The Unofficial Newsletter of Delphi Users - Issue #9 - November 9th, 1995

Well another issue sails by! I apologize for not being able to stick to a rigid publication schedule, but my professional and personal demands on my time sometimes push the newsletter a bit lower on the priority list. As I mentioned in last issue, I will try to get the issues out as quickly as I can, however, I would request that if an issue seems to be overdue, please be patient. Believe it or not, it actually slows me down even more when the issue is a week late and my CompuServe mail starts skyrocketing! When the deadline has come and gone, I think I can actually hear people all across the country, drumming their fingers in eager anticipation...

Typically, each issue of the newsletter is posted to three locations. The first is the *Borland Delphi* forum on CompuServe (**GO DELPHI**) in the "Delphi IDE" file section. If you want the issue as soon as it comes out, then this is the place to look. I also put the issue in the *Informant Communications* forum (**GO ICGFORUM**) in the "Delphi Demo/Share" file section at the same time. Lastly, I take the original source material of the issue and package it up and send it off to a gentleman named *Aaron Richardson* who maintains the Delphi Source web site (**http://www.doit.com/delphi/home.html**). He takes these files and converts them to web pages on the site and also posts the Windows *.HLP files on the sites FTP server. If you have questions about UNDU in general, you can contact me at **76416,1373** on CompuServe. If you have questions about his Web version of UNDU, you can contact Aaron at **aaron@doit.com**.

One final note: The biggest influence on the publication schedule of UNDU is the availability of material contributed by ***you***, the reader. I generally do not have time to provide more than, say, a third of the content of each issue. Above that, I rely on article submissions from all of you out there. Please take advantage of this avenue to express your thoughts, ideas, tricks, and tips about Delphi. It is in my (highly biased) opinion, the best programming language out on the market.

Tips & Tricks

Using Sample Applications

In the previous issue, I offered suggestions on how to use the sample code provided in these issues of the newsletter. I guess I did not succeed in explaining it very well. Let me try once more...

Whenever you see sample code and/or form files in the newsletter, simply do the following to get them up and running in your Delphi environment:

1. *Create a new project*
2. *Save as Unit1 and Project1*
3. *Select all the form data from the newsletter and copy it to the clipboard*
4. *Back in Delphi, select File|Open File*
5. *Change the "List Files of Type" to "Form Files (*.DFM)"*
6. *Open the Unit1 form data. It will be shown in a textual format.*
7. *Paste the text from the clipboard, replacing all the existing text.*
8. *Remove the copyright info at the bottom of the file.*
9. *Save and close the form file.*
10. *Now, select all the unit source data from the newsletter and copy it also to the clipboard.*
11. *Back in Delphi, choose File|Open File.*
12. *Retrieve the Unit1.PAS file.*
13. *Paste the text from the clipboard, replacing all the existing text.*
14. *Remove the copyright info at the bottom of the file.*
15. *Manually change the name of the unit at the top of the form back to "Unit1".*
16. *Save and close the unit file.*
17. *Compile and Run!*

Return to Tips & Tricks

Return to Front Page

```
unit SlideBar;

interface

{$R SLIDEBAR.RES}

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

type
  TBarStyle    = (bsLowered,bsRaised);
  TOrientation = (orVertical,orHorizontal);
  TThumbStyle  = (tsBar1,tsBar2,tsBar3,tsBar4,tsCircle1,tsSquare1,
                  tsDiamond1,tsDiamond2,tsDiamond3,tsDiamond4);
  TSlideBar = class(TCustomControl)
  private
    FFocusColor              : TColor;
    FHandCursor              : Boolean;
    FLabels                  : TStringList;
    FMax,FMin,FPosition      : Integer;
    FOrientation             : TOrientation;
    FStyle                   : TBarStyle;
    FThickness               : Byte;
    FThumbStyle              : TThumbStyle;
    FTicks                   : Boolean;
    FOnChange                : TNotifyEvent;
    ThumbBmp,MaskBmp,BkgdBmp : TBitmap;
    DragVal,HalfTW,HalfTH    : Integer;
    ThumbRect                : TRect;
    TempDC                   : HDC;
    HandPointer              : HCursor;
    OriginalCursor           : HCursor;
    procedure SetLabels(A: TStringList);
    procedure SetMax(A: Integer);
    procedure SetMin(A: Integer);
    procedure SetOrientation(A: TOrientation);
    procedure SetPosition(A: Integer);
    procedure SetStyle(A: TBarStyle);
    procedure SetThickness(A: Byte);
    procedure SetThumbStyle(A: TThumbStyle);
    procedure SetTicks(A: Boolean);
    procedure CMEnter(var Message: TCMGotFocus); message CM_ENTER;
    procedure CMExit(var Message: TCMExit); message CM_EXIT;
    procedure WMGetDlgCode(var Message: TWMGetDlgCode); message WM_GETDLGCODE;
    procedure WMSize(var Message: TWMSize);             message WM_SIZE;
    procedure KeyDown(var Key: Word; Shift: TShiftState); override;
  protected
    Dragging                 : Boolean;
    procedure Paint; override;
    procedure MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
override;
    procedure MouseMove(Shift: TShiftState; X, Y: Integer); override;
    function  NewPosition(WhereX,WhereY: Integer): Integer;
    function  IsVert: Boolean;
    procedure RemoveThumbBar;
    procedure DrawThumbBar;
    procedure DrawTrench;
    procedure SaveBackground;
```

```
    procedure WhereIsBar;
    procedure SetTLColor;
    procedure SetBRColor;
  public
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    function CurrentLabel: String;
  published
    property Enabled;
    property FocusColor: TColor read FFocusColor
                         write FFocusColor default clBlack;
    property HandCursor: Boolean read FHandCursor
                         write FHandCursor default True;
    property Labels: TStringList read FLabels write SetLabels;
    property Max: Integer read FMax write SetMax default 10;
    property Min: Integer read FMin write SetMin default 1;
    property Orientation: TOrientation read FOrientation
                          write SetOrientation default orHorizontal;
    property ParentShowHint;
    property Position: Integer read FPosition write SetPosition default 1;
    property PopupMenu;
    property ShowHint;
    property Style: TBarStyle read FStyle write SetStyle default bsLowered;
    property TabStop default True;
    property TabOrder;
    property Thickness: Byte read FThickness write SetThickness default 1;
    property ThumbStyle: TThumbStyle read FThumbStyle
                         write SetThumbStyle default tsCircle1;
    property Ticks: Boolean read FTicks write SetTicks default True;
    property Visible;
    property OnChange: TNotifyEvent read FOnChange write FOnChange;
    property OnEnter;
    property OnExit;
    property OnKeyDown;
    property OnKeyPress;
    property OnKeyUp;
  end;

procedure Register;

implementation

function MinInt(A,B: Integer): Integer;
begin
  If A > B Then MinInt := B Else MinInt := A;
end;

function MaxInt(A,B: Integer): Integer;
begin
  If A > B Then MaxInt := A Else MaxInt := B;
end;

procedure Register;
begin
  RegisterComponents('Standard', [TSlideBar]);
end;

(******************
 TSlideBar Methods
******************)

constructor TSlideBar.Create(AOwner: TComponent);
```

```
begin
  inherited Create(AOwner);
  Height := 15;
  Width := 100;
  ThumbBmp := TBitmap.Create;
  MaskBmp := TBitmap.Create;
  BkgdBmp := TBitmap.Create;
  HandPointer := LoadCursor(HInstance, 'HandPointer');
  FFocusColor := clBlack;
  FHandCursor := True;
  FLabels := TStringList.Create;
  FMin := 1;
  FMax := 10;
  FOrientation := orHorizontal;
  FPosition := 1;
  FStyle := bsLowered;
  FThickness := 1;
  FTicks := True;
  Dragging := False;
  DragVal := 0;
  ThumbStyle := tsCircle1;
  TabStop := True;
end;

destructor TSlideBar.Destroy;
begin
  FLabels.Free;
  ThumbBmp.Free;
  MaskBmp.Free;
  BkgdBmp.Free;
  inherited Destroy;
end;

procedure TSlideBar.CMEnter(var Message: TCMGotFocus);
begin
  inherited;
  Refresh;
end;

procedure TSlideBar.CMExit(var Message: TCMExit);
begin
  inherited;
  Refresh;
end;

function TSlideBar.IsVert: Boolean;
begin
  IsVert := (Orientation = orVertical);
end;

procedure TSlideBar.KeyDown(var Key: Word; Shift: TShiftState);
var
  b : Integer;
begin
  b := MaxInt(1,(Max-Min) div 10);
  case Key of
    VK_PRIOR : if (Position-b) > Min then
                 Position := Position - b else Position := Min;
    VK_NEXT  : if (Position+b) < Max then
                 Position := Position + b else Position := Max;
    VK_END   : if IsVert then Position := Min else Position := Max;
    VK_HOME  : if IsVert then Position := Max else Position := Min;
    VK_LEFT  : if Position > Min then Position := Position - 1;
```

```
    VK_UP    : if Position < Max then Position := Position + 1;
    VK_RIGHT : if Position < Max then Position := Position + 1;
    VK_DOWN  : if Position > Min then Position := Position - 1;
  end;
end;

procedure TSlideBar.WMGetDlgCode(var Message: TWMGetDlgCode);
begin
  Message.Result := DLGC_WANTARROWS;
  OriginalCursor := GetClassWord(Handle, GCW_HCURSOR);
end;

procedure TSlideBar.WMSize(var Message: TWMSize);
begin
  if Height > Width then
    Orientation := orVertical else Orientation := orHorizontal;
end;

procedure TSlideBar.SetLabels(A: TStringList);
begin
  FLabels.Assign(A);
end;

procedure TSlideBar.SetMin(A: Integer);
begin
  FMin := A;
  Refresh;
end;

procedure TSlideBar.SetMax(A: Integer);
begin
  FMax := A;
  Refresh;
end;

procedure TSlideBar.SetOrientation(A: TOrientation);
begin
  FOrientation := A;
  Refresh;
end;

procedure TSlideBar.SetPosition(A: Integer);
begin
  if csDesigning in ComponentState then
    begin
      if (A >= Min) and (A <= Max) Then FPosition := A;
      Refresh;
    end
  else
    begin
      RemoveThumbBar;
      if (A >= Min) and (A <= Max) Then FPosition := A;
      WhereIsBar;
      SaveBackground;
      DrawThumbBar;
      if Assigned(FOnChange) then FOnChange(Self);
    end;
end;

procedure TSlideBar.SetStyle(A: TBarStyle);
begin
  FStyle := A;
  Refresh;
```

```
end;

procedure TSlideBar.SetThickness(A: Byte);
begin
  If (A > 0) and (A < 6) then
    begin FThickness := A; Refresh; end;
end;

procedure TSlideBar.SetThumbStyle(A: TThumbStyle);
begin
  If ThumbStyle <> A then
    begin
      FThumbStyle := A;
      case ThumbStyle of
        tsBar1     :   ThumbBmp.Handle := LoadBitmap(HInstance,'Bar1');
        tsBar2     :   ThumbBmp.Handle := LoadBitmap(HInstance,'Bar2');
        tsBar3     :   ThumbBmp.Handle := LoadBitmap(HInstance,'Bar3');
        tsBar4     :   ThumbBmp.Handle := LoadBitmap(HInstance,'Bar4');
        tsCircle1  :   ThumbBmp.Handle := LoadBitmap(HInstance,'Circle1');
        tsSquare1  :   ThumbBmp.Handle := LoadBitmap(HInstance,'Square1');
        tsDiamond1 :   ThumbBmp.Handle := LoadBitmap(HInstance,'Diamond1');
        tsDiamond2 :   ThumbBmp.Handle := LoadBitmap(HInstance,'Diamond2');
        tsDiamond3 :   ThumbBmp.Handle := LoadBitmap(HInstance,'Diamond3');
        tsDiamond4 :   ThumbBmp.Handle := LoadBitmap(HInstance,'Diamond4');
      end;
      case ThumbStyle of
        tsBar1     :   MaskBmp.Handle := LoadBitmap(HInstance,'Bar1Mask');
        tsBar2     :   MaskBmp.Handle := LoadBitmap(HInstance,'Bar2Mask');
        tsBar3     :   MaskBmp.Handle := LoadBitmap(HInstance,'Bar3Mask');
        tsBar4     :   MaskBmp.Handle := LoadBitmap(HInstance,'Bar4Mask');
        tsCircle1  :   MaskBmp.Handle := LoadBitmap(HInstance,'Circle1Mask');
        tsSquare1  :   MaskBmp.Handle := LoadBitmap(HInstance,'Square1Mask');
        tsDiamond1 :   MaskBmp.Handle := LoadBitmap(HInstance,'Diamond1Mask');
        tsDiamond2 :   MaskBmp.Handle := LoadBitmap(HInstance,'Diamond2Mask');
        tsDiamond3 :   MaskBmp.Handle := LoadBitmap(HInstance,'Diamond3Mask');
        tsDiamond4 :   MaskBmp.Handle := LoadBitmap(HInstance,'Diamond4Mask');
      end;
      HalfTH := ThumbBmp.Height div 2;
      HalfTW := ThumbBmp.Width div 2;
      Refresh;
    end;
end;

procedure TSlideBar.SetTicks(A: Boolean);
begin
  FTicks := A;
  Refresh;
end;

function TSlideBar.CurrentLabel: String;
begin
  if ((Position-Min+1) <= Labels.Count) and (Position >= Min) then
    CurrentLabel := Labels[Position-Min]
  else
    CurrentLabel := '<Un-Defined>';
end;

function TSlideBar.NewPosition(WhereX,WhereY: Integer): Integer;
var
  H1,W1 : Integer;
begin
  {Calculate the nearest position to where the mouse is located}
  H1 := Height-HalfTH;
```

```
  W1 := Width-HalfTW;
  if IsVert then
    Result := Round(((H1-WhereY)/H1)*(Max-Min)+Min)
  else
    Result := Round((WhereX/W1)*(Max-Min)+Min);
  Result := MinInt(MaxInt(Result,Min),Max);
end;

procedure TSlideBar.MouseUp(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
var
  A,B,C,D,E : Integer;
begin
  if Button <> mbLeft then exit;
  C := Position-1;
  D := Position;
  E := Position+1;
  {B is the center of the ThumbBar}
  if IsVert then B := ThumbRect.Top+HalfTH else B := ThumbRect.Left+HalfTW;
  if Dragging then
    A := NewPosition(X,Y)
  else
    if IsVert then
      if Y < B then A := E else if Y > B then A := C else A := D
    else
      if X < B then A := C else if X > B then A := E else A := D;
  A := MinInt(MaxInt(A,Min),Max);
  Dragging := False;
  Position := A;
end;

procedure TSlideBar.MouseDown(Button: TMouseButton; Shift: TShiftState; X, Y:
Integer);
begin
  SetFocus;
  Dragging := PtInRect(ThumbRect,Point(X,Y));
  If IsVert then DragVal := Y else DragVal := X;
end;

procedure TSlideBar.MouseMove(Shift: TShiftState; X, Y: Integer);
Var
  LastDragVal : Integer;
begin
  if HandCursor then
    SetClassWord(Handle, GCW_HCURSOR, HandPointer)
  else
    SetClassWord(Handle, GCW_HCURSOR, OriginalCursor);
  {Is the left mouse button down and dragging the thumb bar?}
  if (ssLeft in Shift) and Dragging then
    begin
      LastDragVal := DragVal;
      if IsVert then DragVal := Y else DragVal := X;
      {This test eliminates unneccesary repaints}
      if DragVal <> LastDragVal then Position := NewPosition(X,Y);
    end;
end;

procedure TSlideBar.RemoveThumbBar;
begin
  {Place the background bitmap where it was last}
  Canvas.Draw(ThumbRect.Left,ThumbRect.Top,BkgdBmp);
end;
```

```
procedure TSlideBar.DrawThumbBar;
var
  TmpBmp  : TBitMap;
  Rect1   : TRect;
begin
  try
    {Define a rectangle to mark the dimensions of the thumbbar}
    Rect1 := Rect(0,0,ThumbBmp.Width,ThumbBmp.Height);
    {Create a working bitmap}
    TmpBmp := TBitmap.Create;
    TmpBmp.Height := ThumbBmp.Height;
    TmpBmp.Width := ThumbBmp.Width;
    {Copy the background area onto the working bitmap}
    TmpBmp.Canvas.CopyMode := cmSrcCopy;
    TmpBmp.Canvas.CopyRect(Rect1,BkgdBmp.Canvas,Rect1);
    {Copy the mask onto the working bitmap with SRCAND}
    TmpBmp.Canvas.CopyMode := cmSrcAnd;
    TmpBmp.Canvas.CopyRect(Rect1,MaskBmp.Canvas,Rect1);
    {Copy the thumbbar onto the working bitmap with SRCPAINT}
    TmpBmp.Canvas.CopyMode := cmSrcPaint;
    TmpBmp.Canvas.CopyRect(Rect1,ThumbBmp.Canvas,Rect1);
    {Now draw the thumb bar}
    Canvas.CopyRect(ThumbRect,TmpBmp.Canvas,Rect1);
  finally
    TmpBmp.Free;
  end;
end;

procedure TSlideBar.WhereIsBar;
var
  Each          : Real;
  ThumbX,ThumbY : Integer;
begin
  {Calculate where to paint the thumb bar - store in ThumbRect}
  if IsVert then
    begin
      Each := (Height-ThumbBmp.Height)/(Max-Min);
      If Dragging then
        ThumbY := DragVal-HalfTH
      else
        ThumbY := Height-Round(Each*(Position-Min))-ThumbBmp.Height;
      ThumbY := MaxInt(0,MinInt(Height-ThumbBmp.Height,ThumbY));
      ThumbX := (Width-ThumbBmp.Width) div 2;
    end
  else
    begin
      Each := (Width-ThumbBmp.Width)/(Max-Min);
      if Dragging then
        ThumbX := DragVal-HalfTW
      else
        ThumbX := Round(Each*(Position-Min));
      ThumbX := MaxInt(0,MinInt(Width-ThumbBmp.Width,ThumbX));
      ThumbY := (Height-ThumbBmp.Height) div 2;
    end;
  ThumbRect := Rect(ThumbX,ThumbY,ThumbX+ThumbBmp.Width,ThumbY+ThumbBmp.Height);
end;

procedure TSlideBar.SetTLColor;
begin
  {Set the Top/Left color for the trench. Controls raised or lowered styles}
  With Canvas do
    if Style = bsLowered then Pen.Color := clGray else Pen.Color := clWhite;
end;
```

```
procedure TSlideBar.SetBRColor;
begin
  {Set the Bottom/Right color for the trench. Controls raised or lowered styles}
  With Canvas do
    if Style = bsRaised then Pen.Color := clGray else Pen.Color := clWhite;
end;

procedure TSlideBar.DrawTrench;
var
  X1,Y1,X2,Y2 : Integer;
  Each        : Real;
  Tmp,TickPos : Integer;
begin
  {This procedure simply draws the slot that the thumb bar will travel through}
  {including the tick-marks. The bar itself is not drawn.}
  with Canvas do begin
    {Calculate the corners of the trench dependant on orientation}
    if IsVert then
      begin
        X1 := (Width div 2) - (Thickness div 2) - 1;
        X2 := X1 + Thickness + 1;
        Y1 := HalfTH;
        Y2 := Height-ThumbBmp.Height+Y1;
      end
    else
      begin
        X1 := HalfTW;
        X2 := Width-ThumbBmp.Width+X1;
        Y1 := (Height div 2) - (Thickness div 2) - 1;
        Y2 := Y1 + Thickness + 1;
      end;
    Pen.Style := psSolid;
    {Set the color for the Top & Left edges}
    SetTLColor;
    MoveTo(X2,Y1);
    LineTo(X1,Y1);
    LineTo(X1,Y2);
    {Set the color for the Bottom & Right edges}
    SetBRColor;
    LineTo(X2,Y2);
    LineTo(X2,Y1-1);
    {Now do a filled black rectangle in the center if the control has focus}
    with brush do if Focused then Color := FocusColor else Color := clSilver;
    Pen.Style := psClear;
    {Draw the focus highlight}
    Rectangle(X1+1,Y1+1,X2+1,Y2+1);
    Pen.Style := psSolid;
    {Calculate spacing of tick marks}
    Each := 0;
    if Ticks then
      if (Max-Min) > 0 then
        if IsVert then
          Each := (Height-ThumbBmp.Height)/(Max-Min)
        else
          Each := (Width-ThumbBmp.Width)/(Max-Min);
    {Now draw the tick marks}
    if Ticks then
      for Tmp := Min to Max do
        if IsVert then
          begin
            TickPos := Y2-Trunc(Each*(Tmp-Min))-1;
            if Tmp = Max then TickPos := Y1;
```

```
            SetTLColor; MoveTo(X1,TickPos);   LineTo(X1-2,TickPos);
            SetBRColor; MoveTo(X1,TickPos+1); LineTo(X1-2,TickPos+1);
          end
        else
          begin
            TickPos := X1+Trunc(Each*(Tmp-Min));
            if Tmp = Max then TickPos := X2-1;
            SetTLColor; MoveTo(TickPos,Y1);   LineTo(TickPos,Y1-2);
            SetBRColor; MoveTo(TickPos+1,Y1); LineTo(TickPos+1,Y1-2);
          end;
  end;
end;

procedure TSlideBar.SaveBackground;
begin
  {This saves the background image so it can be restored later}
  BkgdBmp.Width := ThumbBmp.Width;
  BkgdBmp.Height := ThumbBmp.Height;
  BkgdBmp.Canvas.CopyRect(Rect(0,0,ThumbBmp.Width,ThumbBmp.Height),
                          Canvas,ThumbRect);
end;

procedure TSlideBar.Paint;
begin
  DrawTrench;
  WhereIsBar;
  SaveBackground;
  DrawThumbBar;
end;

end.
```

Return to Component Cookbook

Return to Front Page

```
/*************************************************************************

slidebar.rc

produced by Borland Resource Workshop

*************************************************************************/


BAR1 BITMAP
{
 '42 4D 9E 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 05 00 00 00 0A 00 00 00 01 00 04 00 00 00'
 '00 00 28 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 08 77 0D DD 0F 87'
 '0D DD 0F 87 08 88 0F 87 08 88 0F 87 00 00 0F 87'
 '00 00 0F 87 00 00 0F F8 00 00 00 00 00 00'
}


BAR1MASK BITMAP
{
 '42 4D 9E 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 05 00 00 00 0A 00 00 00 01 00 04 00 00 00'
 '00 00 28 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 F0 00 F0 00 00 00 0D DD 00 00'
 '0D DD 00 00 08 88 00 00 08 88 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 F0 00 F0 00'
}


BAR2 BITMAP
{
 '42 4D A6 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 06 00 00 00 0C 00 00 00 01 00 04 00 00 00'
 '00 00 30 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 08 77 70 00 0F 88'
 '70 00 0F 88 70 DD 0F 88 70 DD 0F 88 70 88 0F 88'
 '70 88 0F 88 70 00 0F 88 70 00 0F 88 70 00 0F FF'
 '80 00 00 00 00 00'
}


BAR2MASK BITMAP
{
 '42 4D A6 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 06 00 00 00 0C 00 00 00 01 00 04 00 00 00'
 '00 00 30 00 00 00 00 00 00 00 00 00 00 00 00 00'
```

```
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 F0 00 0F 00 00 00 00 00 00 00'
 '00 00 00 00 00 DD 00 00 00 DD 00 00 00 88 00 00'
 '00 88 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 F0 00 0F 00'
}


BAR3 BITMAP
{
 '42 4D 9E 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 0A 00 00 00 05 00 00 00 01 00 04 00 00 00'
 '00 00 28 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 00 01 10 01 08 77'
 '77 77 70 55 55 55 0F 88 88 88 70 55 55 55 0F FF'
 'FF FF 80 55 55 55 00 00 00 00 00 55 55 55'
}


BAR3MASK BITMAP
{
 '42 4D 9E 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 0A 00 00 00 05 00 00 00 01 00 04 00 00 00'
 '00 00 28 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 F0 00 00 00 0F 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 F0 00 00 00 0F 00 00 00'
}


BAR4 BITMAP
{
 '42 4D A6 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 0C 00 00 00 06 00 00 00 01 00 04 00 00 00'
 '00 00 30 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 00 00 00 00 08 77'
 '77 77 77 70 11 11 0F 88 88 88 88 70 44 44 0F 88'
 '88 88 88 70 11 11 0F FF FF FF FF 80 44 44 00 00'
 '00 00 00 00 00 00'
}


BAR4MASK BITMAP
{
 '42 4D A6 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 0C 00 00 00 06 00 00 00 01 00 04 00 00 00'
 '00 00 30 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
```

```
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 F0 00 00 00 00 0F 00 00 00 00'
 '00 00 00 00 11 11 00 00 00 00 00 00 44 44 00 00'
 '00 00 00 00 11 11 00 00 00 00 00 00 44 44 F0 00'
 '00 00 00 0F 00 00'
}


CIRCLE1 BITMAP
{
 '42 4D 96 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 08 00 00 00 08 00 00 00 01 00 04 00 00 00'
 '00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 00 87 77 00 08 88'
 '87 70 0F 88 88 70 0F 88 88 70 0F F8 88 80 00 FF'
 'F8 00 00 00 00 00'
}


CIRCLE1MASK BITMAP
{
 '42 4D 96 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 08 00 00 00 08 00 00 00 01 00 04 00 00 00'
 '00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF 00 00 FF F0 00 00 0F 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 F0 00'
 '00 0F FF 00 00 FF'
}


DIAMOND1 BITMAP
{
 '42 4D 96 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 08 00 00 00 08 00 00 00 01 00 04 00 00 00'
 '00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 00 07 70 00 00 88'
 '77 00 0F 88 87 70 0F F8 88 70 00 FF 88 00 00 0F'
 'F0 00 00 00 00 00'
}


DIAMOND1MASK BITMAP
{
 '42 4D 96 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 08 00 00 00 08 00 00 00 01 00 04 00 00 00'
 '00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
```

```
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF F0 0F FF FF 00 00 FF F0 00'
 '00 0F 00 00 00 00 00 00 00 00 F0 00 00 0F FF 00'
 '00 FF FF F0 0F FF'
}


DIAMOND2 BITMAP
{
 '42 4D D6 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 0C 00 00 00 0C 00 00 00 01 00 04 00 00 00'
 '00 00 60 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 00 00 8F F0 00 00'
 '07 70 00 00 FF F0 00 00 78 87 00 00 E8 F0 00 07'
 '88 88 70 00 EE F0 00 78 88 88 87 00 66 60 07 88'
 '88 88 88 70 0E E0 0F 88 88 88 88 F0 E6 E0 00 F8'
 '88 88 8F 00 EE E0 00 0F 88 88 F0 00 EE E0 00 00'
 'F8 8F 00 00 EE E0 00 00 0F F0 00 00 66 E0 00 00'
 '00 00 00 00 EE E0'
}


DIAMOND2MASK BITMAP
{
 '42 4D D6 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 0C 00 00 00 0C 00 00 00 01 00 04 00 00 00'
 '00 00 60 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF FF F0 0F FF FF 8F F0 FF FF'
 '00 00 FF FF FF F0 FF F0 00 00 0F FF E8 F0 FF 00'
 '00 00 00 FF EE F0 F0 00 00 00 00 0F 66 60 00 00'
 '00 00 00 00 0E E0 00 00 00 00 00 00 E6 E0 F0 00'
 '00 00 00 0F EE E0 FF 00 00 00 00 FF EE E0 FF F0'
 '00 00 0F FF EE E0 FF FF 00 00 FF FF 66 E0 FF FF'
 'F0 0F FF FF EE E0'
}


DIAMOND3 BITMAP
{
 '42 4D BE 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 09 00 00 00 09 00 00 00 01 00 04 00 00 00'
 '00 00 48 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 05 55 55 55 00 00'
 '70 00 00 00 00 00 00 08 77 00 05 55 55 55 00 88'
 '87 70 01 00 00 00 0F F8 88 77 00 00 00 00 00 FF'
 '88 80 00 00 00 00 00 0F F8 00 00 00 00 00 00 00'
 'F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
}


DIAMOND3MASK BITMAP
```

```
{
 '42 4D BE 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 09 00 00 00 09 00 00 00 01 00 04 00 00 00'
 '00 00 48 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF FF 0F FF F5 55 55 55 FF F0'
 '00 FF F0 00 00 00 FF 00 00 0F F5 55 55 55 F0 00'
 '00 00 F1 00 00 00 00 00 00 00 00 00 00 00 F0 00'
 '00 00 F0 00 00 00 FF 00 00 0F F0 00 00 00 FF F0'
 '00 FF F0 00 00 00 FF FF 0F FF F0 00 00 00'
}


DIAMOND4 BITMAP
{
 '42 4D CE 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 0B 00 00 00 0B 00 00 00 01 00 04 00 00 00'
 '00 00 58 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 00 00 00 00 00 00'
 '07 00 00 00 00 00 00 00 87 70 00 05 55 55 00 08'
 '88 77 00 00 00 00 00 88 88 87 70 05 55 55 0F F8'
 '88 88 77 00 00 00 00 FF 88 88 80 00 00 00 00 0F'
 'F8 88 00 00 00 00 00 00 FF 80 00 00 00 00 00 00'
 '0F 00 00 00 00 00 00 00 00 00 00 00 00 00'
}


DIAMOND4MASK BITMAP
{
 '42 4D CE 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 0B 00 00 00 0B 00 00 00 01 00 04 00 00 00'
 '00 00 58 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 FF FF F0 FF FF F0 00 00 FF FF'
 '00 0F FF F0 00 00 FF F0 00 00 FF F5 55 55 FF 00'
 '00 00 0F F0 00 00 F0 00 00 00 00 F5 55 55 00 00'
 '00 00 00 00 00 00 F0 00 00 00 00 F0 00 00 FF 00'
 '00 00 0F F0 00 00 FF F0 00 00 FF F0 00 00 FF FF'
 '00 0F FF F0 00 00 FF FF F0 FF FF F0 00 00'
}


SQUARE1 BITMAP
{
 '42 4D 96 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 08 00 00 00 08 00 00 00 01 00 04 00 00 00'
 '00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 08 77 77 70 0F 88'
 '88 70 0F 88 88 70 0F 88 88 70 0F 88 88 70 0F FF'
```

```
 'FF 80 00 00 00 00'
}


SQUARE1MASK BITMAP
{
 '42 4D 96 00 00 00 00 00 00 00 76 00 00 00 28 00'
 '00 00 08 00 00 00 08 00 00 00 01 00 04 00 00 00'
 '00 00 20 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 10 00 00 00 00 00 00 00 00 00 80 00 00 80'
 '00 00 00 80 80 00 80 00 00 00 80 00 80 00 80 80'
 '00 00 80 80 80 00 C0 C0 C0 00 00 00 FF 00 00 FF'
 '00 00 00 FF FF 00 FF 00 00 00 FF 00 FF 00 FF FF'
 '00 00 FF FF FF 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00'
}


HANDPOINTER CURSOR
{
 '00 00 02 00 01 00 20 20 00 00 06 00 10 00 30 01'
 '00 00 16 00 00 00 28 00 00 00 20 00 00 00 40 00'
 '00 00 01 00 01 00 00 00 00 00 80 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 FF FF FF 00 03 F8 00 00 03 F8 00 00 07 F8'
 '00 00 07 FC 00 00 0F FC 00 00 1B FC 00 00 1B F4'
 '00 00 33 54 00 00 63 50 00 00 03 00 00 00 03 00'
 '00 00 03 00 00 00 03 00 00 00 03 00 00 00 03 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
 '00 00 00 00 00 00 F8 03 FF FF F8 03 FF FF F0 03'
 'FF FF F0 01 FF FF E0 01 FF FF C0 01 FF FF C0 01'
 'FF FF 80 01 FF FF 08 03 FF FF 98 0F FF FF F8 7F'
 'FF FF F8 7F FF FF F8 7F FF FF F8 7F FF FF F8 7F'
 'FF FF FC FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF'
 'FF FF FF FF FF FF'
}
```

Return to Component CookBook
Return To Front Page

The Unofficial Newsletter of Delphi Users - Issue #9 - November 9th, 1995

Developing Windows Applications Using Delphi

Book Review by Bob Swart - CIS: 100434,2072

Title:	Developing Windows Applications Using Delphi
Author:	Paul Penrod
ISBN:	0-471-11017-5 (John Wiley & Sons, Inc)
Pages:	353 (no disk)
Price:	US$ 29.95, CAN$ 41.95

This book is intended for those of us who want to start working with Delphi (and for Windows Programming) and come from a C/C++ background. The book is full with little C examples and equivalent ObjectPascal/Delphi examples, and is therefore a good way for all those C programmers who realised they need to get their hands on Delphi and get up to speed right now!

The book is divided into three sections: introduction & usage, programming and language. The introduction & usage section contains the usual background and information of the most important features of Delphi and the IDE. The programming section demonstrates the use of many of the Delphi components in a 'how-to' fashion by developing a project and then extending it along the way. The language section - the main part of the book - concentrates on the ObjectPascal programming language and OOP in a practical manner. The author explains the basics of OO and uses a lot of C examples to show the differences between C/C++ and ObjectPascal. This part is therefore especially useful to current (past?) programmers of C/C++ or 'plain' Pascal that want to move up to Delphi and ObjectPascal. Other than these three parts there is a good chapter on rousting errors with the integrated debugger of Delphi, which is something we almost forget we can do at all (at times). A nice chapter that shows the ins and outs of the IDE debugger!

The book contains very little (almost zero) database stuff, but that's not strange, since the main objective was to teach OO Windows Programming with Delphi and ObjectPascal for current C or non-OOP programmers. I think for those readers, coming from a C background, or stepping up to OOP, this book certainly has a lot to offer! There is no disk, but the code is never much more than a few lines.

Return to Front Page

Index of Past Issues

Below is a complete index of all principle articles in past issues of the Unofficial Newsletter of Delphi Users. Provided that you have the prior issues in the same directory as this issue, you can click on any of these hotspots to go directly to that article. To return to the index, you can click on the **Back** button, or you can use the **History** list. Once you jump to one of these issues, you can navigate through the issue as you would normally, but you will need to go to the **History** list to get back to this index. There will be an updated index included in all future issues of UNDU.

Issue #4 - May 24, 1995

Issue #5 - June 26, 1995

Issue #6 - July 25, 1995

Issue #7 - August 31, 1995

Issue #8 - October 10, 1995

Issue #9 - November 9, 1995 (This issue)

Return to Front Page

```
unit Qsort;

{TQSort by Mike Junkin 10/19/95.
 DoQSort routine adapted from Peter Szymiczek's QSort procedure which
 was presented in issue#8 of The Unofficial Delphi Newsletter.}

interface

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

type
  TSwapEvent = procedure (Sender : TObject; e1,e2 : word) of Object;
  TCompareEvent = procedure (Sender: TObject; e1,e2 : word; var Action : integer)
of Object;

  TQSort = class(TComponent)
  private
    FCompare : TCompareEvent;
    FSwap : TSwapEvent;
  public
    procedure DoQSort(Sender: TObject; uNElem: word);
  published
    property Compare : TCompareEvent read FCompare write FCompare;
    property Swap : TSwapEvent read FSwap write FSwap;
  end;

procedure Register;

implementation

procedure Register;
begin
  RegisterComponents('Mikes', [TQSort]);
end;

procedure TQSort.DoQSort(Sender: TObject; uNElem: word);
{ uNElem - number of elements to sort }

  procedure qSortHelp(pivotP: word; nElem: word);
  label
    TailRecursion,
    qBreak;
  var
    leftP, rightP, pivotEnd, pivotTemp, leftTemp: word;
    lNum: word;
    retval: integer;
  begin
    retval := 0;
    TailRecursion:
      if (nElem <= 2) then
        begin
          if (nElem = 2) then
            begin
              rightP := pivotP +1;
              FCompare(Sender,pivotP,rightP,retval);
              if (retval > 0) then Fswap(Sender,pivotP,rightP);
            end;
          exit;
        end;
```

```
  rightP := (nElem -1) + pivotP;
  leftP :=  (nElem shr 1) + pivotP;
  { sort pivot, left, and right elements for "median of 3" }
  FCompare(Sender,leftP,rightP,retval);
  if (retval > 0) then Fswap(Sender,leftP, rightP);
  FCompare(Sender,leftP,pivotP,retval);
  if (retval > 0) then Fswap(Sender,leftP, pivotP)
  else
    begin
      FCompare(Sender,pivotP,rightP,retval);
      if retval > 0 then Fswap(Sender,pivotP, rightP);
    end;
  if (nElem = 3) then
    begin
      Fswap(Sender,pivotP, leftP);
      exit;
    end;
  { now for the classic Horae algorithm }
  pivotEnd := pivotP + 1;
  leftP := pivotEnd;
  repeat
    FCompare(Sender,leftP, pivotP,retval);
    while (retval <= 0) do
      begin
        if (retval = 0) then
          begin
            Fswap(Sender,leftP, pivotEnd);
            Inc(pivotEnd);
          end;
        if (leftP < rightP) then
          Inc(leftP)
        else
          goto qBreak;
        FCompare(Sender,leftP, pivotP,retval);
      end; {while}
    while (leftP < rightP) do
      begin
        FCompare(Sender,pivotP, rightP,retval);
        if (retval < 0) then
          Dec(rightP)
        else
          begin
            FSwap(Sender,leftP, rightP);
            if (retval <> 0) then
              begin
                Inc(leftP);
                Dec(rightP);
              end;
            break;
          end;
      end; {while}

  until (leftP >= rightP);
qBreak:
  FCompare(Sender,leftP,pivotP,retval);
  if (retval <= 0) then Inc(leftP);
  leftTemp := leftP -1;
  pivotTemp := pivotP;
  while ((pivotTemp < pivotEnd) and (leftTemp >= pivotEnd)) do
    begin
      Fswap(Sender,pivotTemp, leftTemp);
      Inc(pivotTemp);
      Dec(leftTemp);
```

```
        end; {while}
      lNum := (leftP - pivotEnd);
      nElem := ((nElem + pivotP) -leftP);

      if (nElem < lNum) then
        begin
          qSortHelp(leftP, nElem);
          nElem := lNum;
        end
      else
        begin
          qSortHelp(pivotP, lNum);
          pivotP := leftP;
        end;
      goto TailRecursion;
    end; {qSortHelp }

begin
  if Assigned(FCompare) and Assigned(FSwap) then
  begin
    if (uNElem < 2) then  exit; { nothing to sort }
    qSortHelp(1, uNElem);
  end;
end; { QSort }

end.
```

Return to QSort Article
Return to Component CookBook
Return to Front Page

Delphi-Related Internet Sites

Here is the latest list of Delphi-Related Internet sites that I have come across. Please let me know if you find others!

WWW Sites

The Delphi Source - http://www.doit.com/delphi/
The Delphi Station - http://www.teleport.com/~cwhite/wilddelphi.html
Pierre Mertz Scigraph - http://www.ee.princeton.edu/~phmertz/scigraph/scigraph.html
StarTech Home Page - http://www.neosoft.com/~startech/delphi/delphi.htm
Ann Lynnworth's CGI Components - http://super.sonic.net:80/ann/delphi/cgicomp/
The Dephi Super Page - http://sunsite.icm.edu.pl/~robert/delphi/
The Delphi Connection - http://www.pennant.com/delphi.html
Delphi Hacker's Page - http://www.it.kth.se/~ao/DHC/
Delphi Technical Support - http://loki.borland.com:8080/
Boris Ingram, Cyborg Software - http://www.pcb.co.za/users/borising/cyborg.htm
The Coriolis Group's Delphi Page -
http://www.coriolis.com/coriolis/whatsnew/delphi.htm
Grumpfish, Inc., Productivity and Education - http://www.teleport.com/~grump
The Delphi Bug List - http://www.cybernetics.net/users/bstowers/delphi-bugs.html
The City Zoo - http://www.mindspring.com/~cityzoo/cityzoo.html
The Delphi Online Magazine - http://www.teleport.com/~sig/dol.html
Dave's Delphi Destination - http://vislab-www.nps.navy.mil/~drmcderm/delphi.html
Bill Chosiad's Delphi Resources - http://www.shore.net/~billc/delphi.html
Delphi On-Line Conference - http://www.creativeis.com/talk/working.htm
http://www.widewest.com.au/aerosoft/
Delphi Books and Articles - http://www.iscinc.com/dugbib.html
Daniel Parnell's Delphi Page - http://minyos.xx.rmit.edu.au/~s921878/delphi.html
Damon Wischik's Delphi Page - http://www.statslab.cam.ac.uk/~djw1005/Delphi/
Wool2Wool Software - http://www.webcom.com/~wol2wol/
HyperAct, Inc - http://www.hyperact.com/
C.I.U.P., K.C. Software Headquarters - http://www.webcom.com/~kilgalen/welcome.html
SilverWare Inc (Communication Tools) - http://rampages.onramp.net/~silver
Preferred Solution Ltd's Delphi Component Prize Draw -
http://www.maui.net/~russt/ps.html
Dion Kurczek's Delphi Page - http://www.silicmdr.com
Bill White's Delphi Page - http://www.destek.net/cybermkt/blwhite.htm
Gurnsey Software Company - http://www.scruz.net/~daley/
Kinetic Software Delphi Web Site - http://www.esper.com/kinetic/
EMS Professional Shareware - http://www.wdn.com/ems

User Group's Home Pages

http://super.sonic.net/delphisig/index.html
New York Delphi User Group - http://www.iscinc.com/nydug.html
Auckland Delphi User Group - http://iconz.co.nz/commercial/Borland/
Salt Lake City Delphi Users Group - http://www.xmission.com/~uldata/delphi.html
Philadelphia Delphi Users Group -
http://www.datacraft.db.com/~PhillyDelphi/index.html

Delphi Developers of Dallas (3D)- http://rampages.onramp.net/~jtsabin/3dhome.htm

FTP Sites

ftp://ftp.coriolis.com/delphi/
ftp://sunsite.icm.edu.pl/pub/delphi/
ftp://ftp.sccsi.com/pub/local/moai/
ftp://ftp.sonic.net/pub/mall/moreDelphi/
ftp://ftp.oslohd.no/pub/win3/delphi
ftp://ftp.cybernetics.net/pub/users/bstowers/
ftp://garbo.uwasa.fi/windows/programming/
ftp://parnas.mimuw.edu.pl/pub5/delphi/freeware
ftp://ftp.cdrom.com:/pub/delphi

Thanks to Andrea Mennini - jake@blues.dsnet.it

Return to Front Page

The Unofficial Newsletter of Delphi Users - Issue #9 - November 9th, 1995

Core Concepts with Delphi - Variables and Data Types

by Alan G. Labouseur

Good day, and welcome to the next chapter of Core Concepts. Last month we gained an appreciation for the beauty and ease of Delphi as a high-level, compiled language -- as opposed to an early generation, interpreted language. Now we're going to explore some of the parts of the Delphi programming language that make it beautiful and easy to use. First up is the topic of variables and data types. We'll look at declaring, initializing, and making assignments to variables of some elementary data types.

A variable, simply put, is a named value. For example, a variable called "FirstName" might contain the value "Alan". This means that there is some group of locations in your computer's memory that are identified as "FirstName" and which contain the computer's representation of the value "Alan". So if we were to output the contents of FirstName by issuing the Object Pascal command writeln FirstName; we would expect that output to be "Alan".

Of course FirstName could contain anybody's name, not just mine. The value stored in a variable can be changed. (Thus the name, "variable". If we couldn't change the value, we'd have to call it a "constant".) To get a value into FirstName we use the Object Pascal assignment operator (:=). We could write the following Object Pascal code snippet:

```
FirstName := 'Alan';
Writeln FirstName;      {output is "Alan"}
FirstName := 'Cate';
Writeln FirstName;      {output is "Cate"}
```

We see that we can change a variable's value. Also notice that FirstName can hold only one value at a time. This is a crucial point and cannot be over-stressed. A variable can only hold one value at a time! When you assign it a new value, the old one is lost.

Names are obviously not the only values that you will want to hold in variables. You'll want to store numbers, and you may wish to note whether or not something has been done or some conditions met. Now you might think that you could assign the number 2741 to FirstName. And you would be right in some programming languages, but not in Object Pascal. Once you set up FirstName to hold values like "Alan" and "Cate" you will not be allowed to assign it values like 2741 or True. But how does the compiler know, you ask? How DO it know? Well, we have to tell it.

Before you can assign any value to a variable, you must declare it's existence to the compiler. You must inform the compiler of the variable's name (the identifier) and what type of values you plan on storing in it. You do this by declaring each variable to be of a certain data type. It would be nice, then, if there were several data types since we are interested in dealing with several types of data. Thankfully, there are. Delphi provides many data types for our programming pleasure. I will discuss the more elementary ones here.

To hold data the likes of "Alan" and "Cate" we would use the "string" data type. A Delphi string can be any combination of letters, digits, spaces, and punctuation symbols up to 255 characters in length, bounded by single quotes.

If we want to hold only a single character in a variable we would use the "char" data type. It can hold one letter, digit, or punctuation symbol contained in single quotes. Given this, you can think of a string as a convenient way of handling up to 255 characters "strung" together.

My favorite number, 2741, is an example of the "integer" data type. Integers in Delphi are whole numbers (no fractions or decimals, please) that can be as small as -32768 or as large as +32767.

For those numerical values that require fractions we use the "real" data type. Real numbers can have many decimal places. (Does this imply that integers are not real, and are therefore fake?)

For those issues that are truly black and white, there is the "boolean" data type. It has only two possible

values: True or False. Either it is or it ain't.

Here are some string, integer, real, and boolean declarations...

```
var
  FirstName: String;
  LastName: String;
  Age: Integer;
  GPA: Real;
  Male: Boolean;
```

... and here is some Object Pascal source code initializing them:

```
FirstName := 'James';
LastName := 'Bond';
Age := 41;
GPA := 3.007;
Male := True;
```

All of the above assignment statements are legal because all of the variable identifiers I used are declared in the "var" section above, and all values I assigned are type compatible with their host variables. But how does the compiler know this?

You see, the compiler is sneaky. During compilation, all declared variables are stored in a list called the symbol table. The name and data type are among the items stored. This allows the compiler to check on you when you use these variables later on. (You can't get away with anything!) So when the compiler got to the first assignment, it saw the value, 'James', and noted that it was a string. It then took the variable identifier, FirstName, looked it up in the symbol table, and saw that it was declared as a string, so all was cool. Similarly, the compiler notes that 'Bond' is also a string, Age is an integer as is 41, that GPA is real to match the decimal value assigned to it, and Male is boolean and as such it can accept the value True.

This type compatibility enforcement is not just for initialization, it is ubiquitous. Every variable reference in the program is checked. To demonstrate, here are some more legal uses of these variables.

```
Age := Age + 1;
Male := not Male;
GPA := GPA * 2;
```

Each of these statements conform to the type compatibility of the declared variables.

Here are a few that don't.

```
FirstName := LastName + Male;    {You cannot add a boolean to a string}
Age := Age / 3                   {Division requires real data type variables}
Male := 'Alan';                  {You cannot assign a string to a boolean}
```

The compiler will complain about each of these statements as soon as it sees them. This is a good thing because the chances are that you didn't intend to write them that way. Type compatibility enforcement results in the compiler warning you about errors at compile time, before you or your users uncover the mistakes at runtime. Some languages that aren't strongly typed (such as Ms-Visual Basic) will let you make these mistakes without even a peep. The customer, client, or boss for whom you wrote the software usually isn't so understanding, or quiet!

Here's some more code, with less obvious data type errors:

```
var
  Age: integer;
  Weight: real;
  PObox: string;

begin
  Age := 27;
  Weight := 185.7;
  PObox := '117';
```

```
    Age := Age + PObox;
    {Even though PObox contains what looks like an integer, we have declared
    it as a string.  You cannot use a string in integer addition.}
    Age := Weight;
    {Here we're trying to assign a real to an integer.  But the integer
    cannot hold the decimals, so this is illegal.}
end;
```

We have discussed only the elementary data types. Delphi provides many more that we may discuss in the future. For now, remember that variables are named values, and that they hold only a single value at a time. Every variable must be declared with a name and a data type, and the data type will be strictly enforced throughout your program.

I hope this has been helpful in providing more core information upon which we can build in the future. Feel free, encouraged even, to e-mail me with any comments, questions, or suggestions.

Thank you and goodnight.

About the Author

Alan G. Labouseur is Vice President and co-owner of AlphaPoint Systems, Inc., a custom programming consultancy located in Brewster, NY. Alan has a Masters degree in Computer Science and has been developing custom software for ten years. He is currently working on Clipper and Delphi applications for clients in the NY-NJ-CT area. You can reach Alan through e-mail at AGL007@IX.NETCOM.COM or 70312,2726 here on CompuServe.

Return to Front Page

```
unit CurrEdit;

(*************************************************************************
 This is my first custom control, so please be merciful. I needed a simple
 currency edit field, so below is my attempt. It has pretty good behavior
 and I have posted it up to encourage others to share their code as well.

 Essentially, the CurrencyEdit field is a modified memo field. I have put
 in keyboard restrictions, so the user cannot enter invalid characters.
 When the user leaves the field, the number is reformatted to display
 appropriately. You can left-, center-, or right-justify the field, and
 you can also specify its display format - see the FormatFloat command.
 The field value is stored in a property called Value so you should read
 and write to that in your program. This field is of type Extended.

 If you like this control you can feel free to use it, however, if you
 modify it, I would like you to send me whatever you did to it. If you
 send me your CIS ID, I will send you copies of my custom controls that
 I develop in the future. Please feel free to send me anything you are
 working on as well. Perhaps we can spark ideas!

 Robert Vivrette, Owner
 Prime Time Programming
 PO Box 5018
 Walnut Creek, CA  94596-1018

 Fax: (510) 939-3775
 CIS: 76416,1373
 Net: RobertV@ix.netcom.com

 Thanks to Massimo Ottavini, Thorsten Suhr, Bob Osborn, Mark Erbaugh, Ralf
 Gosch, Julian Zagorodnev, and Grant R. Boggs for their enhancements!

 Please look for this and other components in the "Unofficial Newsletter of
 Delphi Users" posted on the Borland Delphi forum on Compuserve (GO DELPHI)
 in the "Delphi IDE" file section.

*************************************************************************)

interface

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

type
  TCurrencyEdit = class(TCustomMemo)
  private
    DispFormat: string;
    FieldValue: Extended;
    FDecimalPlaces : Word;
    FPosColor : TColor;
    FNegColor : TColor;
    procedure SetFormat(A: string);
    procedure SetFieldValue(A: Extended);
    procedure SetDecimalPlaces(A: Word);
    procedure SetPosColor(A: TColor);
    procedure SetNegColor(A: TColor);
    procedure CMEnter(var Message: TCMEnter);  message CM_ENTER;
    procedure CMExit(var Message: TCMExit);    message CM_EXIT;
    procedure FormatText;
```

```
    procedure UnFormatText;
  protected
    procedure KeyPress(var Key: Char); override;
    procedure CreateParams(var Params: TCreateParams); override;
  public
    constructor Create(AOwner: TComponent); override;
  published
    property Alignment default taRightJustify;
    property AutoSize default True;
    property BorderStyle;
    property Color;
    property Ctl3D;
    property DecimalPlaces: Word read FDecimalPlaces write SetDecimalPlaces default
2;
    property DisplayFormat: string read DispFormat write SetFormat;
    property DragCursor;
    property DragMode;
    property Enabled;
    property Font;
    property HideSelection;
    property MaxLength;
    property NegColor: TColor read FNegColor write SetNegColor default clRed;
    property ParentColor;
    property ParentCtl3D;
    property ParentFont;
    property ParentShowHint;
    property PopupMenu;
    property PosColor: TColor read FPosColor write SetPosColor default clBlack;
    property ReadOnly;
    property ShowHint;
    property TabOrder;
    property Value: Extended read FieldValue write SetFieldValue;
    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

procedure Register;
begin
  RegisterComponents('Additional', [TCurrencyEdit]);
end;

constructor TCurrencyEdit.Create(AOwner: TComponent);
begin
  inherited Create(AOwner);
  AutoSize := False;
  Alignment := taRightJustify;
```

```
  Width := 121;
  Height := 25;
  DispFormat := '$,0.00;($,0.00)';
  FieldValue := 0.0;
  FDecimalPlaces := 2;
  FPosColor := Font.Color;
  FNegColor := clRed;
  AutoSelect := False;
  {WantReturns := False;}
  WordWrap := False;
  FormatText;
end;

procedure TCurrencyEdit.SetFormat(A: String);
begin
  if DispFormat <> A then
    begin
      DispFormat:= A;
      FormatText;
    end;
end;

procedure TCurrencyEdit.SetFieldValue(A: Extended);
begin
  if FieldValue <> A then
    begin
      FieldValue := A;
      FormatText;
    end;
end;

procedure TCurrencyEdit.SetDecimalPlaces(A: Word);
begin
  if DecimalPlaces <> A then
    begin
      DecimalPlaces := A;
      FormatText;
    end;
end;

procedure TCurrencyEdit.SetPosColor(A: TColor);
begin
  if FPosColor <> A then
    begin
      FPosColor := A;
      FormatText;
    end;
end;

procedure TCurrencyEdit.SetNegColor(A: TColor);
begin
  if FNegColor <> A then
    begin
      FNegColor := A;
      FormatText;
    end;
end;

procedure TCurrencyEdit.UnFormatText;
var
  TmpText : String;
  Tmp     : Byte;
  IsNeg   : Boolean;
```

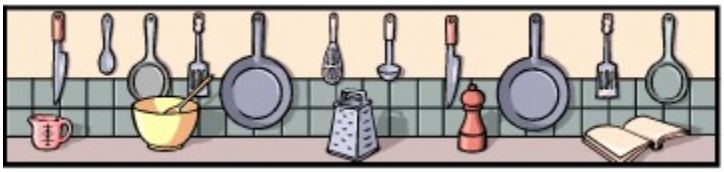
```
begin
  IsNeg := (Pos('-',Text) > 0) or (Pos('(',Text) > 0);
  TmpText := '';
  For Tmp := 1 to Length(Text) do
    if Text[Tmp] in ['0'..'9',DecimalSeparator] then
      TmpText := TmpText + Text[Tmp];
  try
    If TmpText='' Then TmpText := '0.00';
    FieldValue := StrToFloat(TmpText);
    if IsNeg then FieldValue := -FieldValue;
  except
    MessageBeep(mb_IconAsterisk);
  end;
end;

procedure TCurrencyEdit.FormatText;
begin
  Text := FormatFloat(DispFormat,FieldValue);
  if FieldValue < 0 then
    Font.Color := NegColor
  else
    Font.Color := PosColor;
end;

procedure TCurrencyEdit.CMEnter(var Message: TCMEnter);
begin
  SelectAll;
  inherited;
end;

procedure TCurrencyEdit.CMExit(var Message: TCMExit);
begin
  UnformatText;
  FormatText;
  Inherited;
end;

procedure TCurrencyEdit.KeyPress(var Key: Char);
Var
  S : String;
  frmParent : TForm;
  btnDefault : TButton;
  i : integer;
  wID : Word;
  LParam : LongRec;
begin
  {#8 is for Del and Backspace keys.}
  if Not (Key in ['0'..'9','.','-', #8, #13]) Then Key := #0;
  case Key of
    #13 : begin
            frmParent := GetParentForm(Self);
            UnformatText;
            {find default button on the parent form if any}
            btnDefault := nil;
            for i := 0 to frmParent.ControlCount -1 do
              if frmParent.Controls[i] is TButton then
                if (frmParent.Controls[i] as TButton).Default then
                  btnDefault := (frmParent.Controls[i] as TButton);
            {if there's a default button, then make the parent form think it was
pressed}
            if btnDefault <> nil then
              begin
                wID := GetWindowWord(btnDefault.Handle, GWW_ID);
```

```
                LParam.Lo := btnDefault.Handle;
                LParam.Hi := BN_CLICKED;
                SendMessage(frmParent.Handle, WM_COMMAND, wID, longint(LParam) );
              end;
            Key := #0;
          end;
          { allow only one dot in the number }
    '.' : if ( Pos('.',Text) >0 ) then Key := #0;
          { allow only one '-' in the number and only in the first position: }
    '-' : if ( Pos('-',Text) >0 ) or ( SelStart > 0 ) then Key := #0;
  else
    { make sure no other character appears before the '-' }
    if ( Pos('-',Text) >0 ) and ( SelStart = 0 ) and (SelLength=0) then Key := #0;
  end;

  if Key <> Char(vk_Back) then
    begin
     {S is a model of Text if we accept the keystroke.  Use SelStart and
     SelLength to find the cursor (insert) position.}
      S := Copy(Text,1,SelStart)+Key+Copy(Text,SelStart+SelLength+1,Length(Text));
      if ((Pos(DecimalSeparator, S) > 0) and
         (Length(S) - Pos(DecimalSeparator, S) > FDecimalPlaces))  {too many
decimal places}
          or ((Key = '-') and (Pos('-', Text) <> 0))     {only one minus...}
          or (Pos('-', S) > 1)                           {... and only at
beginning}
      then Key := #0;
    end;

  if Key <> #0 then inherited KeyPress(Key);
end;

procedure TCurrencyEdit.CreateParams(var Params: TCreateParams);
var
 lStyle : longint;
begin
  inherited CreateParams(Params);
  case Alignment of
    taLeftJustify  : lStyle := ES_LEFT;
    taRightJustify : lStyle := ES_RIGHT;
    taCenter       : lStyle := ES_CENTER;
  end;
  Params.Style := Params.Style or lStyle;
end;

end.
```

Return to Cooking Up Components

Return to Front Page

The Unofficial Newsletter of Delphi Users - Issue #9 - November 9th, 1995



The Component Cookbook

by Robert Vivrette

Some things old, and some things new...

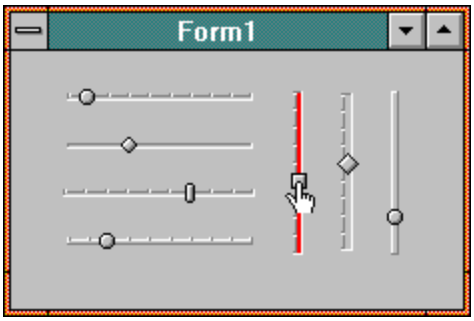
This issue, I am revisiting an old friend: The CurrEdit component was one of my first components I created, and it has draw quite a bit of interest from many of the readers of UNDU. Every week or so, I get another suggestion from someone to improve the behavior of CurrEdit. This issue, I have incorporated some of the updates provided by a few more of the readers.

Updated CurrEdit Source Code

Also, we have in this issue an update to the QSort article that appeared last issue. Check out the QSort Component by Mike Junkin!

And now... TSlideBar!

I recently had an article published in the September 1995 issue of Delphi Informant. In the article, I presented my latest component, the **TSlideBar.** Basically, the SlideBar is an enhanced form of the Windows ScrollBar component. There are properties to control the various aspects of the slider track, the thumb bar, cursor, tick marks and much more.



Below is the source files for **TSlideBar** in various forms. If you would like to learn more about TSlideBar, please request a copy of the September 1995 issue of Delphi Informant (Volume 1 Number 5). The article and others in this issue go into a lot of advanced component design principals. Back issues can be obtained by calling the subscription department at (916) 686-6610. Don't delay, as I understand the first three issues of *Delphi Informant* are out of print at this point, and the other issues are going fast!
Also, I have completed a data-aware version of **TSlideBar** that will appear in Delphi Informant sometime in the future. Please don't ask for a copy of the data-aware version before it gets published... Thanks. (Sorry to tease you like that, but I just wanted everyone to know it was coming!)

The version below includes the SLIDEBAR.PAS file and SLIDEBAR.RC file. You will need to use a program such as Borland Resource Workshop to convert the .RC file into a .RES file so Delphi can compile the unit. I included the RC file because it is textual and not binary.

SlideBar Source Code (SLIDEBAR.PAS)

SlideBar Resource File (SLIDEBAR.RC)

This version is UUEncoded and contains all the files you need (SLIDEBAR.PAS, SLIDEBAR.DCR, SLIDEBAR.RES) to add the **TSlideBar** component into Delphi.

UUEncoded SLIDEBAR.ZIP package

Return to Front Page

The Unofficial Newsletter of Delphi Users - Issue #9 - November 9th, 1995

Object Constructors

by Robert Vivrette

Recently I had a discussion with a gentleman on CompuServe regarding the creation of objects. I realized later that the material we covered might be of interest to the readers of UNDU. The initial question was why you have to create an object like this:

```
MyObject := TMyObject.Create(Self);
```

and not like this:

```
MyObject.Create;
```

The Answer

When you define an object in the TYPE area of code, it is making a "boiler plate" of what makes up the object. It is not actually an object yet. Just the rules of how to make the object. Creating a real object from this boiler plate definition is called "instantiation", meaning "Take this definition and make me one of the objects".

When you declare a variable as of a particular object type in the **Var** section, you are saying "Here is a variable that will eventually be an object of type TMyObject". It is not there yet though. It is like knowing you are going to build a house on a lot, so you clear out an area and put down a foundation for the house. No house yet, but the space is prepared. The same thing is true of the creation of instances of objects in Delphi.

Now, perhaps, you can see why you cannot do...

```
MyObject.Create;
```

MyObject has no substance at this point. Instead you need to assign something into MyObject to initially build it. This is done as...

```
MyObject := TMyObject.Create(Self);
```

Even though the boiler plate is just rules and no substance, it does have the ability to build a copy of itself by means of the **Create** constructor. The methods (procedures) that function like this are called Class methods. That means that they can function in the boiler-plate without an actual copy of the object being created. The Create constructor is the most common type of class method. Depending on which object you are inheriting from, some will require you to pass the object's owner to the constructor; ie. TMyObject.Create(Self). In other cases however the object may not need one.

So... by doing the line...

```
MyObject := TMyObject.Create(Self);
```

... you are telling the object to create a copy of itself, and then pass back a pointer to that new object. Objects in Delphi are generally just pointers. MyObject is only a 4-byte pointer off to some memory location. It is not an actual structure with variables and fields inside. When you try to do this

```
MyObject.Create;
```

... before the object is created, the compiler will choke because MyObject does not have a valid pointer yet, so it can't call any of its methods. The only way of getting a valid pointer is by calling the objects Create constructor which fortunately (and necessarily) can be called from the class definition itself. This class definition (boiler plate) is NOT treated as a "pointer to an object" and so *its* create constructor is available.

Hope this clears things up for those of you who are interested. Let me know if anyone has additional comments on this...

Return to Tips & Tricks

Return to Front Page

The Unofficial Newsletter of Delphi Users - Issue #9 - November 9th, 1995

The Delphi Magazine

By Dr. Bob - Internet: DrBob@pi.net

The contents for The Delphi Magazine Issue 4 (November 1995) are:

Editorial: From issue #5 onwards The Delphi Magazine will come out monthly!

News: Updates from the Delphi world.

Book Review: Delphi Unleashed -- Jeroen Pluimers reviews Charlie Calvert's comprehensive Delphi guide.

Inside TApplication -- Did you know all your Delphi programs have an extra zero sized Window? Nick Hodges shows how to take advantage of TApplication's features, with extra tips from Hallvard Vassbotn too.

Surviving Client/Server" Getting Started With SQL, Part 2 -- Steve Troxell continues his column with the second part of his explanation of the basics of SQL.

Under Construction: Component Help -- Bob Swart shows how to integrate help for developers using your components into Delphi's multihelp environment.

Typecasting Explained: Part 2 -- This time Brian Long covers message crackers, safe object typecasting and assembler typecasting.

Customised Logins -- Xavier Pacheco explains how to customise Delphi's database login features to give your applications that extra zing!

Delphi Internals: Moving Up To 32-Bits -- Dave Jewell shows how to get your code ready for Delphi32, and even start making Windows 95's 32-bit calls with 16-bit Delphi!

Please Call Later... -- The first of a three-part series from Brian Long which aims to illuminate the mysteries of callbacks in Windows and the Borland Database Engine.

Performance Optimisation -- Bob Swart is well known as an expert in making applications go faster; this is the first of a three-part series in which he reveals his secrets...

The Delphi Clinic -- Answers to all your 'How do I...?' queries and all your 'Why don't I...' problems -- Dr. Bob collates the answers.

Tips & Tricks -- Tips in this issue's column include: overcoming Windows resource limits, a technique for implementing huge arrays and hiding a window's title bar.

On The Disk -- A run-down of all the goodies on the subscribers' free disk with this issue, which includes all the source and example files from the articles.

Return to Front Page

The Unofficial Newsletter of Delphi Users - Issue #9 - November 9th, 1995

Using Integer Fields to Store Multiple Data Elements in Tables

by Steve Griffiths - CIS: 102523,27

I spend a lot of my time writing state reporting database applications, for example EMS, Fire and Police incident reports. In the past I was limited to using Foxpro (sorry!) or Paradox. Although I could get the job done in a reasonable time, I was never happy with the appearance or speed of the finished product. In addition, adding a 3.5 Meg runtime to a FoxPro app tends to make installation a real pain!.

Delphi has allowed me to become a 'real' programmer again - it is easier to code a Paradox app in Delphi than it is using Paradox directly, and the resources and components available provide an excellent user interface. (I use and modify the InfoPower Library a lot.)

One thing that all the state reports have in common is an enormous amount of data - literally hundreds of fields. In an early iteration of one of the forms, I ended up with three linked master tables - 600 odd fields before even thinking about the detail tables. Obviously some rethinking was necessary!

The bulk of the fields was taken up with either checkboxes (Yes/No) and limited answers (1 of 10, Yes / No Maybe). By using Delphi's bit manipulation operators in conjunction with Paradox long (32 bit) Field types, I am able to store information for 31 Checkboxes, 15 one of four types, or 7 one of fifteen types in a single field. (The last bit of the field is not used as Paradox integers are signed). Using this technique has allowed me to fit my master data into a single table.

The Demo

On the left hand side of the form are 8 checkboxes and 3 edit fields. On the right hand side are corresponding objects that mimic the left hand objects - checking a checkbox on the left will check it's mate on the right. The edit fields can contain 0 - 3, and again, the partner will update to the same value. The data for all the checkboxes is contained in an 8 bit integer variable (TheNumber), the value of which is shown on the top of the form. Similarly, the data for the Edit Fields is contained in another 8 bit integer (TheEdit), whose value is shown on the bottom.

Checking a checkbox calls the UpDateChecks function, which updates TheNumber. The function in turn calls the UpdateMimics function which uses TheNumber to determine the status of the mimic checkboxes.

When an EditField is changed the UpdateEdits function is called which updates the value of TheEdit. UpdateEdits then calls UpdateEditMimics which uses TheEdit to update the mimic EditFields.

When using the code with a table, the code for the UpdateChecks and UpdateEdits functions would be placed in the Tables BeforePost Event, and the UpdateMimics / UpdateEditMimics Code would go in the DataSources OnDataChange Event. (In a non-demo situation, the update procedures would refer to the original objects).

Now, let's see how it works...

Checkboxes

Binary Revisited... A checkbox has a true or false value. This is a logical value that can be treated as 1 bit. Therefore, as long as we can get at the individual bits of the whole, an 8 bit (unsigned) number can contain the status of 8 checkboxes.

Unfortunately, Delphi does not appear to allow direct binary representation (if I am missing something tell me!) so we must use numbers to represent the bit position. Here is the bit representation of an 8 bit number..

128	**64**	**32**	**16**	**8**	**4**	**2**	**1**
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

To keep the code short I have placed the number representing the bit value of each of the checkboxes into the Tag property of each checkbox. Below is the code for UpdateChecks:

```
procedure TForm1.UpdateChecks(Sender: TObject);
begin
  with Sender as TCheckbox do
    begin
      if Checked = True then
        TheNumber := TheNumber or Tag
      else
        TheNumber := TheNumber and not Tag;
      Label1.Caption := IntToStr(TheNumber);
      UpdateMimics;
    end;
end;
```

By **or**-ing TheNumber with the Tag Value of any given checkbox we are guaranteed that the bit value for that checkbox will be true. Conversely, the **and not** operator will replace that bit with a false. Below is the Code for UpdateMimics:

```
Const  {represents the bit number as an integer}
  Zero  = 1;
  One   = 2;
  Two   = 4;
  Three = 8;
  Four  = 16;
  Five  = 32;
  Six   = 64;
  Seven = 128;

procedure UpdateMimics;
begin
  {Sorts the Contents of the Field to the current Checkboxes}
  {Typically, this code would be attached to a TDataSource OnDataChange Event}
  with Form1 do
    begin
      Checkbox9.Checked  := Boolean(TheNumber and Zero);
      Checkbox10.Checked := Boolean(TheNumber and One);
      Checkbox11.Checked := Boolean(TheNumber and Two);
      Checkbox12.Checked := Boolean(TheNumber and Three);
      Checkbox13.Checked := Boolean(TheNumber and Four);
      Checkbox14.Checked := Boolean(TheNumber and Five);
      Checkbox15.Checked := Boolean(TheNumber and Six);
      Checkbox16.Checked := Boolean(TheNumber and Seven);
    end;
end;
```

As you can see, there is one entry for each Checkbox. Firstly, the checkbox bit value is **and**-ed against the number. The result is cast as a boolean and used to set the checked property of the mimic.

Edit Boxes

This is a little more complex but is explainable. Because our example allow an entry from 0 to 3, we can establish by looking at bits one and two of the bit representation chart that this range of numbers can be stored as two bits (3 = 1 + 2) ... Bits 0 and 1.

By using the SHR (Shift Right) and SHL (Shift Left) operators the value of an edit box can be moved to the bit 0 and 1 positions, compared against the number 3 which represents bits 0 and 1 both being set, and dealt with from there. In this case, the tag property of each edit field represents the number of bits that the number must be shifted to reach the bit 0 and bit 1 positions. Here is the code for the UpdateEdits function:

```
procedure TForm1.UpdateEdits(Sender: TObject);
```

```
var
  TheValue : Byte;
begin
  with Sender as TEdit do
    begin
      if Text = '' then Text := '0';
      if StrToInt(Text) > 3 then Text := '0';
      TheValue := StrToInt(Text);
      TheEdit:= TheEdit and not ((TheEdit shr Tag and 3) shl Tag) or TheValue shl
Tag;
      Label4.Caption := IntToStr(TheEdit);
      UpdateEditMimics;
    end;
end;
```

The meat of the function is all in one line. Here is how it breaks down:

Firstly the contents of TheEdit are shifted right by Tag positions. This aligns the relevant 2 bits with bits 0 and 1. The result is then **and**-ed with 3 to return the previous value of only those bits. This value is then shifted left back to the original position. By using the and not operator the two bits representing the contents of the edit field are set to 0. The contents of TheValue (the new value of the Edit Field) are shifted left by tag positions to align the bits with there proper place in the number, and the two numbers are **or**-ed, which places the value of TheValue into its proper place. Here is the code for UpdateEditMimics:

```
procedure UpdateEditMimics;
begin
  with Form1 do
    begin
      {Sorts the Contents of the Field to the current EditField}
      {Typically, this code would be attached to a TDataSource OnDataChange Event}
      Edit4.Text := IntToStr(TheEdit and 3);
      Edit5.Text := IntToStr(TheEdit shr 2 and 3);
      Edit6.Text := IntToStr(TheEdit shr 4 and 3);
  end;
end;
```

As with UpdateMimics, there is 1 line per object. For each EditField, TheEdit is shifted right by the number of positions necessary to align the data with bits 0 and 1, and then **and**ed with 3 (bits 0 and 1 set) to ignore the other bits. Both the Update functions can be written more elegantly, but are done this way for clarity.

The Editfield range can be changed by assigning more bits per field - 3 bits allows 0 to 8 etc... and changing the shift values accordingly. To seriously save tablespace with 1 of n choice fields, assign the itemsindex property of a stringlist to the table instead of the contents.

Feel free to e-mail if you have questions.

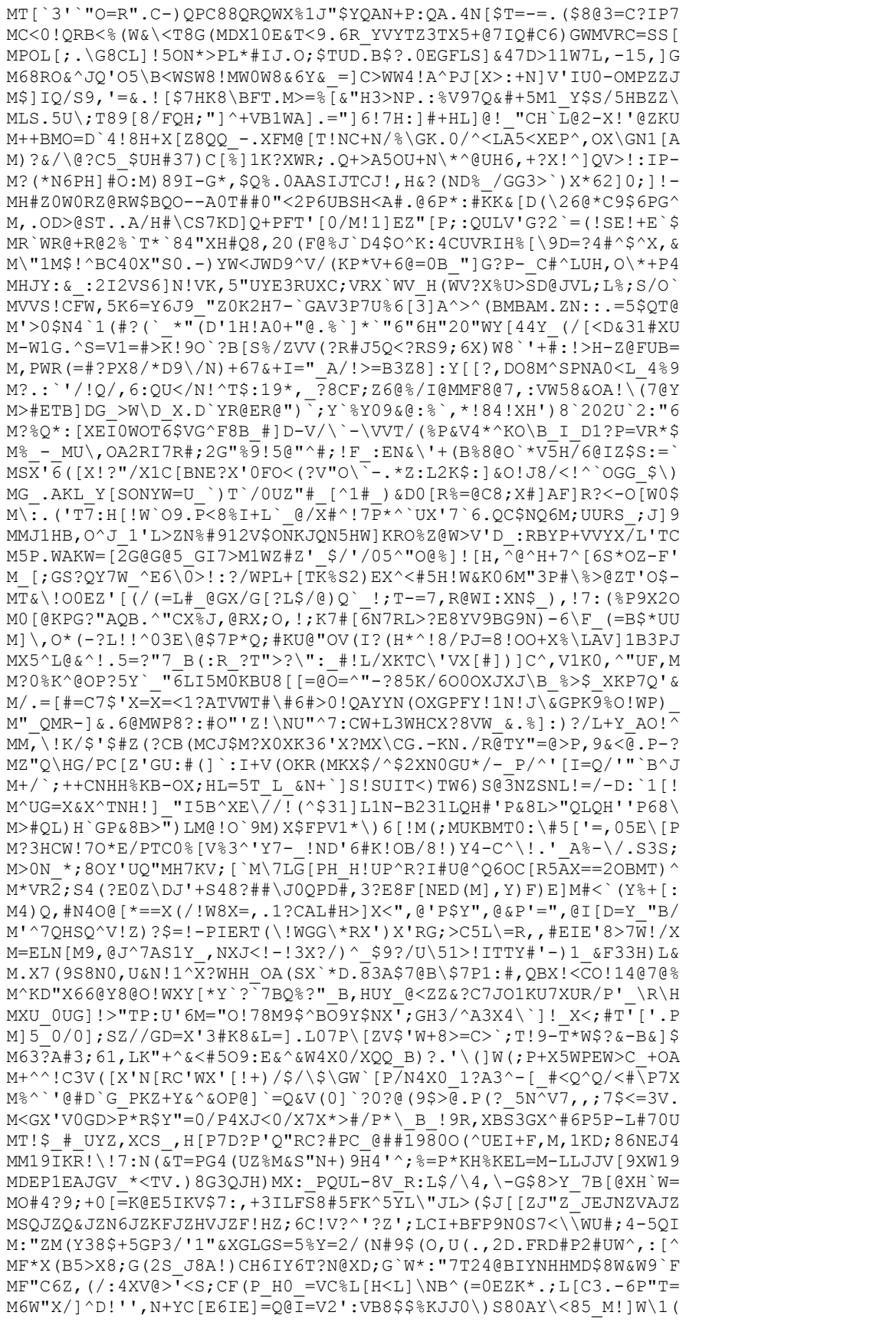
Return to Tips & Tricks

Return to Front Page

```
table
`!"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_
begin 600 SLIDE1.ZIP
M4$L#!!0``@`(`.==81_DO.^"V@\``%Y````,````4TQ)1$5"05(N4$%3M5M;
M;QL[DGXWX/_`A\%$.FX[D9S=ASCQ0I;D"XYE&Y(2R3L8!%0W)1%ND9IFRX['
MR']?WLEFMVX^.P82J<GB5\5B5;%83:T(SL$@Q0FZ@-G9X<'A`28YRJ8P1N+A
M[6]],+B]Z70O6OV3?G?P6S2N&&*'!P`,7MGW'*<L`B-,AJ]+I+X]9#3FWWJ(
M,3@3;>T4,B:^7&5P.<>BLTU)GE$^5.!<TFS!VSH8IG0F1Y(5D[+D'%10#+EP
M@_PU18#_?0.U";NE+RA#231A?8@92NIGDNX^PXCD,,>4"#J:_4!9CF.81C2[
MIAG^-^<+4TT\G*\6$PW+B7/&N30B^=%4'Z?JXS/_:.,L3I'H'OQK!3/^36"$
M?SGCLUA0D@A"_;7IOIZZKY^U$$;W7()8Z*DV;*]83A=:0W5!M,SP,\R18GAY
M2>,5:].49D7>7\!0MIYINFM(DO8J8V6Z"TI3!(DAO(43E++R9#C@(,\PF=UB
MEAOB'OP57?8PB2X?*,-2T9KXAN1HABQ[?RD"5*_+4-O5+<M@UMZ0#N<X?B*(
ML1+IQ6ON4;G%#0!=ER7FB*R2>Z"J>]*>0S)#%6+>T1Q/7[O/?&::6O*Y6"RC
M'F1/XO/B:9;P3SDIG"_@4A-V,CC[`=/H&J;3X4A]7%<I52+V49Q7\!?-A@PM
MEIUVY7RN.VU-),SC@4IO+Q$IL]&$]QF>80+3DBV%A,N,QBA990@,4*Z,JM8J
M&%&]BK('?PDR/=-J$DRVD7@V)9EZSY7TQGJWX4H[D8C&#BO)K%$*4F&':ZB,
MY4E$]UA-+:Q2XBDC+!.U>UVQ?K5GF)EX*Z)`[XKF,DK4S\!"-7/2G]V[8;=?
MA?$+YR4(T1@,']\,2Z-'O2N4=])9FR8HP/"[/*11[^=5=]BYO6K?=[H5>`/\
M[S*2:.08_I^'-[CYWS+2G^BU0U^(A.+?OX`1S9(S,)CC:2Z,4GP.<BA6"M!G
ME&4X4?&`0^0HSE'BG'.&R6QK<'"L'R`F>8#J]_?HBJ'OR]K%*L\IX<+(!O54
M)>$9&$?@T;/3+=!RWO\I\![OKKT#9KHBL=H,[M"+];[1'&5H',D/;W08^-S@
M&R:V]/6Z[Z,%9ZQ"+RQ'IDX&7S9W9HC$\[([PF=T`>.G6497)"F;K9#_AE6!
M"D>^]3?F0M=%WW4M5Y,4QXHHIH3EV2K.:0;:&8(YJK7N7PC*Y"Z_6%*"2%Y6
M<H+LJ([X2E_7KT-[E?&YJD#]!:@@[>1@<^,`7-XERO)7T"5PDB)O\JK9Y2,F
M`P%<X,1/5*I2)?WWDN$<%9*:!$WA*LU!G%ZD,'X*V;FTQAJ!YN=ZMO/SDB/#
M;YBM4,A,[6*%/4PSTTF3@K/[73B^!W]94]8#>9,;)1X,_\:GTFA,2J,Q\4;S
M!SLZ'.QM@,7M4"-Y+1O4Y9CY"(:IGU2'_!^@L*[!G+Y<8YL2N5X=`,+YV:32
M\K4M:V?Z0)>KI3@TA!WKF,LMU]O1-6_UW3)6CX:K/7*$8$,X&>1TN=&,.,U]
MEJ"LU&ZR!I4S:#E<@FME<4UKM>"2B4)F83%-PU;?*.0IEI\]`97XBBPE=$65
M4#L\^;A)0S\PPY.TU&SR[6)^;2S8)./:I\US&41F217-OW!>;M6)0V7'0X88
MJ^SYKI)Y)#>'PP-_0YIAIM@?'N#%,D4+YW>'!S8<]S"Y(7FM%5U4;X,3CB,]
M]68*6N`<7(#A')EAX,LWWM!-&?(:6GR4D<>Q@;_>Q48.DZB:C6VX\-A43MM"
MFC:[@;':AT$.20*SY$,$_F$/P_^L>YBU/TI_',P=G'LHG].$'1Z4Z>IBO+^5
MVE$G&S953V),YD@85U+<A%4F?HWP;"Y5T/@OV3#"23Z7SY]4+#<G0-&FCWR:
ML>S6Y\(UO?:T6-GK']\XQ2V%>C^K7=_P^4(2HPA\\*@^**']G?;+M\(FZV^*
M@JEQ4[/7B3:W#_K"R&U)S%L_P5]*"^K1WSEX<VG3<$'?0:C@PY^+<=>+C8Y6
M11=?8INS\\9+F#+7^@.FHM%;'LNI&.%,6'>PQAZ]',N9D\FV/-O1:CNYS!`J
M6(-KT0;@&O2:NP9G@(Y#V=D\L]YZ(JRR[C/EG=,,L?EV!NN/BWM@VX#DH,NI
MO0M&LDLL1JU8XG,5OOH6P?<X#QX><"+!=@(J0^-$"*(#:2.J\6_'/4SJ(,'/
MH/%)^5D,F3R#`CI5V\6//W\^]&_N^QP23T'-V/SQI`[.1=`&^1Q596*^<]CO
MQUP&),*PW\M!SBRON^YX"`)>1YS75R'X7KR.*GG!7XY7]ZX#-"^]4`(_E&T+
MR/5]K[L-A$N^>=:WW4LS:TOCM+M.EPT'\/T!@!#`J6R=@CR`_LW5]?"O`'3N
M1W?OG8*R_TU>L'.]QC-W37+21TRD;YRQJ-[\'+7NAJU^_WXTD+R#,B&GXG"R
M]"\<K2;VEA1%X*H]^GG=_MX?W/?K6V7=4`OR0\W4;,7G>@MVYEW:>TRX4(:T
M>6O:)-RF$F=I"V@QAF>DUJIO!RV7.AV:WF5;>X3KROJJAZAVZCT1M]1:'7R@
MWCW9K"G1^NL>LPX2JA5[/2;`)G$RD'MF8,>(/Q$26^!<AHXZ@"01CU]E>*FK
M;+>0D;3.S$`GN7CBTDN?2QFJ8%)9A_H+O$L5)E!=DP)5)2[)5QDA2FKVH%17
M(<4^UP8HG=;=]+8O475UW!F`3:[V7/K*<KI_-!%*!)^<!L%_U\/E#I+%UIEE
MO]O<-A;K?5F\-/+K.6BM,[O+8KKIEE:F"EZGR1C4"T7Q4E+7F[T44H53D_FK
MXX&7^7\0HS[4SP*@YKN`FF6@TW<!G9:!/K\+Z',`I!/W/8'TJ`!+O][=$TN/
M"K#,R^#]L,RH:K#FN\":U6"G[P([K0;[_"XP?RV1%\;V\`ISC-K%*03M)L?8
M`ZM9C77Z'JS3:JS/[\'Z7(%5\)!=L?2@"KB"D^P*IP=5P!7\9%<X,V@]7O,]
M>,WU>*?OP3M=C_?Y/7CAZOH^HV\.B)J%]4*5&8NC:;-(-RK0J<2Y2%;*=W;8
M-"O>6;MTP%9J6KM7!=:\IO*SP)H[3?<P.6K414:E4^\V79%<YPGN/*7S+[=3
M^USD$LC1__!Q_UG,]L(1'[Y^)\<=-,4$)><?-L]ISS>@NAIQW8A&C>J*Q%L;
MIO$JU4DO(`AFB.5@:2:<4_`B&,C>A7B-"S`#*8WYB.2W`A>34-9RK.Q(U31E
MN[0.U3PZTUKW3NE*(^Y\V!<)::U6NVX<JWG5/UXWZG_82LF1^*^HSM)@I9>/
MH](X;3J&7-?-=3E&M4>"+A(Y]1:+_<MOXNWBM**+J!UUHF[U`N$I4*@B15Q,
M;M%4US>0>0?1]D_TQ^H\W_';9$O7;SE25&\78C'%RL9(UH/I5#Z90\#O\H*!
M"^O\XM[0R9`NCW3PD.?BH%N(>U1<?5MB=>O?$H-\TQY'C\$J5YF-;'T$7SE3
```

M*9J$Z2HY9,]YH:>M>N3WCHY,%E]BC0.LML4:!UA='TO.K%4VJ5;!FM85E\-3
MVU:K^VN7-#S3&B!5VBT)]Y#?$+%\-;N0D7P7(-=%S>6F:!5>A?Q1J<9K&>\R
MJUUOAQP>_%!^<PM9;IFL\1SOO80SF\&6ZE+DOR()S'#[V&(U2RGK[49Y62J<
M5P71B?+IA+X0N<4DGE?P?ZO%!$Q@]C_&`VN,2<_'1*VQVI<J7*EP<"RHZ)M9
M$O]HO]<:JE%OP[D(&F*70"E>\+GR[V!%"(ICQ&#V"C*T%/>8V&^/D8'Z>EZ0
MJE29#*/`SBE$J6SBMKB'%,9J`YO8H@>8R#1);VXX!R^0@91+)H5N0_(,V8DH
MB-2*P2PJA#YS,7/;;A%45G3D'RZ6XC6=OM79T[<Z!7:C<#/3SB3/7I4ZWE2Z
M`"#(.`DDLQ2)G7H!LR<YSP0O$&&8$F9"NC2IB0[IAHG8,X6;?XH^1<5\+@K2
M0+,.;^K='6?\0K,G87I*CQI6SZCZQ:/I/G&O/P,N13+[4K0HFI6$+E_#186<
M&Z`DI[)C@X@G>H4%2(\F,GF.%X,L%@UGZRBELJ3JS,+KWD@VUDNB+2![6B</
M>,%\>H-^NW77V4VVEDW8-XIFC@6;13,&L56\A];-W7`W`=7UQ1U$M"M:*>,=
M?1'Q*48"[4((:#;H@H<?<BIB,?I:T$N^YYTAS<??@%5NVX7QG/@7>CL(QU7
MI3AC-<''K2FW3JTID`&S.%]P#%A.17`B+J'ZO2Z%+L1]*9UXY:FS\M"9/]J\
MV$;UF\K$S%X=?_3V#YWDVV.DET/YU)JW2LJ%1'\4#EOU^G&U]_L8.I'Z%.G$
MJGH^6MOU>A%A+%6@#B#%$%(O'%<WE^.M+BN!*C2)-VMR7-+D:(,FQ^Y<4Z7"
MB@F'*JL26REL'(Y_W&0SH<J<M<N?%9BMI&#_FL]1%??'HXHM9EN-P%Z'=<[$
MF]69A2X_RA0IEC=4IBKE`[F\E7MB?S<$,OFK'\"[4W4]!#!1)53)RDC$/!5`
M0$+MR4/5$;TK)3IO0>3$NP]SE<%7_9:YV#&:XQQMGYV[T5N>W07-<[KXV)<;
MYW]TANI74?__$RQ<D=91=-R('AO1N!D]-H-?K/CQU8^NBV4D2D$/E*V)K3(Y
M=?R9N#?W*G<3E8*SE`J-PC#8ON`T!7D&GY%(2C.ZFLVEOMXPB=-58C-SSOQ8
MI%GL1+QTDT-QSE`Z%2=IPK$%*W(BQ[X4=.V'E:#J$M.,H,SE:E)-($%+1!)(
M<D`)H.Z%Z.]-)^)"[!(:=K%+>7`='(.:=RO4M#6\.N.X*5/^!CAR%TV]NP;B
M[U%5?5RY1S<WO=`?./C18\.O/99/X)6R>\4#7[BJL'8T+DM8\\N8N\Q=S>!Q
M_=Q=V51X@7TKMV0#FF+391VWZ*M#N@1_!S),H62&S`FI=--?'(:'M"8\PZ9%
MMYC(MD9U6[.^F;6*()R[BB$!>__7!`ZVZ<%Z;8WC(%>C`((I3E.4@(FX%.B=
M2M3M$U-?PE,MF0Q58`X9ER]>&4&DPTRR%9L+3$XLRQ,F&MF`XUU$E-'(BT0#
MG#[;"[O!^K139-]KOW5,?BGY@SF>S5.A&.]\)"=0&S>.A,:/1)P2W_A_]1W6
MWSDX6\)8A`_AW#A^DH<T,V&37)@+_'S*JM(=%+G<3;%S\"G(*JKCP'YIH!M5
MS$'V3G^"W#V<;^7\A(4.U:E17E:B\M*3V9PV3S$(&3*5,?O#-_#8/!YF*Q+K
MW(DST9FG[_!&+B[!-W?=RD<)J3UWM9[:,/N2_+&9]<OCIFLO@1BG*X,(&ZL"
M<;87A-&JI=NLFW'CJ%(W^VAFW#S>03=Z@`Q;3C>N];BY@VZL`A1,$42VAC`F
M3N_P]BF\#A.D$XSWL[#4@!?B-X2,BMI1#`F8B!],R`-;`H3?J_.J*1-LJF48
MFLUED6+!H7B@WJ6`4_FK<_.GS\PVG=^6BYN#OE54\`.XX,Y1Q7VCH")FF,EO
M)X<'_P=02P,$%``"``@`YUUA'_5``O,1*@``D%@```P```!33E14)!4BY$
M0U7-?`EX5$76=E7W[>[;G>Y..CMAWR(80"*(,J@$"#&`2+,D1&48`@02E@22
M@#!B"$N$$!P=T1D<</W\`5$0'9?@,C:(R*(CZGR(RB`P@NP&1`B9#/G>4U6W
M^W:"1K_G?_[GCYX^5>>\=6J]5:?JUB5]4%8JP]]1-H$%$@XDZDF)XK^-29\E
M'4AZH_7*R/&M=C#=2YC.D2VB=U@/6R9'LE_\5\:2K8RMY=LY8ZTLC.VV?&TY
M:VFPA!!#^"HK\9XFV6,BS",=E@4(^:R-K?:SCK1.M3Z@Y%,L3?-MRS=SJA7]
MV?31,PHFYPW,+6;,<?ME1F5A-OOH^26E>3,9Z^UU0<`%;GY)5FG!C!+&TNK;
M\LM2.+:@<,S\67D0SIO0F[NX(?07%TV"\/O+0WA;*1R>5U*2.Y60`_WC>6\A
M=`R:D5M20K(?#Q3S(1)X1W'NK/P"2GWLT#(^7@H'%166%A=1YNMVK.7%0FC+
M*"J>"<G'&9OY,FDOO2!W1M%4R$Y]M9VOE:CA>85S(-EU&V.;26)QCD%M1Y?.
MGY'W'3^':D2!_L%ETQ`CJ1[53L0I#*ES8LF=1??E%>=-E@(!U2>6C,HM*)$R
MF=[B'C.BN""OL#2WM*"H,,-RCCF%>:/_"$52PSR%_\&YJZ@X.Z^XM&!2[@R2
MT!_:UUU4G%E47/![5%W*51X18\;DSYDY4=3@!8A](HO'5!9.T`NF+"B,1K27
MEJ#6J2\H\RY#<N,+RG"Q(>E%$FE,27J31(XG[BPM&510/&E&GC"E!86C9_)
M+99"FQ2Z2DO0'3.+"B<+J;VQ5.3L@`Q]%Y**W'5(>YNEH@1.67WG&&/$KK.>
M8^V9L[O&Z"G98:6J.8?+,LF8SJR.%N(9.F&ML[[G7\7+A?'O&V1CM1>_7K;(
MO=&;Y&+>+[QE7N;9[V%L241&1M&D.26#BF84%>L'82]6F\G+N-!DYA9.'C2G
MN`2:P]#TT!9PSNNL2QP9=^9.S)M1HA^!-%V+3"@@O)8Q/'>>?@RBL1IJI40%
MA?IW$$T3HA[:$F>&OZBD@(:-?A+RQ4(^5EOBSC"-*/TT5$]J<I`LL6>(,:"?
M@?!5C49ANK;$E3$&3_T0CQN^EDH]FKM8"A60[%#PT8_!\TQC5IUL08[8Y"B
M1/\>PCI1EU>I/",*!^7G%D[-TVL@C[;ML(K:Z,+*P)FS]%J(N]K*'&5\K[;$
M,3RW9#I)ZR'M+Z1UD`Z</G4R21G?8<T2TFF0IA?G3LW.G:%KD$ZW447[VY;8
M,W-G3!DS5K=#6"Z$3VI*F*GK$#YN4ZWG%$48E3>I5'=!OLEVBY;"HV%A3-[,
M6>F#]&@(M]LR`2ZW+8F@SO(7%126YA7KL=`<%)K';4N\:-BI!86Y,U17QD-Y
M12B/:4N=H_-*96?Z]"A4Z;IHS(\>&CL=[?))IM[E76U+[4"B@WUZ(F"+@K"A
M"D9EYED*5E#HT]L!-B8(FVV&#;4O]0)FZG"?W@7PS4'X,PHNIH,KMJ41@!OC
MQJ?W!/;K(/8CL^F#MJ4ZL*+W??HM`.HQ!O!'!13S6$?[4C>`P4'DT],`OBD(
M[N"08!I5?+IMJ4>`C8'ETS.!GAA$^Q5:S#*S[:(,8K#Y=#^`#P>!\Q601A_?
M9%OJ&#1\,'69SYT#W/ZHI$B)>U;A7FJ5Q6T_VI?:@9M74.IS3P#,Z6NI8+O,
M,+]CJ7OL\#OR2M-G3!U4-#G/Y\X'>)MOF5%_,W@[NFKL\-$%OP=L%F`7?38%

```
M:Z5+V(DA@.UR+'4,RYN?7G1?H:_5/.`*?><;)&Z(P%D8C23;/#PU&+.#=!KS
M&&U3]40,LPDZU7.^(]WFS\7(]"66P\)_)2Q0.2W2Y?0T04]W#"^:4Y*7-<O7
M:CD@,V)K52;5`J*QMWRK.!>9B&X6/QT<Z4Z13);N$23L$%>G$I[_F82+=)5P
M>-'</%^KU4BX/.Z*2MA9S+]6UBC-LX[TB+OR[@L-PF=H[@WV:[:34!862G!>
M3[</*:'USJ=O`/96GX&=YUP@'O"/[.G>47DS408YU^06^_0M0/X09R"?DVL!
MK*>[T;#WA7#5P$7%&[@]"E>MI[L$KCBO<%*^3P\`=2J(JE&H'QWIWM&Y<[&T
M3)H^M;AH3N%DG[X+R`<3#"36"/'7"O;&YL,?&%(B<MT'5';07II"U3C3733>
M[Q2+B$\_`-26(&J*0LV3J(&C%(I6E(^#J)4*]8Q=LP\JSLLMS4M-/`'$LJAN
M"O%7EQR67A?Z98BN.=+S2N`KS1^92//VUJAZU7_[E:7GG)H;,QY:0DYP/C&1
MOQ/LL:NN=QCU0F>GQS&X,'?BC+S)G/JE2\3UK*$AMM4J4B(%*^?*W9CB\KA"
MBR6G93(KXB#`1`98>!VB&!Y7:/T4EA=''`;PL`E,/`+4)<)CE[,PI^GVQ8@C
M`%VQ28]%62UWBT;P6#$)BQ'V:<0QH#K:0RC*UR,Z!JB"0H&Z'/$=4$--*"Y<
M`<8^C?!$F*9@3G-M9_=IH&<W0D>*(>;Q^G.I-4?G%]V7B:=9U"G;/00)_M(Z
MO+6B0(LC/+KQM(B2+'*?!/291K9]],BY/4Y_T:PYL\B'Y<_$[.6;W&G`IH6W
M:WFTZ#:/'E:"`^Y,P`I1@FY)J[@!13>S36Z/3<S67'BSGC/`?60/:U06*\:>
MQS$F%Q-[T2QA\7K/\]2I?<+KA(>2O8@Z`3FB>')>,:^/7L6S/+UN6L6WF*`-
M^,.H9M=[/,[@XL)I%5GD.0NS/S:J/YXYEA7A<876%DZ+R&;/.8`[.$)@\O<2
MR6..\-C$TB+*NM_S/7!^1[C1%C2Y>CR.[(*2`@QL@63>KD"V;!->JR3*WN/1
M#2>(D_N3XJT!M*91Z[?$SP&WQS&B4*Q8`IGEW074KD;(5OC9[/'8@<2:)8!E
MWGT`[6L$;$US&SI_1*%:8_B^F%5\HW<+<%L:8=M02;T>E\#ZBZE5;;&K^!?>
M:@"K&X';XB?%2V4%.&N6,,M8`*A`(Z3</6CZJ+RI!254+9U:\H$HFB:T9C:Q
MZR,;+-%1_Z\VL6<MFAU/]9#"4@8_D;'/(T-E;+SXV-D_(_^CMNA6/J"]G:V/
ME"B*#VRO!>-G(ZWV47DE<V:4\JL-(1"RRITGLHI"[/:HG\LJ,RH_RIP5-8DY
M*R.>$V7*RA#:@S65H77"E-4^8,1]A7G%[=UB9:`Y_X]15FUTWHPI`^Q"1)LB
MF>)]F<)0_C5,>4DJ'6I;WMDE/##R@`Z'[)$HE*2KKW&29U62&%\PR;.F)'?[
M,GQA1<AV2B750U6XH8&$"U2%7_$])5-8,3P[NX7+E2E;S#8ZOV!*:7N7D)$/
M,B^4*8G([@J?E4]$&U(\U(9'FI1[ERKWWI")76%5C8UNG(1\1'+\KH:2_!A,
MD@KX@&BCHUWL"K;`<O,7!)-(@L<#6&@"TT9!C0H#3"()K@)PC0G<L2FX8Q!<
M#>`N$YCV%G*K&`3/#H*/`?BC"?Q,4\O/!,'16#8ZQ(3`M+60)R%!\$=!<'\`
M1YC`M+UH%V[YQR!X!H`+3&#:7KP07N8.P79^`L`73&#:72P(M^PW]>.7,3MC
MPL;??E?C\6=S"2%Y/)3BIMA[8X?&DEIX=SEXS,CM5>TBA7<C8T/HB0W:[JS&
M=FV,Z6$V);9DIO+ZH*!#K-4R-I77!05HX[BCL1=B+7&4T<`YI:5%A>VCQ$Z`
M//70`^`5LGEB,V[E5,)%>FA:H<(9\;FAPBU2#\C+L6@XE,N`5"$^$,4RXN\A
M/@BE,N)/(Y[.:X/QSQ`?S"\%XW:6%S<@;DY<5GBIJZ]1ZNI&I:YN5&HCWBTN
M6.IJW>C+-^(>C?LR;EV<V>#Y1@;/-S)HBAL&SRN#.^.L$7?FEI2J<P@TR/FP
M&>,_<6'#9I[3/#<,C`]3/J>42?%6^YB9LW#8@?8D(1UV=(^WVNB<(I5?%#(Z
MJ"`K)?$/QJ^)-UE)"HU,;7#NI'ST`8GJ,2XGD&%R@G+0$20T1J,0WHW>,(2R
M?&^%ER\M;-K_(EPY):CLFC`L85:"(\&DW!.:K2TYJ6BD/:;1?#>-9D.0F`#$
MC2AS&.)&E-<0_"9!U>N"$%&]+L9;K6@P7A,$C4^P.LB#@W?,SP6ELN"/)(05
MO":L2S:%*R?H1L$9([>3"0]1.C3E:E%]1/'EBJ]6_!G%-RB^1?%JUD7P@(K[
M%<]1?);B\Q3/9W+I[Z(.E@\H^6'%3RB^3_$)BM<H7JNJEJC2M^,L=(1,)YN*
M1RF>IM29BN\2&QZ,.+%#BU*ECV(]!4_!*2BU2R_%YRI^$,ZOD%MS!$]A>A!/
M7E,?V"%_<S"O9MW!)\(^M407M.\DQ7'6SI*!7P+^--_%'@6_&?)-X#>"OTL;
M&^@_IEJA1`<5/Z-XO6C]GLS-99Q:L!OX+>"#K6G63"6?`_X80A7@7Z+?_@SN
MY;>P3>!3(-\+_@K:Y2!X*O"7P/_!#S`[G,E9E@#K"CX&Y1FN^'APOS:+%X&_
MB?1;+;*^'X&[@Z-`]18/OB'1F.G-2`XSO1')9Z8W(>V8Z0U(*0N]^0APTQN/
M@\STID-CQAL.C87>;$"JWFAT<@3?W<`#_>K>PK;NT7<.21\\<,"H'OX!HY4#
M&9*-&CR:!#X+>;-3$)J*TQQOO(,\SNF(SI!1'DFCV<EF4[2UA7S*^=#^'CNR
MQ!BWNW7KUO%N>VN'P^ZPX\_A=">09TF]O`K'^:Y(_'DBR7-<`]%:E-<5Y=+(
M+7R6#A]4G)R^]:*CD8G/00[;2^BCES'*O*V2HCM%=TKMV;9MV]9,(\?L/?'@
M(65DDD9>UP>([Z2X)T$CQVHOG7;"DBM!(]>)GJE/21WAT,@[VD\3GXJ3`T0C
M[I\J3CX.;2+^A=Q=<"Y=Z&U76Y?+U9YIY-)<I',AA26OI0[Q?U/<':V)HU#>
M!7-9>U2*Z?'MZ:\=_MJP%J9PVU8.II%CTA%#L!.7QLCMZ,H]['I,<)&IJ2Q:
M(U<@E=.DAS<1+-(=D<GZMG;04ML?TH$\FGF=;MW)ND<@U(4EMW:Z-%HBLZ#-
MYC84J6M'C1:R<1#\CGN9-Z(%]K$^IG?$`Z?1*E8(31$5@'74:#TJ1>;S>`+S
M>MGM++Y-&Y;44_W?U>WB&BTLC]!CQA-1(A\;$1')[NW6&;LR"J5VZ\R&:K2^
MT".W&25`EKTU6E->@^!U0T`3^=NTR^2WT_D*9*R-5V^+AHYHV])+[1X513_1
M+#V*M8EBD5Y?Q$@V`H//QP;$=$]%=OUBNZ=&,(UF?;%)M3A@I&/'WS&-YGH=
MIIP6._H/?Z:]Z5'Q8L_-'!;L)$`]+1YL)3UL`>@QBQ=;52_[&G06U`#R6;W8
MS'JQ^8QD(T%300^`5EFC6)TUFL5J,:R'%HNW+'%L+&@::+$6SY[4$MBK6B+;
MJ[5@Q[0D5J>U9-&V5JPKJ#\HR]::30>5@QZWM6&;;&W9=EL[=M#6'ON"#O#@
M.\+E[P3GO#-\[F2XTM?!0^X"Q[<K_-GKL0U+P;ZJ&_8HW;'IZ('MS@TX9$[%
M.G<C/+M>\)-ZP[6Y"2YH'VRQ;H:_<@O<DKY8P'^#A;(?G(M;X0;<AM7^=FP1
```



```
M"R:OY8@K!$VT2<RXC"/??H2NUQB7:X)7<.1KC="M&N-.C7'Q1KY."5VF,:[2
MJ.LV\FU.Z/Z,<7LF=,E&GO"'+LT85V9,5VOD*Z#0;1GCKHQQH2:K\EBUA?V%
M-$O.6*HR[/TRM#*]*D.KS+AZVHXX>%7&U=W;-?934+N":B&H/KH4CT-N\61*
MHQEI-!R+>\?BE<1;C%5FU%=F7WVHK#Y^[#2IW[T7X-2`")>7N5BION2,+F(-
M<[T-<]VI`?^.,KM]K!!-BS2%)X?"6[A?VA_E%W%HDBLK:EERY>-U/Z&L)^75
MGU`R#J6%-_019_IMQII4L3PU`'80R2L.L>2=%8<9O[:1(Y3#OUCRAQ7?82"!
M'<.<@@2G8?W#BI,DVUF!MTGX/0O1SHKOQ2_>I(D<-%Q!-ZK*0U45;>1X&$?\
M2W901U5EZ[OWN%BH$2GS'>\>8?:QIFAM>+0^/(H/*&0TK+%%3C;927OLIAS0
M*PUS72%@*+1[C_YK<*9Q104I?_<T#?TY/&UY1L.*C(;=>[0F<WUX*M$PR55H
MV>04M/(36UWL\A7_"1IG5W>4N9.K;-W;L]L<;*[UI)]7OO_IOWHTK)>5KD('
MI&1<3>Z'9&711M2ORG>:QEERP^O?&>UTTL>-5%:1ZAA[(-J(-DYU+)CJ:U.=
M]4"IKY'5I@DV7C/!,?93Y3K]>Z-8LB9>$<X,ENA>DQI%ENJTH/K6GU>W_EGC
MC8>=*_G#P3H-],$N<U_[IUTT/13Q/'R,B!P&=TKN-[A#6<2T8+7<YB'_$Z-*
MC?+D';<E-_Q6#P=59;AD?L&6"A_"0?VQ:^M72/UR/*S7U)/RTN#$J-(NL"1:
M9T&T#!VC(V>$*M%4H2J<S@^%5=[-)_J)$'KA`G,&2N,:YNH-<VDX7&+XN8B?
M)VM^JAYGKET/>L;G)N+77A(=Q)XU8W_I7WC;K*@XQWZ3X9IC/=E@@5T8/<>$
M\%8GGL7A?'7P80R?8_V-IJN3'7@S`-X<X'!S6;S7'."I9@"GRYK1CV]&?WLS
M^C8_KS<:_)>U;7US;5O?7-O6-]>V]<VU;7US;5O?3-O6-].V]<VT;7TS;5O?
M=/F4\TSB$[AO<?F*G#QT'J9K9=;9^<]/+-^;'S+WK5F%W=/SIA04YDV^G98X
MQH5NT1G&&\]T457[@M,R+7II_;8OO"TT2R](?>AOK"%^;!C&'V;!C;D:=P0;
MY#)Z.EKX9V$R6GFI&%YCI?4:\W0**@TG#N$."-L1K@LM5YE5&?4I&:XG*C,N
M56;7KODZX]*:;S+J3?.6L4Y79ERHS+X(P(4UAW:O2?X&LC6'M@D@>95]D<P-
MJKMFTJ#M:R1MF'O56"%/J,ZDY5-Z!.2E*L?#953,55X6Q4JM)W\(>@69J)((
M5&;4RO4068;J&"UQ;IQJRJ8X'24E='8J&J1>^C+PXTJCY6*E4E/5QH6M[[1Z
M],NH?P"3_R72QLLXN==URKVN)9Y<E>%6N#J%<RO<)1/N9VM/,>%=`A4<>N[P
M>=L8(?,07O2A"T-85D#HE7-[^4!BX-(!.8@3>:@NGN":JO'3$2B@$5&Y-''V
M#N->1XN0DR!+&RL'_NFD,`="ZN*E[E*&5R^]-]3(?C2^AG:_^@O*(KM*X_TR
MKI8FAG6.W^QO-/:G&UY'[S:\[J8?"_TP>O(/A!YAS>2>QAKUC,6FP.O/H@FF
M$F^X'RH[0[N>:4WFE%;^)K.,;.J'RBXBA7[-+8[<235-N*.LWHATNE9&06T7
M$9:V/OS#=KAM">W(>6OQD<RWT=9@E%])S8F"DV1C2SN%I>5F2\$I]==9>E!8
M.F>V5/O++*D1'-(&X79SYZFME.BK$':O,05&&[T9;7YRX=@D\B:-KV;FP9U2
MMMINFN+#)ZUQE=@V$YX0=U9%UE>.-OI0C.E*,43777I).?/@B-,J/F:!J?'U5
M]E73''GR6M.K]#EW57VERO9$RK:6G]]]+_W">-/1:*Z%OV'N)9I)Q`QRJ<EP
M,I`=4%^Q]&'*/'F:_2SN_TJ[T#1;>SKUU[4!DDS[V;*AMK7!VM;^;.^JVEYZ
MJ.Q</.;(6K13$\O6C-JF#[8UXY(_?'RJY]L8;Z%C$V.<5I;5K\Q0K8#C@CGQ
M^>4LO[P<8S5YQP,1AA\3@[5[6D.X\)?:Y/\+FVT,FVU"-D^MS#@3MKYE2=?T
M+.OY[:<GJ@Y0,\NF2WD?R^Q5I<(6.DTNNM*W^*E6MV+*0[>/EAU-8^#J3_8E
M;)+)L+P[F?*N#^9=GZ8&QBE3+4.U:)A;US`72_JI8/QJTWAM*-XH9:T960?_
M*--OQE*>D95EQU=FG`AKN!9RU3E$&[N#P!TWNB0J/\#R`X&09)HMO.1P<M(P
M.=333QW]U*:9<FQ<SP\S+M#/1?JY)!^A\G>_QR"SGGR'A1Y09V"A]>2++%2^
MD<W,?<<KLT_@V3L>]HP?K<S^%L*CYF?\0N7XBY79ETZ/:&;2^+4&S359PT6J
MRHP3X-^Q?ADGRJPG'\3PJ3EM;<BH:32OOXXFO%`U_F)5]B75H(0PLORV,OLX
MLOS6/+&BC<6DVIVS)5\AFW,J">WKY\`UPZ@]9QX6&##G0EV"#LO,](>)PL!5
M&>?,/2CAC80G-_TO"GWU">NVF,\;%=>!49/9J+SG&@WX<S2,P\=QZ&P#Y0J#
M"TGC!,@1OO"W>^%58S[1C?E$_RGWB%R1H.O2U.T)AW1ZJ*SV5WI;\I'[BAZY
M`\SP'<)\()-[H8IM=IU_04@ZBJ9_Z8/1"S_ZN9%^>M%/;Z9>]#'UFH\9+_F,
MP(U&H!<+OMXC._1/FPA;1J"7$>@M`LJN""O;(FS8-T=N-$=ZF2/2EOFO(L#%
M.YZ*@%.\C*D(T)>,\>#EXFI]18"^(KD3O(;^30*6$_#K]&5+3J"+G3XRR`GT
MM-,7*3F!-/`%X)EV>F,!G)U>8^4$)H`O!<^WTY<P.8'EX"O!Z5:\'WR#G;Z-
MR`ELL=.W#3F!*`=]40)[#OKV`O8<].4*[(&G@P?`)X,?<-#G#SF!&@=]5I`3
MJ'70%S8Y`?J<9!;X+-S8[D3Y6.C;#J0#[TWIP,>!'P:_E?4/'$"^/<'IWPE*
M`Z?\,\']XAN0_H$)G+XY88$N-O$->*"=C=HK)W"+3M]L5*2UP'<ANN`;\3U-
M18"^UGD'?`*CUS@5@7P:1>"SQ*NEBL`\\2%!16"Y>)54$7@$_!_@J\47/!5I
M[56_Z.(E94[@I/Q'``);;%2>G$"B*,\PT6]I2CX)Y1P)P8-(%P4Y[FP%5D/^
M?Q"G=ZB;5'P;XNUH+*O^_B=X%_%R4]K!=\*BODXN\5UX1:`G?5;-[P_,%I]9
M2]QTX'*`*P$/0'X?E^-H)?@M].FUPFV$O@L>FU?!"^@?,@`_@+OV?P=O!_EG
MP#T"W&FD0UW8#UR./X^E(D!?&<59I)V>B/OI2R,5O\N2$YA'GR18^@?H>_M"
MQ1]6_&7H*\'_#OY'>N>GXC%6I$<Z!POG46CWON)58T4:M7MK5E&N,]G_SSAH
M?$@<ADK@A";[/P>`Z^@YX/2-4TZ`7C7V9\/2J'WO8,/$Q^U#T4\U*.\()9^L
MY%.5?(:2+U+RI4J.LJ99,5KQ_*71]^E/L\X!'.C@_6O_-+^%/DSI7TY\-\JW
M`?SO3+97&]7>UW,FTO7AH7+GV\2_HR!X)\5[*3Z0GE.=RLD"-4A_3["^.8%J
M&\T')ZLU!XA>>`:<@?*6X"GF0F^@ED$I\]JB)^GYP.\/=<%3\-'*,1_Q]V"
ME^%[$FD'\Y#XZNPF(<?NC\EQ6*SX$L4W*+X5?(/X8$?&OU3\E))?5G&7&L\M
```

```
M%?^-XG<K/D_Q/_-0/?>A?FW1_FE.ZF]#+I^;5HI?;QH?1CIJE\1?&>\#>UOH
M^T_P?8P^F:H([`*?`T[?A2UL)GTGM-=6\9T?\46B_;>*#Z*(OR_:?RO[5LQ7
M6YE-M/]6UEFT_U9\<><5/(]'"5Z!^]/$W^&Q@O^3QPM^E2<*WA;?`A!/L[02
M/-?21N9K:2?XLY8.@F^SR'(=L22K\O80<<;&JW$^$7'ZMUCRT,Y;O.3J#@N4
MV^B#,*E?(O0VS&]27Z7T:Y3^>:%WX7,OJ7])Z=]1^@^$/A+SK]1_K/3_5/KC
M0A^+&Q92?T[IKRJ]@Y.^%<:/U$=RJ6_+I?XZH>^$YT[J;U#Z_DI_A]!W9T.5
M?H32_T[IIPI]'[PLEOHBI5^H],N$OC];H?0/*_W3G-HQEKVE<'\3N$RV3>%V
M*MQ^I?]&Z$>RHTK_G=)?5GIF(?T]^+)6ZG6+U"=:I+Z]T$]BG92^J]+W5?J!
M0C^##5;ZH4I_C])/$OIYN(4K]=.5_GZE7R+T2]F#2E^E]&N4_K^$_H]LG=*_
MJ/1O*_W[0O\4VZGT>Y7^:Z7_E]"_R(XK_6FE_[=%/C^W6:_]?.%S?;%^S\,X
MV:V>0XJO#,U[8I[IABL;]$\3#E!\,/3B'VI3\8F*3P$G;[)`Q8N`.TS?;*OX
M`L7+5?H*%7]2\6=5^N=5?*-*OUG-FUCGQ=>['S<JW\W@SZCY9(-8;R3^N,)?
M4/.D/3C_]1=^5S*M/W`LNJMT`X-VY?S8"GH6+?TLBM^BXID*GZ_R?4K9?R$X
M;PT3ZU=7R*^P#[A%R3UHYRORW_T1ZU@KQ6D]SG&(+Z.%G7$_O:X&,FUTTT;R
M@XJ?!2=_L%;%+5SR9,5OYE(_4,7O5'R6XHN4?J6*/ZWBKZOU]FU.Y=['MW&Y
MKN[EX>T?J_IKH.)C5?^.`Y\'`[DTKL#7*?W?%-^N^G>GZB<[E_WFXE)_C^)%
M7-HKY=+>?"[M;53Z[8KOY-+>'B[M753V:KELWQR+;-]\BU'^80&Z4'6SXG>R
MYN2ROAT@)__H$.('Q!?IPX3?3>U/?KC.)4_$NDOEZ:#B-RJ>#G]F-<IS+^++
M86>NXLNYM+L:G/[5T#=4?">7^?Y;U3/9(GD_BVR7-(MLE\%*?I_B*Y7^CTK_
M)XNT\P^+;+^3"M>@N!7S!-7'897UZFN5]1IBE>4>9Y7UF:+D\Y3\(:NT]Y15
MVGE-\6IE[QUE[Z1*=T6E<VO27KPFY5TT*;]-,]I;]MMB-:Z?5^/\99._$A#7
M"J5?D:3\BD[*K^@1W`_^#U!+`P04``(`"`#G76$?"Z9FBPT#``!'$```#```
M`%-,241%0D%2+E)%4^U7S4[;0!`>AT)2"JQS[B5'CA1X@*881-3RTT"D'I$J
MY>R;N=6W["T]\AA]A!S[$I%\0IRBK50D'Y"V,SNSB7%1VP-V+EWG8[S>M?GF
MF]E9VP:`OS#<#P#6((1M`.@@WB("/*BE\'2SUL)*#@Y-P@.`0KPF_`38NP3X
M?([C\#2>H^4KUA(,(;`V0TP$*:*%UV\4SOGR.Q[L,[0&O.OVW\`.?$4VI-XJ
M8MVI]\(QW"XQ#N=G*4N;IFSHH$OX9S*9D+[NAS*SH<.RZM1:R>94C0@MS4#U
M/7*)$+,[Z5Z\KYFAP0-@<\IH:<;C1G.8X2ZRNQ$F:XB-`KN=RO2+06F'J4`+
M8`Z;%G3<%1WK9&I`%1\Y%6A!24\E3/<*T5Z7B%>=CQ"$02M)DG@P&"BMM;,X
MF*)UXVB%72D?ZV!(V:C^4$?\.#/<+T1Y0R)==3YR3I*`[3;II^,H8HM]JC1I
M%,TK+[,LY6,=3$FGA9#M-EO/;-'W\Y#I0:]_\.&02J06HBV!)]JI2M)1@F5G
M%*.,C%RG8',OH1`3%>LD1\8\D9!TS4UI0-3KGIR=1DN1K1EC=4FP`HY(LA@L
M,E#&R^:9+4,WHVQ9N[EF-#9G1SO*]\*Z*-;IJ\I6\-BP>$3P&O?A.P--S#N8
M&;C6>@3#JZ9;T5N96]GF-H-<ZS',,A1;&[*0C\%9$GR8\7-G6<$ST;U.[UC<
ML9&^<0&P=\;Y;6>&5_M0_O66L+[-^#JRYWD9WX?6/6>8\7.+WM'.]4T\>RGP
MGAU7%+=5MT/%O@1S'W.?WGXQ_Q-.?)TZZU^MN!)"H2)[#R0^=7IAK;+WM+\:
MEWHTKJA/YS]*NYP?=];/Q_O-(R]H!YR(!Z\$WH-/4-T.B.M'["BF0&#]9OUQ
M^<38QWCP!:LY('A!(L#Q*175@D<2ESJ]HJ5"PN*I$]@!`T/S*``NA^[E_<CG
ME.''*3^_>+]_7@,N/@ZZ_:7LJOB2LMA1/2S+OR"VA-WA;\UNP''W-#H_ZYU>
M'O9A)SQRW[<-)-7!#U[ZW*7/X$`H=_Y_!O]S^P502P$"%``4``(`"`#G76$?
MY+SO@MH/``!>0```#`````````!`"`````````4TQ)1$5"05(N4$%34$L!
M`A0`%``"``@`YUUA'_5``O,1*@``D%@```P`````````@````!!```%-,
M241%0D%2+D1#55!+`0(4`!0``@`(`.==81\+IF:+#0,``$<0```,````````
M````(````#\Z``!33$E$14)!4BY215-02P4&``````,``P"N````=CT`````
`
end
size 15930
```

Return to Component Cookbook
Return to Front Page

The Unofficial Newsletter of Delphi Users - Issue #9 - November 9th, 1995

QSort Component

by Mike Junkin - CIS: 71230,272

Most of us are still in the process of adapting to Delphi and learning how to take advantage of its strengths. Its component based architecture and visual tools provide unparalleled ease of use but its easy to miss their importance in how we approach code design.

Issue #8 of the Newsletter contained a code tip **Peter Szymiczek** which explained how to write a generic *Quick Sort* routine. Peter had ported the code from C and cleverly adapted the design so that it would easily work with VCL objects that stored data in lists.

Integrating code with the VCL is an important step but we still have further to go if we want to really take advantage of Delphis strengths. The next step is turning Pascal routines into Delphi components. Components are really Delphi's heart and soul. They provide real code reuse, allow extension by delegation instead of inheritance, and they allow us to really exploit Delphis RAD nature by using the Object Inspector and Form Designer.

Turning the Quick Sort procedure into the TQSort component is a very straight forward task. There are only a few design concerns which require different code than working with regular Object Pascal objects or routines.

Looking through the code the most obvious change you'll see is the switch to property based events for the compare and swap routines. This allows you to easily generate the functions using the Object Inspector but it also necessitates redesigning both routines.

Events in Delphi must be based on procedures because components have to allow for the possibility that the events will be nil. A nil function would have an undefined return value which is not acceptable to Pascals strongly typed nature. We get around this by defining the compare routine as a procedure which passes a var argument. Then we use that argument to hold the results of the comparison.

There's one other change, that is needed in both routines, and thats the inclusion of the Sender parameter. At first glance this may seem like a waste but its actually very important. By passing the Sender we give the developer a way to write one routine that can be used to sort many different objects.

With these two small changes the Quick Sort procedure becomes a easily reusable non-visual component that we can stick on our forms anytime we need to sort a list. Many thanks to Peter for the original code.

QSort Component Source

QSort Demo

Return to Component CookBook

Return to Front Page

Last issue, I inadvertantly indicated that the Keyboard macro function in Delphi is ALT-SHIFT-R and ALT-SHIFT-P. It actually should have read CTRL-SHIFT-R and CTRL-SHIFT-P. Sorry about that!

```
unit Unit1;

interface

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

type
  TForm1 = class(TForm)
    QSort1: TQSort;
    StringGrid1: TStringGrid;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure QSort1Compare(Sender: TObject; e1, e2: Word; var Action: Integer);
    procedure QSort1Swap(Sender: TObject; e1, e2: Word);
    procedure Button1Click(Sender: TObject);
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
     with StringGrid1 do
     begin
          Cells[1,1] := 'the';
          Cells[1,2] := 'brown';
          Cells[1,3] := 'dog';
          Cells[1,4] := 'bit';
          Cells[1,5] := 'me';
     end;
end;

procedure TForm1.QSort1Compare(Sender: TObject; e1, e2: Word;
  var Action: Integer);
begin
     with Sender as TStringGrid do
    begin
      if (Cells[1, e1] < Cells[1, e2]) then
        Action := -1
      else if (Cells[1, e1] > Cells[1, e2]) then
        Action := 1
      else
        Action := 0;
    end; {with}

end;

procedure TForm1.QSort1Swap(Sender: TObject; e1, e2: Word);
var
  s: string[63];  { must be large enough to contain the longest string in the
grid }
  i: integer;
begin
  with Sender as TStringGrid do
    for i := 0 to ColCount -1 do
```

```
      begin
        s := Cells[i, e1];
        Cells[i, e1] := Cells[i, e2];
        Cells[i, e2] := s;
      end; {for}
  end;

  procedure TForm1.Button1Click(Sender: TObject);
  begin
    QSort1.DoQSort(StringGrid1,STringGrid1.RowCount-1);
  end;

  end.
```

Return to QSort Article
Return to Component CookBook
Return to Front Page