



Delphi e Object Pascal

Autor:
Maurício Capobianco Lopes



UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO

DISCIPLINA: Linguagem de Programação Visual
PROFESSOR: Maurício Capobianco Lopes

SUMÁRIO

Introdução

O que é uma Linguagem de Programação Visual

Características das Linguagens de Programação Visual

Tipos de Linguagens Visuais

O Ambiente DELPHI 2.0

Janela FORM

Janela OBJECT INSPECTOR

Janela CODE EDITOR

Janela MAIN

Menu Principal

Projeto

Tipos de Dados

Tipos de Dados NUMÉRICOS INTEIROS

Tipos de dados NUMÉRICOS REAIS e MOEDA

Tipos de dados BOOLEANOS

Tipos de dados ALFANUMÉRICOS

Biblioteca de Classes

Paleta de Componentes

TApplication

TScreen

TPrinter

Componentes

Paleta STANDARD

Paleta ADDITIONAL

Paleta WIN95

A Paleta DIALOGS

Propriedades

Métodos

Eventos

Exceções

Try ... Except

On ... Do

Tipos de Exceção

Try ... Finally

INTRODUÇÃO

Este curso tem por objetivo apresentar uma linguagem de programação visual que tenha como base a programação orientada a objetos e a programação orientada a eventos.

Para cumprir este objetivo foi selecionada uma ferramenta que tenha estas características. A ferramenta selecionada foi o **DELPHI**. O DELPHI é uma linguagem de programação visual desenvolvida pela empresa *Borland*, baseada na linguagem de implementação denominada **Object Pascal**.

Neste curso será estudada a versão 2.0 da linguagem, por se tratar de uma versão totalmente compatível com Windows95® e gerar aplicativos de 32 bits.

Esperamos que ao final deste curso você tenha condições de desenvolver aplicativos utilizando tal linguagem, de modo a aproveitar os seus principais recursos.

Bom Curso!

O QUE É UMA LINGUAGEM DE PROGRAMAÇÃO VISUAL

As **linguagens de programação visual**, ou ferramentas visuais, são linguagens desenvolvidas para ambientes gráficos. Estas linguagens possuem a chamada *sintaxe visual*, pois incorporam figuras, formulários, animações, diagramas ou ícones para facilitar e agilizar o desenvolvimento dos programas.

As **linguagens de programação visual** possuem um ambiente visual baseado em um conjunto de ferramentas e uma interface para acessar a estas ferramentas

CARACTERÍSTICAS DAS LINGUAGENS DE PROGRAMAÇÃO VISUAL

As principais características de uma **Linguagem Visual** são permitir que um sistema seja desenvolvido mais rapidamente do que numa linguagem tradicional e reduzir o tempo de manutenção. Isto ocorre em função dos seguintes aspectos:

- a interface é montada através de componentes já disponíveis;
- torna-se desnecessário escrever uma boa parte do código de um programa em função do mesmo ser gerado automaticamente;
- possuem geradores ou editores de telas, relatórios, etiquetas, consultas, entre outros;
- combinam recursos de análise orientada a objetos e a eventos com gerenciadores de banco de dados.

TIPOS DE LINGUAGENS VISUAIS

Algumas das principais linguagens de Programação Visual disponíveis no mercado são:

Linguagem Visual	Linguagem de Implementação
Borland Delphi	Object Pascal
DBase	DBase
FoxPro	XBase
Microsoft Acces	Acces Basic
Power Builder	Powerscript
Realizer	Basic
SQL Windows	SAL
Visual Basic	Visual Basic
Visual C++	C++
Visual J++	Java
Visual Objects	XBase orientado a objetos
Visual Realia	Cobol

O AMBIENTE DELPHI 2.0

Neste item será estudado o **IDE** (*Integrated Developer Environment* – Ambiente Integrado de Desenvolvimento) do DELPHI.

O DELPHI possui um conjunto de ferramentas que permitem facilitar e agilizar a construção de programas, permitindo uma melhor interação entre o programador e o computador. Suas principais janelas são:

- Janela FORM
- Janela OBJECT INSPECTOR
- Janela CODE EDITOR
- Janela MAIN

JANELA FORM

O FORM é a tela onde o desenvolvedor constrói sua aplicação. A partir de um FORM é que se estabelece a interação USUÁRIO-COMPUTADOR, através de botões, rótulos e outros componentes, estabelecendo-se funções, métodos ou eventos que serão ativados. Os componentes são dispostos dentro da área útil do FORM.



A Janela FORM: interface.

Os quatro tipos de FORM são:

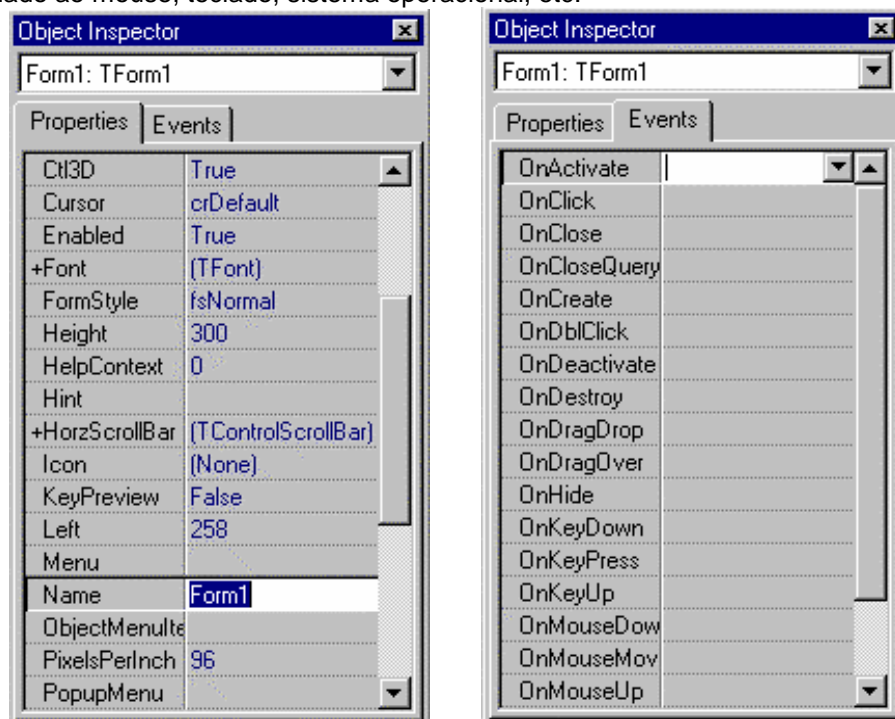
- Normal: é o FORM padrão;
- MDIForm: os FORMS do tipo MDI (Multiple Document Interface), podem conter outros FORMS do tipo **MDIChild**;
- MDIChild: são FORMS que sempre estão subordinados a um FORM **MDIForm**;
- StayOnTop: são FORMS que sempre ficam visíveis na tela.

Para se abrir um FORM utilizam-se os comandos: **Show** ou **ShowModal**. Os FORMS tem um **ModalResult** associado que são dados pelos botões colocados nos mesmos. Este **ModalResult** permite tratar se o FORM foi fechado com **OK**, **Cancel**, **Close**, ...

JANELA OBJECT INSPECTOR

A janela OBJECT INSPECTOR contém propriedades e eventos dos componentes inseridos em um FORM, e do próprio FORM. É na janela *Properties* (**Propriedades**), por exemplo, que se estabelecem as características de cada componente, como nome, fonte, altura, largura, etc. Já na

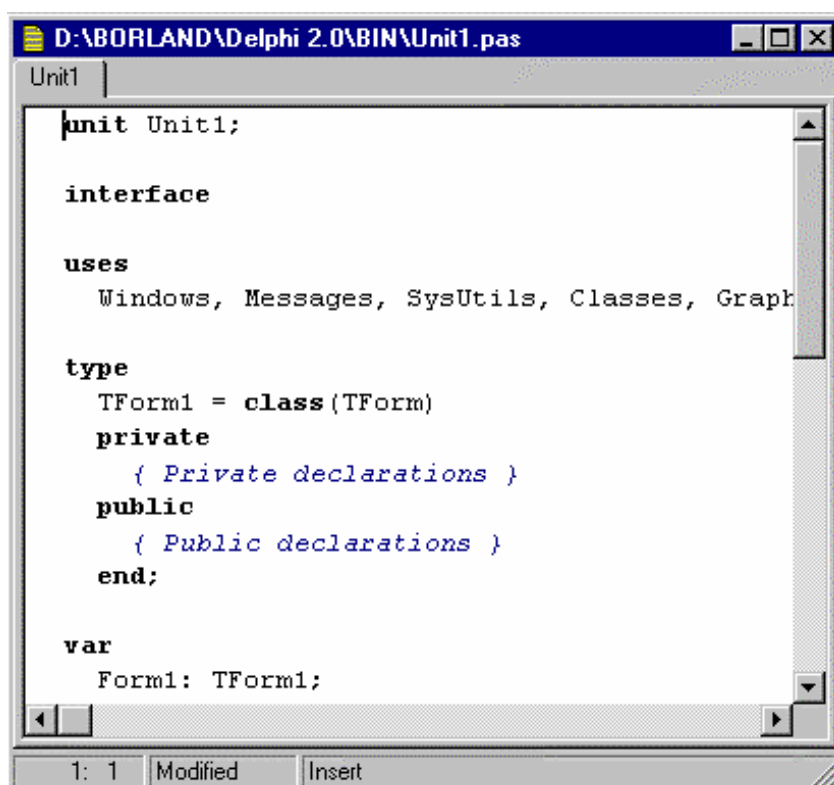
janela *Events (Eventos)* estabelecem-se ações a serem tomadas pelo componente a partir de um evento associado ao mouse, teclado, sistema operacional, etc.



A Janela OBJECT INSPECTOR: propriedades e eventos

JANELA CODE EDITOR

A janela CODE EDITOR, ou editor de código, é onde se desenvolve o programa fonte. É neste editor que se encontra a estrutura sintática propriamente dita da Linguagem DELPHI. Cabe ressaltar, no entanto, que boa parte do código escrito é gerado pela própria linguagem.

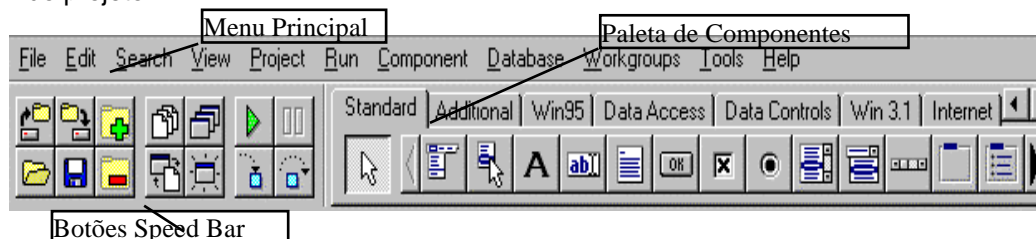


A Janela CODE EDITOR: código fonte da aplicação

JANELA MAIN

A janela MAIN, ou janela principal, controla o funcionamento do DELPHI. Esta janela pode ser dividida em três partes:

- **Menu Principal:** contém as opções de utilização do DELPHI;
- **Botões Speed Bar:** contém botões que agilizam determinadas funções do DELPHI;
- **Paleta de Componentes:** contém os componentes que podem ser utilizados na construção do projeto.



A Janela MAIN: dividida em três partes.

Os Botões SpeedBar e a Paleta de Componentes serão detalhados à medida de sua utilização.

MENU PRINCIPAL

O menu principal contém as opções de utilização do DELPHI. O detalhamento destas opções será feito à medida de sua utilização:

- **File:** permite a manipulação de arquivos do desenvolvedor (PAS, DPR, ...);
- **Edit:** apresenta opções de edição;
- **Search:** apresenta opções de pesquisa e localização;
- **View:** permite verificar detalhes do projeto;
- **Project:** permite adicionar ou remover partes em um projeto, bem como compilá-lo;
- **Run:** apresenta opções de execução e depuração do projeto;
- **Component:** permite a criação ou instalação de novos componentes no DELPHI;
- **Database:** apresenta opções de uso de banco de dados;
- **Workgroups:** permite o controle de versão e gerenciamento de arquivos em ambientes Client/Server;
- **Tools:** permite configurar o ambiente de trabalho, bem com acessar ferramentas externas ao DELPHI;
- **Help:** ajuda do DELPHI.

PROJETO

No DELPHI o “programa principal” tem extensão **DPR**. Um **DPR** é o código de programa, também denominado de projeto, a partir do qual o desenvolvedor construirá sua aplicação. É a partir deste projeto que será gerado o código executável (**EXE**). Grande parte do código, no entanto, fica armazenado em **UNITs** com extensão **PAS**, onde são desenvolvidos os **formulários**.

O projeto pode ser visualizado através da opção do menu **View | Project Source**. Sua estrutura aparece na janela Code Editor e pode-se verificar que é muito semelhante ao programa principal do antigo Turbo Pascal®.

A implementação mínima de um projeto prevê, na grande maioria das situações, a inicialização da aplicação (comando *Application.Initialize*), o uso (cláusula *Uses*) e criação dos formulários (comando *Application.CreateForm*) utilizados na mesma e a execução do projeto (*Application.Run*).

Ao projeto podem ser adicionadas ou retiradas bibliotecas (**Units**) através da opção **Project** do menu.

Estrutura de um Projeto em DELPHI.

```

program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.

```

No DELPHI a definição dos tipos de dados além de permitirem a economia de espaço, seus respectivos tipos de dados são compatíveis com os tipos de dados do DELPHI 1.0 para o Windows 7.0[®] e do DELPHI 1.0 para o Macintosh.

Os principais tipos de dados são:

- **Numéricos**
 - Inteiros
 - Reais
 - Moeda
- **Alfanuméricos**
 - Caracter
 - String
- **Lógicos**
 - Booleanos

Isso é muito importante, pois a linguagem Delphi é compatível com o Pascal e ter compatibilidade com os tipos de dados do Pascal facilita muitas mudanças do PASCAL.

TIPOS DE DADOS NUMÉRICOS INTEIROS

Os principais tipos de dados numéricos inteiros do DELPHI estão definidos na tabela abaixo:

Tipo	Tamanho (em bytes)	Domínio
Byte	1	0..255
ShortInt	1	-127..128
Word	2	0..65535
SmallInt	2	-32768..32768
Cardinal	4	0..2147483647
Integer	4	-2147483648..2147483647
Longint	4	-2147483648..2147483647

Tipos de Dados NUMÉRICOS INTEIROS

TIPOS DE DADOS NUMÉRICOS REAIS E MOEDA

Os principais tipos de dados numéricos reais e moeda do DELPHI estão definidos na tabela abaixo:

Tipo	Tamanho (em bytes)	Domínio
Single	4	$1.5 * 10^{-45}$.. $3.4 * 10^{38}$
Real	6	$2.9 * 10^{-39}$.. $1.7 * 10^{39}$
Double	8	$5.0 * 10^{-324}$.. $1.7 * 10^{308}$
Extended	10	$3.4 * 10^{-4932}$.. $1.1 * 10^{4392}$
Comp [#]	8	-2^{63} .. 2^{62}
Currency		

[#] O tipo COMP é na verdade um inteiro de 64 bits, mas é implementado como um real

Tipos de Dados NUMÉRICOS REAIS e MOEDA

O tipo *Currency* é o tipo MOEDA. Ele tem precisão de quatro casas decimais e é compatível com os bancos de dados que representam dinheiro.

TIPOS DE DADOS BOOLEANOS

Os principais tipos de dados lógicos do DELPHI estão definidos na tabela abaixo:

Tipo	Tamanho (em bytes)	Armazenamento
Boolean	1	tamanho de um byte
ByteBool	1	tamanho de um byte
Bool	2	tamanho de uma palavra
WordBool	2	tamanho de uma palavra
LongBool	4	tamanho de uma palavra dupla

Tipos de Dados BOOLEANOS

A existência de diversos tipos lógicos no DELPHI que armazenam a mesma coisa (TRUE ou FALSE) dá-se pela necessidade de compatibilização com os tipos do Windows®.

TIPOS DE DADOS ALFANUMÉRICOS

Os principais tipos de dados alfanuméricos do tipo **caracter** do DELPHI estão definidos na tabela abaixo:

Tipo	Tamanho (em bytes)	Armazenamento
Char	1	1 caracter ANSI
AnsiChar	1	1 caracter ANSI
WideChar	2	1 caracter Unicode

Tipos de Dados CHARACTER

Os principais tipos de dados alfanuméricos do tipo **string** do DELPHI estão definidos na tabela abaixo:

Tipo	Tamanho (em bytes)	Armazenamento
ShortString	255	ANSIChar
AnsiString	até 3 Gb	ANSIChar
String	255 ou até 3 Gb	ANSIChar
WideString	até 1,5 Gb	WideChar

Tipos de Dados STRING

No DELPHI 2.0 o tipo de dados *String* tem funcionamento idêntico ao *AnsiString*, que é baseado em ponteiros. Para que ele tenha funcionamento igual ao *ShortString*, que é compatível com o antigo *String* do DELPH 1.0 e do PASCAL é necessário incluir a diretiva de compilação {\$H-} no código do programa.

BIBLIOTECA DE CLASSES

Por ser uma linguagem de programação orientada a objetos o DELPHI possui uma **biblioteca de classes** denominada de **VCL (Visual Component Library)**, para auxiliar na construção de programas.

A biblioteca de classes do DELPHI pode ser visualizada no [Browser](#) (no menu principal acesse *View - Browser*).

No DELPHI existem as classes **visíveis** e as **não visíveis**.

CLASSES VISÍVEIS

As classes visíveis são aquelas que podem ser manipuladas através da [paleta de componentes](#), ou seja, são os próprios componentes

CLASSES NÃO-VISÍVEIS

As classes de componentes não visíveis oferecem controles especializados que não são possíveis ou necessários através da interface visual. As classes não visíveis mais comuns são a [TApplication](#), a [TScreen](#) e a [TPrinter](#).

PALETA DE COMPONENTES

A paleta de componentes é a biblioteca de classes que fornece recursos para o desenvolvimento visual em DELPHI. As classes da VCL representadas na paleta de componentes estão separadas por tipos:

- [Standard](#): componentes mais comuns e usados.
- [Additional](#): componentes adicionais também de uso comum.
- [Win95](#): componentes com a aparência do Windows95®.
- [Data Access](#): componentes para comunicação com banco de dados.
- [Data Controls](#): componentes para tratamento de dados vinculados a banco de dados.
- [Win3.1](#): componentes com a aparência do Windows 3.1x®.
- [Dialogs](#): componentes que criam caixas de diálogo comuns no Windows95®.
- [Internet](#): componentes que permitem operações de acesso a Internet.
- [System](#): componentes para aproveitar recursos de sistema do Windows95®.
- [QReport](#): componentes para geração de relatórios.
- [OCX](#): componentes desenvolvidos em outras linguagens mas que podem ser utilizados no DELPHI.
- [Samples](#): componentes de exemplo, cuja documentação vem disponível com o DELPHI.

Os componentes disponíveis na VCL podem ser divididas entre os visuais e os não-visuais.

TAPPLICATION

Quando se inicia um projeto em DELPHI, automaticamente é criada uma variável *Application* que é do tipo *TApplication*. Esta classe permite a interação de sua aplicação com o Windows.

A atribuição de valores às propriedades do *TApplication* pode ser feita apenas em tempo de execução, ou durante o desenvolvimento do projeto na opção do menu *Project - Options*.

Os principais **métodos** da classe *TApplication* são:

Nome	Descrição
Run()	faz com que a aplicação seja executada. Este comando é colocado automaticamente no código fonte do projeto (veja em <i>View - Project Source</i>).
Terminate()	encerra a aplicação.
HelpCommand()	carrega o arquivo de <i>help</i> da aplicação.

Algumas das principais **propriedades** da classe *TApplication* são:

Nome	Descrição
MainForm	contém o <i>form</i> principal da aplicação;
Title	contém o título da aplicação quando a mesma está minimizada;
HelpFile	armazena o nome do arquivo de <i>Help</i> da aplicação;
Icon	contém o ícone que representa a aplicação.

TSCREEN

Quando se inicia um projeto em DELPHI, automaticamente é criada uma variável *Screen* que é do tipo *TScreen*. Esta classe contém as características da tela na qual a aplicação está rodando.

Algumas das principais **propriedades** da classe *TScreen* são:

Nome	Descrição
Cursor	contém o cursor que está em uso;
Cursors	contém a lista dos cursores disponíveis para a aplicação;
Fonts	contém a lista de fontes disponíveis para a aplicação;
Forms	contém a lista de <i>forms</i> da aplicação;
Height	contém a altura da tela (em pixels);
Width	contém a largura da tela (em pixels).

TPRINTER

Quando se inicia um projeto em DELPHI, automaticamente é criada uma variável *Printer* que é do tipo *TPrinter*. Esta classe contém as características da impressora padrão que está sendo utilizada pelo Windows, além de permitir a impressão.

Os principais **métodos** da classe *TPrinter* são:

Nome	Descrição
BeginDoc()	inicia um processo de impressão.
EndDoc()	finaliza um processo de impressão.
Abort()	aborta o processo de impressão.
NewPage()	inicia a impressão em uma nova página.

Algumas das principais **propriedades** da classe *TPrinter* são:

Nome	Descrição
Printers	contém a lista de impressoras instaladas no Windows.
PrinterIndex	contém o índice da impressora ativada.
Orientation	contém a orientação do papel.
PageHeight	contém a altura da página de impressão.
PageWidth	contém a largura da página de impressão.
PageNumber	contém o número da página que está sendo impressa.
Title	contém o título da página impressa.

Para a impressão de dados utilizando o *TPrinter* é necessário utilizar a classe **TCanvas**.

COMPONENTES

COMPONENTES VISUAIS

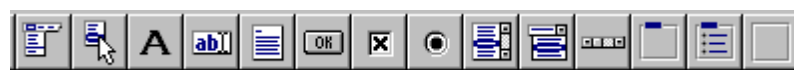
Os componentes visuais podem ter sua forma e tamanho alteradas no formulário (Form), além das propriedades e eventos no Object Inspector. Eles aparecem durante a execução do aplicativo exatamente como foram definidos durante o projeto.

COMPONENTES NÃO-VISUAIS

Os componentes não visuais ficam apenas como a representação de um ícone no formulário (Form), mas suas propriedades e eventos podem ser alteradas no Object Inspector. Eles não aparecem no formulário durante a execução do aplicativo, podendo ser ativados por comandos específicos (por exemplo podemos citar a caixa de diálogo abrir arquivo).









PALETA STANDARD

A paleta Standard contém 14 componentes mais comuns para a construção de aplicações.



Paleta Standard: componentes mais comuns

Os componentes desta paleta são:

	Permite a construção da barra de menus e de menus suspensos.
MainMenu	
	Permite a construção de menus a partir do botão direito do mouse.
PopupMenu	
	Permite colocar textos que não podem ser alterados pelo usuário.
Label	
	Permite a apresentação ou a entrada de dados pelo usuário.
Edit	
	Permite a introdução ou exibição de linhas de texto.
Memo	
	Permite a colocação de botões para seleção de opções por parte do usuário.
Button	
	Permite a colocação de caixa de verificação para a seleção de diversas opções.
CheckBox	
	Permite a colocação de botões de seleção de onde pode ser selecionada apenas uma opção.
RadioButton	



ListBox

Permite criar uma lista de itens que podem ser selecionados.



ComboBox

Permite criar uma lista de itens de onde pode ser selecionado apenas um. Este componente também permite que o usuário digite sua própria opção.



ScrollBar

Permite criar a barras de rolagem verticais ou horizontais, no padrão do Windows®.



GroupBox

Permite agrupar controles como *CheckBox*, *RadioButton*, etc.



RadioGroup

Permite agrupar *RadioButtons* para que se faça a seleção de uma opção.



Panel

Permite agrupar outros componentes. São utilizados para construir barra de *status*, barra de ferramentas, etc. Também são importantes para compatibilizar diferentes tipos de definição de telas.

Os componentes são manipulados através de suas propriedades, métodos e eventos.

PALETA ADDITIONAL

A paleta Additional tem 9 componentes, também de uso comum, mas com algumas funções mais especializadas.



Paleta Additional: mais componentes de uso comum

Os componentes desta paleta são:



BitBtn

Permite a colocação de botões contendo imagens gráficas.



SpeedButton

Permite a criação de barra de ferramentas e conjuntos de botões. Devem ser utilizados juntamente com o componente *Panel*.



MaskEdit

Permite a entrada de dados definindo-se máscaras de leitura.



StringGrid

Permite a apresentação de *strings* em colunas.



DrawGrid

Permite a apresentação de informações em colunas.



Image

Permite a apresentação de imagens gráficas.



Shape

Permite o desenho de figuras geométricas.



Bevel

Permite o desenho de retângulos em relevo.



ScrollBox

Permite a criação de áreas de exibição, com barras de rolagem.

Os componentes são manipulados através de suas propriedades, métodos e eventos.

PALETA WIN95

A paleta Win95 contém 12 componentes para criar aplicações que tenham a aparência do Windows95®.



Paleta Win95: aplicativos com a aparência do Windows95®.

Os componentes desta paleta são:



TabControl

Permite criar páginas que podem ser mudadas por guias ou outros controles.



PageControl

Permite inserir guias para a seleção de páginas.



TreeView

Permite a visualização de dados em forma hierárquica.



ListView

Permite a visualização de dados em formas de listas ou colunas.



ImageList

Permite a criação de listas de imagens.



HeaderControl

Permite a criação de cabeçalhos móveis.



RichEdit

Permite a edição de textos utilizando recursos de formatação.



StatusBar

Permite a criação de barras de status.



TrackBar

Permite a criação de controles deslizantes.



ProgressBar

Permite a criação de barras de progresso.



UpDown

Permite a criação de botões de avanço/retrocesso.



HotKey

Permite a criação de suporte a teclas de atalho.

Os componentes são manipulados através de suas propriedades, métodos e eventos.

A PALETA DIALOGS

A paleta Dialogs contém 8 componentes não visuais para a criação de caixas de diálogo padrão no Windows95©.



Paleta Dialogs: caixas de diálogo prontas.

Os componentes desta paleta são:



OpenDialog

Permite a criação de caixa de diálogo para abrir arquivos.



SaveDialog

Permite a criação de caixa de diálogo para salvar arquivos.



FontDialog

Permite a criação de caixa de diálogo para formatação de fontes.



ColorDialog

Permite a criação de caixa de diálogo para formatação de cores.



PrinterDialog

Permite a criação de caixa de diálogo para impressão.



PrinterSetupDialog

Permite a criação de caixa de diálogo para a configuração das opções de impressão.



FindDialog

Permite a criação de caixa de diálogo para a localização de textos.



ReplaceDialog

Permite a criação de caixa de diálogo para a substituição de textos

Os componentes são manipulados através de suas propriedades, métodos e eventos.

PROPRIEDADES

As propriedades são os atributos da classe ou seja são as suas características.

As **propriedades** mais comuns, presentes nos componentes do DELPHI são:

Nome	Descrição
Align	Determina o alinhamento de controle
Caption	Legenda do componente (com o &, indica atalho)
Name	Nome da instância do componente
Left	Posição esquerda
Top	Posição superior
Height	Altura do componente
Width	Largura do componente
ComponentCount	O número de componentes possuídos
Components	Uma matriz de componentes possuídos
Color	Indica a cor do componente
Font	Indica a fonte (letra) a ser usada no componente
Ctl3D	Define a aparência 2D ou 3D do componente
Enabled	Indica se o componente está ativado ou não
Visible	Indica se o componente está visível ou não
Hint	String utilizada para dicas instantâneas
ShowHint	Mostra ou não as dicas instantâneas
PopupMenu	Menu que será mostrado com o click do botão direito do mouse
TabOrder	Define a ordem de tabulação do componente (tecla TAB)
TabStop	Indica se o componente será ponto de parada para a tecla TAB
HelpContext	Número utilizado para chamar o help sensível ao contexto
Tag	Contém um número inteiro, que pode ser a identificação do componente

MÉTODOS

Os métodos são as ações realizadas pela classe, ou seja, são as funções às quais estão associados um comportamento da classe.

Os **métodos** mais comuns das classes do DELPHI são:

Nome	Descrição
Create	Cria uma nova instância do objeto
Destroy	Destrói a instância do objeto
Show	Torna o componente visível
Hide	Torna o componente invisível
SetFocus	Coloca o foco no componente
Focused	Determina se o componente está com o foco
BringToFront	Coloca o componente na frente dos demais
SendToBack	Coloca o componente atrás dos demais
ScaleBy	Gradua o componente em determinada escala
SetBounds	Muda a posição e o tamanho do componente

EVENTOS

Os **eventos** são blocos de comandos que associam a ocorrência de alguma atividade a uma ação a ser tomada.

Uma linguagem de **programação orientada a eventos** não tem necessariamente um início ou final lógico, isto é, de acordo com uma sequência de comandos, pois as ações só são realizadas a partir do instante que ocorre um evento.

Um evento pode estar associado a:

- clique ou movimento do mouse;
- acionamento do teclado;
- operações sobre janelas (abrir, fechar, criar, redimensionar, mover, ...);
- erros de execução;
- etc.

O DELPHI possui mais de 130 eventos pré-definidos, que estão associados a ocorrência de ações, nas mais diversas classes.

Alguns dos principais **eventos** do DELPHI são:

Nome	Descrição
OnActivate	Ocorre quando o programa ativa o objeto pela primeira vez, ou quando se retorna de um outro aplicativo.
OnChange	Ocorre quando muda o conteúdo de um objeto.
OnClick	Ocorre quando o usuário dá um clique no botão esquerdo do mouse.
OnClose	Ocorre quando o objeto é fechado.
OnCreate	Ocorre quando o objeto é criado.
OnDblClick	Ocorre quando é feito um duplo clique com o botão esquerdo do mouse.
OnDeactivate	Ocorre quando se sai do objeto.
OnDestroy	Ocorre quando se elimina um objeto.
OnDragDrop	Ocorre quando um objeto é arrastado para outro objeto e solto.
OnDragOver	Ocorre quando um objeto é arrastado para cima de outro objeto.
OnDropDown	Ocorre quando se abre um objeto <i>ComboBox</i> ou <i>ListBox</i> .
OnEnter	Ocorre quando o objeto recebe o foco.
OnException	Ocorre quando ocorre um erro de execução na aplicação.
OnExit	Ocorre quando o objeto perde o foco.
OnHelp	Ocorre quando é solicitado a abertura de um arquivo de ajuda.
OnHide	Ocorre quando o objeto passa a ser oculto.
OnKeyDown	Ocorre quando o usuário pressiona uma tecla, incluindo SHIFT, ALT e INSERT.
OnKeyPress	Ocorre quando o usuário pressiona uma tecla ASCII.
OnKeyUp	Ocorre quando o usuário solta uma tecla.
OnMinimize	Ocorre quando se minimiza uma janela.
OnMouseDown	Ocorre quando o usuário clica em um botão do mouse e o cursor é posicionado sobre a área clicada.
OnMouseMove	Ocorre quando o usuário move o cursor dentro da área selecionada.
OnMouseUp	Ocorre quando o usuário solta um botão do mouse.
OnPopup	Ocorre quando se ativa um menu popup com o botão direito do mouse.
OnResize	Ocorre quando se muda o tamanho do objeto.
OnRestore	Ocorre quando se restaura uma janela que foi minimizada.
OnRun	Ocorre quando uma aplicação inicia sua execução.
OnShow	Ocorre antes que o objeto se torne visível.
OnTimer	Ocorre em intervalos periódicos de tempo.

EXCEÇÕES

Exceções são erros que ocorrem durante a execução do programa. Existem diversos tipos de exceções, que podem ocorrer nas seguintes operações:

- alocação de memória;
- criação e utilização de objetos;
- cálculo de expressões matemáticas;
- manipulação de arquivos;
- utilização de recursos do sistema operacional.

Algumas das principais características das exceções no DELPHI são:

- a instância da exceção fornece informações sobre o tipo de problema ocorrido;
- o usuário pode definir suas próprias exceções, utilizando a classe **Exception**;
- para forçar a execução de uma exceção utiliza-se a cláusula **Raise**;
- os tipos de exceção normalmente devem iniciar com a letra **E**.
- uma exceção gera um evento denominado **OnException**.

As exceções podem ser tratadas através das instruções:

- Try ... Except
- Try ... Finally

TRY ... EXCEPT

O bloco **try ... except** é utilizado para definir um código de programa a ser executado no momento que ocorrer uma exceção.

Sintaxe:

```
try
  { Bloco de comandos}
except
  {Bloco de comandos a ser executado quando ocorrer uma exceção}
end;
```

Exemplo: tratamento de uma divisão por zero.

```
.
Valor1 := StrToInt (Edit1.Text);
Valor2 := StrToInt (Edit2.Text)
try
  Resultado := Valor1 div Valor2;
except
  ShowMessage ('Divisão por Zero');
end;
.
```

No entanto, pode ocorrer mais de uma exceção dentro de um mesmo bloco. Neste caso é necessário utilizar o comando on ... do.

ON ... DO

O comando **on ... do** permite tratar mais de um tipo de exceção de um bloco try ... except.

Sintaxe:

```
On
  {Tipo de exceção}
Do
```

{Bloco de comandos da exceção}

Exemplo: tratamento de uma divisão por zero e da digitação de um caracter em um campo que deveria ser numérico.

```
.
try
  Valor1 := StrToInt (Edit1.Text);
  Valor2 := StrToInt (Edit2.Text)
  Resultado := Valor1 div Valor2;
except
  on EDivByZero do
    ShowMessage ('Divisão por Zero');
  on EInOutError do
    ShowMessage ('Os valores devem ser numéricos');
end;
```

TIPOS DE EXCEÇÃO

Os principais tipos de exceção da RTL (*RunTime Library*) do DELPHI, a serem tratadas nos blocos **on ... do** são:

Nome	Descrição
EaccessViolation	ocorre quando se tenta acessar uma região de memória inválida (ex: tentar atribuir valor a um ponteiro cujo conteúdo é <i>nil</i>).
EconvertError	ocorre quando se tenta converter um string em um valor numérico (ex: utilizar a função StrToInt em uma letra).
EdivByZero	ocorre na divisão de um número por zero.
EinOutError	ocorre numa operação incorreta de I/O (ex: abrir um arquivo que não existe).
EintOverFlow	ocorre quando o resultado de um cálculo excedeu a capacidade do registrador alocado para ele (para variáveis inteiras).
EinvalidCast	ocorre quando se tenta realizar uma operação inválida com o operador as (ex: tentar usar um Sender com uma classe que não corresponde a seu tipo).
EinvalidOp	ocorre quando se detecta uma operação incorreta de ponto flutuante.
EinvalidPointer	ocorre quando se executa uma operação inválida com um ponteiro (ex: tentar liberar um ponteiro duas vezes).
EoutOfMemory	ocorre quando se tenta alocar memória mas já não existe mais espaço suficiente.
Eoverflow	ocorre quando o resultado de um cálculo excedeu a capacidade do registrador alocado para ele (para variáveis de ponto flutuante).
ErangeError	ocorre quando uma expressão excede os limites para a qual foi definida (ex: tentar atribuir 11 ao índice de um vetor que pode ir no máximo até 10).
EstackOverflow	ocorre quando o sistema não tem mais como alocar espaço de memória na <i>Stack</i> .
Eunderflow	ocorre quando o resultado de um cálculo é pequeno demais para ser representado como ponto flutuante.
EZeroDivide	ocorre quando se tenta dividir um valor de ponto flutuante por zero.

TRY ... FINALLY

O bloco **try ... finally** permite que um bloco de comandos seja executado, mesmo quando houver uma exceção.

Sintaxe:

```
try
  { Bloco de comandos}
```

```
finally  
    {Bloco de comandos a ser executado mesmo quando ocorrer uma exceção}  
end;
```

Exemplo: garantir que um arquivo será fechado mesmo quando ocorrer uma exceção.

```
assign (arq, 'teste.txt');  
try  
    reset (arq)  
    {outros comandos sobre o arquivo}  
finally  
    close (arq)  
end;
```