Executando um programa em DOS e fechando sua janela em seguida

Quando voce exeuta um programa DOS no Windows95, sua janela permanece aberta até ser fechada pelo usuário.
Para executar um programa DOS que fecha sua janela após a execução, deve ser especificado command.com /c programa"na linha de comando. Usando a função da API WinExec para executar um programa chamado progdos.exe, a chamada deve ser:

```
WinExec('command.com /c progdos.exe',sw_ShowNormal);
```

Se o programa deve ser executado sem que seja visualizado pelo usuário, o segundo parâmetro deve ser sw_Hide.
Deve ser especificada a extensão .com senão o programa não será executado.

Mudando o texto de um TEdit no seu evento OnChange

Se o texto de um TEdit for mudado no seu evento OnChange, este evento será chamado recursivamente até acabar com o espaço de pilha. Para fazer isso, deve-se setar o evento OnChange para NIL antes de mudar o texto, voltando ao original depois, desta maneira:

```
procedure Edit1Change(Sender : TObject);
begin
  Edit1.OnChange := NIL;
```

```
  if Edit1.Text = 'Texto' then
    Edit1.Text := 'Novo Texto';
  Edit1.OnChange := Edit1Change;
end;
```

Encolhendo o executável

Em Delphi 1.0, marcando a checkbox Optimize for size and load time, em Options/Project/Linker não funciona (aparece uma mensagem de disco cheio, mesmo com muito espaço). Delphi 1.0 vem com um programa DOS, W8LOSS, que faz o mesmo. Para usá-lo, deve-se teclar o seguinte:

```
W8LOSS programa.exe
```

Isto encolherá o executável em aproximadamente 20%, diminuindo o tempo de carga.

Pegando a linha e coluna atuais de um memo

Para pegar a linha e coluna de um memo voce deve utilizar o seguinte:

```
With Memo1 do begin
  Line := Perform(EM_LINEFROMCHAR,SelStart, 0);
  Column := SelStart - Perform(EM_LINEINDEX, Line, 0);
end;
```

Mudando o fonte de um menu

Para mudar o fonte de um menu voce deve esquecer o Menu Designer e criar os itens usando a função da API AppendMenu, utilizando o parâmetro MF_OWNERDRAW, no evento OnCreate da Form. Então use as mensagens WM_MEASUREITEM e WM_DRAWITEM para desenhar os menus.

Configurações internacionais

Normalmente o Delphi pega os formatos de data/hora, moeda e formato numérico da Configuração internacional do Painel de Controle. Isto pode levar a erros quando avaliando datas, numeros ou listas. Para evitar estes erros, voce pode mudar as constantes definidas no Delphi, como DecimalSeparator, ShortDateFormat e outros desta maneira:

```
DecimalSeparator := '.';
ShortDateFormat := 'mm/dd/yy';
```

Isto terá precedência sobre a configuração padrão. Para uma lista completa das variáveis, procure em Currency Formatting Variables na ajuda do Delphi.

Extraindo ícones de um executável

Para extrair ícones de um executável, deve-se usar a função da API ExtractIcon. Ela usa 3 parâmetros:

Instance - Instancia da aplicação
FileName - Nome do executável. Deve ser um PChar

NumIcon - Número do ícone a ser recuperado. Se for Word(-1), a função retorna a quantidade de ícones no executável.

Como acessar o ambiente do DOS

Para acessar as variáveis de ambiente do DOS, deve-se usar a função da API GetDosEnvironment. Ela retorna um PChar que pode ser avaliado.

Filtrando e classificando tabelas em Delphi 1.0

Para filtrar e classificar tabelas em Delphi 1.0, voce pode usar um arquivo QBE (Query by Example) como propriedade TableName de uma TTable. Isto permite filtrar, classificar ou unir tabelas usando o componente TTable. Arquivos QBE podem ser criados no Database Desktop.

Acessando botões do DBNavigator

O DBNavigator tem uma matriz protegida Buttons que acessa os seus botões. Criando um descendente do DBNavigator permite acessá-los e mudar suas propriedades. TBSNavigator é um componente que permite mudar seu estado Enabled ou sua figura.

Fazendo a tecla Enter funcionar como Tab

Para fazer a tecla Enter funcionar como Tab é um processo de 3 passos

1) Setar a propriedade KeyPreview da Form para True
2) Setar a propriedade Default de todos os botões da Form para False
3) Criar um evento OnKeyPress para a Form como este:

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key = #13 then begin
    Key := #0;
  if (Sender is TDBGrid) then
    TDBGrid(Sender).Perform(WM_KeyDown,VK_Tab,0)
  else
    Perform(Wm_NextDlgCtl,0,0);
  end;
end;
```

Compactando tabelas

Para compactar (remover fisicamente todos registros apagados) de uma tabela Paradox deve-se utilizar o seguinte código

```
procedure ParadoxPack(Table : TTable);
var
  TBDesc : CRTblDesc;
  hDb: hDbiDb;
  TablePath: array[0..dbiMaxPathLen] of char;
begin
  FillChar(TBDesc,Sizeof(TBDesc),0);
  with TBDesc do begin
    StrPCopy(szTblName,Table.TableName);
    StrPCopy(szTblType,szParadox);
    bPack := True;
  end;
  Table.Open;
  hDb := nil;
  Check(DbiGetDirectory(Table.DBHandle, True, TablePath));
  Table.Close;
  Check(DbiOpenDatabase(nil, 'STANDARD', dbiReadWrite,
    dbiOpenExcl,nil,0, nil, nil, hDb));
  Check(DbiSetDirectory(hDb, TablePath));
  Check(DBIDoRestructure(hDb,1,@TBDesc,nil,nil,nil,False));
  Table.Open;
```

```
end;
```

Para compactar tabelas Dbase use o seguinte comando

```
DBIPackTable(Table1.DBHandle,Table1.Handle,nil,nil,True);
```

Trabalhando com grids multiseleções

As grids multiseleções do Delphi 2.0 têm uma propriedade não documentada chamada SelectedRows, uma lista de TBookmark. Você pode utilizá-la com um código como este:

```
With DbGrid1 do begin
  for i := 0 to Pred(SelectedRows.Count) do begin
    DataSource.DataSet.Bookmark := SelectedRows[i];
    {o dataset está posicionado na seleção.}
  end;
end;
```

Listando todas janelas abertas

Para listar (pegar) todas janelas abertas, deve-se usar a funcao da API EnumWindows, que usa uma funcao Callback, com dois parametros, um Handle para a janela e um ponteiro. Voce pode usá-la com um código semelhante a este (Este lista as janelas abertas, mesmo invisíveis, em uma listbox):

```
function EnumWindowsProc(Wnd : HWnd;Form : TForm1) : Boolean; Export; {$ifdef Win32}
StdCall; {$endif}
var
  Buffer : Array[0..99] of char;
begin
  GetWindowText(Wnd,Buffer,100);
  if StrLen(Buffer) <> 0 then
    Form.ListBox1.Items.Add(StrPas(Buffer));
  Result := True;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  EnumWindows(@EnumWindowsProc,LongInt(Self));
end;
```

Convertendo a primeira letra de um EditBox para maiúsculas

Para converter a primeira letra de um EditBox para maiúsculas este código pode ser utilizado:

```
procedure TForm1.Edit1Change(Sender: TObject);
var
  OldStart : Integer;
begin
  With Edit1 do
    if Text <> '' then begin
      OnChange := NIL;
      OldStart := SelStart;
      Text := UpperCase(Copy(Text,1,1))+LowerCase(Copy(Text,2,Length(Text)));
      SelStart := OldStart;
      OnChange := Edit1Change;
    end;
end;
```

Fazendo pesquisa incremental numa tabela

Para fazer pesquisa incremental numa tabela usando um EDIT, deve-se colocar o seguinte código em seu evento OnChange:

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
  With Edit1 do
    if Text <> '' then
      Table1.FindNearest([Text]);
end;
```

Fazendo busca incremental numa Query

Para fazer pesquisa incremental numa query usando um EDIT, deve-se colocar o seguinte código em seu evento OnChange: (isto funciona somente em D2/D3):

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
  With Edit1 do
    if Text <> '' then begin
      Query1.Filter := 'codigo = '''+Edit1.Text+'''';
      Query1.FindFirst;
    end;
end;
```

Outra maneira poderia ser

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
  With Edit1 do
    if Text <> '' then
      Query1.Locate('codigo',Edit1.Text,[loPartialKey]);
end;
```

Criando janelas não retangulares (D2/D3)

Para criar uma janela não retangular, voce deve criar uma Região do Windows e usar a função da API SetWindowRgn, desta maneira (isto funciona apenas em D2/D3):

```
var
  hR : THandle;
begin
  {cria uma Região elíptica}
  hR := CreateEllipticRgn(0,0,100,200);
  SetWindowRgn(Handle,hR,True);
end;
```

Colaboração de : Serkan Aksu

Detectando a finalização do Windows

Para detectar a finalização do Windows, deve-se capturar a mensagem WM_ENDSESSION. Estes passos devem ser tomados:
Declarar uma rotina de manipulação de mensagens na sessao private de sua form:

```
procedure WMEndSession(var Msg : TWMEndSession); message WM_ENDSESSION;
```

Adicionar a procedure à seção implementation de sua unit:

```
procedure TForm1.WMEndSession(var Msg : TWMEndSession);
begin
  if Msg.EndSession = TRUE then
    ShowMessage('O Windows está finalizando ' + #13 + 'às ' +
      FormatDateTime('c', Now));
  inherited;
end;
```

Movimentando o ponteiro do mouse

Para movimentar o ponteiro do mouse sem intervenção do usuário, deve-se usar um TTimer e colocar o seguinte código em seu evento OnTimer:

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
   pt:tpoint;
begin
  getcursorpos(pt);
  pt.x := pt.x + 1;
  pt.y := pt.y + 1;
  if pt.x>=screen.width-1 then setcursorpos(0,pt.y);
  if pt.y>=screen.height-1 then setcursorpos(pt.x,0);
end;
```

Posicionando o cursor numa linha de um Memo ou RichEdit

Para posicionar o cursor em uma linha de um Memo ou RichEdit, deve-se utilizar o seguinte:

```
With Memo1 do
  SelStart := Perform(EM_LINEINDEX, Linha, 0);
```

Movendo uma form sem barra de título

Para movimentar uma form sem barra de título, deve-se tratar a mensagem WM_NCHITTEST, desta maneira:

```
type
  TForm1 = class(TForm)
  public
    procedure WMNCHitTest(var M: TWMNCHitTest); message WM_NCHitTest;
  end;

var
  Form1: TForm1;

implementation
{$R *.DFM}

procedure TForm1.WMNCHitTest(var M: TWMNCHitTest);
begin
  inherited;
  if M.Result = htClient then {se o mouse foi clicado na form}
    M.Result := htCaption; {faz o Windows pensar que foi na barra de título}
```

```
end;
```

Alinhando itens do menu principal à direita

Para alinhar itens do menu principal à direita, deve-se utilizar o seguinte código:

```
{Isto justifica todos itens à direita do selecionado}

procedure SetJustify(Menu: TMenu; MenuItem: TMenuItem; Justify: Byte);
{$IFDEF WIN32}
var
  ItemInfo: TMenuItemInfo;
  Buffer: array[0..80] of Char;
{$ENDIF}
begin
{$IFDEF VER80}
  MenuItem.Caption := Chr(8) + MenuItem.Caption;
{$ELSE}
  ItemInfo.cbSize := SizeOf(TMenuItemInfo);
  ItemInfo.fMask := MIIM_TYPE;
  ItemInfo.dwTypeData := Buffer;
  ItemInfo.cch := SizeOf(Buffer);
  GetMenuItemInfo(Menu.Handle, MenuItem.Command, False, ItemInfo);
  if Justify = 1 then
    ItemInfo.fType := ItemInfo.fType or MFT_RIGHTJUSTIFY;
  SetMenuItemInfo(Menu.Handle, MenuItem.Command, False, ItemInfo);
{$ENDIF}
end;
```

Mostrar e alterar resoluções de vídeo

Para mostrar as resoluções de vídeo disponíveis, deve-se usar a função da API EnumDisplaySettings: ela pega todos os modos de vídeo disponíveis.
Para alterar os modos, deve-se usar a função ChangeDisplaySettings, que muda a resolução de vídeo e quantidade de cores.

Detectando quando o mouse está sobre um componente

Quando o mouse entra na área de um componente, Delphi envia a mensagem CM_MouseEnter. Quando sai, o Delphi envia uma mensagem. Tratando estas duas mensagens, pode-se detectar quando o mouse está sobre um componente. O componente Fly-Over label mostra esta técnica.

Detectando a finalização do Windows

Quando o Windows está finalizando, ele envia a mensagem WM_QueryEndSession para todas aplicações abertas. Para detectar (e impedi-la), deve-se definir um manipulador para esta mensagem. Ponha esta definição na seção privada de sua form:

```
    procedure WMQueryEndSession(var Msg : TWMQueryEndSession); message
    WM_QueryEndSession;
```

E ponha este método na seção implementation da unit:

```
    procedure TForm1.WMQueryEndSession(var Msg : TWMQueryEndSession);
    begin
      if MessageDlg('Fecha o Windows ?', mtConfirmation, [mbYes,mbNo], 0) = mrNo then
        Msg.Result := 0
      else
```

```
    Msg.Result := 1;
end;
```

Desenhando com tipos diferentes de linhas

O Windows permite desenhar linhas onde cada pixel é outro tipo de primitiva ou desenho com a função LineDDA. Ela precisa de uma função "callback", que será chamada quando um pixel deve ser desenhado. Ali podem ser postas as rotinas de desenho. A rotina a seguir desenha um retângulo a cada 4 pixels:

```
TForm1 = class(TForm)
  procedure FormCreate(Sender: TObject);
  procedure FormPaint(Sender: TObject);
public
  DrawNow : Integer;
end;

var
  Form1: TForm1;

procedure DrawPoint(x,y : Integer;lpData : LParam); stdcall;

implementation

{$R *.DFM}

procedure DrawPoint(x,y : Integer;lpData : LParam);
begin
  with TObject(lpData) as TForm1 do begin
    if DrawNow mod 4 = 0 then
      Canvas.Rectangle(x-2,y-2,x+3,y+3);
    Inc(DrawNow);
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  DrawNow := 0;
end;

procedure TForm1.FormPaint(Sender: TObject);
begin
  LineDDA(0,0,Width,Height,@DrawPoint,Integer(Self));
end;
```

Chamando a caixa de diálogo de conexão da rede DialUp

Para chamar a caixa de diálogo da conexão da rede DialUp, pode-se usar o WinExec, como a seguir:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  winexec(PChar('rundll32.exe rnaui.dll,RnaDial '+Edit1.Text),sw_show);
end;
```

Acessando Membros Protegidos de um Objeto

Normalmente, você não pode acessar membros protegidos de um objeto. Mas Delphi deixa você acessar membros

protegidos se o objeto foi definido na mesma unit. Assim, você pode definir um objeto derivado e fazer a mudança de tipo onde quer usar o membro protegido. Seria algo assim:

```
THackControl = class(TCustomEdit)
end;
```

Depois de definir esta classe, você pode acessar todos membros protegidos de TCustomEdit, com ebjetos de classe derivadas com um código como esse:

```
THackControl(MyEdit).Color := clBlack;
```

Escondendo janelas filhas minimizadas

Para esconder janelas filhas minimizadas, basta capturar a mensagem WM_Size, desta maneira:

```
type
  TForm1 = class(TForm)
  public
    procedure WMSize(var M : TWMSIZE);Message WM_Size;
  end;

implementation

procedure TForm1.WMSize(var M:TWMSIZE);
begin
  if M.SizeType=Size_Minimized then
    showwindow(Handle,Sw_Hide);
end;
```

Escondendo e mostrando a barra de tarefas do Windows

Para esconder a barra de tarefas do Windows, usa-se um código como esse:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  hTaskBar :Thandle;
begin
  hTaskBar := FindWindow('Shell_TrayWnd',Nil);
  ShowWindow(hTaskBar,Sw_Hide);
end;
```

Para mostrar a barra de tarefas do Windows, usa-se este código:

```
procedure TForm1.Button2Click(Sender: TObject);
var
  hTaskBar :Thandle;
begin
  hTaskBar := FindWindow('Shell_TrayWnd',Nil);
  ShowWindow(hTaskBar,Sw_Normal);
end;
```

Senha de tabelas Paradox no programa

Para dar a senha de uma tabela Paradox no programa, pode-se usar este código:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Session.AddPassWord('MyPass');
```

```
  Table1.Acive := True;
end;
```

Executando ações padrão de um Ole Container

Para executar ações padrão de um Ole Container (abrir um documento Word ou Excel ou rodar uma apresentação (Powerpoint), pode-se usar este código:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  OleContainer1.DoVerb(ovprimary);
end;
```

Compactando tabelas

Para compactar (remover fisicamente todos registros apagados) de uma tabela Paradox deve-se utilizar o seguinte código:

```
procedure ParadoxPack(Table : TTable);
var
TBDesc : CRTblDesc;
hDb: hDbiDb;
TablePath: array[0..dbiMaxPathLen] of char;

begin
FillChar(TBDesc,Sizeof(TBDesc),0);
with TBDesc do begin
StrPCopy(szTblName,Table.TableName);
StrPCopy(szTblType,szParadox);
bPack := True;
end;

hDb := nil;
Check(DbiGetDirectory(Table.DBHandle, True, TablePath));

Table.Close;
Check(DbiOpenDatabase(nil, 'STANDARD', dbiReadWrite,
dbiOpenExcl,nil,0, nil, nil, hDb));
Check(DbiSetDirectory(hDb, TablePath));
Check(DBIDoRestructure(hDb,1,@TBDesc,nil,nil,nil,False));

Table.Open;
end;
```

Verifica validade de CGC e CPF

```
unit CPFeCGC;

interface

function cpf(num: string): boolean;
function cgc(num: string): boolean;

implementation

uses SysUtils;

function cpf(num: string): boolean;
```

```
var
n1,n2,n3,n4,n5,n6,n7,n8,n9: integer;
d1,d2: integer;
digitado, calculado: string;

begin
n1:=StrToInt(num[1]);
n2:=StrToInt(num[2]);
n3:=StrToInt(num[3]);
n4:=StrToInt(num[4]);
n5:=StrToInt(num[5]);
n6:=StrToInt(num[6]);
n7:=StrToInt(num[7]);
n8:=StrToInt(num[8]);
n9:=StrToInt(num[9]);

d1:=n9*2+n8*3+n7*4+n6*5+n5*6+n4*7+n3*8+n2*9+n1*10;
d1:=11-(d1 mod 11);

if d1>=10 then d1:=0;
d2:=d1*2+n9*3+n8*4+n7*5+n6*6+n5*7+n4*8+n3*9+n2*10+n1*11;
d2:=11-(d2 mod 11);

if d2>=10 then d2:=0;
calculado:=inttostr(d1)+inttostr(d2);
digitado:=num[10]+num[11];

if calculado=digitado then
cpf:=true
else

cpf:=false;
end;

function cgc(num: string): boolean;

var
n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12: integer;
d1,d2: integer;
digitado, calculado: string;

begin
n1:=StrToInt(num[1]);
n2:=StrToInt(num[2]);
n3:=StrToInt(num[3]);
n4:=StrToInt(num[4]);
n5:=StrToInt(num[5]);
n6:=StrToInt(num[6]);
n7:=StrToInt(num[7]);
n8:=StrToInt(num[8]);
n9:=StrToInt(num[9]);
n10:=StrToInt(num[10]);
n11:=StrToInt(num[11]);
n12:=StrToInt(num[12]);
d1:=n12*2+n11*3+n10*4+n9*5+n8*6+n7*7+n6*8+n5*9+n4*2+n3*3+n2*4+n1*5;
d1:=11-(d1 mod 11);
```

```
if d1>=10 then d1:=0;
d2:=d1*2+n12*3+n11*4+n10*5+n9*6+n8*7+n7*8+n6*9+n5*2+n4*3+n3*4+n2*5+n1*6;
d2:=11-(d2 mod 11);

if d2>=10 then d2:=0;
calculado:=inttostr(d1)+inttostr(d2);
digitado:=num[13]+num[14];

if calculado=digitado then
cgc:=true
else

cgc:=false;
end;

end.
```

Gera número por extenso

```
unit Ext;

interface

function extenso (valor: real): string;

implementation

uses
SysUtils, Dialogs;

function extenso (valor: real): string;

var
Centavos, Centena, Milhar, Milhao, Texto, msg: string;

const
Unidades: array[1..9] of string = ('Um', 'Dois', 'Tres', 'Quatro', 'Cinco',
'Seis', 'Sete', 'Oito', 'Nove');
Dez: array[1..9] of string = ('Onze', 'Doze', 'Treze', 'Quatorze', 'Quinze',
'Dezesseis', 'Dezessete', 'Dezoito', 'Dezenove');
Dezenas: array[1..9] of string = ('Dez', 'Vinte', 'Trinta', 'Quarenta',
'Cinquenta', 'Sessenta', 'Setenta',
'Oitenta', 'Noventa');
Centenas: array[1..9] of string = ('Cento', 'Duzentos', 'Trezentos',
'Quatrocentos', 'Quinhentos', 'Seiscentos',
'Setecentos', 'Oitocentos', 'Novecentos');

function ifs(Expressao: Boolean; CasoVerdadeiro, CasoFalso: String): String;

begin
if Expressao
then Result:=CasoVerdadeiro

else Result:=CasoFalso;
end;
```

```
function MiniExtenso (trio: string): string;

var
Unidade, Dezena, Centena: string;

begin
Unidade:='';
Dezena:='';
Centena:='';

if (trio[2]='1') and (trio[3]<>'0') then
begin
Unidade:=Dez[strtoint(trio[3])];
Dezena:='';
end

else
begin
if trio[2]<>'0' then Dezena:=Dezenas[strtoint(trio[2])];
if trio[3]<>'0' then Unidade:=Unidades[strtoint(trio[3])];
end;

if (trio[1]='1') and (Unidade='') and (Dezena='')
then Centena:='cem'

else
if trio[1]<>'0'
then Centena:=Centenas[strtoint(trio[1])]

else Centena:='';
Result:= Centena + ifs((Centena<>'') and ((Dezena<>'') or (Unidade<>'')), ' e ', '')
+ Dezena + ifs((Dezena<>'') and (Unidade<>''),' e ', '') + Unidade;
end;

begin
if (valor>999999.99) or (valor<0) then
begin
msg:='O valor está fora do intervalo permitido.';
msg:=msg+'O número deve ser maior ou igual a zero e menor que 999.999,99.';
msg:=msg+' Se não for corrigido o número não será escrito por extenso.';
showmessage(msg);
Result:='';
exit;
end;

if valor=0 then
begin
Result:='';
Exit;
end;

Texto:=formatfloat('000000.00',valor);
Milhar:=MiniExtenso(Copy(Texto,1,3));
Centena:=MiniExtenso(Copy(Texto,4,3));
Centavos:=MiniExtenso('0'+Copy(Texto,8,2));
Result:=Milhar;
```

```
if Milhar<>'' then
if copy(texto,4,3)='000' then
Result:=Result+' Mil Reais'

else
Result:=Result+' Mil, ';
if (((copy(texto,4,2)='00') and (Milhar<>'')
and (copy(texto,6,1)<>'0')) or (centavos=''))

and (Centena<>'') then Result:=Result+' e ';
if (Milhar+Centena <>'') then Result:=Result+Centena;
if (Milhar='') and (copy(texto,4,3)='001') then
Result:=Result+' Real'

else
if (copy(texto,4,3)<>'000') then Result:=Result+' Reais';
if Centavos='' then
begin
Result:=Result+'.';
Exit;
end

else
begin
if Milhar+Centena='' then
Result:=Centavos

else
Result:=Result+', e '+Centavos;
if (copy(texto,8,2)='01') and (Centavos<>'') then
Result:=Result+' Centavo.'

else
Result:=Result+' Centavos.';
end;
end;

end.
```

Veja abaixo um exemplo para que o seu Form não seja redimensionado

Inclua o código abaixo em um Form.

```
type
TForm1 = class(TForm)
private
{ Private declarations }
procedure WMGetMinMaxInfo(var Msg: TWMGetMinMaxInfo);
message WM_GETMINMAXINFO;
procedure WMInitMenuPopup(var Msg: TWMInitMenuPopup);
message WM_INITMENUPOPUP;
procedure WMNCHitTest(var Msg: TWMNCHitTest);
message WM_NCHitTest;
public
{ Public declarations }
end;
var
```

```
Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.WMGetMinMaxInfo(var Msg: TWMGetMinMaxInfo);
begin
inherited;
with Msg.MinMaxInfo^ do
begin
ptMinTrackSize.x:= form1.width;
ptMaxTrackSize.x:= form1.width;
ptMinTrackSize.y:= form1.height;
ptMaxTrackSize.y:= form1.height;
end;
end;
procedure TForm1.WMInitMenuPopup(var Msg: TWMInitMenuPopup);
begin
inherited;
if Msg.SystemMenu then
EnableMenuItem(Msg.MenuPopup, SC_SIZE, MF_BYCOMMAND or MF_GRAYED)
end;
procedure TForm1.WMNCHitTest(var Msg: TWMNCHitTest);
begin
inherited;
with Msg do
if Result in [HTLEFT, HTRIGHT, HTBOTTOM, HTBOTTOMRIGHT,HTBOTTOMLEFT, HTTOP,
HTTOPRIGHT, HTTOPLEFT] then
Result:= HTNOWHERE
end;
```

Faço a indexação da minha tabela e reorganizo os registro e quando deleto um determindado registro esta num DBGrid, o Delphi me retorna um erro. (Index out of date). Porque acontece isto e como posso resolver este problema?

Provavelmente o problema acima esteja ocorrendo porque o arquivo não tem um índice primário. Crie o índice primário o provavelmente o seu problema estará resolvido. Caso o problema persista entre em contato conosco.

Estou desenvolvendo um programa em Delphi para controle de um Hardware, e estou precisando de uma rotina igual ao "Delay" do Pascal mas sendo em ('Microsegundos'), não sei se é possível no windows, me deram uma dica que era possivel atraves da Kernel32 mas não achei nada.

O Sr. pode utilizar a função Sleep, por exemplo:

Sleep(2000);

Estou usando o banco de dados (Paradox), e quando estou executando o aplicativo, fazendo uma inclusão de cliente ao cair a energia perde-se todos os dados incluidos desde a execução. Estou usando o comando Post para salvar os registro. Como solucionar esse problema ?

Um dos problemas dos programadores Delphi é salvar as informações fisicamente no disco rígido.

Quando estamos trabalhando com o programa as informações ficam retidas no buffer, o que, em caso de queda de energia ou até mesmo se o usuário fechar o Windows com a aplicação aberta resulta na perda dos dados, que foram processados na execução atual do sistema. Para resolver o problema, basta acrescentar no evento AfterPost de cada componente Table as linhas de código que estão abaixo. Na lista de Uses acrescente a unit DBIProcs.

Dessa forma, você não precisa temer perder os seus dados por uma falha elétrica ou pela quebra do sistema (como um erro GPF, por exemplo), após atualizar o banco de dados.

```
implementation
```

```
USES DBIProcs;
{$R *.DFM}
procedure TForm1.Table1AfterPost(DataSet: TDataSet);
begin
DBISaveChanges(Table1.handle);
end;
```

Como Justificar Texto Num RichEdit utilizando O Delphi 3.0?
Nós ainda não temos informações sobre justificar um texto conforme o Sr. está precisando. Mas estamos pesquisando a respeito e tão logo tenhamos alguma novidade entraremos em contato através de e-mail

Como posso instalar um aplicativo em um computador que não tenha o Delphi?
Para se instalar um aplicativo feito em Delphi o Sr. irá precisar utilizar um instalador. Por exemplo, no Cd do Delphi o Sr. poderá encontrar o InstallShield que é um instalador de programas. Com ele o Sr. irá gerar os discos de instalação da sua aplicação.

Existe pacotes que contenham componentes para comunicação serial e emulação de terminais VT52 / VT 100 e outros?
Nós temos um componente QUE TAMBÉM ESTÁ EM NOSSA REVISTA EM CD-ROM que poderá lhe ser útil conforme a dúvida acima
Vocês tem alguma coisa para porta serial e porta paralela?
O mesmo componente Async32 poderá lhe ser útil.

Procura e substituição de string num campo memo

```
Procedure TForm1.Button1Click (Sender: TObject);
Begin
FindReplace(Edit1.Text,Edit2.Text, Memo1);
end;

Procedure FindReplace (const Enc, subs: String; Var Texto: TMemo);
Var
i, Posicao: Integer;
Linha: string;
Begin
For i:= 0 to Texto.Lines.count - 1 do
begin
Linha := Texto. Lines[i];
Repeat
Posicao:=Pos(Enc,Linha);
If Posicao > 0 then
Begin
Delete(Linha,Posicao,Length(Enc));
Insert(Subs,Linha,Posicao);
Texto.Lines[i]:=Linha;
end;
until Posicao = 0;
end;
end;
```

Migrando do VB para o DELPHI
Muitos usuários de Visual Basic 5 estão atualmente migrando para o Delphi tendo em vista alguns problemas apresentadas pelo VB. Exclua aí os excelentes arquivos de HELP do VB contra os fraquíssimos do Delphi (se alguém puder me explicar por que a Borland faz isso com os usuários...). Alguns usuários têm reclamado das constantes "congeladas" que o VB faz. Alguns dizem que no VB5 desenvolver uma aplicação que diz "OI" é simples, mas fazer funcionar aplicativos que funcionavam nas versões anteriores não é tarefa fácil. "...geralmente VB4 congela quando em tempo de processamento ou o arquivo compilado trava quando

roda...", diz um usuário. Alguns dizem que pode ser um problema de Driver de Vídeo, mas como nós, programadores ainda vamos encontrar tempo para verificar isso? Não seria mais fácil destinar este tempo para aprender uma nova linguagem? Para facilitar esta migração, a própria BORLAND envia junto com o CD do Delphi 3 um arquivo chamado VBDELPHI.HLP, no diretório VB_Chelp. Este arquivo não é perfeito (o que poderíamos esperar...) mas ajuda. Outro arquivo que pode ser útil é o Vb2Delphi.Doc, que contém conceitos gerais e também existe um sub-diretório que contém exemplos de arquivos escritos em VB re-escritos em Delphi. Para as pessoas que possuem muitos sistemas desenvolvidos em VB, uma dica é a página Delphi Super Page. Lá existe um arquivo chamado "BASICS", o qual possui algumas funções do Basic re- escritas em Pascal. Muitas das funções estão disponíveis diretamente no Delphi, mas é interessante
ver o que é feito quando se necessita transportar código de VB para DELPHI.

Vb x Delphi - A verdade sobre o Delphi
Se você é programador de Visual Basic, para que se importar com o que podem fazer os programadores Delphi? No mínimo, isto daria uma idéia de que se deve esperar das próximas versões do Visual Basic.

Não estou falando de um compilador de verdade. É lógico que os programadores VB anseiam por um destes há anos. E embora o Delphi seja um deles, a presença ou ausência de um compilador não altera o desempenho da maioria dos aplicativos escritos em qualquer uma dessas linguagens.

INVEJA DOS OBJETOS - O que o Delphi tem que o VB não tem? Objetos. O Delphi emprega um modelo dos objetos consistente que abrange todos os seus aspectos. Enquanto os módulos de Classe e os recursos de construção de objetos do Visual Basic 4.0 são adições recentes derivadas de uma ferramenta originalmente não orientada a objetos, o Delphi é orientado a objetos desde o começo.

O modelo de objetos do Delphi permite obter mais resultados com menos códigos e desenvolver aplicativos melhores com maior rapidez e menor número de erros. Um exemplo: para adicionar uma lista de fontes da impressora a uma caixa de listagem no VB 4.0, você executaria este código:

```
Dim I
For I=0 to Printer.FontCount - 1
List1.AddItem Printer.Fonts (I)
Next I
```

No Delphi 2.0, você pode obter os mesmos resultados com uma instrução:

```
List1.Itens := Printer.fonts;
```

No Delphi, a propriedade Items (itens) de um objeto de caixa de listagem e a propriedade Fonts (fontes) do objeto Printer (impressora) são indicadores de objetos Tstrings, que mantêm uma lista de cadeias de caracteres e são semelhantes a conjuntos de cadeias (assim como as propriedades Items dos componentes Memo, RichtEdit e Outline). Sendo um Tstring tão bom quanto o outro, você pode atribuir a lista de fontes à caixa de listagem, a seu critério.

No VB, por outro lado, a lista de fontes e os itens da lista de alternativas são conjuntos de cadeias, características de um produto que anteriormente não era orientado a objetos. Uma vez que estes conjuntos não são objetos, não há como fazer referência a toda a lista de fontes ou ao conteúdo da caixa de listagem no VB. Seu único recurso é copiar cada item da lista de fontes para a caixa desejada.

A Microsoft introduziu um objeto Collection, que permite fazer referência a um conjunto de itens como uma unidade no VB 4.0. Entretanto, você só pode utilizar objetos Colletion com componentes Windows 95, portanto eles são inúteis quando você está trabalhando com elementos de linguagem (por exemplo, uma lista suspensa ou objeto Printer) anteriores ao Windows 95.

COMPONENTES EXPANDIDOS - Outra grande vantagem do Delphi em relação ao VB é que ele permite personalizar e expandir componentes, além de criar outros totalmente novos. Embora os módulos de Classe do VB 4.0 ainda nem começou a resolver a hereditariedade e a capacidade de reutilização.

Digamos que você precisa de uma lista de fontes em vários aplicativos e deseja que estas listas classifiquem as fontes, exibindo-as em duas colunas e opermita seleções múltiplas. No VB, seria necessário definir cada uma destas propriedades todas as vezes que uma caixa de listagem fosse adicionada a um formulário;e, em seguida, adicionar o código necessário para preencher a caixa com a lista de fontes Print. O Delphi, porém, permite criar um componente FontList reutilizável. A declaração de um novo componente TFontList como um tipo TListBox faz com que o TFontList do Delphi possa herdar, automaticamente, todas as propriedades, métodos e eventos do componente padrão. Portanto, é preciso elaborar códigos apenas para as coisas que estão sendo alteradas.

Você pode fazer tudo isto elaborando código VB. E se quisesse que o componente FontList exibisse o nome de cada fonte usando a família de fontes que ele representa? Não é possível fazer isto numa caixa de listagem VB sem utilizar controles personalizados adicionais. Mas no Delphi, é só especificar que seu FontList é uma caixa de listagem proprietária e, depois, adicionar códigos ao manipulador de eventos InDrawItem do componente para exibir cada nome de fonte com sua própria família de fontes.

Embora os usuários VB possam sentir-se ludibriados agora, é óbvio que a Microsoft fará todo o possível para eliminar a vantagem do Delphi. O VB não vai se tornar um ambiente totalmente orientado a objetos, mas vai chegar perto, incorporando mais elementos a cada nova versão. Quanto à capacidade de reutilização, pode ter certeza de que já vimos a última versão do VB, que é incapaz de criar controles ActiveX.

Paul Bonner
Colaborador da Windows Sources
E com isso respondo a diversas perguntas e mostro que estou absolutamente certo e que não iremos de forma nenhuma trabalhar com o Visual Basic em minha Home-Page ok?

Veja abaixo como mover um componente em Run-time

No exemplo abaixo deve ser incluído um componente Button. Para testar este exemplo mantenha a tecla CTRL pressionada clique com o mouse no componente Button. Feito isto, basta arrastar o componente Button para qualquer lado.

```
type
TForm1 = class(TForm)
Button1: TButton;
procedure Button1MouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure Button1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure Button1MouseUp(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
private
{ Private declarations }
public
{ Public declarations }
MouseDownSpot : TPoint;
Capturing : bool;
end;
var
Form1: TForm1;

implementation
{$R *.DFM}

// Evento OnMouseDown do Form
procedure TForm1.Button1MouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
if ssCtrl in Shift then begin
```

```
SetCapture(Button1.Handle);
Capturing := true;
MouseDownSpot.X := x;
MouseDownSpot.Y := Y;
end;
end;

// Evento OnMouseMove do Form
procedure TForm1.Button1MouseMove(Sender:
TObject; Shift: TShiftState; X, Y: Integer);
begin
if Capturing then begin
Button1.Left:= Button1.Left-(MouseDownSpot.x-x);
Button1.Top:= Button1.Top - (MouseDownSpot.-y);
end;
end;

// Evento OnMouseUp do Form
procedure TForm1.Button1MouseUp(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
begin
if Capturing then begin
ReleaseCapture;
Capturing := false;
Button1.Left := Button1.Left - (MouseDownSpot.x -x);
Button1.Top := Button1.Top - (MouseDownSpot.y - y);
end;
end;
```

Veja abaixo como verificar se a tecla TAB foi pressionada

Para testar o exemplo abaixo inclua alguns componentes Edit em seu form.

```
type
TForm1 = class(TForm)
Edit1: TEdit;
Edit2: TEdit;
Edit3: TEdit;
procedure FormCreate(Sender: TObject);
private
{ Private declarations }
procedure ProcessaMsg(var Msg: TMsg; var Handled: Boolean);
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.DFM}
procedure TForm1.ProcessaMsg(var Msg: TMsg; var Handled: Boolean);
begin
if Msg.message = WM_KEYDOWN then
begin
if Msg.wParam = VK_TAB then
Caption := 'Tecla Tab';
end;
end;
```

```
// Evento OnCreate
procedure TForm1.FormCreate(Sender: TObject);
begin
Application.OnMessage := ProcessaMsg;
end;
```

Executar um AVI no Form

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin

with MediaPlayer1 do
begin
FileName := 'c:\windows\help\scroll.avi';
Open;
Display := Form2;
Form2.Show;
Play;
end;

end;
```

Colocar zeros a esquerda de um valor digitado no componente Edit

```
procedure TForm1.Edit1Exit(Sender: TObject);
begin
Edit1.Text := FormatFloat('000000',StrToFloat(Edit1.Text));
end;
```

Cancelar o pressionamento de uma tecla

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
if Key = ',' then
Key := #0;
end;
```

Obs. O exemplo acima cancela o pressionamento de uma virgula em um componente EDIT.

Utilizar o MessageBox com parâmetros

```
var
Button: Integer;
Mensagem1 : Array[0..79] of Char;
Mensagem2 : Array[0..79] of Char;
begin
StrPCopy(Mensagem1, Edit1.Text + ' ' + Edit2.Text);
StrPCopy(Mensagem2, Edit3.Text + ' ' + Edit4.Text);
Button := Application.MessageBox (Mensagem2,Mensagem1, MB_YESNOCANCEL+
mb_DefButton1+MB_ICONQUESTION);
end;
```

Retorna a cor de um determinado componente no formato string

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
// Retorna a cor do form
```

```
Caption := ColorToString(Form1.Color);
// Muda a cor do form
Form1.Color := StringToColor('clBlack');
end;
```

Verifica se existe o diretório

```
procedure TForm1.Button1Click(Sender: TObject);
begin
if DirectoryExists(Edit1.Text) then
Label1.Caption := Edit1.Text + ' exists'
else
Label1.Caption := Edit1.Text + ' does not exist';
end;
```

Bloquear a tecla Ctrl+Del do DBGrid

```
procedure TForm1.DBGrid1KeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
begin
if ((Shift = [ssCtrl]) and (key = vk_delete)) THEN
Abort;
end;
```

Para criar uma janela não retangular

Você deve criar uma Região do Windows e usar a função da API SetWindowRgn, desta maneira (isto funciona apenas em
D2/D3):

```
var
hR : THandle;
begin {cria uma Reigião elíptica}
hR := CreateEllipticRgn(0,0,100,200);
SetWindowRgn(Handle,hR,True);
end;
```

Fecha todos os arquivos

```
var
i: integer;
begin
with Session do
for i:= 0 to DatabaseCount - 1 do
Databases[I].Close;
end;
```

Hint com quebra de linhas

Para incluir mais de uma linha no Hint você deve utilizar o evento OnMouseMove de cada componente. Veja abaixo como ficará o código em um Edit por exemplo.

```
procedure TForm1.Edit1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
begin
Edit1.hint := 'Primeira Linha'+#13+'Segunda Linha'+#13+
'Terceira Linha'+#13+'Quarta Linha';
end;
```

Obs. Não esquecer de mudar para TRUE o evento ShowHint.

Imprimir diretamente para a impressora sem passar pelo gerenciador de impressão

```
procedure TForm1.Button1Click(Sender: TObject);
var
F : TextFile;
i : integer;

begin
AssignFile(F,'LPT1');
Rewrite(F);
i := 0;
Writeln(F,'Teste de impressao - Linha 0');
Writeln(F,'Teste de impressao - Linha 1');
Writeln(F,#27#15+'Teste de Impressão - Linha 2');
Writeln(F,'Teste de impressao - Linha 3');
Writeln(F,#27#18+'Teste de Impressão - Linha 4');
Writeln(F,'Teste de impressao - Linha 5');
Writeln(F,#12); // Ejeta a página
CloseFile(F);
end;
```

Fechar um aplicativo Delphi a partir de outro aplicativo Delphi

```
procedure TForm1.Button1Click(Sender: TObject);
var
Win : THandle;

begin
Win := FindWindow(nil,'Form1');
if Win <> 0 then
PostMessage(Win,WM_CLOSE,0,0)
else
ShowMessage('Programa não encontrado');
end;
```

Obs. No exemplo acima foi utilizado o POSTMESSAGE para enviar uma mensagem WM_CLOSE para a janela principal.

Mostrar o HINT num Panel

```
procedure TForm1.FormCreate(Sender: TObject);
begin
Application.OnHint := DisplayHint;
end;
procedure TForm1.DisplayHint(Sender: TObject);
begin
Panel1.Caption := Application.Hint;
end;
```

Obs. Não é necessário Atribuir True para o ShowHint

Executando um programa em DOS e fechando sua janela em seguida

Quando você executa um programa DOS no Windows95, sua janela permanece aberta até ser fechada

pelo usuário.

Para executar um programa DOS que fecha sua janela após a execução, deve ser especificado command.com /c programa" na linha de comando. Usando a função da API WinExec para executar em programa chamado progdos.exe, a chamada deve ser:

WinExec('command.com /c progdos.exe',sw_ShowNormal);

Obs. Se o programa deve ser executado sem que seja visualizado pelo usuário, o segundo parâmetro deve ser sw_Hide. Deve ser especificada a extensão com senão o programa não será executado.

Retornar o nome do usuário que está editando o registro

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
try
Table1.Edit;
except on E:EDBEngineError do
if E.Errors[0].ErrorCode = 10241 then
begin
ShowMessage('Mensagem de erro'+E.Errors[0].Message );
ShowMessage( 'Arquivo com erro'+E.Errors[1].Message );
ShowMessage( 'Nome do usuario'+ E.Errors[2].Message );
end;
end;
end;
```

Retornar o nome do usuário que está com a tabela Exclusiva

```
procedure TForm1.BitBtn1Click(Sender: TObject);

begin
try
Table1.Close;
Table1.Exclusive := True;
Table1.Open;
except on E:EDBEngineError do
if E.Errors[0].ErrorCode = 10243 then
begin
ShowMessage('Mensagem de erro'+E.Errors[0].Message );
ShowMessage( 'Arquivo com erro'+E.Errors[1].Message );
ShowMessage( 'Nome do usuario'+ E.Errors[2].Message );
end
end;

end;
```

Configuração do BDE para ambiente de rede

Para o seu aplicativo feito em Delphi rodar em rede, você deve instalar o BDE em todas as estações. No BDE de cada estação, você deve colocar no parâmetro NET DIR do drive PARADOX o local onde estão as bases de dados e na PATH do Alias especificar o caminho das base de dados. Mas muita atenção, todas as estações devem estar com a mesma configuração do BDE. Veja o exemplo abaixo para configuração do parâmetro NET DIR do drive PARADOX e o PATH do Alias.

Estação n.1
NET DIR F:\
Path do Alias F:\DIRETORIO

Estação n.2
NET DIR F:\
Path do Alias F:\DIRETORIO

Estação n.3
NET DIR F:\
Path do Alias F:\DIRETORIO

Não é aconselhável que os aplicativos feitos em Delphi 1, sejam executados no servidor da base de dados, pois o PARADOX apresenta problemas de corrupção de arquivos e índices neste caso. É aconselhável que no servidor você coloque somente as bases de dados. Mas caso você tenha necessidade de utilizar o servidor você pode utilizar uma solução alternativa para o problema do PARADOX, esta solução esta sendo satisfatória na maioria dos casos. Digamos que a letra do drive de rede que você vai acessar o servidor, seja a letra "F:", então, faça o seguinte: Coloque a linha abaixo no arquivo AUTOEXEC.BAT, do servidor.

SUBST F: C:

Configure o BDE do servidor para que ele acesse o drive "F:"
Esta linha deverá ser colocada apenas no servidor, com isso você passa a ter em seu servidor, um drive virtual para acessar o
drive C:, evitando o problema do PARADOX.
No Delphi 2 e Delphi 3, você deve utilizar um instalador de programas. No CD do Delphi 2 e Delphi 3 existe um instalador
chamado InstallShield para fazer a instalação e configuração do aplicativo e do BDE.

Veja abaixo os exemplos da configuração do BDE p/ Delphi 2 e 3:

Servidor Estação 1
NET DIR \\SERVIDOR\C NET DIR \\SERVIDOR\C
PATH DO ALIAS \\SERVIDOR\C\DIRETORIO PATH DO ALIAS \\SERVIDOR\C\DIRETORIO
LOCAL SHARE TRUE LOCAL SHARE FALSE

Estação 2 Estação 3
NET DIR \\SERVIDOR\C NET DIR \\SERVIDOR\C
PATH DO ALIAS \\SERVIDOR\C\DIRETORIO PATH DO ALIAS \\SERVIDOR\C\DIRETORIO
LOCAL SHARE FALSE LOCAL SHARE FALSE

DICA:
O executável pode ser colocado em cada máquina da rede, diminuindo assim o tráfego de rede.

Como criar um Form de Apresentação (Splash Form) como o do WORD?

Para você criar um pequeno Form de apresentação enquanto seu programa é carregado ou enquanto sua aplicação gera indices, etc. Crie seu Form de Apresentação (ApresForm) e depois no menu View o opção Project Source, inclua o seguinte código:

```
program Mastapp;
uses
Forms,

Apres in 'APRES.PAS' {ApresForm},
Aplicacao01 in 'APLIC01.PAS' {Aplic01Form},
Aplicacao02 in 'APLIC02.PAS' {Aplic02Form};
{$R *.RES}
begin
ApresForm := TApresForm.Create(Application);
```

```
ApresForm.Show;
ApresForm.Update;
Application.CreateForm(TAplic01Form, Aplic01Form);
Application.CreateForm(TAplic01Form, Aplic02Form);
ApresForm.Hide;
ApresForm.Free;
Application.Run;
end.
```

Verifica se o Form, já esta ativo, Delphi1, Delphi2 e Delphi3

```
procedure TForm1.Button1Click(Sender: TObject);
var Found,i : Integer;
begin
Found := -1;
for i := 0 to Screen.FormCount - 1 do
if Screen.Forms[i] is TForm2 then
Found := i;
if Found >= 0 then
Screen.Forms[Found].Show
else
begin
Form2 := TForm2.Create(Self);
Form2.Show;
end;

end;
```

Converter a primeira letra de um Texto em maiúscula

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
with Sender as TEdit do
if (SelStart = 0) or (Text[SelStart] = ' ') then
if Key in ['a'..'z'] then
Key := UpCase(Key);
end;
```

Mostrar as fontes TrueType instaladas no Windows

Para testar o exemplo abaixo inclua em seu formulário um componente ListBox, um componente Label e um componente ListBox.

```
// Evento OnClick do componente LisBox

procedure TForm1.ListBox1Click(Sender: TObject);
begin

{ Atribui a propriedade Caption do componente Label o nome da fonte selecionada apenas para
visualização}

Label1.Caption := ListBox1.Items[ListBox1.ItemIndex];

{ Atribui ao componente Label1 na propriedade Name da propriedade Font o nome da fonte selecionada
para que o componente Label para utilizar a mesma fonte }
Label1.Font.Name := ListBox1.Items[ListBox1.ItemIndex];
```

```
end;

// Evento OnClick do componente Button.

procedure TForm1.Button1Click(Sender: TObject);
begin
{Carrega as fontes instaladas no Windows para o componente ListBox}
ListBox1.Items := Screen.Fonts;
end;
```

ShowMessage com quebra de linhas

```
procedure TForm1.Button1Click(Sender: TObject);
var
MSG : String;
begin
MSG := 'Mensagem da Primeira Linha'+#13+'Mensagem da Segunda Linha'+#13+'Mensagem da
Terceira Linha';
ShowMessage(MSG);
end;
```

ATENÇÃO. A quebra foi possível através do codigo #13.

Veja abaixo como retornar informações do ambiente DOS

No exemplo abaixo deve ser incluído no form um componente Button, um componente StringGrid e um componente ListBox.

```
type
TForm1 = class(TForm)
ListBox1: TListBox;
Button1: TButton;
StringGrid1: TStringGrid;
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

// Evento OnClick do componente Button
procedure TForm1.Button1Click(Sender: TObject);

var
Env : PChar;
i : Integer;
S : String;
PosEq : Integer;
begin
Env := GetEnvironmentStrings;
With ListBox1,StringGrid1 do begin
```

```
While Env^ <> #0 do begin
Items.Add(StrPas(Env));
Inc(Env,StrLen(Env)+1);
end;
RowCount := Items.Count;
for i := 0 to Pred(Items.Count) do begin
PosEq := Pos('=',Items[i]);
Cells[0,i] := Copy(Items[i],1,PosEq-1);
Cells[1,i] :=
Copy(Items[i],PosEq+1,Length(Items[i]));
end;
end;

end;
```

Veja abaixo como colocar um componente ComboBox em um componente StringGrid

Inclua no seu Form um componente ComboBox e um componente StringGrid.

```
type
TForm1 = class(TForm)
StringGrid1: TStringGrid;
ComboBox1: TComboBox;
procedure FormCreate(Sender: TObject);
procedure ComboBox1Change(Sender: TObject);
procedure ComboBox1Exit(Sender: TObject);
procedure StringGrid1SelectCell
(Sender: TObject; Col, Row: Integer;
var CanSelect: Boolean);
private
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;
implementation
{$R *.DFM}
// Evento OnCreate do Form
procedure TForm1.FormCreate(Sender: TObject);
begin
{ Ajusta a altura do ComboBox com a altura da linha do StringGrid}
StringGrid1.DefaultRowHeight := ComboBox1.Height;
{Esconde o ComboBox}
ComboBox1.Visible := False;
end;
// Evento OnChange do componente ComboBox
procedure TForm1.ComboBox1Change
(Sender: TObject);
begin
StringGrid1.Cells[StringGrid1.Col,StringGrid1.Row]
:= ComboBox1.Items[ComboBox1.ItemIndex];
ComboBox1.Visible := False;
StringGrid1.SetFocus;
end;
// Evento OnExit do componente ComboBox
procedure TForm1.ComboBox1Exit
```

```
(Sender: TObject);
begin
StringGrid1.Cells[StringGrid1.Col,StringGrid1.Row]
:= ComboBox1.Items[ComboBox1.ItemIndex];
ComboBox1.Visible := False;
StringGrid1.SetFocus;
end;
// Evento OnSelectCell do componente StringGrid
procedure TForm1.StringGrid1SelectCell(Sender: TObject; Col, Row: Integer;
var CanSelect: Boolean);
var
R: TRect;
begin
if ((Col = 3) AND
(Row <> 0)) then begin
R := StringGrid1.CellRect(Col, Row);
R.Left := R.Left + StringGrid1.Left;
R.Right := R.Right + StringGrid1.Left;
R.Top := R.Top + StringGrid1.Top;
R.Bottom := R.Bottom + StringGrid1.Top;
ComboBox1.Left := R.Left + 1;
ComboBox1.Top := R.Top + 1;
ComboBox1.Width := (R.Right + 1) - R.Left;
ComboBox1.Height := (R.Bottom + 1) - R.Top;
ComboBox1.Visible := True;
ComboBox1.SetFocus;
end;
CanSelect := True;

end;
```

Veja como retornar o coluna ativa do DBGrid

```
unit Unit1;
interface

uses
Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,
Db, DBTables, Grids, DBGrids;
type
TForm1 = class(TForm)
DBGrid1: TDBGrid;
Table1: TTable;
DataSource1: TDataSource;
procedure DBGrid1ColEnter(Sender: TObject);
end;
var
Form1: TForm1;
implementation
{$R *.DFM}

procedure TForm1.DBGrid1ColEnter(Sender: TObject);
begin
Caption := DBGrid1.SelectedField.FieldName;
end;
```

Como mover o conteudo da calculadora do Windows para um Edit?

Neste exemplo deve ser incluído um componente Timer.

```
type
TForm1 = class(TForm)
Timer1: TTimer;
procedure Timer1Timer(Sender: TObject);
procedure FormPaint(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure FormResize(Sender: TObject);
private
Hour, Minute, Second: Word; {hora corrente}
XCenter, YCenter, Radius: Integer; {tamanho atual do formulário}

public
procedure DrawHand (XCenter, YCenter, Radius,
BackRadius: Integer; Angle: Real);
end;

var
Form1: TForm1;

implementation

{$R *.DFM}
// Evento OnTimer do componente Timer
procedure TForm1.Timer1Timer(Sender: TObject);
var
HSec: Word; {valor temporário, não utilizado}
begin
{obtém a hora do sistema}
DecodeTime (Time, Hour, Minute, Second, HSec);
Refresh;
end;

// Evento OnPaint do componente Form
procedure TForm1.FormPaint(Sender: TObject);
var
Angle: Real;
I, X, Y, Size: Integer;
begin
{calcula o centro do formulário}
XCenter := ClientWidth div 2;
YCenter := ClientHeight div 2;
if XCenter > YCenter then
Radius := YCenter - 10
else
Radius := XCenter - 10;
{0. Desenha o marcador de horas}
Canvas.Pen.Color := clYellow;
Canvas.Brush.Color := clYellow;
Size := Radius div 50 + 1;
for I := 0 to 11 do
begin
Angle := 2 * Pi * I / 12;
X := XCenter - Round (Radius * Cos (Angle));
```

```
Y := YCenter - Round (Radius * Sin (Angle));
Canvas.Ellipse (X - Size, Y - Size, X +
Size, Y + Size);
end;

{1. Desenha o ponteiro dos minutos}
Canvas.Pen.Width := 2;
Canvas.Pen.Color := clBlue;
Angle := 2 * Pi * Minute / 60;
DrawHand (XCenter, YCenter, Radius * 90 div 100, 0, Angle);
{2. Desenha o ponteiro das horas: percentual dos minutos adicionado à hora para mover o ponteiro
suavemente}
Angle := 2 * Pi * (Hour + Minute / 60) / 12;
DrawHand (XCenter, YCenter,
Radius * 70 div 100, 0, Angle);
{3. Desenha o ponteiro dos segundos}
Canvas.Pen.Width := 1;
Canvas.Pen.Color := clRed;
Angle := 2 * Pi * Second / 60;
DrawHand (XCenter, YCenter, Radius,
Radius * 30 div 100, Angle);
end;

procedure TForm1.DrawHand (XCenter, YCenter,
Radius, BackRadius: Integer; Angle: Real);
begin
Angle := (Angle + 3*Pi/2);
Canvas.MoveTo (
XCenter - Round (BackRadius * Cos (Angle)),
YCenter - Round (BackRadius * Sin (Angle)));
Canvas.LineTo (
XCenter + Round (Radius * Cos (Angle)),
YCenter + Round (Radius * Sin (Angle)));
end;

// Evento OnCreate do Form
procedure TForm1.FormCreate(Sender: TObject);
begin
{lê as horas antes do formulário ser exibido}
Timer1Timer (self);
end;

// Evento OnResize do Form
procedure TForm1.FormResize(Sender: TObject);
begin
Refresh;
end;
```

Veja como criar um contador de página para um relatório desenvolvido no QuickReport 2.0

```
var
Form1: TForm1;
i : integer;
implementation
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
```

```
begin
i := 0 ;
QuickRep1.Prepare;
QrLabel2.Caption := IntToStr(i);
QuickRep1.Preview;
end;

procedure TForm1.QuickRep1StartPage(Sender: TQuickRep);
begin
i := i + 1;
Form2.Label1.caption := IntToStr(i);
end;
```

Veja como alterar a data e hora do sistema

```
procedure TForm1.Button1Click(Sender: TObject);
begin
SetNewTime(1998,2,10,18,07);
end;

function SetNewTime(Ano, Mes, Dia, hour, minutes: word): Boolean;

var
st:TSYSTEMTIME;
begin
GetLocalTime(st);
st.wYear := Ano;
st.wMonth := Mes;
st.wDay := Dia;
st.wHour := hour;
st.wMinute := minutes;
if not SetLocalTime(st) then
Result := False
else
Result := True;
end;
```

Veja como incluir o evento onClick no DBgrid

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, Grids, DBGrids, DB, DBTables;

type
thack = class(tcontrol);
TForm1 = class(TForm)
Table1: TTable;
DataSource1: TDataSource;
DBGrid1: TDBGrid;
Button1: TButton;
procedure Button1Click(Sender: TObject);
procedure FormClick(Sender: TObject);
private
```

```
{ Private declarations }
public
{ Public declarations }
end;
var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
THack(dbgrid1).controlstyle := THack(dbgrid1).controlstyle + [csClickEvents];
THack(dbgrid1).OnClick := Form1.OnClick;
end;

procedure TForm1.FormClick(Sender: TObject);
begin
ShowMessage('Teste');
application.processmessages;
end;

end.
```

Veja como jogar uma imagem direto para um campo da tabela

```
procedure TForm1.Button1Click(Sender: TObject);
var
BMP: TBitMap;
begin
BMP := TBitMap.Create;
if OpenPictureDialog1.Execute then
begin
if Table1.State in [dsInsert, dsEdit] then
begin
BMP.LoadFromFile(OpenPictureDialog1.FileName);
Table1Graphic.Assign( BMP );
end;
end
end;
```

[Delphi Updates] e [Delphi News]

http://www.borland.com/delphi/press/3rdparty/marotz.html=3/29/99 - borland.com And Marotz, Inc. Team For Record Delphi Certification
http://www.zdnet.co.uk/pcmag/labs/1999/04/visual/004.html=3/23/99 - Delphi 4 Wins PC Magazine UK Editors Choice Award
http://www.borland.com/techvoyage/jediinitiative.html=3/8/99 - The Borland Jedi Initiative
http://www.borland.com/midas/press/1998/midas2.html=12/22/98 - Inprise Corporation Announces New Version Of MIDAS
http://www.borland.com/delphi/chat/chat121798.html=12/17/98 - Delphi Chat Transcript
http://www.borland.com/about/press/1998/toolsfct.html=11/20/98 - toolsfactory Announces Class Explorer For Inprise's Award-Winning Enterprise Tools
http://www.infoworld.com/cgi-bin/displayStat.pl?pageone/news/features/iw100/98iw100.beloit.htm=9/30/98 - Delphi Application Wins InfoWorld 100
http://www.borland.com/programs/training/aectrain.html=9/17/98 - Delphi Training Centers
http://www.borland.com/delphi/news/=9/17/98 - See the latest Delphi 4 reviews

http://www.borland.com/delphi/books/=6/15/98 - Delphi 4 Books

[Borland News]
http://www.borland.com/conf99/=5/17/99 - Inprise and borland.com Conference 1999, July 17-22, Philadelphia, Pennsylvania
http://www.borland.com/=5/6/99 - Open Letter to Inprise Customers
http://www.borland.com/jbuilder/press/3rdparty/inlineborland.html=4/28/99 - Borland and InLine Software Sign Online Distribution Agreement for InLine's EJB Development Tools
http://www.borland.com/jbuilder/press/1999/jbldr3.html=4/19/99 - Borland Announces Borland JBuilder 3

[Inprise Events]
http://www.softbite.com/a_dtrain.htm=10/12/98 - Delphi World Tour, Delphi Training and Seminars (US)
http://www.borland.com/programs/usergroups/calendar.html#delphi=9/17/98 - Delphi User Group Calendar (US)

[Delphi White Papers]
http://www.borland.com/delphi/papers/del4app.html= - Making Every Application Development Project a Success with Delphi 4
http://www.borland.com/delphi/papers/del4ent/= - Radically Simplifying Distributed Enterprise Development Using Delphi 4
http://www.borland.com/delphi/papers/fundel4/= - Fun with Delphi 4

[Delphi Case Studies]
http://www.borland.com/delphi/cases/suprmemo.html=12/3/98 - SuperMemo World - Education
http://www.borland.com/delphi/cases/remy.html=10/26/98 - Remy Cointreau - Retailing
http://www.borland.com/delphi/cases/mincom.html=10/26/98 - Mincom Pty Ltd. - Asset Management
http://www.borland.com/europe/norben/case_sky.htm=10/26/98 - Sky Shops - Sales
http://www.borland.com/europe/norben/case_urk.htm=10/26/98 - Urk Fish Market - Agribusiness
http://www.borland.com/delphi/cases/tempus.html= - Tempus Software, Inc. - Health Care
http://www.borland.com/midas/cases/mndhs3.html= - Minnesota Department of Human Services - Government
http://www.borland.com/midas/cases/dcimmunz/= - District of Columbia Department of Health - Government
http://www.borland.com/delphi/cases/= - More Delphi Case Studies

[A Sip from the Firehose]
http://www.borland.com/firehose/1999/02-25-99.html=3/1/99 - What do developers really want?
http://www.borland.com/firehose/1999/02-22-99.html=2/22/99 - Will the Real Frank Borland Please Stand Up?
http://www.borland.com/firehose/1999/02-08-99.html=2/8/99 - borland.com - A community of developers
http://www.borland.com/firehose/1998/12-31-98.html=12/31/98 - The Year in Review 1998
http://www.borland.com/firehose/1998/12-8-98.html=11/30/98 - David I Does Comdex Las Vegas
http://www.borland.com/firehose/1998/10-29-98.html=10/28/98 - E-mail Blasting with Borland Delphi 4
http://www.borland.com/firehose/= - More from David I's Sip from the Firehose

[Tech Voyage]
http://www.borland.com/techvoyage/articles/ADOBasics/ADOBasics.html=1/5/99 - Accessing Database Using ADO and Delphi
http://www.borland.com/techvoyage/articles/MtsAnatomy/MtsAnatomy.html=12/3/98 - Anatomy of an MTS Server
http://www.borland.com/delphi/books/del4unleashed/mtstransact/=9/1/98 - Working with MTS Transactions
http://www.borland.com/techvoyage/DirectX.pas=9/1/98 - Totally unofficial DirectX 5 headers for Delphi 3.0 and Delphi 4.0 (download pas file)
http://www.borland.com/delphi/books/del4unleashed/chapter2/=6/27/98 - Delphi 4 IDE and VCL Enhancements
http://www.borland.com/techvoyage/= - More from Charlie Calvert's Tech Voyage

[Code Central]

http://www.borland.com/codecentral/repository/=5/18/99 - Borland CodeCentral(tm) Repository
http://www.borland.com/codecentral/client/=5/18/99 - Borland CodeCentral Client beta now available
http://www.borland.com/codecentral/midas/1999/constraints/=2/11/99 - Dynamic Constraints in MIDAS
http://www.borland.com/midas/papers/licensing/=2/4/99 - When Do I Need to Buy a MIDAS License?
http://www.borland.com/midas/startclient/=10/15/98 - Getting started with the MIDAS Client for Java-Part I
http://www.borland.com/codecentral/= - More Code Central

[Cobb Corner - Feature Article]
http://www.borland.com/delphi/news/zd/1999/may99/=5/1/99 - Setting an Example
http://www.borland.com/delphi/news/zd/1999/apr99/=4/1/99 - Memory mapped files
http://www.borland.com/delphi/news/zd/1999/mar99/=3/1/99 - Creating a Performance Object Explorer
http://www.borland.com/delphi/news/zd/1999/feb99/=2/1/99 - Tricking Windows
http://www.borland.com/delphi/news/zd/1999/jan99/=1/1/99 - Why use Version Control?
http://www.borland.com/delphi/news/cobb/= - Previous Articles

[Cobb Corner - Tip of the Week]
http://www.borland.com/delphi/deltips/1999/tip042699.html=4/26/99 - Duplication of object properties
http://www.borland.com/delphi/deltips/1999/tip041999.html=4/19/99 - Creating a template to add a comment block
http://www.borland.com/delphi/deltips/1999/tip041299.html=4/12/99 - Docking Delphi windows as tabbed pages
http://www.borland.com/delphi/deltips/1999/tip040599.html=4/5/99 - Changing the Order of your projects
http://www.borland.com/delphi/deltips/1999/tip032999.html=3/29/99 - Quickly obtain the Lpt Port Adress
http://www.borland.com/delphi/deltips/1999/tip032299.html=3/22/99 - Quick Printing in Delphi
http://www.borland.com/delphi/deltips/1999/tip031599.html=3/15/99 - Make a Form Showmodal Using Show
http://www.borland.com/delphi/deltips/1999/tip030899.html=3/8/99 - Shifting Blocks of Code Right or Left
http://www.borland.com/delphi/deltips/1999/tip030199.html=3/1/99 - Keystroke Shortcut in Delphi 4
http://www.borland.com/delphi/deltips/= - More tips

[Tech Tips]
http://www.borland.com/devsupport/delphi/qanda/= - Frequently Asked Questions (FAQ's)

Dicas para Delphi

Autor: André Luiz de Souza Fernandes