Módulo 4

Object Pascal II

ESTRUTURAS DE DECISÃO

If

O if é uma estrutura de decisão usada para realizar instruções em determinadas condições. O if é considerado uma só instrução, por isso, só encontramos o ponto-e-vírgula no final. O else é opcional.

```
if Opn.Execute then
  Img.Picture.LoadFromFile(Opn.FileName);

if Nota < 5 then
  ShowMessage('Reprovado')
else
  ShowMessage('Aprovado');
```

Case

Permite que o fluxo da execução seja desviado em função de várias condições de acordo com o valor do argumento, que tem que ser ordinal, caso o valor do argumento não corresponda a nenhum dos valores listados, podemos incluir um else.

```
case Ch of
  ' ': ShowMessage('Espaço');
  '0'..'9': ShowMessage('Dígito');
  '+', '-', '*', '/': ShowMessage('Operador');
else
  ShowMessage('Caractere especial');
end;

case CbbBorda.ItemIndex of
  0: BorderStyle := bsDialog;
  1: BorderStyle := bsSingle;
  2: BorderStyle := bsSizeable;
end;
```

ESTRUTURAS DE REPETIÇÃO

While

O laço while executa uma instrução até que uma condição seja falsa.

```
I := 10;
while I >= 0 do
begin
  ShowMessage(IntToStr(I));
  Dec(I);
end;
```

For

O laço for executa uma instrução um número determinado de vezes, incrementando uma variável de controle automaticamente a cada iteração. Caso seja preciso que a contagem seja decremental, pode-se usar downto em vez de to.

```
for I := 1 to ComponentCount do
  ShowMessage('O ' + IntToStr(I) + 'º Componente é ' + Components[I - 1].Name);
```

```
for I := Length(Edit1.Text) downto 1 do
  ShowMessage(Edit1.Text[I]);
```

Repeat

O laço repeat executa instruções até que uma condição seja verdadeira.

```
I := 1;
repeat
  S := InputBox('Acesso', 'Digite a senha', '');
  Inc(I);
  if I > 3 then
    Halt;
until S = 'fluminense';
```

Quebras de Laço

Em qualquer um dos laços mostrados podemos usar o procedimento Break para cancelar a repetição e sair do laço, podemos também forçar a próxima iteração com o procedimento Continue.

```
I := 1;
while true do
begin
  Inc(I);
  if I < 10000000 then
    Continue;
  ShowMessage('Chegamos a dez milhões');
  Break;
end;
```

TIPOS DEFINIDOS PELO USUÁRIO

O usuário também pode declarar tipos não definidos pelo Delphi. Essas declarações são feitas na seção type, da interface ou implementation, sendo que na implementation esses tipos não poderão ser usados em outras Units. Dificilmente você terá que definir tipos, a não ser classes, pois os tipos padrão do Delphi são o bastante para a maioria das aplicações.

Strings Limitadas

Caso se deseje limitar o número de caracteres que uma string pode receber, podemos criar um tipo de string limitada.

```
TNome = string[40];
TEstado = string[2];
```

Tipo Sub-Faixa

É um subconjunto de um tipo ordinal e possui as mesmas propriedades do tipo original.

```
TMaiusculas = 'A'..'Z';
TMes = 1..12;
```

Enumerações

Define uma seqüência de identificadores como valores válidos para o tipo. A cada elemento da lista de identificadores é associado internamente um número inteiro, iniciando pelo número 0, por isso são chamados de tipos enumerados.

```
TBorderIcon = (biSystemMenu, biMinimize, biMaximize, biHelp);
TDiaSemana = (Seg, Ter, Qua, Qui, Sex, Sab, Dom);
```

Ponteiros

Ponteiros armazenam endereços de memória, todas as classes em Object Pascal são implementadas como ponteiros, mas raramente o programador vai precisar usá-los como tal.

```
TIntPtr: ^Integer;
```

Records

O tipo record é uma forma de criar uma única estrutura com valores de diferentes tipos de dados. Cada um dos dados de um record é chamado de campo.

```
TData = record
  Ano: Integer;
  Mes: TMes;
  Dia: Byte;
end;

var
  Festa: TData;
begin
  Festa.Ano := 1997;
  Festa.Mes := Mai;
  Festa.Dia := 8;
end;
```

Arrays

Arrays fornecem uma forma de criar variáveis que contenham múltiplos valores, como em uma lista ou tabela, cujos elementos são do mesmo tipo. Veja abaixo alguns exemplos de arrays de dimensões variadas.

```
TTempDia = array [1..24] of Integer;
TTempMes = array [1..31, 1..24] of Integer;
TTempAno = array [1..12, 1..31, 1..24] of Integer;

var
  TD: TTempDia;
  I: Integer;
begin
  for I := 1 to 24 do
    TD[I] := StrToIntDef(InputBox('Temperaturas', 'Digite a temperatura na hora '
      + IntToStr(I), ''), 30);
end;
```

Um array pode ser definido como constante tipada, onde todos os seus elementos devem ser inicializados.

```
FAT: array[1..7] of Integer = (1, 2, 6, 24, 120, 720, 5040);
```

O tipo dos elementos de um array pode ser qualquer um, você pode ter uma array de objetos, de conjuntos, de qualquer tipo que quiser, até mesmo um array de arrays.

```
TTempMes = array [1..31] of TTempDia;
TBtnList = array [1..10] of TButton;
```

Sets

São conjuntos de dados de um mesmo tipo, sem ordem, como os conjuntos matemáticos. Conjuntos podem conter apenas valores ordinais, o menor que um elemento pode assumir é zero e o maior, 255.

```
TBorderIcons = set of TBorderIcon;
BorderIcons := [biSystemMenu, biMinimize];

if MesAtual in [Jul, Jan, Fev] then
  ShowMessage('Férias');
```

Os conjuntos podem ser definidos como constantes ou constantes tipadas, como abaixo.

```
DIG_HEXA = ['0'..'9', 'A'..'Z', 'a'..'z'];
DIG_HEXA: set of Char = ['0'..'9', 'A'..'Z', 'a'..'z'];
```

PROCEDIMENTOS, FUNÇÕES E MÉTODOS

As ações de um objeto devem ser definidas como métodos. Quando a ação não pertence a um objeto, como uma transformação de tipo, essa ação deve ser implementada em forma de procedimentos e/ou funções.

Procedimentos

Procedimentos são sub-rotinas, que realizam uma tarefa e não retornam um valor. A declaração de um procedimento é feita na seção interface e a definição, na seção implementation. Ao chamar o identificador do procedimento, com os parâmetros necessários, esse procedimento será executado. Veja abaixo o exemplo de uma unit com a implementação um procedimento.

```
unit Tools;

interface
  procedure ErrorMsg(const Msg: string);
implementation

uses Forms, Windows;

procedure ErrorMsg(const Msg: string);
begin
  Application.MessageBox(PChar(Msg), 'Operação inválida', MB_ICONERROR);
end;

end.
```

Funções

Funções são muito semelhantes a procedimentos a única diferença é que as funções retornam um valor. O tipo do valor de retorno deve ser informado no cabeçalho da função. Na implementação da função deve-se atribuir o valor de retorno à palavra reservada Result ou ao identificador da função. Pode-se então usar a função em expressões, atribuições, como parâmetros para outras funções, em qualquer lugar onde o seu valor possa ser usado.

```
function Average(A, B: Double): Double;
begin
  Result := (A + B) / 2;
end;
```

Métodos

Métodos são funções ou procedimentos que pertencem a alguma classe, passando a fazer parte de qualquer objeto dessa classe. Na implementação de um método precisamos indicar qual a classe à qual ele pertence. Para chamar um método em algum lugar não pertencente à sua classe, como procedimentos, funções ou métodos de outras classes, deve ser indicado o objeto que deve executar o método. Os métodos usam os mesmos níveis de encapsulamento dos atributos.

```
type
  TFrmMsg = class(TForm)
    LblMsg: TLabel;
    BtnOk: TButton;
    ImgMsg: TImage;
  public
    procedure ShowMsg(const Msg: string);
  end;

procedure TFormMsg.ShowMsg(const Msg: string);
begin
  LblMsg.Caption := Msg;
  ShowModal;
end;
```

Parâmetros

Existem três tipos de passagem de parâmetros, que devem ser indicados na declaração da função ou procedimento. Parâmetros de tipos diferentes devem ser separados por ponto e vírgula.

```
function MultiStr(const S: string;  N: Double; var Erro: Integer): string;
```

Quando não é indicado o tipo de passagem, é passado o valor do parâmetro, como constante.

Ao usar a palavra-chave var, não será enviado o valor do parâmetro e sim uma referência a ele, tornando possível mudar o valor do parâmetro no código do procedimento.

Como alternativa você pode passar um parâmetro por referência constante, para isso use a palavra const antes da declaração do parâmetro.

With

Usado para facilitar o acesso às propriedades e métodos de um objeto.

```
with Edt do
begin
  CharCase := ecUpperCase;
  MaxLenght := 10;
  PasswordChar := '*';
  Text := 'Brasil';
end;
```

Self

Self é usado quando se quer referenciar a instância atual da classe. Se você precisar referenciar a instância atual de uma classe, é preferível usar Self em vez de usar o identificador de um Objeto, isso faz com que o código continue funcionando para as demais instâncias da classe e em seus descendentes.

CRIANDO E DESTRUINDO OBJETOS

Antes de tudo, você deve declarar o objeto, se quiser referenciá-lo. Para criá-lo, use o método Create, que é um método de classe. Para você usar um método de classe, referencie a classe, não o Objeto, como mostrado abaixo.

```
var
  Btn: TBitBtn;
begin
  Btn := TBitBtn.Create(Self);
  With Btn do begin
    Parent := Self;
    Kind := bkClose;
    Caption := '&Sair';
    Left := Self.ClientWidth - Width - 8;
    Top := Self.ClientHeight - Height - 8;
  end;
end;
```

Porém, se você não precisar referenciar o Objeto, poderia criar uma instância sem referência.

```
with TBitBtn.Create(Self) do
begin
  Parent := Self;
  Kind := bkClose;
  Caption := '&Sair';
  Left := Self.ClientWidth - Width - 8;
  Top := Self.ClientHeight - Height - 8;
end;
```

Para destruir um objeto, use o método Free. Para Forms, é recomendado usar o Release, para que todos os eventos sejam chamados.

O parâmetro do método Create é usado apenas em Componentes, para identificar o componente dono. Ao criar Forms, poderíamos usar o Objeto Application.

```
FrmSobre := TFrmSobre.Create(Application);
FrmSobre.ShowModal;
FrmSobre.Release;
```

Para criar objetos não componentes, você não precisa de nenhum parâmetro no método Create.

```
var
  Lst: TStringList;
begin
  Lst := TStringList.Create;
  Lst.Add('Alô, Teresinha!');
  Lst.Add('Uhh uhh...');
```

```
  Lst.SaveToFile('Teresinha.txt');
  Lst.Free;
end;
```

RTTI

Run Time Type Information é a informação de tipo dos objetos em tempo de execução. O operador is é usado para fazer comparações e o operador as é usado para fazer um TypeCast seguro com objetos.

```
for I := 0 to ComponentCount - 1 do
  if Components[I] is TEdit then
    TEdit(Components[I]).Clear;
(Sender as TEdit).Color := clYellow;
```

EXERCÍCIOS (Resolva todos os exercícios e concorra a um brinde no final do curso)

01) Faça um programa para encontrar as raízes de uma equação de 2º grau cujos coeficientes sejam informados pelo usuário. Para relembrar, a fórmula é ax2 + bx + c = 0, o delta é Δ = b2 - 4ac e as raízes são dadas como x1 = (- b + RAIZ(Δ))/2a e x2 = (- b - RAIZ(Δ))/2a.

02) Faça uma função que defina se o ano é bissexto ou não. Sabendo que para ser bissexto, o ano precisa ser divisível por 4 e, além disso, não ser divisível por 100. Se for divisível por 100 o ano tem que ser divisível por 400 para ser bissexto.

03) Faça um programa que peça a data de nascimento do usuário, converta o texto dessa data para uma forma de ano com 4 dígitos e mostre a idade do usuário.

ADQUIRA JÁ A SÉRIE DE CDS COMPLETE

Essencial para qualquer técnico ou usuário de computador.

100 documentos (apostilas, livros, tutoriais, manuais...)
Programas para:
- Desempenho
- Manutenção
- Recuperação
- Segurança

Clientes VPN
Compressores de Executáveis
Emuladores
Programas Essenciais
Pacote Escritório
Ferramentas para BOOT
Ferramentas para HD
Máquina Virtual
Diversos utilitários.

Quer Passar no Vestibular? Este CD vai te ajudar.

Diversos documentos de atualidades, biologia, matematica, português, geografia, história, química, idiomas, física.

Diversas provas de vestibular.
Simulados.

423 E-Books para você ler e passar no vestibular.
Livros dos mais consagrados autores.

Os e-books estão separados por autores e por períodos literários: barroco, modernismo, humanismo etc.

Veja detalhes no site: www.alberteije.com.

Ação	**56**	**Derrote diversos inimigos em jogos de 1ª e 3ª pessoa.**
Corrida	**24**	**Vários jogos de corrida.**
Esportes	**58**	**Hockey, Baseball, Bicicross, Boliche, Golf, Futebol e muito mais.**
Cassino	**81**	**Diversos jogos de cartas e cassino.**
Naves	**24**	**Jogos de nave em 1ª e 3ª pessoa.**
Diversos	**120**	**Jogos de estratégia, raciocínio, aventura e muitos outros.**
Essenciais	**13**	**Acrobat, Divx, Emule, FireFox, Thunderbird, Winamp, WinRAR, Instalador de Codecs, FlashPlayer e outros.**

Acesse o site www.alberteije.com e veja mais detalhes sobre cada CD e como Adquirí-los.