Um tutorial com as 100 melhores dicas selecionadas a dedo em toda a Internet para facilitar ainda mais o seu apredinzado em Delphi.

Autor: **Marcelo Jaloto Machado**
mjaloto@bol.com.br

Índice

1) MOSTRAR E ESCONDER O BOTÃO INICIAR

Crie um *sub-diretório* chamado "**Botão Iniciar**" utilizando o *windows explorer.*
Depois abra o DELPHI; feche o projeto que estiver aberto usando a opção *Close all* dentro do menu *File* e crie um novo projeto utilizando a opção *New Application* também no menu *File*.

a) Mude as seguintes propriedades do Form1:
Name : *frmEsconderMostrar*
Caption : *Programa para Esconder e Mostrar o Botão Iniciar*
Position : *poScreenCenter*
BorderStyle : *bsDialog*
Height : *104*
Width : *403*

b) Insira *dois* Botões no formulário: na Paleta de Componentes *Standard* - *Button*

c) Mude as seguintes propriedades do Button1:
Name : *btnEsconder*
Caption : *Esconder o Botão Iniciar*
Width : *177*

d) Mude as seguintes propriedades do Button2:
Name : *btnMostrar*
Caption : *Mostrar o Botão Iniciar*
Width : *177*

OBS : Salve o projeto no *sub-diretório* que você criou:

e) A *Unit1* salve com o nome de *untEsconderMostrar* e o *Project1* com o nome de *EsconderMostrar*

f) Na parte interface da unit (untEsconderMostrar) abaixo da clausula uses inclua a definição da procedure

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

Procedure MostrarEsconderIniciar(Estado:Boolean); *{inclua esta linha}*

Na parte implementation da unit (untEsconderMostrar) inclua a procedure MostrarEsconderIniciar:

implementation

```
{$R *.DFM}

procedure MostrarEsconderIniciar(Estado:Boolean);
Var taskbarhandle, buttonhandle : HWND;
begin
        taskbarhandle := FindWindow('Shell_TrayWnd', nil);
        buttonhandle := GetWindow(taskbarhandle, GW_CHILD);
        If Estado = True Then
                ShowWindow(buttonhandle, SW_RESTORE) {mostra o botão}
        Else
                ShowWindow(buttonhandle, SW_HIDE); {esconde o botão}
end;
```

2) MOSTRAR E ESCONDER A BARRA DE TAREFAS

Crie um *sub-diretório* chamado "**Barra de Tarefa**" utilizando o *windows explorer.*
Feche o projeto que estiver aberto usando a opção *Close all* dentro do menu *File* e crie um novo projeto utilizando a opção *New Application* também no menu *File*.

a) Mude as seguintes propriedades do Form1:
- Name : *frmBarraTarefa*
- Caption : *Programa para Esconder e Mostrar a Barra de Tarefa*
- Position : *poScreenCenter*
- BorderStyle : *bsDialog*
- Height : *104*
- Width : *403*

b) Insira *dois* Botões no formulário: na Paleta de Componentes *Standard* - *Button*

c) Mude as seguintes propriedades do Button1:
- Name : *btnEsconder*
- Caption : *Esconder a Barra de Tarefa*
- Width : *177*

d) Mude as seguintes propriedades do Button2:
- Name : *btnMostrar*
- Caption : *Mostrar a Barra de Tarefa*
- Width : *177*

OBS : Salve o projeto no *sub-diretório* que você criou:

e) A ***Unit1*** salve com o nome de *untBarraTarefa* e o ***Project1*** com o nome de *BarraTarefa.*

f) Na parte interface da unit (untBarraTarefa) abaixo da clausula uses inclua a definição da procedure

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

procedure EscondeBarraTarefa(EstadoBarra: Boolean);*{inclua esta linha}*

Na parte implementation da unit (untBarraTarefa) inclua a procedure EscondeBarraTarefa:

implementation

*{$R *.DFM}*

```
procedure EscondeBarraTarefa(EstadoBarra: Boolean);
var wndHandle : THandle;
        wndClass : array[0..50] of Char;
begin
        StrPCopy(@wndClass[0],'Shell_TrayWnd');
        wndHandle := FindWindow(@wndClass[0], nil);
        If EstadoBarra=True Then
                ShowWindow(wndHandle, SW_RESTORE) {Mostra a barra de tarefas}
        Else
                ShowWindow(wndHandle, SW_HIDE); {Esconde a barra de tarefas}
end;
```

3) PEGANDO O NOME DO USUARIO E A EMPRESA DO WINDOWS

Crie um *sub-diretório* chamado "**Usuário**" utilizando o *windows explorer.*
Feche o projeto que estiver aberto usando a opção *Close all* dentro do menu *File* e crie um novo projeto utilizando a opção *New Application* também no menu *File*.

a) Mude as seguintes propriedades do Form1:
Name : *frmEmpresausuario*

Caption : *Programa para ler do Windows nome do Usário e Empresa*
Position : *poScreenCenter*
BorderStyle : *bsDialog*
Height : *123*
Width : *441*

b) Insira *um* Botões no formulário: na Paleta de Componentes *Standard* - *Button*

c) Mude as seguintes propriedades do Button1:
Name : *btnUsuario*
Caption : *Pegar nome do Usuário e Empresa no Windows*
Width : *241*
d) Insira *duas* Caixas de Edição no formulário: na Paleta de Componentes *Standard* - *Edit*

e) Mude as seguintes propriedades do Edit1:
Name : *EdtUsuario*
Text : *vazio*
Width : 417
f) Mude as seguintes propriedades do Edit2:
Name : *EdtEmpresa*
Text : *vazio*
Width : 417

OBS : Salve o projeto no *sub-diretório* que você criou:

g) A ***Unit1*** salve com o nome de *untEmpresaUsuario* o ***Project1*** com o nome de *Usuario.*

h) Na parte **uses** da interface da unit (*untEmpresaUsuario*) insira a clausula : **Registry**

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, **Registry**;

No evento **onClick** do botão *btnUsuario* inclua as seguintes linhas de código:

```
procedure TForm1.Button1Click(Sender: TObject);
var
    reg: TRegIniFile;
begin
     reg := TRegIniFile.create('SOFTWARE\MICROSOFT\MS SETUP (ACME)\');
    EdtUsuario.Text := reg.ReadString('USER INFO','DefName','');
```

```
        EdtEmpresa.Text := reg.ReadString('USER INFO','DefCompany','');
        reg.free;
end;
```

4) COMO ARRASTAR UM FORM SEM CLICAR NO CAPTION?

Quando você pressiona o botão do mouse, o Windows identifica a posição da tela onde o cursor estava no momento do clique. Se a posição é igual a área do Caption do Form, o Windows ativa o modo de movimentação do Form permitindo que este seja arrastado. Portanto, a maneira mais fácil de solucionar esta questão é "enganar" o Windows.

Neste exemplo vamos considerar que o usuário poderá arrastar o Form ao clicar na área cliente deste Form:

a) Crie uma nova aplicação;
b) Adicione a seguinte declaração na seção private do Form:

```
        procedure WMNCHitTest(var M: TWMNCHitTest); message wm_NCHitTest;
```

c) Adicione o código deste procedimento na seção **implementation** do **Form**:

```
procedure TForm1.WMNCHitTest(var M: TWMNCHitTest);
begin
        inherited;                              { ativa a herança da mensagem }
        if M.Result = htClient then             { o clique foi na área cliente? }
                M.Result := htCaption;      { se sim, faz o Windows pensar que foi no
Caption. }
        end;
```

Este exemplo tratou o clique na área cliente. Você pode alterar este código para suas necessidades. Eis os possíveis valores para o **Result**:

VALOR - Local do clique
HTBORDER - Borda da janela que não tem a borda de tamanho
HTBOTTOM - Borda horizontal inferior da janela
HTBOTTOMLEFT - Canto inferior esquerdo da janela
HTBOTTOMRIGHT - Canto inferior direito da janela
HTCAPTION - Barra de Título(Caption)
HTCLIENT - Área cliente
HTERROR - igual ao HTNOWHERE, a diferença é que produz um beep indicando erro
HTGROWBOX - Caixa de tamanho (igual ao HTSIZE)
HTHSCROLL - Barra de rolagem horizontal
HTLEFT - Borda esquerda da janela
HTMENU - Em um menu
HTNOWHERE - Plano de fundo da janela ou linha de divisão entre janelas
HTREDUCE - Botão minimizar

HTRIGHT - Borda direita da janela
HTSIZE - Caixa de tamanho (igual ao HTGROWBOX)
HTSYSMENU - Botão de Sistema/Fechar da janela MDIChild
HTTOP - Borda horizontal superior da janela
HTTOPLEFT - Canto superior esquerdo da janela
HTTOPRIGHT - Canto direito superior da janela
HTTRANSPARENT - Janela em segundo plano
HTVSCROLL - Barra de rolagem vertical
HTZOOM - Botão maximizar

5) 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;
```

6) ESCONDENDO A APLICAÇÃO DA BARRA DE TAREFAS

Para fazer com que o ícone da aplicação em Delphi desapareça da Barra de Tarefas, execute o código a seguir:

```
var
        H : HWnd;
begin
        H := FindWindow(Nil,'Project1');
        if H <> 0 then ShowWindow(H,SW_HIDE);
end;
```

7) OS COMANDOS INC E DEC

Você sabia que existe uma opção para a comum linha de comando:
Variavel:=Variavel+1; ?

O comando INC e DEC permitem agilizar o processamento do seu sistema. Para isso substitua a linha acima por:

INC(variavel);

ou

DEC(variavel) se você quiser diminuir ao invés de aumentar 1.

8) COMO FAZER UM BEEP NO COMPUTADOR

```
messageBeep(0);
```

9) COMO FAZER UMA PAUSA POR UM PERÍODO DETERMINADO

NumSec é o tempo em segundos de espera

```
var
        NumSec SmallInt;
        StartTime: TDateTime;
begin
        StartTime := now;
        NumSec:=10;
repeat
        Application.ProcessMessages;
        until Now > StartTime + NumSec * (1/24/60/60);
end;
```

10) DESABILITANDO O SPLASH SCREEN DO REPORT SMITH

1 - Localize o arquivo RS_RUN.INI (no diretório do Windows);
2 - Na seção [ReportSmith] inclua a linha seguinte:
ShowAboutBox=0
3 - Na seção [RS_RunTime] inclua a linha seguinte:
ShowAboutBox=0
4 - Não se esqueça de distribuir com o seu aplicativo o referido arquivo INI.

11) LENDO O VOLUME DO HD

```
Function ExtractDiskSerial(Drive:String):String;
Var
        Serial:DWord;
        DirLen,Flags: DWord;
        DLabel : Array[0..11] of Char;
begin
        GetVolumeInformation(PChar(Drive+':\'),dLabel,12,@Serial,DirLen,Flags,nil,0);
        Result := IntToHex(Serial,8);
end;
```

12) DESCOBRINDO O NÚMERO SERIAL DO HD

```
 procedure TForm1.Button1Click(Sender: TObject);
 var
        SerialNum : pdword;
```

```
        a, b : dword;
        Buffer  : array [0..255] of char;
begin
        if GetVolumeInformation('c:\', Buffer, SizeOf(Buffer), SerialNum, a, b, nil, 0) then
                Label1.Caption := IntToStr(SerialNum^);
end;
```

13) PARA SABER SOMENTE O PATH DA APLICAÇÃO

```
ExtractFilePath( Application.ExeName )
```

14) INTERCEPTAR AS TECLAS DE FUNÇÃO (F1, F2, F3...)

Primeiro, coloque a propriedade KeyPreview do formulário como TRUE. Depois, insira este código no evento OnKeyDown do formulário:

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift: TShiftState);
begin
if Key = VK_F5 then
        showMessage('I pressed the F5 key');
end;
```

Você também pode usar as variáveis VK_F1 até VK_F12 referentes as outras teclas de função.

15) TRADUZINDO A MENSAGEM: "DELETE RECORD?"

Quando clicamos sobre o botão de deleção no DBNavigator (o do sinal de menos) surge uma box com a mensagem "Delete Record?" com botões Ok e Cancel. Para fazer aparecer a mensagem em português deverá selecionar o componente Table e mudar a propriedade ConfirmDelete para False e no evento da tabela BeforeDelete colocar o seguinte (flaviojr@cyber.com.br):

```
procedure TForm1.Table1BeforeDelete(DataSet:TDataSet);
begin
    if MessageDlg('Eliminar o Registro?',mtConfirmation,[mbYes,mbNo],0)<>mrYes then
Abort;
end;
```

16) INCLUIR UM PREVIEW PERSONALIZADO NO QUICK REPORT

No relatório, criar a procedure SHOWPREVIEW contendo:

```
Procedure Showpreview;
begin
        preview.showmodal;
end;
```

Onde preview é o nome do form criado para preview.
Não esquecer de incluir o nome da procedure na cláusula uses.
Após isso, deve-se incluir no evento CREATE do formulario principal ou do relatório o direcionamento do objeto Qprinter, com a seguir:

```
qprinter.onpreview:=showpreview;
```

Isto faz com que toda vez que se desejar exibir um preview, o programa abra a rotina 'showpreview', que abre o formulário criado, chamado 'preview'.

17) EXECUTANDO PROGRAMAS EXTERNOS

Se você precisa abrir programas externos no seu aplicativo DELPHI, como a calculadora do Windows, por exemplo, inclua a seguinte linha no seu programa:

```
WinExec('calc.exe', sw_show);
```

'calc.exe' é o nome do programa. Caso queira abrir um outro programa, altere este nome.

18) UTILIZANDO A TECLA ENTER PARA SALTAR DE CAMPO

Insira este código em um evento OnKeyPress de um controle de edição:

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
        If Key = #13 Then
        Begin
                SelectNext(Sender as tWinControl, True, True );
                Key := #0;
        end;
end;
```

19) TOCANDO UM SOM WAV SEM O MEDIA PLAYER

Acrescente MMSystem na Uses do começo na Unit

Utilize a API SndPlaySound();
Para interromper o som sem ele acabar de tocar utilize a API PlaySound(nil,0,0);

Ex: SndPlaySound('c:\teste.wav', snd_ASync);

PS: snd_Loop serve para repetir continuamente o som.

20) OBTER O DIRETÓRIO ONDE SEU PROGRAMA ESTÁ INSTALADO

Crie uma váriavel do tipo String e insira a seguinte linha no evento ou função desejada do formulário:

ExtractFilePath(Application.Exename);

Retorna-rá o path atual do arquivo .EXE do seu programa.

21) COMO BLOQUEAR UM ARQUIVO EM AMBIENTE DE REDE

É uma dica simples mas muito importante !!!
Quando você programar visando uma rede e quiser bloquear um arquivo é só chamar o metodo "edit" da Tabela que estiver usando.
EX: Table1.edit;
PS: Se o registro já estiver bloqueado ocorrerá um erro, então você deve fazer o seguinte :

```
try { para verificar o erro }
        Table1.edit;
exception on TDBEngineError do { o erro..}
        MessageDlg('Registro ja esta sendo usado...!', mtInformation, [ mbOk ], 0 );
end;
```

22) USANDO ENTER PARA MUDAR DE CAMPO DE UM DBGRID

```
If ( Chr(Key) <> #13) Then Exit;
If ( DBGrid1.SelectedIndex + 1 <> DBGrid1.FieldCount ) Then
        DBGRid1.SelectedIndex := DBGRid1.SelectedIndex + 1;
```

23) FUNÇÃO PARA OBTER O NÚMERO DO REGISTRO ATUAL

```
Function Recno(Dataset: TDataset): Longint;
var
        CursorProps: CurProps;
        RecordProps: RECProps;
begin
        { Return 0 if dataset is not Paradox or dBASE }
        Result := 0;
        with Dataset do
```

```
        begin
                if State = dsInactive then DBError(SDataSetClosed);
                Check(DbiGetCursorProps(Handle, CursorProps));
                UpdateCursorPos;
                try
                        Check(DbiGetRecord(Handle, dbiNOLOCK, nil, @RecordProps));
                        case CursorProps.iSeqNums of
                                0: Result := RecordProps.iPhyRecNum; { dBASE }
                                1: Result := RecordProps.iSeqNum; { Paradox }
                        end;
                except
                        on EDBEngineError do
                        Result := 0;
        end;
        end;
end;
```

24) ENVIANDO UM ARQUIVO PARA A LIXEIRA

```
uses ShellAPI;

Function DeleteFileWithUndo(sFileName : string ) : boolean;

var
        fos : TSHFileOpStruct;

begin

        FillChar( fos, SizeOf( fos ), 0 );
        With fos do
        begin
                wFunc := FO_DELETE;
                pFrom := PChar( sFileName );
                fFlags := FOF_ALLOWUNDO
                or FOF_NOCONFIRMATION
                or FOF_SILENT;
        end;
        Result := ( 0 = ShFileOperation( fos ) );
end;
```

25) CARREGAR UM CURSOR ANIMADO (*.ANI)

```
const
        cnCursorID1 = 1;

begin
```

```
        Screen.Cursors[ cnCursorID1 ] :=
        LoadCursorFromFile('c:\win95\cursors\cavalo.ani' );
        Cursor := cnCursorID1;
end;
```

PS: O arquivo CAVALO.ANI deverá existir no diretório apontado.

26) TRANSFERIR O CONTEÚDO DE UM MEMO PARA O MEMOFIELD

```
var
        t: TTable;
begin
        t := TTable.create(self);
        with t do
        begin
                DatabaseName := 'MyAlias'; {Nome do Alias}
                TableName := 'MyTbl.db';
                open;
                edit;
                insert;
                FieldByName('TheField').assign(Memo1.lines);
                post; { Requerido!!!}
                close;
        end;
end;
```

27) CAPTURANDO O CONTEÚDO DO DESKTOP

Coloque o código abaixo no evento FormResize do Formulário.

```
procedure TForm1.FormResize(Sender: TObject);
var
        R : TRect;
        DC : HDc;
        Canv : TCanvas;
begin
        R := Rect( 0, 0, Screen.Width, Screen.Height );
        DC := GetWindowDC( GetDeskTopWindow );
        Canv := TCanvas.Create;
        Canv.Handle := DC;
        Canvas.CopyRect( R, Canv, R );
        ReleaseDC( GetDeskTopWindow, DC );
end;
```

28) ESCREVENDO UM TEXTO DIAGONAL USANDO O CANVAS

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

```
begin
        with Form1.Canvas do begin
                Font.Name := 'Arial';
                Font.Size := 24;
                tf := TFont.Create;
                tf.Assign(Font);
                GetObject(tf.Handle, sizeof(lf), @lf);
                lf.lfEscapement := 450;
                lf.lfOrientation := 450;
                tf.Handle := CreateFontIndirect(lf);
                Font.Assign(tf);
                tf.Free;
                TextOut(20, Height div 2, 'Texto Diagonal!');
        end;
 end;
```

29) EXTRAIR UM ICONE DE UM DETERMINADO APLICATIVO

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

Instance - Instância 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 do executável.

Coloque ShellAPI em uses no começo da unit.

```
procedure TForm1.Button1Click(Sender: TObject);
var
        IconIndex : word;
        h : hIcon;
begin
        IconIndex := 0;
        h                                                                       :=
        ExtractAssociatedIcon(hInstance,'C:\WINDOWS\NOTEPAD.EXE',IconINdex);
        DrawIcon(Form1.Canvas.Handle,10,10,h);
end;
```

30) ALINHANDO ITEMS DO MENU 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;
```

31) ABRIR AUTOMATICAMENTE SEU NAVEGADOR PADRÃO E CARREGAR A PÁGINA DETERMINADA PELO LINK

1º Declare o procedure na seção PUBLIC da unit.

```
   procedure JumpTo(const aAdress: String);
```

2º Coloque a cláusula ShellAPI na uses no início da unit.

```
procedure TForm1.JumpTo(const aAdress: String);
var
        buffer: String;
begin
        buffer := 'http://' + aAdress;
        ShellExecute(Application.Handle,        nil,        PChar(buffer),        nil,        nil,
SW_SHOWNORMAL);
end;

procedure TForm1.Label1Click(Sender: TObject);
begin
        JumpTo('www.geocities.com/SiliconValley/Way/1497');
end;
```

32) COPIAR REGISTROS DE UMA TABELA PARA OUTRA INCLUINDO VALORES NULL

```
procedure TtableCopiaRegistro(Origem, Destino: Ttable);
begin
        with TabelaOrig do
         begin
                {Inicia um contador para os campos da TabelaOrig}
```

```
                for i := 0 to FieldCount -1 do
                {Este if verifica se o campo da TabelaOrig é NULL, se for, atribui seu valor
ao
                   campo da TabelaDest}
                 if not Fields[i].IsNull then TabelaDest.Fields[i].Assign(Fields[i]);
        end; {end with}
end;
```

Este exemplo funcionará com todos tipos de campos se você tiver acabado de criar a TabelaDest.
Para criar um dado valor NULL : Fields[i].Clear

33) DELETAR ARQUIVOS DE UM DIRETÓRIO COM O CARACTERE CURINGA '*'

```
procedure TForm1.SpeedButton1.Click(Sender: TObject);
var
    SearchRec: TSearchRec;
    Result: Integer;
begin
    Result:=FindFirst('c:\teste\*.*', faAnyFile, SearchRec);
    while result=0 do
            begin
                    DeleteFile('c:\teste\'+SearchRec.Name);
                    Result:=FindNext(SearchRec);
            end;
end;
```

34) CAPTURAR A LISTA DE ALIASES DISPONÍVEIS

Tudo que você precisa é de um componente TSession, um componente TListBox e uma String List.

Defina a propriedade SessionName do TSession para 'Session'. Utilize o seguinte código:

```
procedure TForm1.Button3Click(Sender: TObject);
var
        MyStringList: TStringList;
         i: integer;
```

```
begin
        MyStringList := TStringList.Create;
        Session.GetAliasNames(MyStringList);
        for I := 0 to MyStringList.Count - 1 do
                ListBox1.Items.Add(MyStringList[I]);
end;
```

Utilize o Help do TSession e consulte seus métodos para ver por exemplo como capturar o diretório ou caminho de um Alias com o método 'GetAliasParams'.

35) ABRIR E FECHAR A BANDEJA DO DRIVE DE CD-ROM

```
{Para Abrir:}
mciSendString('Set cdaudio door open wait', nil, 0, handle);

{Para Fechar:}
mciSendString('Set cdaudio door closed wait', nil, 0, handle);
```

36) UTILIZANDO O CODE EXPLORER

A versão 4 do Borland Delphi está recheada de novos recursos em várias áreas do produto (IDE, Internet, linguagem, aplicações multi-tier, entre outras).

Uma das primeiras coisas que notamos quando abrimos o editor de código do Delphi 4 é a presença de um painel ancorado na lateral esquerda, contendo todos os tipos, classes, propriedades, métodos, variáveis globais, rotinas globais e interfaces contidos na unit selecionada. Esse painel é chamado Code Explorer e seu objetivo é tornar mais fácil a navegação entre as units do projeto e automatizar alguns processos envolvidos na criação de classes.

O Code Explorer também permite que você navegue diretamente para as declarações que são apresentadas nele, bastando dar um duploclique com o mouse sobre a declaração desejada. Novas declarações podem ser feitas também usando o Code Explorer, facilitando o desenvolvimento do código. Por exemplo, vamos supor que você queira criar uma nova função chamada Calculo com dois parâmetros do tipo real e que retornará também um real. Para isso, deve-se seguir os seguintes passos:

Selecione a pasta Variable/Constants dentro do Code Explorer. Dê um clique com o botão direito do mouse e selecione New no menu que aparecerá, como mostra a figura a seguir.

O Code Explorer apresentará um novo item que permite identificar qual o tipo de declaração que está sendo feita, por meio da informação que o programador passar a ele. No nosso exemplo, digitaremos o cabeçalho da função Calculo como segue :

```
function Calculo(x, y : real) : real;
```

Após digitar a declaração e pressionar <Enter> o Code Explorer criará automaticamente o cabeçalho da função na seção Interface e a sua implementação na seção Implementation da unit, como mostra a figura seguinte, evitiando assim que ocorreram erros de declaração no interior do código.

Um outro recurso que o Code Explorer fornece é conhecido como Class Completion. O programador pode, dentre outras facilidades, criar apenas o básico da declaração de uma propriedade e, com o simples toque de um atalho no teclado, o Code Explorer completará a declaração. Como exemplo, vamos declarar uma propriedade chamada Cor do tipo TColor dentro da nossa classe TForm 1.

```
type
        TForm1=class(TForm)
        private
                {declarações privadas}
        public
                {declarações públicas}
                property Cor:TColor;
  end;
```

Agora com o cursor posicionado sobre a declaração da propriedade, pressionamos <Ctrl><Shift>C e o Code Explorer completará toda a declaração da estrutura da classe.

37) COPIANDO ARQUIVOS VIA DELPHI

```
Function CopiaArquivo(scrname,destname:string):byte;
var
        source,destination:file;
        buffer:array[1..1024] of byte;
        readcnt,writecnt:word;
        pname,dname,fname,ename:String;
{USO:
R:=COPIAARQUIVO('C:\diretorio\FILE.EXT','C:\diretorio\FILE.EXT');
Devolve 0=Ok, 1=Erro no Origem, 2=Erro no Destino, 3=Disco Cheio}
begin
        AssignFile(source,scrname);
        Try
                Reset(source,1);
        Except
                CopiaArquivo:=1;
                Exit;
        end;

        If destname[length(destname)]='\' then
        begin
                pname:=scrname;
```

```
            destname:=destname+separa(scrname,'\',Ocorre(scrname,'\')+1);
        end;
        AssignFile(destination,destname);
        Try
            Rewrite(destination,1);
        Except
            CopiaArquivo:=2;
            Exit;
        end;

        Repeat
            BlockRead(source,buffer,sizeof(buffer),readcnt);
            Try
                BlockWrite(destination,buffer,readcnt,writecnt);
            Except
                CopiaArquivo:=3; {Disco Cheio?}
                Exit;
            end;
        until (readcnt=0) or (writecnt<>readcnt);
        CloseFile(destination);
        CloseFile(source);
        CopiaArquivo:=0;
end;
```

38) ABRIR UM TCOMBOBOX SEM CLICÁ-LO

```
ComboBox1.DroppedDown := True;
```

39) MUDAR A COR DA CÉLULA ATIVA DO DBGRID

A rotina abaixo deverá ser colocada no evento OnDrawDataCell, do DBGrid.

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

40) COMO INCREMENTAR 1 MÊS NUMA DATA

IncMonth(Data, 1);

No exemplo, a variável Data é do tipo TDateTime.

41) VERIFICAR SE EXISTE DISQUETE NO DRIVE

```
unit UTestaDrive;

interface

uses
	Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

type
	TForm1 = class(TForm)
		Button1: TButton;
		procedure Button1Click(Sender: TObject);
		function TemDiscoNoDrive(const drive : char): boolean;

	private
		{ Private declarations }
	public
		 { Public declarations }
end;

var
	Form1: TForm1;

implementation

{$R *.DFM}

function TForm1.TemDiscoNoDrive(const drive : char): boolean;
var
	DriveNumero : byte;
	EMode : word;
begin
	result := false;
	DriveNumero := ord(Drive);
	if DriveNumero >= ord('a') then
		dec(DriveNumero,$20);
	EMode := SetErrorMode(SEM_FAILCRITICALERRORS);
	try
		if DiskSize(DriveNumero-$40) = -1 then
			Result := true
		else
```

```
                messagebeep(0);
        finally
                SetErrorMode(EMode);
        end;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
        if TemDiscoNoDrive('a') then
                ShowMessage('Tem disco No drive A:')
        else
                ShowMessage('Não tem Disco no Drive A:');
end;

end.
```

42) ACESSAR O AMBIENTE DOS

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

43) EXECUTAR UM PROG. DOS E FECHAR 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 um programa chamado proddos.exe, a chamada deve ser:

```
WinExec('c:\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.

44) INSTANCIAR UMA ÚNICA VEZ UM EXECUTÁVEL CORRESPONDENTE A UMA DETERMINADA APLICAÇÃO

No Microsoft® Windows®, existe uma tabela do sistema chamada ‘atom table’. Esta tabela armezena strings com seus correspondentes identificadores. Existem

várias funções da API do windows, chamadas de ‘atom functions’, que permitem que uma aplicação insira, apague, procure por uma determinada ‘atom string’, etc.. O código abaixo garante que somente uma instância do executável de sua aplicação estará carregado em memória. O form1 seria o form principal da aplicação:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    {Procura na tabela para verificar se o programa já está rodando}

    {Substitua a string ‘MINHA STRING’ por uma de sua conveniência}
    if GlobalFindAtom(‘MINHA STRING') = 0 then
        {zero significa não encontrar}
        atom := GlobalAddAtom(‘MINHA STRING’)
    else

    begin

        {Se o programa já estiver rodando, então mostrar a mensagem e parar}
        MessageDlg(‘A aplicação já encontra-se em execução!!', mtWarning, [mbOK], 0);
        Halt;
    end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
    {Remove o item da tabela, de modo que a aplicação possa ser executada novamente}
    GlobalDeleteAtom(atom);
end;
```

45) MODIFICAR VÁRIAS PROPRIEDADES DE UM OBJETO AO MESMO TEMPO

Utilize o comando with, desse modo:

```
With Edit1 do
begin
        Text := ' @Delphi';
        Widht := 30;
        Enabled := True;
end;
```

46) PARA EMITIR UM SOM QUANDO O ENTER FOR PRESSIONADO

No Evento OnKeyPress de um Edit:

```
if Key = Chr(VK_RETURN) then
        Perform(WM_NEXTDLGCTL,0,0);
```

47) 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 erro 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 digitar o seguinte:

W8LOSS programa.exe

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

48) CRIAR ALIAS VIA PROGRAMAÇÃO

Paradox:

```
Session.AddStandardAlias('SeuAlias', edtPath.text, 'Paradox');
Session.SaveConfigFile;
```

49) DESABILITAR AS TECLAS DE ACESSO (CTRL+ALT+DEL), (ALT+TAB), (CTRL+ESC)

```
var
        OldValue : LongBool;
 begin
        {liga a trava}
         SystemParametersInfo(97, Word(True), @OldValue, 0);
        {desliga a trava}
        SystemParametersInfo(97, Word(False), @OldValue, 0);
end;
```

50) FAZER UM SPLASH SCREEN

```
form2:=tform2.create(application);
form2.show;
form2.update;

form2.hide;
```

```
form2.free;
Application.Run;
```

Obs: apagar a primeira linha, 'Application.Initialize'.

51) FUNÇÃO PARA ENCRIPTAR UMA STRING

```
function encrypt( dummy: Pchar):Pchar;
var
        x: Integer;
        w: Word;
        s: String;
        c: Char;
begin
        s:=StrPas(dummy);
        w:=StrLen(dummy);
        for x:=1 to w do
        begin
                c:=s[x];
                c:=char ( ord (c) xor 159);
                s[x]:=c;
        end;
        StrPCopy(dummy,s);
        encrypt:=dummy;
end;
```

52) FUNÇÃO REVERTER UMA STRING

```
function TForm1.StrReverse(MyString : string) : String;
var
        i: integer;
        HelpString: string;
begin
        HelpString := '';
        for i := 1 to Length(MyString) do
        HelpString := MyString[i]+HelpString;
        Result := HelpString;
end;
```

53) FUNÇÃO DE CONVERTER UM NÚMERO INTEIRO PARA BINÁRIO

```
{Integer to Binary}
function IntToBin(Value: LongInt;Size: Integer): String;
var
```

```
        i: Integer;
  begin
        Result:='';
        for i:=Size downto 0 do
        begin
                if Value and (1 shl i)<>0 then
                begin
                        Result:=Result+'1';
                end
                else
                begin
                        Result:=Result+'0';
                end;
        end;
  end;
```

54) FUNÇÃO DE CONVERTER UM NÚMERO BINÁRIO PARA INTEIRO

```
  {Binary to Integer}
  function BinToInt(Value: String): LongInt;
  var
        i,Size: Integer;
  begin
        Result:=0;
        Size:=Length(Value);
        for i:=Size downto 0 do
        begin
                if Copy(Value,i,1)='1' then
                begin
                Result:=Result+(1 shl i);
                end;
        end;
  end;
```

55) FUNÇÃO PARA ENCRIPTAR E DESCRIPTAR UMA STRING

```
  const
        StartKey     = 981;  {Start default key}
        MultKey   = 12674;   {Mult default key}
        AddKey    = 35891;   {Add default key}
{Encriptar}
  function Encrypt(const InString: string; StartKey,MultKey,AddKey: Integer): string;
  var
        I: Byte;
  begin
        Result := '';
```

```
        for I := 1 to Length(InString) do
        begin
                Result := Result + CHAR(Byte(InString[I]) xor (StartKey shr 8));
                StartKey := (Byte(Result[I]) + StartKey) * MultKey + AddKey;
        end;
  end;
{Descriptar}
  function Decrypt(const InString: string; StartKey,MultKey,AddKey: Integer): string;
  var
        I: Byte;
  begin
        Result := '';
        for I := 1 to Length(InString) do
        begin
                Result := Result + CHAR(Byte(InString[I]) xor (StartKey shr 8));
                StartKey := (Byte(InString[I]) + StartKey) * MultKey + AddKey;
        end;
  end;
```

56) UMA ROTINA PARA VERIFICAR ERROS EM TODA A APLICAÇÃO

Para tratar erros de forma genérica, em todo o seu programa, insira no método ON CREATE do formulário principal a linha:

Application.OnException:=RotinaGeral;

RotinaGeral é uma procedure na qual deverá constar o código para verificar e enviar as mensagens de erro do seu sistema.

57) ENVIANDO INFORMAÇÕES DIRETO PARA A IMPRESSORA

Muitas vezes torna-se necessário, ou até mesmo, imprescindível que você envie informações diretamente para a impressora, uma vez que a utilização da impressão típica do Windows é um pouco demorada e o uso do driver Genérico/Somente Texto não é muito confiável.

Uma boa solução para enviar informações diretamente para a impressora é usar o seguinte código:

```
Procedure TForm1.Button1Click(Sender: Object);
var
        Imp: TextFile;
begin
        AssignFile(Imp, 'LPT1');
        Rewrite(Imp);
        Write(Imp, 'Isto vai sair na impressora');
        CloseFile(Imp);
end;
```

Desta forma será possível, inclusive, utilizar os códigos de configuração da impressora. Para a impressora padrão Epson, por exemplo, você poderia utilizar algo assim:

```
Write(Imp, #27#69 + 'Teste' + #27#70); { impressão em negrito }
Write(Imp, #15 + 'Teste' + #18); { impressão no modo condensado }
Write(Imp, #12); { salto de página }
```

58) ENVIANDO CARACTERES DIRETAMENTE AO BUFFER DA IMPRESSORA

Ao trabalharmos com impressão, em certos casos desejamos alterar o comportamento da impressora.

Algumas opções, principalmente em impressoras matriciais, são obtidas através do envio dos chamados "códigos de escape" para a impressora (por exemplo, alterar espaçamento entre as linhas (#45), tipo de fonte (#18, #23), etc). Em versões 16-bit do Windows, isso não era complicado, mas agora, nas versões 32-bit, o acesso direto ao hardware não é mais possível.

Portanto, para enviarmos caracteres diretamente a impressora, devemos utilizar o "escape" chamado "PASSTHROUGH" do Windows e enviarmos a informação desejada diretamente.

Na documentação do Win32 SDK este escape é dado como obsoleto, mas enquanto utilizarmos impressoras matriciais que necessitem de "códigos escape" para certas funcionalidades, ele será necessário.

Ao utilizar impressoras Postscript tenha cuidado, pois nem sempre esta técnica irá funcionar. Em impressoras matriciais, você pode enviar qualquer tipo de caracteres que achar necessário.

Abaixo segue um código exemplificando o envio de uma string qualquer diretamente ao buffer da impressora:

```
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  end;
```

```
var
  Form1: TForm1;

implementation

{$R *.DFM}

uses Printers;

type
{ Tipo requerido pelo PASSTHROUGH }
TBufferImpressora = record
  TamanhoBuffer: Word;
  Buffer: array [0..255] of Char;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
        Buff: TBufferImpressora;
        TestePasstrough: Integer;
        strFoo: string;
begin
  { Primeiro devemos checar se o "escape" PASSTHROUGH é suportado. Para isso, executamos a função "Escape" passando o QUERYESCSUPPORT. Caso o driver suporte, ela irá retornar um valor maior que 0 }
        TestePasstrough := PASSTHROUGH;
        if Escape(Printer.Handle, QUERYESCSUPPORT, SizeOf(PASSTHROUGH),
@TestePasstrough, nil) > 0 then
        begin
                { Inicializamos o driver }
                Printer.BeginDoc;
                 { Informação qualquer a enviar diretamente para a impressora }
                 strFoo := 'Passthrough string';
                { Copia da string para a estrutura }
                StrPCopy(Buff.Buffer, strFoo);
                { Indicamos o tamanho da informação}
                Buff.TamanhoBuffer := StrLen(Buff.Buffer);
                { Enviamos o "escape" }
                Escape(Printer.Canvas.Handle, PASSTHROUGH, 0, @Buff,nil);
                { Descarregamos... }
                Printer.EndDoc;
        end;
end;

end.
```

59) IMPRIMIR DIRETAMENTE PARA A IMPRESSORA SEM PASSAR PELO GERENCIADOR DE IMPRESSÃO

```
procedure TForm1.Button1Click(Sender: TObject);
var
        F : TextFile;
        i : integer;
begin
        AssignFile(F,&rsquo;LPT1');
        Rewrite(F);
        i := 0;
        Writeln(F,&rsquo;Teste de impressao - Linha 0');
        Writeln(F,&rsquo;Teste de impressao - Linha 1');
        Writeln(F,#27#15+&rsquo;Teste de Impressão - Linha 2');
        Writeln(F,&rsquo;Teste de impressao - Linha 3');
        Writeln(F,#27#18+&rsquo;Teste de Impressão - Linha 4');
        Writeln(F,&rsquo;Teste de impressao - Linha 5');
        Writeln(F,#12); // Ejeta a página
        CloseFile(F);
end;
```

60) CRIAR BARRA DE STATUS COM SUPORTE ÀS CAIXAS DE EDIÇÃO DO WINDOWS 95

No evento OnCreate do Form:
Application.OnHint := DisplayHint;
Crie (e defina) uma Procedure DisplayHint (no Form Principal):
StatusBar1.Panels[0].Text := Application.Hint;

61) CAPTURANDO UMA TELA DO WINDOWS

```
procedure TForm1.Button1Click(Sender: TObject);
var
        DeskTopDC: HDc;
        DeskTopCanvas: TCanvas;
        DeskTopRect: TRect;
begin
        DeskTopDC := GetWindowDC(GetDeskTopWindow);
        DeskTopCanvas := TCanvas.Create;
        DeskTopCanvas.Handle := DeskTopDC;
        DeskTopRect := Rect(0,0,Screen.Width,Screen.Height);
        Form1.Canvas.CopyRect(DeskTopRect,DeskTopCanvas,DeskTopRect);
        ReleaseDC(GetDeskTopWindow,DeskTopDC);
end;
```

62) COMPACTAR DE UMA TABELA PARADOX

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;
```

63) CONFIGURAÇÕES INTERNACIONAIS

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

```
DecimalSeparator :='.';

ShortFormatDate :='mm/dd/yy';
```

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

64) LISTANDO TODAS AS JANELAS ABERTAS

Para listas (pegar) todas as janelas abertas, deve-se usar a função API EnumWindows, que usa uma função Callback, com dois parâmetros, um Handle para a janela e um ponteiro. Você pode usá-la como 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;
```

65) PRIMEIRA LETRA DE UM EDITBOX MAIÚSCULA

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)));
          OnChange :=Edit1.Change;
        end;
  end;
```

66) DESLIGAR E LIGAR O MONITOR (OFF/ON)

Turn monitor off

```
SendMessage(Application.Handle, WM_SYSCOMMAND, SC_MONITORPOWER, 0);
```

Turn monitor on

```
SendMessage(Application.Handle, WM_SYSCOMMAND, SC_MONITORPOWER, -1);
```

67) INVERTENDO OS BOTÕES DO MOUSE

Adicionar à cláusula Uses, Shellapi.

Digitar os seguintes parâmetros:

Swapmousebutton(true) ;

Para voltar ao normal basta mudar para False.

68) MUDAR O PAPEL DE PAREDE DO WINDOWS

Primeiro deve-se adicionar à cláusula Uses, ShellApi.

Em seguida deve-se digitar os seguinte comandos:

```
procedure TForm1.FormCreate(Sender: TObject);
var
        Arquivo: String;
begin
        Arquivo:='c:\windows\nuvens.bmp';
        SystemParametersInfo(SPI_SetDeskWallPaper, 0, PChar(Arquivo), 0);
end;
```

69) ACESSANDO ARQUIVOS PARADOX EM REDE

Arquivos Paradox podem ser compartilhados em rede. Para que isto ocorra devemos:

Adicionar o Database Engine Configuration (BDE Config);
Selecionar a página Drivers;
Selecionar o driver PARADOX e alterar o parâmetro NET DIR para o local onde serão gravados os arquivos de controle para compartilhamento. Por exemplo, "G:\MEUAPLIC", onde G corresponde ao drive de rede e MEUAPLIC, o diretório onde está o aplicativo (executável);
Depois selecionar a página System;
Alterar o parâmetro LOCAL SHARE para TRUE. Após isto o BDE controlará o compartilhamento de arquivos Paradox em rede.

70) 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
        Width Edit1 do
                if Text <>'' then
                        Table1.FindNearest([Text]);
end;
```

71) INCLUIR MAIS DE UMA LINHA NO HINT

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 := &lsquo;Primeira Linha&rsquo;+#13+&rsquo;Segunda
Linha&rsquo;+#13+
      &lsquo;Terceira Linha&rsquo;+#13+&rsquo;Quarta Linha&rsquo;;
end;
```

Obs. Não esquecer de mudar para TRUE a propriedade ShowHint.

72) COMO SABER SE O APLICATIVO JÁ FOI ABERTO

Esta dica só funciona em Delphi 1.0 e não no win95, mas como bom micreiro, coloquei também a versão para W95, que segue abaixo:

No Projeto (.DPR):

```
uses  windows, messages,     // necessarios acrescentar
  Forms, .....

var
        Hwnd: THandle;   // variavel necessaria
begin
         Hwnd:=FindWindow('O seu TForm', 'O caption do seu form principal');
```

```
        if Hwnd = 0 then begin
                Application.Initialize;
        end
        else begin
                if not IsWindowVisible(Hwnd) then begin     // se minimizado
                        ShowWindow(Hwnd, SW_SHOWNORMAL);  //mostra
                        PostMessage(Hwnd, WM_USER, 0, 0);   //restaura
                end;
                SetForegroundWindow(Hwnd);     // visivel
end;
```

No Programa principal coloque:

```
private
        { Private declarations }
public
        procedure WMUser(var msg: TMessage); message WM_USER;  //definicao
        { Public declarations }
end;

implementation

{$R *.DFM}

procedure TPrincipal.WMUser(var msg: TMessage); //uso de fato.
begin
        Application.restore;
end;
```

73) 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.

74) VERIFICAR SISTEMA OPERACIONAL

```
unit sobreManager;

interface

uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
    Buttons, ExtCtrls;

type
  TfrmSobreManager = class(TForm)
    btnOK: TButton;
    ProductName: TLabel;
```

```
    Version: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Copyright: TLabel;
    Panel1: TPanel;
    Image2: TImage;
    GroupBox1: TGroupBox;
    Bevel1: TBevel;
    stOSVersao: TStaticText;
    stOSBuilder: TStaticText;
    stOS: TStaticText;
    stOSService: TStaticText;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  frmSobreManager: TfrmSobreManager;

implementation

{$R *.DFM}

// VERIFICA E APRESENTA AS INFORMAÇÕES do SISTEMA OPERACIONAL (FaC)

procedure TfrmSobreManager.FormCreate(Sender: TObject);
var
  verInfo : TOsVersionInfo;
  str     : String;
  I       : Word;
begin
  verInfo.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  if GetVersionEx(verInfo) then begin
    stOSVersao.Caption := 'Versão : '+ IntToStr(verInfo.dwMajorVersion) +
                          IntToStr(verInfo.dwMinorVersion);
    stOSBuilder.Caption := 'Compilação : '+IntToStr(verInfo.dwBuildNumber);
    str := 'Sistema Operacional : ';
    case verInfo.dwPlatformId of
      VER_PLATFORM_WIN32s : stOS.Caption := str +'Windows 95';
      VER_PLATFORM_WIN32_WINDOWS  : stOS.Caption := str +'Windows 95 Osr2 /
98';
      VER_PLATFORM_WIN32_NT : stOS.Caption := str +'Windows NT';
    end;

    str := '';
```

```
    for I := 0 to 127 do
      str := str + verInfo.szCSDVersion[I];
    stOSService.Caption := 'Informações Adicionais : '+ str;
  end
end;

end.
```

75) NOME DO USUÁRIO LOGADO NA REDE

Existe na API do BDE uma funcao chamada DbiGetNetUserName, que retorna o nome do usuário logado. Tente o seguinte:

```
function GetUserName:string;
var
        Nome: array[0..64] of char;
begin
        if DbiGetNetUserName(Nome) <> DBIERR_NONE then Nome:='';
                Result:=StrPas(Nome);
end;
```

76) CAPTURAR O NOME DAS TABELAS DE UM BANCO DE DADOS

Crie um Alias para o seu banco da dados(Fizemos com um banco do Interbase). Adicione o seguinte código ao seu programa(Neste caso o código foi feito no evento OnActivate do Form1):

```
    procedure TForm1.FormActivate(Sender: TObject);
    Var
        MyStringList: TStringList;
        i: Integer;
    begin
        try
                MyStringList:= TStringList.Create;
                Session.GetTableNames('SeuAlias','*.*',False, False, MyStringList);
                For i:= 1 To MyStringList.Count-1 do
                        ListBox1.Items.Add(MyStringList.Strings[i]);
        finally
                MyStringList.Free;
        end;
    end;
```

O método GetTableNames trabalha com cinco parâmetros:

1-Item do tipo String=Nome do Alias do banco desejado.

2-Item do tipo String=Especifique um filtro para retornar somente as tabelas desejadas. Podem incluir símbolos(Wildcards) como por exemplo '*'.
3-Item do tipo Boolean=Para pesquisas em Paradox e dBASE, defina True para incluir a extensão do arquivo como parte do nome da tabela.
Para tabelas SQL, defina False.
4-Item do tipo Boolean=Defina True para tabelas SQL para receber também o nome das tabelas de sistema as quais contém a estrutura dos dados. Defina False para tabelas Paradox e dBASE.
5-Item do tipo TStringList=Nome do StringList onde serão guardados os nomes das tabelas.

77) TABELA DOS CARACTERES ESPECIAIS UTILIZADOS COMO MÁSCARA:

Caracter Definições
! Faz com que a digitação da máscara fique parada no primeiro caracter, fazendo com que os caracteres digitados que se movam. Ex: !;0;_
> Todos os caracteres digitados serão convertidos para maiúsculas. Ex: >aaa;0;_
< Todos os caracteres digitados serão convertidos para minúsculas. Ex:
<> Anula o uso dos caractes > e <, ou seja, utilizado para cancelar a opção de máscara para os caracteres a direita. Ex: >aaa<>aaa;0;_
\ Utilizado para marcar determinado caracter não especial como fixo. Ex: !\(999\)000-0000;0;_
L Exige caracteres alfabéticos obrigatórios para a posição, do tipo A-Z, a-z. Ex: LLL;1;_
l Somente caracteres alfabéticos para a posição, mas não-obrigatórios, do tipo A-Z, a-z. Ex: lll;1;_
A Exige caracteres alfanuméricos obrigatórios para a posição, do tipo A-Z, a-z, 0-9. Ex: AAA;1;_
a Somente caracteres alfanuméricos para a posição, mas não-obrigatórios, do tipo A-Z, a-z, 0-9. Ex: aaa;1;_
C Requer um caracter obrigatório para a posição. Ex: CCC;1;_
c Permite o uso de qualquer caracter para a posição, limitando apenas o número de caracteres. Ex: ccc;1;_
0 Exige caracteres numéricos obrigatórios para a posição, do tipo 0-9. Ex: 000;1;_
9 Somente caracteres numéricos para a posição, não-obrigatórios, do tipo 0-9. Ex: 999;1;_
Somente caracteres numéricos para a posição e o uso dos sinais de - ou +, não-obrigatórios. Ex: ###;1;_
: Utilizado como separador de horas, minutos e segundos.
/ Utilizado como separador de dia, mês e ano.

78) TRADUZIR CAPTIONS E BOTÕES DA MESSAGEDLG

Para traduzir algumas as mensagens do Delphi que aparecem nos botões e nas caixas de avisos da função MessageDlg, você necessita dos arquivos de recursos do Delphi (*.RC).

Possuo o Delphi 2 - Developers, que vem acompanhado de alguns destes arquivos de recursos. No meu caso, eles estão gravados em C:\DELPHI2\SOURCE\VCL.

Os arquivos *.RC são arquivos "só texto", e contêm diversas mensagens utilizadas nos programas compilados no Delphi. O exemplo que se segue realiza alterações nos arquivos CONSTS.RC e DBCONSTS.RC. A alteração de outros arquivos "RC" pode ser feita de modo similar. Para maiores detalhes, envie-me um e-mail (paulosd@dglnet.com.br), ou, melhor ainda, consulte o livro "Dominando o Delphi" (edição para o Delphi 1), de Marcos Cantù, ed. Makron Books. O assunto "Usando recursos de tabelas de strings" está no capítulo 21, página 876. Não sei se a edição para o Delphi 2 cobre este assunto.

a) Faça uma cópia dos arquivos CONSTS.RC e DBCONSTS.RC em um diretório seguro, para o caso de algo sair errado.

b) Pelo mesmo motivo, faça uma cópia dos arquivos CONSTS.RES e DBCONSTS.RES, que estão no diretório LIB do Delphi. No meu caso, o diretório destes arquivos é C:\DELPHI2\LIB.

c) Use o Bloco de Notas para abrir e alterar os arquivos CONSTS.RC e DBCONSTS.RC. (O Edit também serviria; entretanto, para acentuação correta no Windows, o Bloco de Notas é melhor).

Você só deve alterar as strings que estão entre aspas. Não altere o nome das constantes, que estão no início de cada linha.
Por exemplo, localize o seguinte bloco, em CONSTS.RC:

```
SCancelButton, "Cancel"
SYesButton, "&Yes"
SNoButton, "&No"
```

Altere para:

```
SCancelButton, "Cancelar"
SYesButton, "&Sim"
SNoButton, "&Não"
```

Não é necessário alterar todas as mensagens. Se desejar, altere apenas aquelas que você utiliza em seus sistemas. Lembre-se de salvar as alterações efetuadas.

d) Acione o prompt do DOS, e execute do seguinte modo o compilador de recursos do Delphi 2 (BRC32.EXE), que está no diretório BIN do Delphi (no meu caso, C:\DELPHI2\BIN):

```
C:\DELPHI2\BIN\BRC32 -R CONSTS.RC
C:\DELPHI2\BIN\BRC32 -R DBCONSTS.RC
```

(No Delphi 1, o compilador tem o seguinte nome: BRC.EXE).

e) Os dois comandos anteriores irão gerar os arquivos CONSTS.RES e DBCONSTS.RES. Copie os dois "*.RES" para o diretório LIB do DELPHI (no meu caso C:\DELPHI2\LIB)

f) Crie uma aplicação no Delphi que utilize a função MessageDlg, e botões "BitBtn". Ao rodar o seu programa, as mensagens já devem aparecer traduzidas.

79) ÚLTIMO ACESSO DE UM ARQUIVO

```
unit Uultimoacesso;

{object Form1: TForm1
  Left = 230
  Top = 186
  Width = 435
  Height = 167
  Caption = 'Ultimo Acesso'
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -16
  Font.Name = 'Arial'
  Font.Style = []
  PixelsPerInch = 96
  TextHeight = 18
  object Label1: TLabel
    Left = 6
    Top = 11
    Width = 53
    Height = 18
    Caption = 'Arquivo'
  end
  object Label2: TLabel
    Left = 6
    Top = 58
    Width = 101
    Height = 18
    Caption = 'Último Acesso'
  end
  object EdArquivo: TEdit
    Left = 6
    Top = 28
    Width = 281
    Height = 26
    TabOrder = 0
  end
  object BtSeleciona: TButton
    Left = 226
    Top = 82
```

```
    Width = 87
    Height = 31
    Caption = 'Seleciona'
    TabOrder = 1
    OnClick = BtSelecionaClick
  end
  object EdUltimoAcesso: TEdit
    Left = 6
    Top = 82
    Width = 204
    Height = 26
    TabOrder = 2
  end
  object ODSelecionaArquivo: TOpenDialog
    Left = 352
    Top = 8
  end
end
}

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls;

type
  TForm1 = class(TForm)
    EdArquivo: TEdit;
    BtSeleciona: TButton;
    Label1: TLabel;
    Label2: TLabel;
    EdUltimoAcesso: TEdit;
    ODSelecionaArquivo: TOpenDialog;
    procedure BtSelecionaClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

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

```
var
  FileHandle    : THandle;
  LocalFileTime : TFileTime;
  DosFileTime   : DWORD;
  LastAccessdTime : TDateTime;
  FindData      : TWin32FindData;
  NomeArquivo   : array[0..255] of char;
begin
  if OdSelecionaArquivo.Execute then
  begin
    EdArquivo.Text := OdSelecionaArquivo.FileName;
    StrPCopy(NomeArquivo,OdSelecionaArquivo.FileName);
    FileHandle := FindFirstFile(NomeArquivo, FindData);
    if FileHandle = INVALID_HANDLE_VALUE then
    begin
      Windows.FindClose(Handle);
      if (FindData.dwFileAttributes and FILE_ATTRIBUTE_DIRECTORY) = 0 then
      begin
        FileTimetoLocalFileTime(FindData.ftLastWriteTime, LocalFileTime);
        FileTimeToDosDateTime(LocalFileTime, LongRec(DosFileTime).Hi,
                              LongRec(DosFileTime).Lo);
        LastAccessdTime := FileDateToDateTime(DosFileTime);
        EdUltimoAcesso.Text := DateTimeToStr(LastAccessdTime);
      end;
    end;
  end;
end;
end.
```

80) OBTENDO AS INFORMAÇÕES DE VERSÃO DOS ARQUIVOS

Um dos recursos disponibilizados pelo Delphi é a customização das informações de versão a serem "anexadas" na linkagem.

Pouco utilizado, este recurso é muito interessante, pois possibilita o cadastro de diversas informações sobre o arquivo gerado, como: número de versão, nome do produto, nome interno do arquivo, nome da empresa, etc.

Podemos alterar as informações na página "Version Info", da página "Project Options":

Atenção com o item "Auto-increment build number": ele só será incrementado automaticamente quando for executada a opção "Build All" para compilar o projeto.

Porém, não existem rotinas "prontas" para obtermos estas informações. É necessário fazermos chamadas diretamente a API Win32, mais espeficamente, para as funções como a "GetFileVersionInfo" e a "VerQueryValue".

Abaixo encontramos uma função, a "FileVerInfo", que exemplifica o processo de obtenção das informações. Ela irá retornar "True" caso o arquivo informado no parâmetro "FileName" possuir as informações de versão, e devolverá por referência um "TStringList" contendo as informações.

//Código

```
uses
   Windows, Messages, SysUtils, Classes, Graphics, Controls,
   Forms, Dialogs, StdCtrls;
type
   TForm1 = class(TForm)
     Memo1: TMemo;
     Button1: TButton;
     procedure Button1Click(Sender: TObject);
   end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
function FileVerInfo(const FileName: string; var FileInfo: TStringList): Boolean;

const
    Key: array[1..9] of string =('CompanyName', 'FileDescription', 'FileVersion', 'InternalName', 'LegalCopyright', 'OriginalFilename', 'ProductName', 'ProductVersion', 'Comments');
  KeyBr: array [1..9] of string = ('Empresa', 'Descricao', 'Versao do Arquivo', 'Nome Interno', 'Copyright', 'Nome Original do Arquivo', 'Produto', 'Versao do Produto', 'Comentarios');
var
  Dummy : THandle;
  BufferSize, Len : Integer;
  Buffer : PChar;
  LoCharSet, HiCharSet : Word;
  Translate, Return : Pointer;
  StrFileInfo, Flags : string;
  TargetOS, TypeArq : string;
  FixedFileInfo : Pointer;
  i : Byte;

begin
  Result := False;
  { Obtemos o tamanho em bytes do "version  information" }
  BufferSize := GetFileVersionInfoSize(PChar(FileName), Dummy);
  if BufferSize <> 0 then
```

```
  begin
    GetMem(Buffer, Succ(BufferSize));
    try
      if GetFileVersionInfo(PChar(FileName), 0, BufferSize,
        Buffer) then
      { Executamos a função "VerQueryValue" e conseguimos informações sobre o
idioma /character-set }
      if VerQueryValue(Buffer, '\VarFileInfo\Translation', Translate, UINT(Len)) then
      begin
        LoCharSet := LoWord(Longint(Translate^));
        HiCharSet := HiWord(Longint(Translate^));
        for i := 1 to 9 do
        begin
          { Montamos a string de pesquisa }
          StrFileInfo  :=  Format('\StringFileInfo\0%x0%x\%s',  [LoCharSet,  HiCharSet,
Key[i]]);
          { Adicionamos cada key pré-definido }
          if VerQueryValue(Buffer,PChar(StrFileInfo), Return, UINT(Len)) then
          FileInfo.Add(KeyBr[i] + ': ' + PChar(Return));
        end;
        if VerQueryValue(Buffer,'\',FixedFileInfo, UINT(Len))
          then
          with TVSFixedFileInfo(FixedFileInfo^) do
        begin
          Flags := '';
          {Efetuamos um bitmask e obtemos os "flags" do arquivo}
        if (dwFileFlags and VS_FF_DEBUG) = VS_FF_DEBUG then
            Flags := Concat(Flags,'*Debug* ');
        if (dwFileFlags and VS_FF_SPECIALBUILD) = VS_FF_SPECIALBUILD then
              Flags := Concat(Flags, '*Special Build* ');
        if (dwFileFlags and VS_FF_PRIVATEBUILD) = VS_FF_PRIVATEBUILD then
              Flags := Concat(Flags, '*Private Build* ');
        if (dwFileFlags and VS_FF_PRERELEASE) = VS_FF_PRERELEASE then
              Flags := Concat(Flags, '*Pre-Release Build* ');
        if (dwFileFlags and VS_FF_PATCHED) = VS_FF_PATCHED then
               Flags := Concat(Flags, '*Patched* ');
        if Flags <> '' then FileInfo.Add('Atributos: ' + Flags);
        TargetOS := 'Plataforma (OS): ';
        { Plataforma }
        case dwFileOS of
          VOS_UNKNOWN :
            TargetOS := Concat(TargetOS, 'Desconhecido');
          VOS_DOS :
            TargetOS := Concat(TargetOS, 'MS-DOS');
          VOS_OS216 :
            TargetOS := Concat(TargetOS, '16-bit OS/2');
          VOS_OS232 :
            TargetOS := Concat(TargetOS, '32-bit OS/2');
          VOS_NT    :
```

```
          TargetOS := Concat(TargetOS, 'Windows NT');
        VOS_NT_WINDOWS32, 4:
          TargetOS := Concat(TargetOS, 'Win32 API');
        VOS_DOS_WINDOWS16:
          TargetOS := Concat(TargetOS, '16-bit Windows ',
          'sob MS-DOS');
        else
           TargetOS := Concat(TargetOS, 'Fora do Padrão. Código: ', IntToStr(dwFileOS));
      end;

      FileInfo.Add(TargetOS);
      TypeArq := 'Tipo de Arquivo: ';
      { Tipo de Arquivo }
      case dwFileType of
        VFT_UNKNOWN :
          TypeArq := Concat(TypeArq,'Desconhecido');
        VFT_APP : TypeArq := Concat(TypeArq,'Aplicacao');
        VFT_DLL : TypeArq := Concat(TypeArq,'Dynamic-Link Lib.');
        VFT_DRV : begin
          TypeArq := Concat(TypeArq,'Device driver - Driver ');
          case dwFileSubtype of VFT2_UNKNOWN : TypeArq := Concat (TypeArq,
'Desconhecido');
            VFT2_DRV_PRINTER : TypeArq := Concat(TypeArq,'de Impressao');
            VFT2_DRV_KEYBOARD : TypeArq := Concat(TypeArq,'de Teclado');
            VFT2_DRV_LANGUAGE : TypeArq := Concat(TypeArq,'de Idioma');
            VFT2_DRV_DISPLAY : TypeArq := Concat(TypeArq,'de Vídeo');
            VFT2_DRV_MOUSE : TypeArq := Concat(TypeArq,'de Mouse');
            VFT2_DRV_NETWORK : TypeArq := Concat(TypeArq,'de Rede');
            VFT2_DRV_SYSTEM : TypeArq := Concat(TypeArq,'de Sistema');
            VFT2_DRV_INSTALLABLE : TypeArq := Concat(TypeArq,'Instalavel');
            VFT2_DRV_SOUND : TypeArq := Concat(TypeArq,'Multimida');
          end;
        end;
        VFT_FONT : begin
          TypeArq := Concat(TypeArq,'Fonte - Fonte ');
          case dwFileSubtype of VFT2_UNKNOWN : TypeArq := Concat(TypeArq,
'Desconhecida');
            VFT2_FONT_RASTER : TypeArq := Concat(TypeArq,'Raster');
            VFT2_FONT_VECTOR : TypeArq := Concat(TypeArq,'Vetorial');
            VFT2_FONT_TRUETYPE : TypeArq := Concat(TypeArq,'TrueType');
          end;
        end;
        VFT_VXD : TypeArq := Concat(TypeArq,'Virtual Device');
        VFT_STATIC_LIB
          : TypeArq := Concat(TypeArq,'Static-Link Lib.');
      end;
      FileInfo.Add(TypeArq);
    end;
  end;
```

```
    finally
      FreeMem(Buffer, Succ(BufferSize));
      Result := FileInfo.Text <> '';
    end;
  end;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
   StrLst: TStringList;
begin
  StrLst := TStringList.Create;
  try
    FileVerInfo('C:\WINDOWS\SYSTEM\TAPI.DLL', StrLst);
    Memo1.Lines.Assign(StrLst);
  finally
    StrLst.Free;
  end;
end;
```

81) ACERTA PADRÃO DE DATA

```
procedure AcertaPadraoData;

const

  arrShortDayNames: array[1..7] of string[3]    =
('Dom','Seg','Ter','Qua','Qui','Sex','Sab');
  arrLongDayNames: array[1..7] of string[15]    =
('Domingo','Segunda','Terça','Quarta','Quinta', 'Sexta', 'Sábado');
  arrShortMonthNames: array[1..12] of string[3] =
('Jan','Fev','Mar','Abr','Mai','Jun','Jul','Ago',
'Set','Out','Nov','Dez');
  arrLongMonthNames: array[1..12] of string[15] = ('Janeiro','Fevereiro',
'Março','Abril','Maio',
'Junho','Julho','Agosto', 'Setembro','Outubro','Novembro','Dezembro');

var
        ii: integer;

begin
        ShortDateFormat := 'dd/mm/yyyy';
        DecimalSeparator := '.';
        ThousandSeparator := ',';

        for ii := 1 to 7 do begin
                ShortDayNames[ii] := arrShortDayNames[ii];
                LongDayNames[ii] := arrLongDayNames[ii];
```

```
        end;

        for ii := 1 to 12 do begin
                ShortMonthNames[ii] := arrShortMonthNames[ii];
                LongMonthNames[ii] := arrLongMonthNames[ii];
        end;

end;
```

82) COMO COLOCAR UM BITMAP NUM COMBOBOX

-Ajuste a propriedade Style do ComboBox para csOwnerDrawVariable.

```
var

  Form1: TForm1;
  Bmp1, Bmp2, Bmp3: TBitmap;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);

begin

        Bmp1:=TBitmap.Create;
        Bmp.Loadfromfile('c:\chip16.bmp');
        Bmp1:=TBitmap.Create;
        Bmp.Loadfromfile('c:\zoom.bmp');
        Bmp1:=TBitmap.Create;
        Bmp.Loadfromfile('c:\disk.bmp');
        ComboBox1.Items.AddObject('Chip',Bmp1);
        ComboBox1.Items.AddObject('Zoom',Bmp2);
        ComboBox1.Items.AddObject('Disk',Bmp3);

end;

procedure TForm1.ComboBox1DrawItem(Control: TWinControl; Index: Integer; Rect:
TRect; State: TOWnerDrawState);

var
        Bitmap: TBitmap;
        Offset: Integer;

begin

        with (Control as TComboBox).Canvas do begin
```

```
                FillRect(Rect);
                Bitmap:= TBitmap(ComboBox1.Items.Objects[index]);
                if Bitmap nil then begin
                    BrushCopy(Bounds(Rect.Left + 2, Rect.Top + 2, Bitmap.Width,
                    Bitmap.Height), Bitmap, Bounds(0, 0, Bitmap.Width, Bitmap.Height),
clRed);
                    Offset: Bitmap.width + 8;
                end;
                TextOut(Rect.Left + Offset, Rect.Top, ComboBox1.Items[index]);
        end;

end;

procedure TForm1.ComboBox1MeasureItem(Control: TWinControl; Index: Integer; var
Height: Integer);
begin
        Height:=20;
end;
```

83) ADICIONANDO UM BOOKMARKS

Bookmarks permitem ao programador "memorizar" um determinado local da tabela para que possa retornar mais tarde, é muito simples e fácil de usar já que existem apenas três métodos que lhe permitem utilizar este recurso.

Para marcar um determinado local em uma tabela necessitamos criar uma nova instancia de TBookmark e executar o método GetBookMark de uma TTable.

```
var
        bmLocalImportante : TBookmark;
begin
        bmLocalImportante := table.GetBookMark;
```

{Para retornar a este local em particular a qualquer momento deve-se utiliza o método GotoBookMark() , este método recebe como parâmetro o TBookmark recebido como retorno de GetBookMark.}

```
        table1.GotoBookMark(bmLocalImportante);
```

{Após utilizar o Bookmark para o que desejar é importante que a memória utiliza por este recurso seja novamente liberada para o sistema,para executar esta operação utilize o método FreeBookMark.}

```
        table1.FreeBookMark(bmLocalImportante);
```

{Podem ser criados vários Bookmarks para uma mesma tabela, sendo este numero limitado apenas pela quantidade de memória livre no equipamento.}

end;

OBS:*Mas cuidado com o uso indevido deste recurso, cada instancia de TBookMark reserva uma determinada porção de memória que só será novamente liberada para ser reutilizada após a execução de um FreeBookmark. Se vários Bookmarks forem criados e não liberados podem comprometer a execução do programa.*

84) INSERINDO UM COMBOBOX NUM DBGRID

1. insira um Datasource, um DBGrid e dois Table's no form

2. link o Table1 com Datasource1 e DBGrid1

3. defina um banco de dados, uma tabela e ative o Table1

4. defina também para o Table2, mas use uma tabela diferente

5. adicione todos os campos do Table1 através do Fields Editor

6. mude a propriedade Visble para False do campo do Combobox

7. dê um clique com o botão direito do mouse sobre o Fields Editor e escolha New Field...

8. especifique os parametros para o novo campo
   a) Name: <algum nome>
   b) Type: <tipo do campo>
   c) Size: <tamanho>
   d) Field type: Lookup
   e) Key Field: <campo que receberá o valor escolhido no combobox&
   f) DataSet: Table2
   g) LookUpKeys: <campo listado no combo>
   h) Result Field: <campo que será mostrado para o usuário no Combobox>

9. Execute a aplicação.

85) COMO CONECTAR UMA UNIDADE DE REDE

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

var
        NRW: TNetResource;
begin
        with NRW do
```

```
        begin
                dwType := RESOURCETYPE_ANY;
                lpLocalName := 'G:';
                lpRemoteName := '\\servidor\c';
                lpProvider := '';
        end;
        WNetAddConnection2(NRW, 'MyPassword', 'MyUserName',
CONNECT_UPDATE_PROFILE);

end;
```

86) CONFIGURAR UMA REDE NOVELL

Configurar uma rede Novell para trabalhar com Delphi é muito facíl, mais facíl que um rede ponto a ponto como Windows 95 ...

Primeiro precisamos de um diretório onde será criado o NET FILE do Paradox, normalmente um diretório partindo do raiz do servidor será o suficiente, vamos usar o drive padrão da Novell para exemplificar o caso ...

F: <-- drive da rede

F:\NETDIR <-- o diretório do NET FILE

Para o diretório onde serão armazenados os arquivos do seus sistema não é necessário nenhum cuidado especial, podemos assumir alguma configuração como a que segue ...

F:\SISTEMA\DADOS <-- diretório de dados

Para quem trabalha todos os dias com uma rede Novell existe uma armadilha que pode causar umas noites mal dormidas :(...

Uma das coisas que muitos tentam para economizar um pouco do HD local é colocar o executável na rede, sem grandes problemas desde que você tenha um cuidado básico. Sempre que você executa um programa que trata com base Paradox o PRIVATE DIR fica sendo o diretório onde este executável se encontra, no caso de você colocar este executável em um diretório da rede para ser acessado por muitos como fica este diretório PRIVATE ??? jé não é mais tanto PRIVATE né ...

Para resolver este problema você deve "programaticamente" alterar o Private Dir para um diretório local,para isso você tem de adicionar as seguintes linhas de código ao iniciar a sua aplicação.

```
  Session.PrivateDir := 'C:\WINDOWS\TEMP';
```

Para ter acesso a Session voce tem de incluir a unit DB na clausula

Uses do seu projeto.

Este cuidado pode prevenir varias travadas sem razão aparente em redes Novell.

87) CONFIGURAÇÃO DE REDE WINDOWS 95/98 COM DELPHI

Vamos exemplificar com tres maquinas, 1 servidor chamado SERV e duas maquinas clientes. Claro que isto pode ser incrementado de acordo com suas necessidades :).

No servidor:

diretório real D:\SISTEMA\DADOS

compartilhe o subdiretorio D:\SISTEMA\DADOS como um recurso chamado Servidor_H

no AUTOEXEC.BAT incluir → Subst H: D:\SISTEMA\DADOS

Nas maquinas clientes:

mapear drive de rede H: como \\SERV1\Servidor_H

No fim deste processo você terá o drive H: como sendo o seu Drive de rede para o Sistema, este drive estará presente em todas as maquinas e pode ser utilizado como seu NET DIR.

No BDE Configuration:

NET DIR: H: (Para o driver Paradox)

Em seu Alias:

Path: H:\DADOS\

Em System:

Local Share: True

Depois disto você configurar o BDE de forma igual em todas as maquinas da rede.

88) CRIAR UM ARQUIVO EM TEMPO DE EXECUÇÃO

Criar um arquivo em tempo de execução é relativamente simples, você tem que criar uma instancia do objeto TTable, esse objeto (de uma lida no Help TTable e suas propriedades e metodos) tem um metodo de criação e um de Criar tabela.

Depois disso é só definir as propriedades da nova tabela ...

```
DatabaseName := 'c:\lista';
TableName := 'Produtos.dbf';
TableType := ttDbase;
```

os campos da tabela ...

```
   Add('codigo', ftString,7, false);
   Add('Nome', ftString, 45, false);
```

e os indices ...

```
   Add('prod1', 'codigo', []);
   Add('prod2', 'Fornecedor', []);
```

com todos os dados devidamente setados ...

```
   CreateTable;
```

```
Procedure TMainForm.Inicializa;

var
        Table1 : TTable;
begin

  { Criar componente TTable }
  Table1 := TTable.create(Application);

  { Definições de Campos e criação do arquivo }

  with Table1 do begin
        DatabaseName := 'c:\lista';
        TableName := 'Produtos.dbf';
        TableType := ttDbase;
        with FieldDefs do begin
                Clear;
                Add('codigo', ftString,7, false);
                Add('Nome', ftString, 45, false);
                Add('Fornecedor', ftString, 5,false );
                Add('Custo', ftCurrency, 0, false );
                Add('Venda', ftCurrency, 0, false );
        end;
        with IndexDefs do begin
                Clear;
                Add('prod1', 'codigo', []);
                Add('prod2', 'Fornecedor', []);
        end;
        CreateTable;
  end;

end;
```

Utilizando o tipo ftCurrency, formato de valores do sistema financeiro o Delphi cria um campo Dbase com N,20,4

89) CONTROLE SOBRE DIGITAÇÃO

Quando alguém esta digitando algum valor que posteriormente será utilizado para calculo alguns cuidados são necessários, esse procedimento ValidaKey deve ser ligado no OnChange do TDBEdit para checar qual foi a tecla digitada.

```
procedure ValidaKey(Const Sender:TObject; var key: char);
begin
        if not(key in ['0'..'9','.',',',#8,#13]) then key := #0;
        if key in [',','.'] then key := DecimalSeparator;
        if key = DecimalSeparator then
                if pos(key,TEdit(Sender).Text) <> 0 then key := #0;
end;
```

```
        if not(key in ['0'..'9','.',',',#8,#13]) then key := #0;
```

Se algum numero, ponto, virgula, BackSpace ou Enter for digitado então pode passar normalmente, caso contrario a tecla pressionada é ignorada.

```
        if key in [',','.'] then key := DecimalSeparator;
```

Se ponto ou virgula, assume como separador decimal.

```
         if key = DecimalSeparator then
                if pos(key,TEdit(Sender).Text) <> 0 then key := #0;
```

O separador decimal so pode ser digitado uma unica vez, na tentativa de uma segunda digitação ignora-se o símbolo.

Observem que o mais importante aqui é o conceito utilizado, o fato de se interceptar os caracteres digitados pelo usuário e poder filtrar esses caracteres para evitar uma entrada de dados inconsistente. O exemplo de numero e símbolos não é conclusivo, uma vez que o mesmo efeito poderia ter sido obtido com a aplicação de uma mascara.

90) CRIAR ARQUIVO DBF COM INDICES COMPOSTOS

Quem trabalha com DBF's já encontrou problemas para criar índices compostos, ou seja um índice que tenha mais de um campo, em tempo de execução.

Isto ocorre por dois motivos, embora exista na internet documentação sobre o assunto, dizendo ser possível, ainda não encontrei ninguém que tenha tido sucesso nesta tarefa. O outro motivo é a existência de um parâmetro não

documentado necessário para arquivos DBF's ixExpression que informa o Delphi que se trata de um índice com mais de um campo.

Como trabalhar com DBF's é comum para a maioria dos programadores que vem do Clipper acho interessante solucionar este problema da melhor forma possível.

Assim a criação do indice fica para logo depois da criação da tabela, como mostra o segmento de codigo abaixo.

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

procedure TForm1.Button1Click(Sender: TObject);

var
  Table1 : TTable;
begin

  { Criar arquivos }
  Table1 := TTable.create(Application);

  { Cria arquivo }
  with Table1 do begin
    Active := False;
    DatabaseName := 'C:\';
    TableName := 'teste';
    TableType := ttdBASE;
    with FieldDefs do begin
      Clear;
      Add('Name1', ftString, 20, False);
      Add('Name2', ftString, 20, False);
      Add('Name3', ftString, 20, False);
      Add('Name4', ftString, 20, False);
    end;
    CreateTable;
    AddIndex('Indice1', 'Name1 + Name2', [ixExpression]);
    AddIndex('Indice2', 'Name2 + Name3', [ixExpression]);
    AddIndex('Indice3', 'Name3 + Name4', [ixExpression]);
  end;

end;
```

Na pratica criar os índices com AddIndex() logo após CreateTable não vai influenciar em nada o seu programa.

91) SISTEMAS EM DELPHI PARA LINUX

Baixe na Internet um dos melhores compiladores de Pascal para DOS/OS2/Linux. Suporta praticamente todo o set de opções do Borland Pascal e vai além, dando suporte ao modelo de objetos do Object Pascal e otimiza pra 386/486/Pentium/Pentium Pro/MMX e Pentium II. O código é todo gerado para modo protegido e, devido a isso, não possui quaisquer limites para arrays ou ponteiros. Mas o melhor de tudo é que ele é GRATIS!!!

Para pega-lo vá o endereço abaixo:

http://www.brain.uni-freiburg.de/~klaus/fpc/

92) DESENHAR UM BITMAP NO FORMULÁRIO

```
var
  Form1: TForm1;
  Bmp: TBitmap;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
        Bmp:=TBitmap.Create;
        Bmp.Loadfromfile('c:\windows\nuvens.bmp');
end;

procedure TForm1.TForm1.FormPaint(Sender: TObject);
begin
        Canvas.Draw(50,50,Bmp);
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
        Bmp.Free;
end;
```

93) EVITANDO A SAÍDA DE FORMULÁRIO

No evento OnCloseQuerie do form escreva o seguinte código:

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
        CanClose:=False;
        if messagebox(handle,'Deseja realmente fechar esta janela ?', 'Aviso',
mb_IconInformation + mb_YesNo + mb_DefButton2 ) = idYes then
                CanClose := True;
end;
```

94) DESABILITANDO SIMULTANEAMENTE AS TECLAS (ALT + F4)

No evento OnCloseQuerie do form escreva o seguinte código:

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);
begin
        CanClose:=False;
end;
```

95) FILTRANDO REGISTROS

O caminho mais fácil e rápido para implementar filtros em tabelas é utilizando o evento OnFilterRecord, este evento é chamado sempre que um registro for lido do arquivo pelo componente TTable.

OnFilterRecord é declarado como do tipo TFilterRecordEvent

```
procedure(DataSet: TDataSet; var Accept: Boolean) of object;

property OnFilterRecord: TFilterRecordEvent;
```

A variável Accept deverá ser manipulada internamente em OnFilterRecord, assim para que o registro seja mostrado o valor de Accept tem de ser true que é o default.

Tradicionalmente aplicar um filtro a uma tabela, qualquer que seja a tabela é um processo muito lento, assim é interessante desenvolver rotinas rápidas e fáceis de serem executadas.

```
procedure TForm1.Table1FilterRecord(DataSet: TDataSet; var Accept: Boolean);
begin
        Accept := (Dataset as TTable).
         FieldByName('Dupr_valida').AsBoolean;

end;
```

Mas pode-se colocar varias verificações para decidir se um deter- minado registro deve ou não ser filtrado.

```
procedure TForm1.Table1FilterRecord(DataSet: TDataSet;  var Accept: Boolean);
begin
        with (Dataset as TTable) do
                if not ( (FieldByName('Dupr_Valida').AsBoolean) and
(FieldByName('Dupr_DataVenc').AsString = '05/09/97') ) then
                        accept := false;
end;
```

O filtro pode ser ligado ou desligado a qualquer momento setando a propriedade Filtered da tabela, o evento OnFilterRecord só é chamado se a propriedade Filtered estiver com o valor true.

96) COMO SABER SE UM FORM JÁ ESTA CRIADO

Bem saber se um form já esta ou não criado, melhor dizer instanciado, não é um problema muito critico, abaixo pode-se ver uma unit padrão criada com um form vazio no Delphi, esta unit alem de declarar o nova classe TForm1 cria também uma variável Form1 do tipo TForm1. Isto é muito importante que seja observado, um variável para um tipo "FORM" nada mais é que um ponteiro, ou seja ela apenas mostra em que local da memória esta a instancia do seu form, enquanto o seu form não existir este ponteiro deve apontar para lugar nenhum, ou seja, este ponteiro estará guardando o valor "NIL".

```
unit Unit1;

interface

uses

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

type

  TForm1 = class(TForm)

  private
    { Private declarations }
  public
    { Public declarations }
```

```
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

end.
```

{Partindo deste principio podemos verificar se um form foi ou não instanciado verificando o valor guardado em FORM1, se este valor for diferente de "NIL" significa que o Form já foi instanciado... bem a coisa não é assim tão simples, imagine que um amigo mudou-se para São Paulo e lhe passou seu novo endereço, você recebe e anota em sua agenda... depois de dois meses ele resolve que não quer mais morar em São Paulo e vai embora, ok o fato de seu amigo ir embora por si só não vai fazer com que o endereço dele se auto apague da sua agenda, assim sendo quando o form for destruído o ponteiro continuara guardando o endereço de memória onde o form estava e ai o nosso método de controle vai pro brejo :(

A forma mais limpa e automática para se contornar este problema nos obriga a codificar o seguinte no evento OnDestroy do Form ...}

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

begin
        Form1 := nil;
end;
```

assim, quando o "FORM1" for destruído ele apaga o seu endereço junto, ótimo não é mesmo ....

Assim quando for instanciar um form utilize a seguinte verificação ...

```
if Form2 = nil then
        Form2 := TForm2.Create(Self);
        Form2.Show;
```

... interessante que o Show quando um form já esta criado tem o efeito de um BringToFront.

Bem tudo isso resolve parte do problema, a outra parte tem de ser resolvida por você estruturando o seu programa de forma aos controles funcionem de acordo.

1 - Este controle não funciona para forms com múltiplas instancias, a não ser que você crie uma variável para cada instancia. Pessoalmente eu nunca usei isso, se um form pode ter múltiplas instancias em MDI então controle por ActiveMDIChild e se for SDI então não sei porque ter mais de uma instancia.

2 - Quando for criar um novo form não crie variáveis desnecessariamente, utilize a variável que já esta sendo criada na unit do Form.

Ex: Dados FORM1 e FORM2

Apenas FORM1 esta no AUTO-CREATE

Quando no uses de FORM1 for referenciada a unit UNIT2 a variável FORM2 estará acessível, use-a.

FORM2 := TFORM2.CREATE(SELF);

3 - Quando um Form é mostrado com ShowModal este tipo de controle não se aplica já que será impossível mostrar qualquer outro form.

97) NÃO REDIMENSIONAR O FORMULÁRIO

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;
```

98) CRIANDO FORMS DINAMICAMENTE (SDI)

Com certeza criar os forms que vc vai utilizar em tempo de execução é uma das grandes "sacadas" do Delphi, uma tarefa não tão complicada mas com varias alternativas de como ser executada.

Vamos supor inicialmente que vc esta trabalhando com uma aplicação SDI (Single Document Interface) ou documento de interface simples, utilizando essa forma de desenvolvimento vc tem janelas sobre janelas, o Delphi é um exemplo disso, para criar um form em tempo de execução siga os seguintes passos:

1- Crie um projeto novo (SDI), o Delphi automaticamente cria o Form1.
2- Crie um Form Novo, ele recebera o nome de Form2.
3- Va em Options/Project, vc vai ver dois list boxes, o da esquerda contêm os forms que devem ser criados automaticamente quando sua aplicação for iniciada, os dois forms devem estar ai, pois bem mova Form2 para o List Box da direita que deve conter os forms que ficam disponiveis porém não são automaticamente criados.
(importante) Nesse momento você retirou do Delphi a obrigação de criar o Form2,se vc não o criar e em algum momento fizer referencia a ele isso deve causar um erro.
4- Ok, esqueca temporariamente Form2, em Form1 crie um botao e digite o código abaixo em seu evento TForm1.Button1Click

```
implementation
```

```
{$R *.DFM}

uses Unit2;

procedure TForm1.Button1Click(Sender: TObject);
begin
        Form2 := TForm2.Create(self);
        Form2.Show;
end;

end.
```

5- Observe o uses criado logo após {$R *.DFM}, ele deve fazer referencia a Unit de Form2, que no caso do meu exemplo chama-se Unit2.

Pronto, com esses pequenos cuidados vc ja estara criando seus forms em tempo de execução :)

Agora uma dica interessante, se vc executar esse programa vc vai ver que quando Form2 for criado o usuario pode clicar em Form1 e esse recebera o foco, coisa que pode nao ser interessante.

Para evitar este tipo de comportamento da sua aplicação utilize ShowModal e não Show para chamar o segundo form...

```
Form2 := TForm2.Create(self);
Form2.ShowModal;
```

O ShowModal vai fazer com que a aplicação fique com Form2 permanentemente em primeiro plano, Form1 não poderá ser acessado até que Form2 sejá fechado.

99) CRIANDO FORMS DINAMICAMENTE (MDI)

A coisa muda um pouquinho quando voce esta trabalhando em um projeto MDI, basicamente o controle que voce tem de executar é o mesmo, mas as caracteristicas do projeto podem lhe trair ...

Primeiro que em um projeto MDI não podem existir forms não visiveis, ou sejá, assim que um determinado form é criado ele já se torna visivel,não é necessario um SHOW ou SHOWMODAL para fazer isso, portanto não existe outro caminho senão criar todos os forms de sua aplicação em RUN TIME.

Todos os passos descritos no exemplo acima são validos aqui também, mas para que o seu projeto caracterize-se como MDI você tem de mudar a propriedade FormStyle dos forms do projeto como segue :

- O form principal da aplicação tem de estar como fsMDIForm.
- Os outros forms que fazem parte da aplicação como fsMDIChild.

Algum form especial, como o form onde o usuario configura a impressora, deve ser deixado como fsNormal.

Vá até Project/Options e deixe apenas o form principal como AutoCreate.

(importante) Nesse momento vc retirou do Delphi a obrigação de criar o Form2,se vc não o criar e em algum momento fizer referencia a ele isso deve causar um erro.

Pois bem, como uma das caracteristicas de uma aplicação MDI é o form principal conter os outros forms não podemos ficar colocando botões para testar a criação dinamica desses forms, melhor utilizar um componente do tipo Menu e codificar a chamada aos outros forms nele ...

implementation

*{$R *.DFM}*

uses Unit2;

procedure TForm1.Form21Click(Sender: TObject);
begin
Form2 := TForm2.Create(self);
end;

end.

Uma das primeiras coisas diferentes é que não é mais necessário usar o Show :) ...

Alguns cuidados devem ser tomados quando trabalhamos com uma aplicação MDI.

Primeiro não existe porque ficar discutindo como evitar que o usuário fique abrindo muitos forms ao mesmo tempo, já que está é a principal qualidade de um projeto MDI, caso voce não queira este tipo de comportamento pare de ler e volte para SDI ...

100) DEFINIDO O TAMANHO MÍNIMO E MÁXIMO DE UM FORM

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private declarations }
    procedure WMGetMinMaxInfo(var MSG: TMessage); message WM_GetMinMaxInfo;
  public
    { Public declarations }
  end;
```

```
var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.WMGetMinMaxInfo(var MSG: TMessage);
begin
        inherited;
        with PMinMaxInfo(MSG.lparam)^ do begin
                ptMinTRackSize.X := 300;
                ptMinTRackSize.Y := 150;
                ptMaxTRackSize.X := 350;
                ptMaxTRackSize.Y := 250;
        end;
end;

end.
```