
  end
  object Button2: TButton
    Left = 104
    Top = 16
    Width = 89
    Height = 33
    Caption = 'Find Bkmk'
    TabOrder = 1
    OnClick = Button2Click
  end
  object Button3: TButton
    Left = 200
    Top = 16
    Width = 89
```

```
    Height = 33
    Caption = 'Replace'
    TabOrder = 2
    OnClick = Button3Click
  end
  object Button4: TButton
    Left = 296
    Top = 16
    Width = 89
    Height = 33
    Caption = 'Print'
    TabOrder = 3
    OnClick = Button4Click
  end
  object DCC: TDdeClientConv
    ConnectMode = ddeManual
  end
end
```

Q: How do I flush the data buffer to the table? (i.e. hard write to disk)

A: This will allow you to commit changes to disk without incurring the performance hit of LocalShare.

```
Table1.UpdateCursorPos;
Check(dbiSaveChanges(Table1.Handle));
Table1.CursorPosChanged;
```

Q: How do I paint with a cross-hatched brush?

A:

```
var
  TheMask: TBitmap;
  TheRect: TRect;
begin
  TheMask := TBitmap.Create;
  TheRect := rect(0, 0, image1.width, image1.height);
  try
    with TheMask do begin
      Monochrome := False;
      Width := image1.width;
      Height := image1.height;
      canvas.brush.color := clBlack;
      canvas.brush.style := bsDiagCross;
      canvas.Ellipse(0, 0, image1.width, image1.height);
      canvas.CopyMode := cmSrcAnd;
      canvas.copyrect(TheRect, image1.canvas, TheRect);
   end;
   image1.canvas.CopyMode := cmSrcCopy;
   image1.Canvas.CopyRect(TheRect, TheMask.Canvas, TheRect);
  finally
    TheMask.free;
  end;
end;
```

COMPUTERS & ELECTRONICS (as depicted in movies, naturally)

Word processors never display a cursor.

You never have to use the space-bar when typing long sentences.

All monitors display inch-high letters.

High-tech computers, such as those used by NASA, the CIA, or some such governmental institution, will have easy to understand graphical interfaces. Those that don't, have incredibly powerful text-bases command shells that can correctly understand and execute commands typed in plain English. (Corollary: you can gain access to any information you want by simply typing "ACCESS ALL OF THE SECRET FILES" on any keyboard.)

Likewise, you can infect a computer with a destructive virus by simply typing "UPLOAD VIRUS" (see "Fortress").

All computers are connected. You can access the information on the villain's desktop computer, even if it's turned off.

Powerful computers beep whenever you press a key or whenever the screen changes. Some computers also slow down the output on the screen so that it doesn't go faster than you can read. The *really* advanced ones also emulate the sound of a dot-matrix printer.

All computer panels have thousands of volts and flash pots just underneath the surface. Malfunctions are indicated by a bright flash, a puff of smoke, a shower of sparks, and an explosion that forces you backwards.

People typing away on a computer will turn it off without saving the data.

A hacker can get into the most sensitive computer in the world before intermission and guess the secret password in two tries.

Any PERMISSION DENIED has an OVERRIDE function (see "Demolition Man" and countless others).

Complex calculations and loading of huge amounts of data will be accomplished in under three seconds. Movie modems usually appear to transmit data at the speed of two gigabytes per second.

When the power plant/missile site/whatever overheats, all the control panels will explode, as will the entire building.

If a disk has got encrypted files, you are automatically asked for a password when you

try to access it.

No matter what kind of computer disk it is, it'll be readable by any system you put it into. All application software is usable by all computer platforms The more high-tech the equipment, the more buttons it has (Aliens). However, everyone must have been highly trained, because the buttons aren't labelled.

Most computers, no matter how small, have reality-defying three- dimensional,active animation, photo-realistic graphics capability.

Laptops, for some strange reason, always seem to have amazing real-time video phone capabilities and the performance of a CRAY Supercomputer.

Whenever a character looks at a VDU, the image is so bright that it projects itself onto his/her face (see "Alien", "2001").

Q: How do I use arrays in a Delphi DLL with VB? (from a compuserve post)

I'm trying to get a DLL written in Delphi (TESTDLL.DLL) to reference an array created in Visual Basic. When I run it, I get a GPF in TESTDLL.DLL.

Code in VB is:

```
' Start of VB Code ************************************
' Form level declarations
Declare Function ReadArray Lib "TestDLL.dll" (TestArr As Any) As Integer
Dim TestArray(3) As Integer  ' Declares an array[0..3] of integer in VB

Sub Form_Load

  TestArray(0) = 2
  TestArray(1) = 4
  TestArray(2) = 6
  TestArray(3) = 8

End Sub

Sub cmdDelphi_Click ()
Dim iResult As Integer

  iResult = ReadArray(TestArray(0))  ' Documentation says this is the way to
get VB to
                                     ' pass a pointer to an array
  MsgBox "Back once again with result = " & iResult

End Sub
' End of VB Code  ******************************************
```

Code in Delphi is:

```
{Start of Delphi Code ************************************}
interface
.
.

function ReadArray(var TestArray: array of integer): integer; export;
.
.

implementation

function ReadArray(var TestArray: array of integer): integer;
var
  iCount : integer;
begin

  for iCount := 0 to 3 do
  begin
    ShowMessage( 'Element ' + InttoStr(iCount) + ' = ' +
IntToStr(TestArray[iCount]));
  end;
  Result := 4;
```

```
end;
{End of Delphi Code ************************************}
```

The DLL manages to read VB integers and VB strings OK. But it won't cope with arrays. Any offers on what I'm doing wrong?

A: You declare the parameter on the Delphi side as an open array, but it isn't one! An open array parameter in Delphi consists of a pointer to the array data and an additional word that gives the actual size of the passed array, 6 bytes in total. What VB provides is only the pointer to the data, so you have a mismatch in parameter list sizes, which is a sure recipe for GPF.

Try this one, modified from your code:

```
' Start of VB Code ************************************
' Form level declarations
Declare Function ReadArray Lib "TestDLL.dll" (TestArr As Any, ByVal Size as
Integer) As Integer

Dim TestArray(3) As Integer  ' Declares an array[0..3] of integer in VB

Sub Form_Load

  TestArray(0) = 2
  TestArray(1) = 4
  TestArray(2) = 6
  TestArray(3) = 8

End Sub

Sub cmdDelphi_Click ()
Dim iResult As Integer

  iResult = ReadArray(TestArray(0), 4)  ' Documentation says this is the way
to get VB to
                                      ' pass a pointer to an array
  MsgBox "Back once again with result = " & iResult

End Sub
' End of VB Code  ****************************************

Code in Delphi is:
{Start of Delphi Code ************************************}
interface
.
.
Type
  TIntArray = Array [1..High(Word) div Sizeof(Integer)] of Integer;
function ReadArray(var TestArray: TIntArray; size: Integer): integer; export;
.
.

implementation
```

```
function ReadArray;
var
  iCount : integer;
begin

  for iCount := 1 to size do
  begin
    ShowMessage( 'Element ' + InttoStr(iCount) + ' = ' +
IntToStr(TestArray[iCount]));
  end;
  Result := 4;
end;
{End of Delphi Code ************************************}
```

Q: How do I find one string inside another *with wildcards*?

A:

```
function Match(Source, Pattern: PChar): Boolean;
{ Returns true if the string Source matches the Pattern, which may contain
wildcards * and ?. }

function RMatch(s: PChar; i: Integer; p: PChar; j: Integer): Boolean;
Var
  matched        : Boolean;
  k              : Integer;
begin
  if p[0]=#0 then
    RMatch :=  TRUE
  else while TRUE do
    if (s[i]=#0) and (p[j]=#0) then begin
      RMatch :=  TRUE;
      exit
    end
    else
      if p[j] = #0 then begin
        RMatch :=  FALSE;
        exit
      end
      else if (p[j] = '*') then begin
        k := i;
        if (p[j + 1] = #0) then begin
          RMatch :=  TRUE;
          exit
        end
        else while TRUE do begin
          matched := RMatch(s, k, p, j + 1);
          if matched OR (s[k] = #0) then begin
            RMatch :=  matched;
            exit;
          end; {if}
          inc(k);
        end; {while}
      end {if}
      else if ((p[j] = '?') and (s[i] <> #0)) OR (UpCase(p[j]) =
UpCase(s[i])) then begin
        inc(i);
        inc(j);
      end
      else begin
        RMatch :=  FALSE;
        exit
      end;
end;

begin {Match}
  Match :=  RMatch(Source, 0, Pattern, 0);
end;
```

Q: How can I get information about window's tasks?

A:

```
unit Unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Grids;

type
  TForm1 = class(TForm)
    StringGrid1: TStringGrid;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure StringGrid1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
uses ToolHelp;

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
Var
  te: TTaskEntry;
  MoreTasks: Boolean;
  count: Integer;
  buffer: Array [0..80] of Char;
begin
  With StringGrid1 Do Begin
    FixedCols := 0;
    Cells[0,0] := 'hTask'; {Headings for the information.}
    Cells[1,0] := 'hInst';
    Cells[2,0] := 'hModule';
    Cells[3,0] := 'ModuleName';
    Cells[4,0] := 'Filename';
    te.dwSize := Sizeof(te);
    count := 1;
    MoreTasks := TaskFirst( @te );
    While MoreTasks Do Begin
      If RowCount <= count Then RowCount := count + 1;
      Cells[0, count] := '$' + IntToHex( te.hTask, 4 );
      Cells[1, count] := '$' + IntToHex( te.hInst, 4 );
      Cells[2, count] := '$' + IntToHex( te.hModule, 4 );
      Cells[3, count] := StrPas( te.szModule );
      GetModuleFilename( te.hModule, buffer, 81 );
```

```
      buffer[80] := #0;
      Cells[4, count] := StrPas( buffer );
      MoreTasks := TaskNext( @te );
      inc( count );
    end;
  end;
end;

procedure TForm1.StringGrid1Click(Sender: TObject);
begin
  with StringGrid1 do {To see the strings that don't fit the cell.}
    caption := cells[col, row];
end;

end.
```

ChartFX

Here is a bit of code that does something with ChartFX. There is not much out there in the way of examples, so this may help someone.

```
unit Unit1;
 interface

 uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, VBXCtrl, Chart2fx, ChartFx;

 type
  TForm1 = class(TForm)
    ChartFX1: TChartFX;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

 var
  Form1: TForm1;

 implementation

 {$R *.DFM}

const NumberOfSalesReps = 24;

procedure TForm1.FormCreate(Sender: TObject);
var
  i : integer;
begin
  randomize;
  with ChartFX1 do begin
    OpenData[COD_VALUES] := MAKELONG(1, NumberOfSalesReps);
    ThisSerie := 0; {this sets the Y-Values}
    for i := 0 to NumberOfSalesReps - 1 do Value[i] := random(100);
    CloseData[COD_VALUES] := 0;

    {this sets the X-Values}
    OpenData[COD_XVALUES] := MAKELONG(1, NumberOfSalesReps);
    for i := 0 to NumberOfSalesReps - 1 do XValue[i] := random(100);
    CloseData[COD_XVALUES] := 0;
  end;
 end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  FormCreate(Sender); {Just to watch the values change.}
end;
```

```
end.
```

```
write(f, #12); {#12 = FormFeed}
```

General VB related questions:

How Do I pass a struct (**by reference**) from VB to a DLL created in Delphi?
How do I close a file that was opened in a DLL (Delphi made) and called from VB?
How do I emulate a VB control array?
How do I use arrays in a Delphi DLL with VB?
How do I create a form with the equivelent of the visual basic 'FIXED DOUBLE' border with NO caption bar?

Q: How do I create a form with the equivelent of the visual basic 'FIXED DOUBLE' border with NO caption bar?

A: Just override CreateParams thus:

```
  private
    { Private declarations }
    procedure CreateParams(VAR Params: TCreateParams); override;
  ...
procedure TForm1.CreateParams(VAR Params: TCreateParams);
begin
  Inherited Createparams(Params);
  WITH Params DO
    Style := (Style OR WS_POPUP OR WS_BORDER) AND NOT WS_DLGFRAME;
end;
```

Q: How do I place the mouse anywhere on the form that I want?

A: Note: Using ClientToScreen() makes it so that the XY coordinates are relative to the window rather than the whole screen.

```
procedure PlaceMouse(x, y: word);
var
  tp: TPoint;
begin
  tp := ClientToScreen(point(x, y));
  SetCursorPos(tp.x, tp.y);
end;
```

Q: How do I size a form to fit a bitmap?

A: Note: The TImage is aligned to the client area.

```
unit Bumps;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, ExtCtrls, Menus;

type
  TForm1 = class(TForm)
    Image1: TImage;
    OpenDialog1: TOpenDialog;
    procedure Image1DblClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Image1DblClick(Sender: TObject);
begin
  OpenDialog1.FileName := '*.bmp';
  OpenDialog1.InitialDir := 'C:\WINDOWS';
  if OpenDialog1.execute then begin
    image1.picture.LoadFromFile(OpenDialog1.FileName);
    ClientWidth := image1.Picture.width;
    ClientHeight := image1.Picture.Height;
    {This next line is not needed if the TImage is aligned to the client
area.}
    SetBounds((screen.width - width) div 2, (screen.height - height) div 2,
width, height);
  end;
end;

end.
```

Q: How do I disable the screensaver?

A:

```
unit Nosaver;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
    procedure AppMessage(var msg: TMsg; var handled: boolean);
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.AppMessage(var msg: TMsg; var handled: boolean);
begin
  if (msg.message = wm_SysCommand) and (msg.wParam = sc_ScreenSave) then
    handled := true;  {TRUE = Disable the screensaver.}
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
  application.OnMessage := AppMessage;
end;

end.
```

Q: How do I detect the pentium bug?

A:

```
const
  a: single = 4195835.0;
  b: single = 3145727.0;
var
  c: double;
begin
{$U-}
  c := a / b;
{$U+}
  if a - c * b > 1.0 then ShowMessage('BUG!')
  else ShowMessage('OK');
end;
```

Q: How do I broadcast changes to WIN.INI?

A: The API function SendMessage() takes several parameters. First is the window handle; HWnd_Broadcast is correct for that. Next comes the message, WM_WININICHANGE. The last two parameters are the wParam and lParam (word parameter and long parameter) for the message. For this particular message, the wParam must be 0, and the lParam is the address of a string containing the name of the section that the changes were in. If lParam is NIL (zero), it means the windows receiving this message shoudl check ALL sections for changes, but that's slow; don't send 0 unless you've made changes in many sections.

So it might go like this:

```
var
  s: array[0..40] of char;
begin
  StrCopy(s, 'Desktop');
  SendMessage(HWnd_Broadcast, wm_WinIniChange, 0, LongInt(@s));
```

Q: How do I know which line number I am currently on in a TMemo?

A: The trick is to use the em_LineFromChar message. Try this:

```
{Note: First line is zero.}
ShowMessage('Line Number: ' + IntToStr(Memo1.Perform(em_LineFromChar, $FFFF,
0)));
```

to get row AND col you might do this:

```
Row := SendMessage(Memo1.Handle, EM_LINEFROMCHAR, Memo1.SelStart, 0);
Col := Memo1.SelStart - SendMessage(Memo1.Handle, EM_LINEINDEX, Row, 0);
```

Q: How do I manage disk volume labels in Delphi?

A: This is the source code for a unit that is useful for getting, setting, and deleting volume labels from a floppy or hard disk. The code for getting a volume label uses the Delphi FindFirst function, and the code for setting and deleting volume labels involves calling DOS interrupt 21h, functions 16h and 13h respectively. Since function 16h isn't supported by Windows, it must be called through DPMI interrupt 31h, function 300h.

```
unit VolLabel;

interface

uses Classes, SysUtils, WinProcs;

type
  EInterruptError = class(Exception);
  EDPMIError = class(EInterruptError);
  Str11 = String[11];

procedure SetVolumeLabel(NewLabel: Str11; Drive: Char);
function GetVolumeLabel(Drive: Char): Str11;
procedure DeleteVolumeLabel(Drv: Char);

implementation

type
  PRealModeRegs = ^TRealModeRegs;
  TRealModeRegs = record
    case Integer of
      0: (
        EDI, ESI, EBP, EXX, EBX, EDX, ECX, EAX: Longint;
         Flags, ES, DS, FS, GS, IP, CS, SP, SS: Word);
      1: (
        DI, DIH, SI, SIH, BP, BPH, XX, XXH: Word;
        case Integer of
          0: (
            BX, BXH, DX, DXH, CX, CXH, AX, AXH: Word);
          1: (
            BL, BH, BLH, BHH, DL, DH, DLH, DHH,
            CL, CH, CLH, CHH, AL, AH, ALH, AHH: Byte));
  end;

  PExtendedFCB = ^TExtendedFCB;
  TExtendedFCB = Record
    ExtendedFCBflag : Byte;
    Reserved1       : array[1..5] of Byte;
    Attr            : Byte;
    DriveID         : Byte;
    FileName        : array[1..8] of Char;
    FileExt         : array[1..3] of Char;
    CurrentBlockNum : Word;
    RecordSize      : Word;
    FileSize        : LongInt;
    PackedDate      : Word;
    PackedTime      : Word;
```

```
    Reserved2       : array[1..8] of Byte;
    CurrentRecNum   : Byte;
    RandomRecNum    : LongInt;
  end;

procedure RealModeInt(Int: Byte; var Regs: TRealModeRegs);
{ procedure invokes int 31h function 0300h to simulate aa real mode }
{ interrupt  from protected mode. }
var
  ErrorFlag: Boolean;
begin
  asm
    mov ErrorFlag, 0       { assume success }
    mov ax, 0300h          { function 300h }
    mov bl, Int            { real mode interrupt to execute }
    mov bh, 0              { required }
    mov cx, 0              { stack words to copy, assume zero }
    les di, Regs           { es:di = Regs }
    int 31h                { DPMI int 31h }
    jnc @@End              { carry flag set on error }
  @@Error:
    mov ErrorFlag, 1       { return false on error }
  @@End:
  end;
  if ErrorFlag then
    raise EDPMIError.Create('Failed to execute DPMI interrupt');
end;

function DriveLetterToNumber(DriveLet: Char): Byte;
{ function converts a character drive letter into its numerical equiv. }
begin
  if DriveLet in ['a'..'z'] then
    DriveLet := Chr(Ord(DriveLet) -32);
  if not (DriveLet in ['A'..'Z']) then
    raise EConvertError.CreateFmt('Cannot convert %s to drive number',

                                  [DriveLet]);
  Result := Ord(DriveLet) - 64;
end;

procedure PadVolumeLabel(var Name: Str11);
{ procedure pads Volume Label string with spaces }
var
  i: integer;
begin
  for i := Length(Name) + 1 to 11 do
    Name := Name + ' ';
end;

function GetVolumeLabel(Drive: Char): Str11;
{ function returns volume label of a disk }
var
  SR: TSearchRec;
  DriveLetter: Char;
  SearchString: String[7];
  P: Byte;
begin
```

```
  SearchString := Drive + ':\*.*';
  { find vol label }
  if FindFirst(SearchString, faVolumeID, SR) = 0 then begin
    P := Pos('.', SR.Name);
    if P > 0 then begin                    { if it has a dot... }
      Result := '           ';             { pad spaces between name }
      Move(SR.Name[1], Result[1], P - 1);  { and extension }
      Move(SR.Name[P + 1], Result[9], 3);
    end
    else begin
      Result := SR.Name;                   { otherwise, pad to end }
      PadVolumeLabel(Result);
    ennd;
  end
  else
    Result := '';
end;

procedure DeleteVolumeLabel(Drv: Char);
{ procedure deletes volume label from given drive }
var
  CurName: Str11;
  FCB: TExtendedFCB;
  ErrorFlag: WordBool;
begin
  ErrorFlag := False;
  CurName := GetVolumeLabel(Drv);        { get current volume label }
  FillChar(FCB, SizeOf(FCB), 0);         { initialize FCB with zeros }
  with FCB do begin
    ExtendedFCBflag := $FF;              { always }
    Attr := faVolumeID;                  { Volume ID attribute }
    DriveID := DriveLetterToNumber(Drv); { Drive number }
    Move(CurName[1], FileName, 8);       { must enter volume label }
    Move(CurName[9], FileExt, 3);
  end;
  asm
    push ds                              { preserve ds }
    mov ax, ss                           { put seg of FCB (ss) in ds }
    mov ds, ax
    lea dx, FCB                          { put offset of FCB in dx }
    mov ax, 1300h                        { function 13h }
    Call DOS3Call                         { invoke int 21h }
    pop ds                               { restore ds }
    cmp al, 00h                          { check for success }
    je @@End
  @@Error:                               { set flag on error }
    mov ErrorFlag, 1
  @@End:
  end;
  if ErrorFlag then
    raise EInterruptError.Create('Failed to delete volume name');
end;

procedure SetVolumeLabel(NewLabel: Str11; Drive: Char);
{ procedure sets volume label of a disk.  Note that this procedure }
{ deletes the current label before setting the new one.  This is }
{ required for the set function to work. }
```

```
var
  Regs: TRealModeRegs;
  FCB: PExtendedFCB;
  Buf: Longint;
begin
  PadVolumeLabel(NewLabel);
  if GetVolumeLabel(Drive) <> '' then           { if has label... }
    DeleteVolumeLabel(Drive);                   { delete label }
  Buf := GlobalDOSAlloc(SizeOf(PExtendedFCB));  { allocate real buffer }
  FCB := Ptr(LoWord(Buf), 0);
  FillChar(FCB^, SizeOf(FCB), 0);                { init FCB with zeros }
  with FCB^ do begin
    ExtendedFCBflag := $FF;                     { required }
    Attr := faVolumeID;                         { Volume ID attribute }
    DriveID := DriveLetterToNumber(Drive);      { Drive number }
    Move(NewLabel[1], FileName, 8);             { set new label }
    Move(NewLabel[9], FileExt, 3);
  end;
  FillChar(Regs, SizeOf(Regs), 0);
  with Regs do begin                            { SEGMENT of FCB }
    ds := HiWord(Buf);                          { offset = zero }
    dx := 0;
    ax := $1600;                                { function 16h }
  end;
  RealModeInt($21, Regs);                       { create file }
  if (Regs.al <> 0) then                        { check for success }
    raise EInterruptError.Create('Failed to create volume label');
end;

end.
```

Q: How efficient is inc(i)?

Here is a message that I posted on compu-serve. It was sent in response to a post from one user to another where he suggested that calling the procedure `inc(i)` was not as efficient as `i := i + 1;`

Here is the post: "One small optimization you can do , especially if your not using pointers is Kill the call to INC , every function call needs to allocate an activation record , local variables,... all of which suck time so why not just x := x + 1."

Here was my answer:

This is my chance to plug how great Delphi is.

```
x := x + 1;
```

yields the following assembly code:

```
mov  ax,  [bp-02]
inc  ax
mov  [bp-02], ax
```

Now, `inc(i)` does it this way:

```
inc  word ptr [bp-02]
```

The point here is that Delphi is *really* great at doing things fast, FAST *FAST*! Not all function calls slow things down.

Q: How can I tell which row and column is currently selected?

A:

```
unit Rowcol;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, DBTables, Grids, DBGrids;

type
  TForm1 = class(TForm)
    DBGrid1: TDBGrid;
    DataSource1: TDataSource;
    Table1: TTable;
    procedure DBGrid1ColEnter(Sender: TObject);
    procedure DataSource1DataChange(Sender: TObject; Field: TField);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure ShowRowCol;
  end;
  THack = class(TDBGrid);

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.ShowRowCol;
begin
  caption := 'Row: ' + IntToStr(THack(DBGrid1).row) + '  ' +
    'Col: ' + IntToStr(THack(DBGrid1).col);
end;

procedure TForm1.DBGrid1ColEnter(Sender: TObject);
begin
  ShowRowCol;
end;

procedure TForm1.DataSource1DataChange(Sender: TObject; Field: TField);
begin
  if (Sender as TDataSource).State = dsBrowse then ShowRowCol;
end;

end.
```

Q: How do I do a ShowMessage on the OnExit() of a TEdit and still get a cursor on the next component?

A: Doing a ShowMessage() on the exit of a component confuses the program as to just where the focus is supposed to be. This is a workaround.

Warning: Be careful to avoid infinite loops. You don't want to go to another component that has its own OnExit() routine that calls another one, etc, etc.

```
unit unit1;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    procedure Edit1Exit(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Edit1Exit(Sender: TObject);
begin
  ShowMessage((sender as TEdit).name);
  PostMessage(Handle, WM_NextDLGCTL, 0, 0); {next control}
  PostMessage(Handle, WM_NextDLGCTL, 1, 0); {previous control}
end;

end.
```

(From a post on Compu-serve by Neil Rubenking.)

There was some talk a while back about the fact that the Comp data type is a second-class citizen - there are no routines to go from Comp to string and back. I'm working on a remedy, and I have a unit that's in pretty good shape. It includes CompToStr, CompToHex, StrToComp, and the helper functions CMod and CDiv, which implement MOD and DIV for Comp.

I found out something interesting in implementing the CMod and CDiv functions. It appears that the division operation on Comp variables *ROUNDS* the result rather than truncating it as one might expect.

I've found some ODD things happening at the very ends of the Comp range. E.g. the FIRST time I attempt to use CompToStr on a string with the value $7FFF FFFF FFFF FFFD (broken up for clarity), I get a floating point exception without a specific location in my program. But if I try, try again I get no exception. WEIRD! Anyway, take a look at this unit and see if you find any cool uses for it.

You WILL realize from looking at this that the format of a Comp is simply two double-words jammed together. In effect, the high Dword is a LongInt and the low DWord is an unsigned double-word. I really don't know why Delphi and Object Pascal treat Comp as floating-point??

```
 unit Compfunc;

 interface
 TYPE
   CompAsTwoLongs = RECORD
      LoL, HiL : LongInt;
   END;
 CONST Two32TL: CompAsTwoLongs = (LoL:0; HiL:1);
 VAR   Two32: Comp ABSOLUTE Two32TL;

{Some operations fail erratically for values at the extreme  ends of the
range of Comp}
CONST MaxCompTL: CompAsTwoLongs = (LoL:$FFFFFFF0; HiL:$7FFFFFFF);
VAR   MaxComp: Comp ABSOLUTE MaxCompTL;

FUNCTION CMod(Divisor, Dividend: Comp): Comp;
FUNCTION CDiv(Divisor: Comp; Dividend: LongInt): Comp;
FUNCTION CompToStr(C: Comp): String;
FUNCTION CompToHex(C: Comp; Len: Integer): String;
FUNCTION StrToComp(const S : String): Comp;

implementation
USES SysUtils;

FUNCTION CMod(Divisor, Dividend: Comp): Comp;
VAR Temp : Comp;
BEGIN
```

```
  {Note: / operator for Comps apparently ROUNDS
   result rather than truncating}
  Temp := Divisor / Dividend;
  Temp := Temp * Dividend;
  Result := Divisor - Temp;
  IF Result < 0 THEN Result := Result + Dividend;
END;

FUNCTION CDiv(Divisor: Comp; Dividend: LongInt): Comp;
BEGIN
  Result := Divisor / Dividend;
  IF Result * Dividend > Divisor THEN
    Result := Result - 1;
END;

FUNCTION CompToStr(C: Comp): String;
VAR Posn : Integer;
BEGIN
  IF C > MaxComp THEN
    Raise ERangeError.Create('Comp too large for conversion to string');
  IF C < 0 THEN Result := '-'+CompToStr(-C)
  ELSE
    BEGIN
      Result := '';
      Posn := 0;
        WHILE TRUE DO
          BEGIN
            Result := Char(Round($30 + CMod(C,10)))+Result;
            IF C < 10 THEN Break;
            C := CDiv(C,10);
            Inc(Posn);
            IF Posn MOD 3 = 0 THEN Result := ','+Result;
          END;
    END;
END;

FUNCTION CompToHex(C: Comp; Len: Integer): String;
BEGIN
  IF (CompAsTwoLongs(C).HiL = 0) AND (Len <= 8) THEN
    Result := IntToHex(CompAsTwoLongs(C).LoL, Len)
  ELSE
    Result := IntToHex(CompAsTwoLongs(C).HiL, Len-8) +
      IntToHex(CompAsTwoLongs(C).LoL, 8)
END;

FUNCTION StrToComp(const S : String): Comp;
VAR Posn : Integer;
BEGIN
  IF S[1] = '-' THEN
    Result := -StrToComp(Copy(S,2,Length(S)-1))
  ELSE
    IF S[1] = '$' THEN {Hex string}
      try
        IF Length(S) > 9 THEN
          BEGIN
            {If string is invalid, exception raised by StrToInt}
            Result := StrToInt('$'+Copy(S,Length(S)-7, 8));
```

```
            IF Result < 0 THEN Result := Result + Two32;
            {If string is invalid, exception raised by StrToInt}
            CompAsTwoLongs(Result).HiL :=
              StrToInt(Copy(S,1,Length(S)-8))
          END
        ELSE
          BEGIN
            {If string is invalid, exception raised by StrToInt}
            Result := StrToInt(S);
            IF Result < 0 THEN Result := Result + Two32;
          END;
      except
        ON EConvertError DO Raise
          EConvertError.Create(S+' is not a valid Comp');
      end
    ELSE {Decimal string}
      BEGIN
        Posn := 1;
        Result := 0;
        WHILE Posn <= Length(S) DO
          CASE S[Posn] OF
            ',': Inc(Posn);
            '0'..'9': BEGIN
              Result := Result * 10 + Ord(S[Posn])-$30;
              Inc(Posn);
            END;
            ELSE Raise EConvertError.Create(S+
              ' is not a valid Comp');
          END;
      END;
END;

end.
```

Q: How do I write to a printer port?

A:

```
var
  Lpt2: TextFile;
begin
  AssignFile(Lpt2, 'LPT2');
  Rewrite(Lpt2);
  WriteLn(Lpt2, 'a few characters');
  CloseFile(Lpt2);
end;
```

Q: How do I fill a graphic field from a BMP file?

A: The TGraphicField descends from TBlobField which has a LoadFromFile() method.

```
unit Blobs;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, DB, DBTables, Menus, StdCtrls, DBCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    DBNavigator1: TDBNavigator;
    DBImage1: TDBImage;
    DataSource1: TDataSource;
    Table1: TTable;
    Button1: TButton;
    OpenDialog1: TOpenDialog;
    Table1TheBlob: TGraphicField; {from a custom PW table}
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
  if OpenDialog1.execute then begin
    table1.edit;
    Table1TheBlob.LoadFromFile(OpenDialog1.FileName);
    table1.post;
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  OpenDialog1.InitialDir := 'c:\windows';
end;

end.
```

TCalendar

How do I decide whether or not to hightlight a day in a calendar?

Q: How do I decide whether or not to hightlight a day in a calendar?

A: This is a custom component that overrides the DrawCell. It changes the brush and pen colors, then calls the inherited DrawCell method, then resets the colors.

```
unit New_cal;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Grids, Calendar;

type
  TEMS = class(TCalendar)
  private
    { Private declarations }
    FAllowHighlighting: boolean;
  protected
    { Protected declarations }
  public
    { Public declarations }
    procedure DrawCell(ACol, ARow: Longint; ARect: TRect; AState:
TGridDrawState); override;
    constructor Create(AOwner: TComponent); override;
  published
    { Published declarations }
    property AllowHighlighting: boolean read FAllowHighlighting write
FAllowHighlighting;
  end;

procedure Register;


implementation

procedure Register;
begin
  RegisterComponents('Samples', [TEMS]);
end;

constructor TEMS.Create(AOwner: TComponent);
begin
  AllowHighlighting := true;
  DefaultDrawing := false;
  inherited create(AOwner);
end;

procedure TEMS.DrawCell(ACol, ARow: Longint; ARect: TRect; AState:
TGridDrawState);
var
  OldCanvas: TCanvas;
begin
  inherited DrawCell(ACol, ARow, ARect, AState);
  if AllowHighlighting and (gdSelected in AState) then begin
    OldCanvas := TCanvas.create;
```

```
    OldCanvas.Brush.color := canvas.brush.color;
    OldCanvas.font.color := canvas.font.color;
    canvas.brush.color := clWhite;
    canvas.font.color := clBlack;

    inherited DrawCell(ACol, ARow, ARect, AState);

    canvas.font.color := OldCanvas.font.color;
    canvas.brush.color := canvas.brush.color;
    OldCanvas.free;
  end;
end;

end.
```

Q: How do I justify the text in a TEdit?

A: Here is a custom component that will do the job:

```
unit EditJust;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, StdCtrls, Menus, DsgnIntF;

type
  TMyCustomEdit = class (TCustomEdit)
  private
    FAlignment: TAlignment;
    FOldAlignment : TAlignment;
    FTextMargin : integer;
    function CalcTextMargin : integer;
    procedure CMEnter(var Message: TCMEnter); message CM_ENTER;
    procedure CMExit(var Message: TCMExit);   message CM_EXIT;
    procedure WMPaint(var Message: TWMPaint); message WM_PAINT;
    procedure SetAlignment(Value: TAlignment);
  protected
    property Alignment: TAlignment read FAlignment write SetAlignment default
taLeftJustify;
  public
    constructor Create(AOwner: TComponent); override;
  end;

  TEditJust = class (TMyCustomEdit)
  published
    property Alignment;
    property AutoSize;
    property BorderStyle;
    property Color;
    property Ctl3D;
    property DragCursor;
    property DragMode;
    property Enabled;
    property Font;
    property HideSelection;
    property ParentColor;
    property ParentCtl3D;
    property ParentFont;
    property ParentShowHint;
    property PopupMenu;
    property ReadOnly;
    property ShowHint;
    property TabOrder;
    property Visible;
    property OnChange;
    property OnClick;
    property OnDblClick;
    property OnDragDrop;
    property OnDragOver;
```

```
    property OnEndDrag;
    property OnEnter;
    property OnExit;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
    property OnMouseDown;
    property OnMouseMove;
    property OnMouseUp;
  end;

procedure Register;

implementation

constructor TMyCustomEdit.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  FTextMargin := CalcTextMargin;
end;

function TMyCustomEdit.CalcTextMargin : integer;
{
This was borrowed from TDBEdit.  It calculates a pixel
offset from the edge of the control to the text
(a margin) used in the paint routine.
}
var
  DC: HDC;
  SaveFont: HFont;
  I: Integer;
  SysMetrics, Metrics: TTextMetric;
begin
  DC := GetDC(0);
  GetTextMetrics(DC, SysMetrics);
  SaveFont := SelectObject(DC, Font.Handle);
  GetTextMetrics(DC, Metrics);
  SelectObject(DC, SaveFont);
  ReleaseDC(0, DC);
  I := SysMetrics.tmHeight;
  if I > Metrics.tmHeight then I := Metrics.tmHeight;
  Result := I div 4;
end;

procedure TMyCustomEdit.SetAlignment(Value: TAlignment);
begin
  if FAlignment <> Value then begin
    FAlignment := Value;
    Invalidate;
  end;
end;

procedure TMyCustomEdit.CMEnter(var Message: TCMEnter);
begin
  inherited;
  FOldAlignment := FAlignment;
  Alignment := taLeftJustify;
```

```
end;

procedure TMyCustomEdit.CMExit(var Message: TCMExit);
begin
  inherited;
  Alignment := FOldAlignment;
end;

procedure TMyCustomEdit.WMPaint(var Message: TWMPaint);
{borrowed from TDBEdit}
{paints the text in the appropriate position}
var
  Width, Indent, Left, I: Integer;
  R: TRect;
  DC: HDC;
  PS: TPaintStruct;
  S: string;
  Canvas: TControlCanvas;
begin
  {let the existing code handle left justify}
  if (FAlignment = taLeftJustify) then begin
    inherited;
    Exit;
  end;

  try
    Canvas := TControlCanvas.Create;
    Canvas.Control := Self;
    DC := Message.DC;
    if DC = 0 then DC := BeginPaint(Handle, PS);
    Canvas.Handle := DC;

    Canvas.Font := Font;
    with Canvas do begin
      R := ClientRect;
      if (BorderStyle = bsSingle) then begin
        Brush.Color := clWindowFrame;
        FrameRect(R);
        InflateRect(R, -1, -1);
      end;
      Brush.Color := Color;
      S := Text;
      Width := TextWidth(S);
      if BorderStyle = bsNone then Indent := 0
      else Indent := FTextMargin;
      if FAlignment = taRightJustify then Left := R.Right - Width - Indent
      else Left := (R.Left + R.Right - Width) div 2;
      TextRect(R, Left, Indent, S);
    end;
  finally
    Canvas.Handle := 0;
    if Message.DC = 0 then EndPaint(Handle, PS);
  end; {try}
end;

procedure Register;
begin
```

```
  RegisterComponents ('Samples', [TEditJust] );
end;

end.
```

TMediaPlayer

How do I size an AVI display to fit a panel?
How can I tell the default display size of an AVI file?

Q: How do I size an AVI display to fit a panel?

A: Note: the DisplayRect property is modified when the avi file is opened. Therefore, setting or getting the DisplayRect must be done *after* the file is opened.

```
begin
  with MediaPlayer1 do begin
    DeviceType := dtAutoSelect;
    visible := false;
    FileName := InputBox('AVI', 'Enter AVI file name', 'c:
\windows\borland.avi');
    display := panel1;
    open;
    DisplayRect := rect(0, 0, panel1.width, panel1.height); {This is it!}
    rewind;
    play;
  end;
end;
```

Q: How can I tell the default display size of an AVI file?

A: Note: the DisplayRect property is modified when the avi file is opened. Therefore, setting or getting the DisplayRect must be done *after* the file is opened.

```
var
  t: trect;
begin
  with MediaPlayer1 do begin
    DeviceType := dtAutoSelect;
    visible := false;
    FileName := InputBox('AVI', 'Enter AVI file name', 'c:
\windows\borland.avi');
    display := panel1;
    open;
    t := DisplayRect;
    caption := IntToStr(t.left) + ' : ' + IntToStr(t.top) + ' : ' +
IntToStr(t.right) + ' : ' + IntToStr(t.bottom); {This is it!}
    rewind;
    play;
  end;
end;
```

API

CreateFont

CreateFont

```
var
  dc: HDC;
  TheFont: HFont;
begin
  dc := GetDC(handle);
  TheFont := CreateFont(24, 16, 0, 0, 400, 0, 0, 0,
    Oem_CharSet, Out_Default_Precis, Clip_Default_Precis,
    Default_Quality, Default_Pitch OR FF_Script, 'script');
  SelectObject(dc, TheFont);
  TextOut(dc, 10, 10, 'Lloyd is the greatest!!!', 24);
  ReleaseDC(handle, dc);
  DeleteObject(TheFont);
end;
```