Módulo 3

Object Pascal I

INTRODUÇÃO

Essa lição tem por objetivo apresentar a linguagem que o Delphi utiliza: Object Pascal. Tenha em mente que o Delphi não é uma linguagem, mas sim uma Ferramenta de Desenvolvimento que utiliza a linguagem Object Pascal.

CONCEITOS DE PROGRAMAÇÃO ORIENTADA A OBJETOS

Antes de partir para a linguagem propriamente dita, vamos aprender alguns conceitos básicos de Programação Orientada a Objetos.

Classe: Definição de tipo dos objetos, modelo de objeto.
Objeto: Instância de classe, variável cujo tipo é uma classe.
Atributos: Variáveis de instância. São os dados de um objeto.
Métodos: Funções e procedimentos de um objeto.
Propriedades: Apelido usado para evitar o acesso direto aos atributos de um objeto, onde podemos especificar métodos que serão usados para ler e atribuir seus valores a esses atributos.
Mensagens: Chamada de métodos, leitura e atribuição de propriedades.
Encapsulamento: Conjunto de técnicas usadas para limitar o acesso aos atributos e métodos internos de um objeto.
Herança: Possibilidade de criar uma classe descendente de outra, aproveitando seus métodos, atributos e propriedades.
Ancestral: Super classe ou classe de base, a partir da qual outras classes podem ser criadas.
Descendente: Subclasse.
Hierarquia de Classes: Conjunto de classes ancestrais e descendentes, geralmente representadas em uma árvore hierárquica.
Polimorfismo: Capacidade de redefinir métodos e propriedades de uma classe em seus descendentes.

ESTRUTURA DE UNITS

Vamos examinar o código gerado para um novo Form, identificando as principais seções de uma Unit típica. Abra o Delphi e crie uma nova aplicação. Observe na Unit principal as seguintes cláusulas.

Unit: A primeira declaração de uma unit é seu identificador, que é igual ao nome do arquivo.
Interface: Seção interface, onde ficam declarações que podem ser usadas por outras Units.
Uses: Na cláusula uses fica a lista de Units usadas.
Type: Na cláusula type fica a definição de tipos, aqui temos a declaração da classe do Form.
Var: Na cláusula var são declaradas as variáveis, aqui temos a declaração da instância do Form.
Implementation: Na seção implementation ficam as definições dos métodos.
End: Toda Unit termina com um end a partir de onde qualquer texto é ignorado.

VARIÁVEIS

No Delphi, toda variável tem que ser declarada antes de ser utilizada. As declarações podem ser feitas após a palavra reservada var, onde são indicados o nome e o tipo da variável. Os nomes de variáveis não podem ter acentos, espaços ou caracteres especiais como &, $ ou % e o primeiro caractere de um nome de variável tem que ser uma letra ou um sublinhado (_).

VARIÁVEIS GLOBAIS

As variáveis abaixo são globais, declaradas na Interface da Unit. Podem ser acessadas por qualquer Unit usuária.

```
var
   I: Integer;
   Usuario: string;
   A, B, Soma: Double;
```

```
>>
   Ok: Boolean;
```

VARIÁVEIS LOCAIS

As variáveis abaixo são locais ao método, ou seja, elas só existem dentro do método, não podem ser acessadas de fora, mesmo que seja na mesma Unit. Na verdade essas variáveis são criadas quando o método é chamado e são destruídas quando ele é encerrado, seu valor não é persistente.

```
procedure TFrmExemplo.BtnTrocarClick(Sender: TObject);
var
   Aux: string;
begin
   Aux := EdtA.Text;
   EdtA.Text := EdtB.Text;
   EdtB.Text := Aux;
end;
```

ATRIBUTOS

Os atributos são variáveis de instância. Para declarar um atributo em uma classe basta definir o identificador e o tipo do atributo na declaração da classe, feita na seção type da Interface da Unit, como abaixo.

```
type
   TFrmSomar = class(TForm)
   private {Private declarations}
      A, B: Double;
   public {Public declarations}
      Soma: Double;
   end;
```

ENCAPSULAMENTO

Os principais níveis de visibilidade dos atributos e métodos de uma classe são mostrados abaixo.

Nível	Visibilidade
Private	Os itens declarados nesse nível só podem ser acessados na mesma unit.
Public	Nesse nível, qualquer unit usuária poderá acessar o item.
Protected	Os itens só poderão ser acessados em outra unit se for em uma classe descendente.
Published	É o nível default, igual ao Public, mas define propriedades e eventos usados em tempo de projeto.

CLASSES

Classes são tipos de objetos, uma classe é declarada na cláusula type da seção interface e os métodos são definidos na seção implementation. Examine o código de um Form para identificar os elementos de sua classe.

```
interface
type
  TFrmSomar = class(TForm)
    EdtA: TEdit;
    EdtB: TEdit;
    BtnSoma: TButton;
    procedure BtnSomaClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

implementation
```

```
procedure TFrmSoma.BtnSomaClick(Sender: TObject);
begin
  ShowMessage(EdtA.Text + EditB.Text);
end;
```

OBJETOS

Um Objeto é tratado como uma variável cujo tipo é uma classe. A declaração de objetos é igual à declaração de uma variável simples, tendo no lugar do tipo a classe do objeto.

```
var
  FrmSomar: TFrmSomar;
```

LITERAIS

Valores literais são valores usados em atribuições e expressões. Cada tipo tem uma sintaxe diferente.

Tipo	Definição
Inteiro	Seqüência de dígitos decimais (0 a 9), sinalizados ou não.
Inteiro Hexadecimal	Seqüência de dígitos hexadecimais (0 a F), precedidos por um cifrão ($).
Real	Igual ao tipo Inteiro, mas pode usar separador decimal e notação científica.
Caractere	Letra entre apóstrofos ou o caracter # seguido de um número inteiro entre 0 e 255 (ASCII).
String	Seqüência de caracteres delimitados por apóstrofos.

CONSTANTES

São declaradas na seção const, podem ser usadas como variáveis, mas não podem ser alteradas. Geralmente o nome das constantes é escrito em letras maiúsculas e na declaração dessas constantes não é indicado o tipo.

```
const
  G = 3.94851265E-19;
  NUM_CHARS = '0123456789';
  CR = #13;
  SPACE = ' ';
  MAX_VALUE = $FFFFFFFF;
```

CONSTANTES TIPADAS

Na verdade, constantes tipadas são variáveis inicializadas com valor persistente, que podem ser alteradas normalmente, como qualquer variável. A única diferença de sintaxe entre constantes tipadas e simples é que o tipo da constante é indicado explicitamente na declaração. Se uma constante tipada for declarada localmente, ela não será destruída quando o método for encerrado. Para diferenciar das constantes normais, costuma-se declarar estas com letras de caso variável, como abaixo.

```
const
  Cont: Integer = 1;
  Peso: Double = 50.5;
  Empresa: string = 'Linux Informática';
```

INSTRUÇÕES

Os programas são compostos por instruções, que são linhas de código executável. Exemplos de instruções simples são atribuições, mensagens entre objetos, chamadas de procedimentos, funções e métodos, como mostradas abaixo. As instruções podem ser divididas em várias linhas, o que indica o fim de uma instrução é o ponto e vírgula no final. Quando uma instrução é quebrada, costuma-se dar dois espaços antes das próximas linhas, para melhorar a leitura do código.

```
Caption := 'Linux Informática!';
Form2.ShowModal;
Application.MessageBox('Você executou uma operação ilegal, o programa será
finalizado.','Falha geral', MB_ICONERROR);
```

Você pode usar várias instruções agrupadas em uma instrução composta, como se fosse uma só instrução. Uma instrução composta delimitada pelas palavras reservadas begin e end. Toda instrução, simples ou composta, é terminada com um ponto-e-vírgula.

```
if CheckBox1.Checked then
begin
  ShowMessage('O CheckBox será desmarcado.');
  CheckBox1.Checked := False;
end;
```

ESTILO DE CODIFICAÇÃO

As instruções e todo o código de uma Unit devem ser distribuídos para facilitar o máximo a leitura. Para isso, podemos usar a indentação, geralmente de três espaços para indicar os níveis de código. Procure criar um estilo próprio, que melhor se molde à sua realidade. Se for desenvolver em grupo, é melhor que todos usem o mesmo estilo para evitar confusões.

COMENTÁRIOS

Existem 3 estilos de comentário no Delphi, como mostrado abaixo:

```
(* Comentário do Pascal Padrão *)
{ Comentário do Turbo Pascal }
// Comentário de linha do C++
```

Cuidado com as diretivas de compilação, pois elas são delimitadas por chaves e podem ser confundidas com comentários. A diretiva de compilação mostrada abaixo é incluída em todas as Units de Forms.

```
{$R*.DFM}
```

TIPOS DE DADOS PADRÃO

O Delphi trata vários tipos de dados padrão, segue uma descrição sucinta desses tipos.

TIPOS INTEIROS

São tipos numéricos exatos, sem casas decimais. O tipo Integer é o tipo inteiro padrão.

Tipo	Tamanho em Bytes	Valor Mínimo	Valor Máximo
ShortInt	1	-128	127
SmallInt	2	-32768	32767
Longint	4	-2147483648	2147483647
Byte	1	0	255
Word	2	0	65535
Integer	4	-2147483648	2147483647
Cardinal	4	0	2147483647

TIPOS REAIS

São tipos numéricos com casas decimais. O tipo Double é o tipo real padrão.

Tipo	Tamanho em Bytes	Valor Mínimo	Valor Máximo	Dígitos Significativos
Real	6	10-39	1038	11-12
Single	4	10-45	1038	7-8
Double	8	10-324	10308	15-16
Extended	10	10-4932	104932	19-20
Comp	8	-1018	1018	19-20
Currency	8	-1012	1012	19-20

TIPOS TEXTO

Os tipos texto podem operar com caracteres simples ou grupos de caracteres. O tipo texto padrão é o tipo String.

Tipo	Descrição
Char	Um único caractere ASCII.
String	Texto alocado dinamicamente. Pode ser limitado a 255 caracteres conforme configuração.
PChar	String terminada em nulo (#0), usada geralmente nas funções da API do Windows.

O operador + pode ser usado para concatenar strings e você pode usar uma variável do tipo string como uma lista de caracteres.

```
ShowMessage('5ª letra do título da janela: ' + Caption[5]);
Label1.Text := '2ª letra do Edit: ' + Edit1.Text[2];
```

Existem várias funções de manipulação de strings, veja algumas das mais importantes mostradas abaixo.

Função	Descrição
AnsiCompareText	Compara 2 strings sem sensitividade de maiúsculas/minúsculas.
AnsiLowerCase	Converte todas as letras de uma string para minúsculas.
AnsiUpperCase	Converte todas as letras de uma string para maiúsculas.
Copy	Retorna parte de uma string.
Delete	Apaga parte de uma string.
Insert	Insere uma string em outra.
Length	Número de caracteres de uma string.
Pos	Posição de uma string em outra.
Trim	Remove todos os espaços de uma string.
TrimLeft	Remove os espaços à esquerda de uma string.
TrimRight	Remove os espaços à direita de uma string.
Format	Formata uma string com uma série de argumentos de vários tipos.

Por exemplo, para comparar o texto de dois Edits, poderíamos usar a função AnsiCompareText.

```
if AnsiCompareText(EdtA.Text, EdtB.Text) = 0 then
  ShowMessage('O texto dos dois Edits é igual.');
```

A função Format é especialmente útil na formatação de strings, veja alguns exemplos.

```
ShowMessage(Format('O número %d é a parte inteira do número %f.', [10, 10.5]));
ShowMessage(Format('Este texto%sfoi formatado%susando o caractere #%d.', [#13, #13,
13]));
ShowMessage(Format('O preço do livro %s é %m.', ['Como Programar em Delphi', 50.7]));
```

Um detalhe que deve ser observado é que as propriedades dos objetos não podem ser usadas como variáveis em funções. Veja a declaração do procedimento Delete no help.

```
procedure Delete(var S: string; Index, Count:Integer);
```

Digamos que você deseje apagar as 5 primeiras letras de um Edit, como a string do Delete é variável, não poderia usar o código abaixo.

```
Delete(Edit1.Text, 1, 5);
```

Para você poder fazer a operação desejada, teria que usar uma variável como variável auxiliar.

```
var
  S: string;
begin
  S := Edit1.Text;
  Delete(S, 1, 5);
  Edit1.Text := S;
end;
```

TIPOS ORDINAIS

Tipos ordinais são tipos que tem uma seqüência incremental, ou seja, você sempre pode dizer qual o próximo valor ou qual o valor anterior a um determinado valor desses tipos. São tipos ordinais o Char, os tipos inteiros, o Boolean e os tipos enumerados. Algumas rotinas para ordinais são mostradas abaixo.

Função	Descrição
Dec	Decrementa variável ordinal.
Inc	Incrementa variável ordinal.
Odd	Testa se um ordinal é ímpar.
Pred	Predecessor do ordinal.
Succ	Sucessor do ordinal.
Ord	Ordem de um valor na faixa de valores de um tipo ordinal.
Low	Valor mais baixo na faixa de valores.
High	Valor mais alto na faixa de valores.

Por exemplo, use o código abaixo no evento OnKeyPress de um Edit e veja o resultado.

```
Inc(Key);
```

BOOLEAN

Variáveis do tipo Boolean podem receber os valores lógicos True ou False, verdadeiro ou falso. Uma variável Boolean ocupa 1 byte de memória.

TDATETIME

O tipo TDateTime guarda data e hora em uma estrutura interna igual ao tipo Double, onde a parte inteira é o número de dias desde 31/12/1899 e a parte decimal guarda a hora, minuto, segundo e milisegundo. As datas podem ser somadas ou subtraídas normalmente.
Existem várias rotinas de manipulação de datas e horas, usadas com o tipo TDateTime, veja algumas abaixo.

Rotina	Descrição
Date	Retorna a data do sistema.
Now	Retorna a data e hora do sistema.
Time	Retorna a hora do sistema.
DayOfWeek	Retorna o dia da semana de uma data especificada.
DecodeDate	Decodifica um valor TDateTime em Words de dia, mês e ano.
DecodeTime	Decodifica um valor TDateTime em Words de hora, minuto, segundo e milisegundos.
EncodeDate	Retorna um TDateTime a partir de Words de dia, mês e ano.
EncodeTime	Retorna um TDateTime a partir de Words de hora, minuto, segundo e milisegundos.

No help de cada uma das funções acima você vai encontrar alguns exemplos, veja os colocados abaixo.

```
if DayOfWeek(Date) = 1 then
  ShowMessage('Hoje é Domingo, pé de cachimbo!')
else
  ShowMessage('Hoje não é Domingo, pé de cachimbo!');

var
  A, M, D: Word;
begin
  DecodeDate(Date, A, M, D);
  ShowMessage(Format('Dia %.2d do mês %.2d de %d.', [D, M, A]));
end;
```

VARIANT

Tipo genérico, que pode atribuir e receber valores de qualquer outro tipo. Evite usar variáveis do tipo Variant, pois o uso dessas variáveis pode prejudicar a performance do programa, além de diminuir a legibilidade do código fonte e a integridade do executável, veja o trecho de código abaixo e note como esse tipo de variável tem um comportamento estranho.

```
var
  V1, V2, V3: Variant;
begin
  V1 := True;
  V2 := 1234.5678;
  V3 := Date;
  ShowMessage(V1 + V2 + V3);
end;
```

CONVERSÕES DE TIPO

Freqüentemente você vai precisar converter um tipo de dado em outro, como um número em uma string. Para essas conversões você pode usar duas técnicas, o TypeCasting e as rotinas de conversão de tipos.

TYPECASTING

TypeCast é uma conversão direta de tipo, usando o identificador do tipo destino como se fosse uma função. Como o Delphi não faz nenhuma verificação se a conversão é válida, você deve tomar um certo cuidado ao usar um TypeCast para não criar programas instáveis.

```
var
  I: Integer;
  C: Char;
  B: Boolean;
begin
  I := Integer('A');
  C := Char(48);
  B := Boolean(0);
  Application.MessageBox(PChar('Linguagem de Programação' + #13 + 'Delphi 3'),
'Linux Informática',MB_ICONEXCLAMATION);
end;
```

ROTINAS DE CONVERSÃO

As principais rotinas de conversão estão listadas na tabela abaixo. Caso você tente usar uma dessas rotinas em uma conversão inválida, pode ser gerada uma exceção.

Rotina	Descrição
Chr	Byte em Char.
StrToInt	String em Integer.
IntToStr	Integer em String.
StrToIntDef	String em Integer, com um valor default caso haja erro.
IntToHex	Número em String Hexadecimal.
Round	Arredonda um número real em um Integer.
Trunc	Trunca um número real em um Integer.
StrToFloat	String em Real.
FloatToStr	Real em string.
FormatFloat	Número real em string usando uma string de formato.
DateToStr	TDateTime em string de data, de acordo com as opções do Panel de Controle.
StrToDate	String de data em TDateTime.
TimeToStr	TDateTime em String de Hora.
StrToTime	String de hora em TDateTime.
DateTimeToStr	TDateTime em string de data e hora.
StrToDateTime	String de data e hora em TDateTime.
FormatDateTime	TDateTime em string usando uma string de formato.
VarCast	Qualquer tipo em outro usando argumentos do tipo Variant.
VarAsType	Variante em qualquer tipo.
Val	String em número, real ou inteiro.
Str	Número, real ou inteiro, em String.

Veja alguns exemplos de como usar essas rotinas. Conversão de dados é uma operação muito comum na programação em Object Pascal, seria interessante dar uma olhada no help de cada uma das funções acima.

```
var
  I: Integer;
  D: Double;
  S1, S2: string;
begin
  D := 10.5;
  I := Trunc(D);
  S1 := FloatToStr(D);
  S2 := IntToStr(I);
  ShowMessage(S1 + #13 + S2);
end;
```

```
var
  A, B, Soma: Double;
begin
  A := StrToFloat(EdtA.Text);
  B := StrToFloat(EdtB.Text);
  Soma := A + B;
  ShowMessage(Format('%f + %f = %f', [A, B, Soma]);
end;
```

EXPRESSÕES

Uma expressão é qualquer combinação de operadores, variáveis, constantes, valores literais e chamadas de funções que resultem em um valor de determinado tipo. Uma expressão é usada sempre que precisamos de um valor que possa ser obtido por uma expressão.

```
  A + 12 * C
  Date - 4
  StrToInt(Edit1.Text + Edit2.Text)
  StrToDate(Edit2.Text) - StrToDate(Edit1.Text)
  12 * A / 100
  A < B
```

OPERADORES

Os operadores são usados em expressões e a ordem em que as expressões são executadas depende da precedência desses operadores. Veja abaixo a lista de operadores em ordem descendente de precedência.

Operador	Descrição
Operadores Unários	
@	Endereço
not	Não booleano ou bit voltado para não
Operadores Multiplicativos e de direção de Bit	
*	Multiplicação ou interseção de conjuntos
/	Divisão de Real
div	Divisão de Inteiro
mod	Resto de divisão de Inteiros
as	TypeCast seguro quanto ao tipo (RTTI)
and	E booleano ou bit voltado para e
shl	Deslocamento de bits à esquerda
shr	Deslocamento de bits à direita
Operadores Aditivos	
+	Adição ou união de conjuntos
-	Subtração ou diferença de conjuntos
or	Ou booleano ou bit voltado para ou
xor	Ou exclusivo booleano ou bit voltado para ou exclusivo

Operadores Relacionais	
=	Igual
<>	Diferente
<	Menor
>	Maior
<=	Menor ou igual
>=	Maior ou igual
in	Pertinência a conjuntos
is	Compatibilidade de tipos (RTTI)

Para forçar uma expressão de menor precedência a ser executada antes, você pode usar os parênteses, como mostrado abaixo.

```
(5 - 2) * 3;
(A > B) and (A < C)
```

Para fazer potenciação, use a função Power, abaixo temos que A é igual a A elevado a 4.

```
A := Power(A, 4);
```

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

01) Faça um Form, com um Edit e botões para passar o texto do Edit para maiúsculas e minúsculas.

02) Faça um programa que sugira que o usuário pare de trabalhar quando o relógio do sistema não estiver em horário comercial.

03) Sendo I um número inteiro, R um real e S uma string com um texto numérico como encontrar o resultado da soma dos 3 valores, e de que tipo será esse resultado? Faça isso no Delphi.

Obs: Envie-me por e-mail os códigos-fontes desses exercícios para participar do sorteio do brinde.