

## Capítulo 4

### ***Estudo Detalhado de Componentes***

Ao final do capítulo, você estará apto a:

- Explicar o que são componentes visuais e não-visuais
- Explicar o que são propriedades, eventos e métodos
- Descrever três maneiras de manipular ou personalizar componentes

### **Estudo Detalhado de Componentes**

#### ***Overview***

Componentes são blocos de construção de uma aplicação. Este capítulo cobre os principais aspectos de componentes: propriedades, eventos e métodos. O estudo dos componentes também inclui várias maneiras de arranjá-los, alinhá-los e definir propriedades de componentes.

Este capítulo também enfatiza a utilização do help on-line, pois é uma fonte extensiva sobre componentes.

## ***Descrição de Um Componente***

### **Introdução**

Aplicações do Delphi são construídas utilizando-se componente. Um componente é um objeto que pode ser manipulado para construir e personalizar uma aplicação. Mesmo um form, embora não seja encontrado na Component Palette, é um componente. Ele é um componente que pode conter outros componentes. Genericamente, o termo componente refere-se a itens encontrados na Component Palette. Este significado aplica-se no decorrer do curso.

Os componentes são categorizados em componente visuais e não-visuais.

### **Componentes Visuais**

Componentes Visuais aparecem durante a execução da mesma forma como aparecem durante o design. Exemplos são botões e edit fields.



O Delphi utiliza o termo controle, como um significado para um componente visual que possa ser visualizado quando a aplicação estiver sendo executada. Os controles são divididos em dois grupos: controles ajanelados e não ajanelados. Consulte o Apêndice A: Controles no Delphi, para mais detalhes.

### **Componentes Não-visuais**

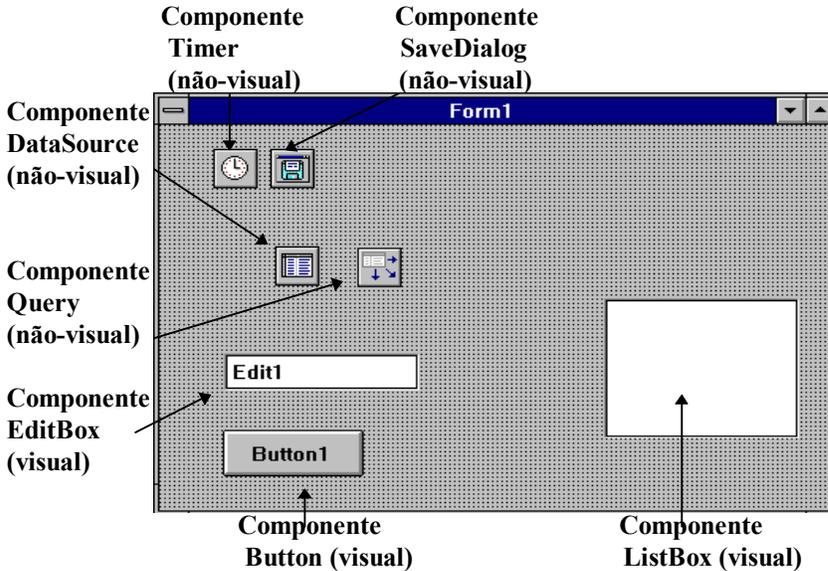
Componentes não-visuais aparecem durante o design como pequenas figuras em um form. Entretanto, eles:

- Fazem com que um quadro de dialogo seja exibido quando chamado. Um exemplo é o componente SaveDialog.

- Não aparecem em momento nenhum durante a execução. Exemplos são os componentes Timer, DataSource, e Table.

### Exemplos de Componentes Não-Visuais

A figura a seguir mostra exemplos de componentes Visuais e não-visuais:



### Código Gerado para um Form

Como mencionado, o Delphi permite criar uma aplicação com um mínimo de codificação. Quando você inicializa o Delphi, um novo form é automaticamente gerado e o Delphi cria um arquivo unit correspondente. O código correspondente ao form pode ser visualizado na janela do code editor. A janela code editor geralmente é posicionada atrás do form default, Form1. código é gerado quando você adiciona um componente e um evento, como você verá posteriormente neste capítulo. A janela do code editor aparece, como segue:

```

UNIT1.PAS

type
  TForm1 = class(TForm)
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

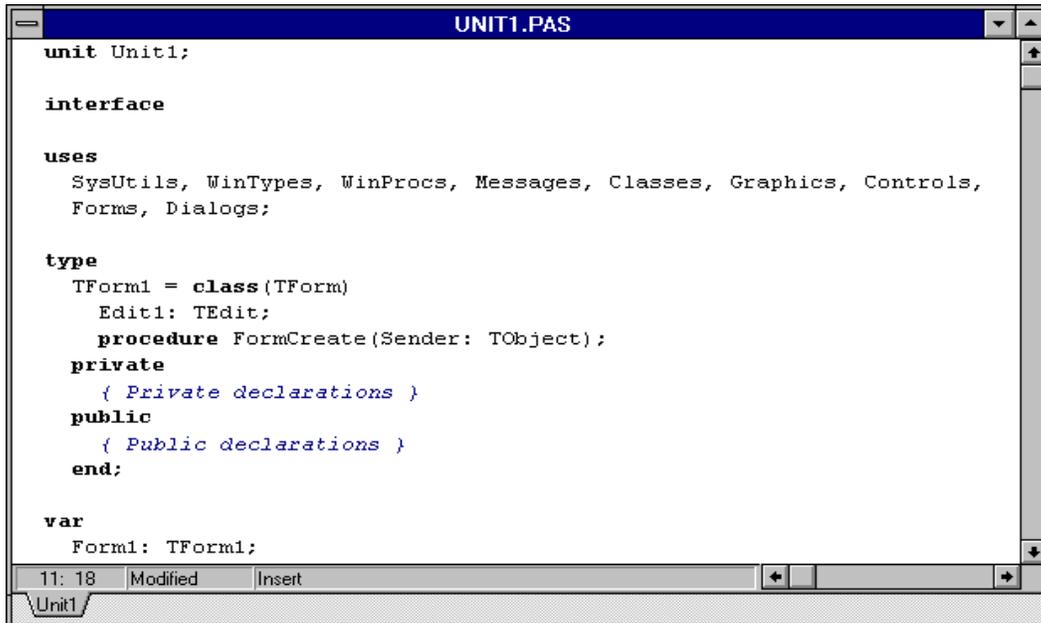
implementation

{$R *.DFM}

end.
    
```

## Código Gerado Adicionando um Componente

Quando você adiciona um componente ao form, o Delphi gera o código para isto, como mostrado no exemplo a seguir, com um componente Edit:



```
unit Unit1;

interface

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

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
```

## Aspectos de um Componente

Um componente Delphi possui os três seguintes aspectos:

- Propriedades
- Eventos
- Métodos

Cada Aspecto é explicado nas seções a seguir:

### Propriedades

Propriedades são atributos ou campos de componente. Propriedades podem ser definidas durante a execução ou design. As propriedades controlam a forma como um componente se comporta e sua aparência em uma aplicação. Por exemplo, um botão é um componente que você pode adicionar a um form. Uma das muitas propriedades de um botão é a propriedade Caption. Definir a propriedade Caption altera o texto exibido no botão.

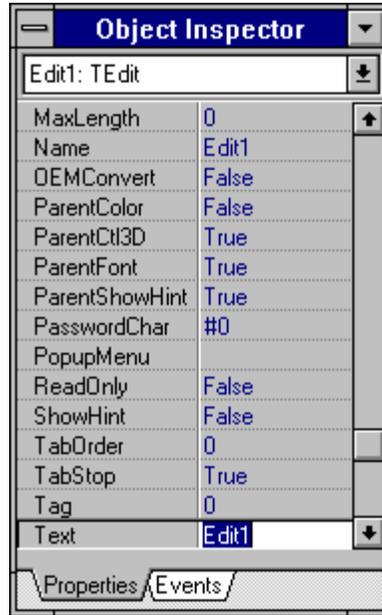
### Influenciando o Comportamento de um Componente

O Object Inspector exibe as propriedades de um componente. O Object Inspector é a conexão entre a aparência visual do componente e o código que o Delphi gera para fazer com que a aplicação seja executada. O Object Inspector é utilizado para definir propriedades durante o design.

### Exemplo:

#### Propriedade para o Component Edit

As propriedades são listadas na página Properties do Object Inspector. O exemplo a seguir mostra parte desta lista para o componente Edit adicionado ao form.



**Exemplo:**

**Descrição das Propriedades para o Componente Edit**

Muito embora as propriedades no Object Inspector variem para cada componente selecionado, uma vez que você entenda as propriedades de um componente, você pode aplicar este conhecimento para outros componentes. A tabela a seguir descreve a lista completa das propriedades do componente Edit. Esta lista é uma representação das propriedades comumente encontradas para os componentes do Delphi.

Propriedade	Descrição
<b>AutoSelect</b>	Determina se o texto é selecionado automaticamente quando o usuário mover-se ate o componente Edit ou Memo utilizando tab
<b>AutoSize</b>	Determina se um componente tem o tamanho reajustado automaticamente para acolher seu conteúdo
<b>BorderStyle</b>	Determina o tipo de borda exibido para um componente
<b>CharCase</b>	Determina se o estilo dos caracteres exibidos sera em maiúsculas, minúsculas ou misturadas
<b>Color</b>	Determina a seguir: <ul style="list-style-type: none"> <li>• A cor de fundo de form</li> <li>• A cor de um controle ou figura</li> </ul>
<b>Ctl3D</b>	Determina se o controle possui um visual tridimensional ou bidimensional
<b>Cursor</b>	Determina a imagem que o ponteiro do mouse assume (tal como seta ou I-beam) quando passar por área coberta pelo componente
<b>DragCursor</b>	Determina a imagem que o cursor assume (tal como seta ou I-beam) quando passar por sobre um componente que aceite arrasto
<b>DragMode</b>	Determina o componente drag-and-drop de um componente
<b>Enabled</b>	Determina se o componente responde a eventos do mouse, teclado ou timer
<b>Font</b>	Determina os atributos(cor, tamanho, estilo, ou nome) do seguinte: <ul style="list-style-type: none"> <li>• Texto escrito sobre ou dentro de um componente ou objeto</li> <li>• Texto enviado a impressora</li> </ul>
<b>Height</b>	Determina o tamanho vertical de um componente ou objeto
<b>HelpContext</b>	Determina um número único para cada tela a ser chamada no Help sensível ao contexto
<b>HideSelection</b>	Determina se o texto selecionado mantém-se selecionado quando o objeto perder o foco

<b>Propriedade</b>	<b>Descrição</b>
<b>Hint</b>	Determina a string de texto que aparecerá quando o evento OnHint ocorrer (quando o cursor passar sobre um componente ou item de menu, seu significado será exibido)
<b>Left</b>	Determina a localização horizontal em pixels do lado esquerdo de : <ul style="list-style-type: none"> <li>• Um componente em relação ao form ou painel ou outros objetos container</li> <li>• Um form em relação a tela</li> </ul>
<b>MaxLength</b>	Determina o número máximo de caracteres que um usuário pode digitar em um componente Edit ou Memo. Zero(0) significa sem limite.
<b>Name</b>	Determina um nome único para um componente ou objeto
<b>OEMConvert</b>	Determina se o texto é convertido para caracteres OEM
<b>ParentColor</b>	Determina onde o componente procurará pela informação sobre sua cor, como segue: <ul style="list-style-type: none"> <li>• Se o valor form True, o componente utiliza a propriedade Color do componente pai.</li> <li>• Se o valor form False, o componente utiliza sua própria propriedade Color.</li> </ul>
<b>ParentCtl3D</b>	Determina onde o componente procurará pela informação sobre seu visual tridimensional, como segue: <ul style="list-style-type: none"> <li>• Se o valor form True, o componente utiliza a propriedade tridimensional do componente pai.</li> <li>• Se o valor form False, o componente utiliza sua própria propriedade tridimensional.</li> </ul>
<b>ParentFont</b>	Determina onde um componente procurará pela informação sobre seu fonte, como segue: <ul style="list-style-type: none"> <li>• Se o valor form True, o componente utilizará a propriedade Font do componente pai.</li> <li>• Se o valor form False, o componente utilizará sua própria propriedade Font.</li> </ul>
<b>ParentShowHint</b>	Determina onde um controle procurará se o Help hint deve ser exibido, como segue: <ul style="list-style-type: none"> <li>• Se o valor form True, o controle utiliza a propriedade ShowHint do componente pai.</li> <li>• Se o valor form False, o controle utiliza sua própria propriedade ShowHint.</li> </ul>
<b>PassWordChar</b>	Determina se um componente Edit ou Memo exibe caracteres especiais (ao invés do texto real) quando uma senha form digitada
<b>PopupMenu</b>	Identifica o nome do menu pop-up que aparecerá quando um dos seguintes acontecer: <ul style="list-style-type: none"> <li>• O usuário seleciona um componente e pressiona o botão direito do mouse.</li> <li>• O método PopUp de um menu pop-up é executado.</li> </ul>
<b>ReadOnly</b>	Torna um componente read-only durante a execução, para que o usuário possa alterar o valor do campo ou do dataset
<b>ShowHint</b>	Determina se o Help está habilitado ou não, para a aplicação, como segue: <ul style="list-style-type: none"> <li>• Se o valor form True, Help Hints está habilitado.</li> <li>• Se o valor form False, Help Hints está desabilitado.</li> </ul>
<b>TabOrder</b>	Indica a posição do componente na ordem tab do container, a ordem na qual um componente recebe o foco quando a tecla tab é pressionada.
<b>TabStop</b>	Determina se um usuário pode pressionar tab até o componente
<b>Tag</b>	Cria um local disponível para armazenar valor integer como parte de um componente. A propriedade Tag, embora não utilizada pelo Delphi, está disponível para necessidades especiais do usuário.
<b>Text</b>	Especifica a string de texto exibida em um componente ou outro objeto
<b>Top</b>	Determina o posicionamento vertical em pixels do canto superior esquerdo de: <ul style="list-style-type: none"> <li>• Um componente em relação ao form, painel ou outro controle container</li> <li>• Um form em relação a tela</li> </ul>
<b>Visible</b>	Determina se um componente aparece na tela
<b>Width</b>	Determina o tamanho horizontal do componente e outros objetos

## Utilizando Editores de Propriedades para Definir ou Alterar uma Propriedade

O Object Inspector oferece diversas maneiras de exibir propriedades e suas variações para que você possa defini-las ou alterá-las. Estes mecanismos são chamados de editores de propriedades.

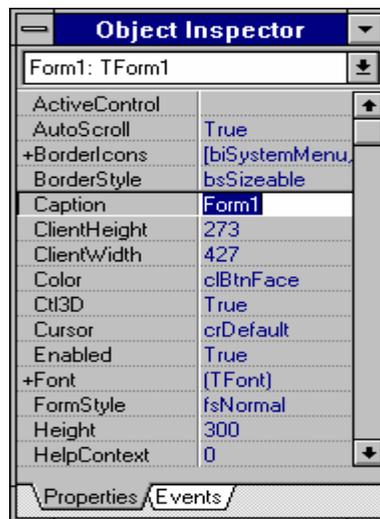


Se você criar seu próprio componente, você pode utilizar os mesmos editores de propriedades para definir as propriedades de seu componente. Os editores são objetos do Delphi. Os editores são:

- Editor simples
- Editor drop-down list
- Editor com quadro de dialogo
- Editor com propriedades aninhadas

### Editor Simples

O editor simples permite digitar um novo valor no lugar de um valor default para tipos numérico e strings alfanuméricas. No exemplo a seguir, o valor default da propriedade Caption é Form1. Você pode digitar um novo nome em seu lugar. O Delphi checa pela validade do valor para certificar-se de que uma string numérica não foi digitada no lugar de uma string alfanumérica ou vice-versa.



### Editor Drop-Down List

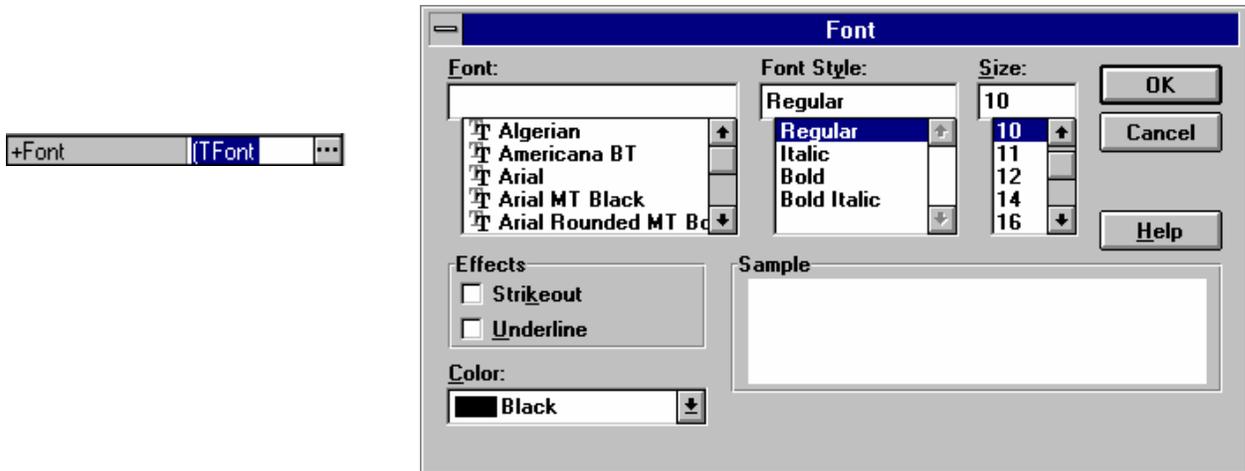
Uma seta para baixo() após algum valor na coluna Values indica que um valor é limitado a uma lista de escolhas. Color e Cursor são exemplos de propriedades que possuem uma drop-down list.

### Editor com Quadro de Diálogo

Reticências (...) indicam que existe um quadro de opções. O exemplo a seguir mostra um quadro de dialogo utilizado para definir diversos atributos para o tipo de objeto TFont. Font e Color são exemplos de propriedades que possuem um quadro de diálogo.



A propriedade Color não possui reticências para indicar existência de um quadro de dialogo. Você deve dar um duplo-clique para exibi-lo.

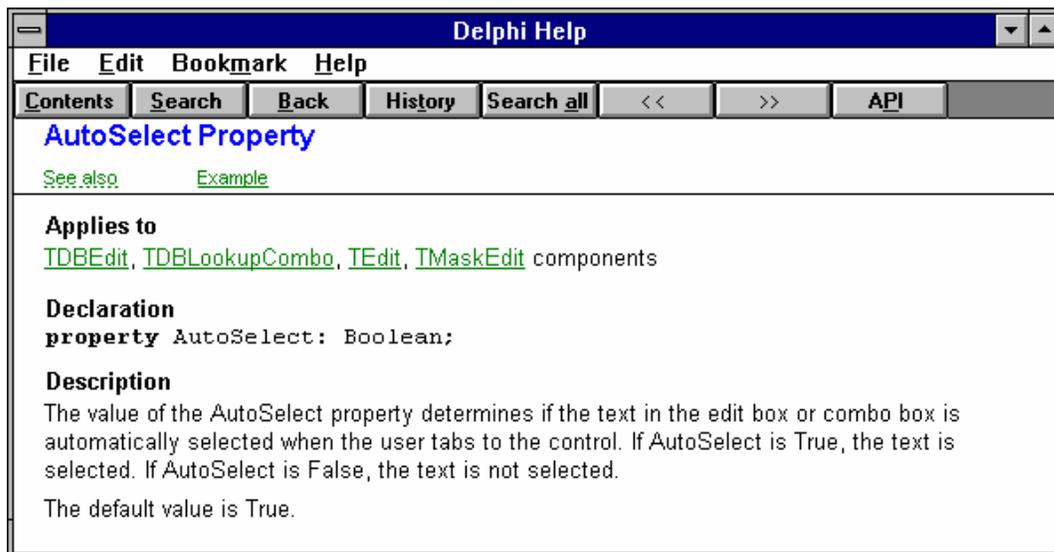


### Editor com Propriedades Aninhadas

Algumas propriedades possuem menus aninhados (Submenus ocultos) de propriedades. Um sinal de mais (+) na frente do nome de tal propriedade indica que um ou mais menus aninhados de propriedades existem para aquela propriedade. Dando um clique sobre o sinal para exibir o submenu, um sinal de menos substitui o sinal de mais. Um sinal de menos (-) indica que não existe nenhum submenu adicional. Font é um exemplo de propriedade que possui um menu aninhado. O sinal de mais no submenu +Style indica que a propriedade Style possui um menu aninhado.

### Utilizando a Tecla F1 para Uma Descrição Completa de uma Propriedade

A importância de Help sensível ao contexto como característica do Delphi é aparente quando você precisa de uma descrição completa de uma propriedade. Pressionando a tecla F1 sobre a propriedade e seu(s) valor(es), como na propriedade AutoSelect, como segue:



### Tópicos Relacionados Através da Tecla F1

Dando um clique sobre a opção See Also (em verde no texto) liga propriedade com tópicos relacionados. Uma lista de tópicos é exibida, como no exemplo a seguir, da propriedade AutoSelect:

**See Also**  
[AutoSize](#) property  
[SelLength](#) property  
[SelStart](#) property  
[SelText](#) property  
[Text](#) property

### Exemplos de Códigos Através da Tecla F1

Dando um clique sobre a opção Example (em verde, no texto) liga a propriedade a um exemplo de código relacionado para propriedade AutoSelect.

O Help on-line inclui o comando Copy que permite copiar o texto do Help e cola-lo na aplicação. Isto pode ser especialmente útil se você encontrar exemplo de código que queira utilizar em sua aplicação. Para utilizar o comando Copy, exiba o tópico de Help que queira copiar, e no menu Edit, selecione Copy.

### Passos para Definir Propriedades Durante o Design

Execute os passos para definir propriedades durante o design:

Passo	Ação
1	Dê um clique sobre uma página da Component Palette, e de um clique sobre o componente que você queira, como por exemplo o componente Edit.
2	De um clique sobre a área do form onde você queira inserir o componente. O componente aparece no form com o componente Edit. O nome do componente e o tipo do objeto Edit1: TEdit é colocado no Object Selector. A coluna Values exibe o nome Edit1 como o texto default da propriedade Text.
3	Utilize um editor de propriedade para alterar o valor da propriedade. As alterações feitas para a maioria das propriedades aparecem imediatamente no form.

## Eventos

### Introdução

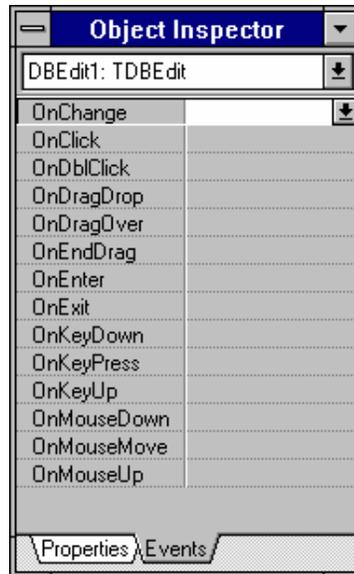
Evento é outra característica de um componente. Eventos são ações de usuários ou ocorrências do sistema que um componente possa reconhecer, tal como um clique de mouse.

### Event Handler

Quando você seleciona um evento para um componente, o Delphi gera um heading de procedure e bloco de código. O código que você escreve especifica como um componente deve responder a um evento, e é chamado de event handler. Event handler são procedures especializadas. O Object Inspector permite especificar quais procedures estão associadas a determinados eventos.

### Exemplo: Eventos para o Componente Edit

Os eventos estão listados na página Events do Object Inspector. O exemplo a seguir mostra uma lista de eventos para o componente Edit:



### Exemplo : Descrição dos Eventos para o Componente Edit

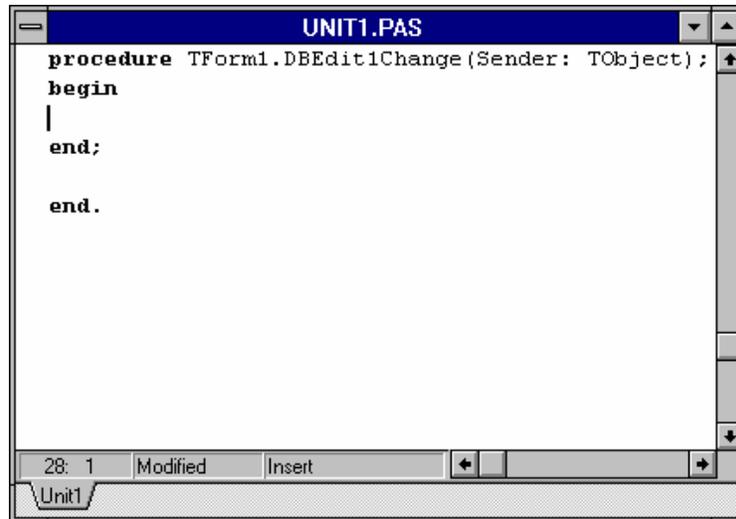
Muito embora os eventos no Object Inspector variem para cada tipo de componente, uma vez que você entenda os eventos para um componente, você pode aplicar este conhecimento para outros componentes.

A tabela a seguir lista os eventos e suas descrições para o componente Edit:

Evento	Descrição
<b>OnChange</b>	Ocorre quando um determinado valor de um objeto ou componente é alterado
<b>OnClick</b>	Ocorre quando o usuário dá um clique sobre o componente
<b>OnDblClick</b>	Ocorre quando o usuário dá um duplo-clique sobre um componente
<b>OnDragDrop</b>	Ocorre quando o usuário solta um objeto sendo arrastado
<b>OnDragOver</b>	Ocorre quando o usuário arrasta um objeto sobre um componente
<b>OnEndDrag</b>	Ocorre quando termina o arrasto de um objeto
<b>OnEnter</b>	Ocorre quando um componente torna-se ativo
<b>OnExit</b>	Ocorre quando o foco de input passa de um componente para outro
<b>OnKeyDown</b>	Ocorre quando o usuário pressiona qualquer tecla quando um componente possui o foco
<b>OnKeyPress</b>	Ocorre quando o usuário pressiona uma única tecla de caractere
<b>OnKeyUp</b>	Ocorre quando o usuário solta uma tecla que estava pressionada
<b>OnMouseDown</b>	Ocorre quando o usuário pressiona o botão mouse enquanto o ponteiro do mouse estiver sobre um componente
<b>OnMouseMove</b>	Ocorre quando o usuário move o ponteiro do mouse quando o ponteiro do mouse estiver sobre o componente
<b>OnMouseUp</b>	Ocorre quando o usuário solta o botão do mouse quando o ponteiro do mouse estiver sobre um componente

### Heading de de Procedure Gerado Através de um Duplo-Clique em um Evento

No Object Inspector, um duplo-clique na coluna Values próximo ao evento gera um heading de procedure para o código do evento e o associa ao evento, como mostrado no exemplo a seguir, no evento OnChange do componente Edit. O cursor é posicionado entre o par begin...end para que você possa digitar o código para o comportamento desejado.



### Chamando Procedures com Parâmetros

Um event handler pode ter parâmetros. As instruções dentro da procedure podem utilizar os valores passados como parâmetros quando o programa é executado. Os valores são tratados como variáveis declaradas dentro da procedure para passagem de dados. Os parâmetros aparecem entre parêntesis após o nome da procedure, como mostrado na figura anterior.

### Passos para Adicionar um Enevt Handler

Execute os passos a seguir para adicionar um event handler ao componente Edit:

Passo	Ação
1	Após modificar uma ou mais propriedades de um componente, de um clique sobre a aba da página Events para exibi-la. A página Events exibe todos os eventos reconhecidos pelo componente selecionado, como no exemplo anterior.
2	Selecione um evento e de um duplo-clique na coluna Values próximo ao evento. O Delphi gera um event handler (heading da procedure e o bloco de código) na janela do code editor. O cursor é posicionado dentro do par <b>begin...end</b> .
3	Dentro do par <b>begin...end</b> , digite as instruções que você quer que o Delphi execute quando o componente receber o evento. O Delphi adiciona uma instrução de procedure na porção interface do arquivo unit. Se você alterar o nome de event handler, o Delphi altera o nome onde quer que apareça dentro no código-fonte.

## Métodos

### Introdução

Alem de propriedades e eventos, um componente possui métodos. Como os componentes são objetos, eles herdam não somente propriedades e eventos, mas também métodos. Em geral, um método é uma procedure ou função associada a um componente. Nos termos de programação, um método é uma procedure ou função declarada como parte de um objeto.

### Exemplo: Descrição dos Métodos de um Componente Edit

A tabela a seguir lista os métodos e suas descrições de um componente Edit:

Método	Descrição
--------	-----------

<b>BeginDrag</b>	Uma procedure que inicia com o arrasto de um controle <ul style="list-style-type: none"> <li>• Se o valor do parâmetro Immediate for True, o ponteiro do mouse altera para o valor da propriedade DragCursor e o arrasto inicia imediatamente.</li> <li>• Se o valor do parâmetro Immediate for False, o ponteiro do mouse não muda de valor e o arrasto inicia somente após o usuário mover o ponteiro do mouse cinco pixels.</li> </ul>
<b>BringToFront</b>	Uma procedure que põe um componente ou form na frente de outros componentes ajanelados ou não, ou forms dentro de seu componente pai ou form.
<b>ClassName</b>	Uma função que retorna o nome da classe de um objeto
<b>ClassParent</b>	Uma função que retorna a classe que seja pai de um objeto
<b>ClassType</b>	Uma função que retorna o tipo de classe de um objeto
<b>Create</b>	Um construtor que inicializa um objeto ou componente de acordo com certas procedures padrão. Um construtor é declarado com a palavra reservada constructor.
<b>Clear</b>	Uma procedure que deleta itens ou texto de um controle
<b>CleanSelection</b>	Uma procedure que deleta o texto selecionado de um componente Edit ou Memo
<b>CopyToClipboard</b>	Uma procedure que copia uma seleção ao Clipboard
<b>CutToClipboard</b>	Uma procedure que deleta a seleção de um controle e a copia para o Clipboard
<b>Destroy</b>	Um destrutor que destrói um objeto, controle ou componente, e libera a memória que estava alocada a ele. Um destrutor é declarado com a palavra destructor
<b>Dragging</b>	Uma função que especifica se um objeto está sendo arrastado
<b>EndDrag</b>	Uma procedure que termina o arrasto de um objeto
<b>Free</b>	Uma procedure que destrói um objeto e libera a memória alocada para ele
<b>GetSelTextBug</b>	Uma função que copia para um buffer o texto selecionado de um componente Edit ou Memo, e retorna o número de caracteres copiados
<b>GetTextLen</b>	Uma função que retorna o tamanho do texto de um controle
<b>Hide</b>	Uma Procedure que opera somente durante a execução e torna um form ou controle invisível através da definição da propriedade Visible para False, do form ou controle
<b>PasteFromClipboard</b>	Uma procedure que copia texto do Clipboard para um controle
<b>Refresh</b>	Uma procedure que limpa qualquer imagem que esteja na tela e depois redesenha o controle inteiro
<b>ScaleBy</b>	Uma procedure que reajusta um controle a um percentual de seu tamanho anterior
<b>ScrollBy</b>	Uma procedure que rola o conteúdo de um form ou controle ajanelado
<b>SelectAll</b>	Uma procedure que seleciona o bloco de texto inteiro em um controle
<b>SendToBack</b>	Uma procedure que coloca um componente, ajanelado ou não, por tras de todos os outros componentes dentro de um componente pai ou form
<b>SetBounds</b>	Uma procedure que define as propriedades Left, Right, Height e Width de um componente aos valores passados com os parâmetros ALeft, ARight, AHeight e AWidth
<b>SetFocus</b>	Uma procedure que põe o foco de input no controle
<b>SetSelTextBuf</b>	Uma procedure que define texto selecionado em um componente Edit ou Memo ao texto na string null-terminated para o qual o buffer aponta
<b>SetTextBuf</b>	Define o texto em um componente para o texto do buffer
<b>Show</b>	Uma procedure que torna um form ou controle visual definindo sua propriedade Visible para True
<b>Update</b>	Uma procedure que redesenha um componente

### **Chamando Métodos**

Você pode chamar métodos(funções ou procedures) utilizando a notação de ponto (.), conhecida como dot notation. Por exemplo para chamar um método show:

*Edit.Show*

Métodos, diferente de propriedades, nunca são ativados durante o design. Eles são procedures ou funções que operam no componente.

### **Utilizando a Tecla F1 para uma Descrição Completa dos Métodos**

Pressionando **F1** sobre o nome de um determinado componente será mostrado uma lista dos métodos, bem como as propriedades e eventos associados àquele componente. Cada componente é listado pelo seu tipo de objeto. Você também pode selecionar **Topic Search** no menu **Help** para pesquisar qualquer componente.

Você pode querer pesquisar um componente com o qual queira trabalhar para tornar-se familiar com todos os seus métodos disponíveis.

## ***Manipulando e Personalizando Componentes***

### **Introdução**

Adicionar componente ao form é somente uma parte da tarefa em utiliza-los. Eles precisam ser manipulados e personalizados para melhor utilização e apresentação no form. As operações a seguir estão envolvidas :

- Definir propriedades comuns a múltiplos componentes
- Reajustar um componente
- Adicionar múltiplas cópias de um componente
- Agrupar componentes
- Alinhar componentes

### **Operações Adicionais**

As operações sobre componentes a seguir não são cobertas no curso pois são idênticas no MS Windows:

- Deletar
- Mover
- Recortar e colar
- Copiar

### **Utilizando o Menu Edit**

A maioria das opções para manipulação dos componentes estão no menu Edit.

### **Selecionando Um Componente e Vários Componentes**

Quando um componente é selecionado, ele é realçado com pequenos quadrados negros ao redor de suas bordas. A seleção de componentes varia, se você quer selecionar um ou vários componentes. Você pode selecionar um componente das seguintes maneiras:

- Dando um clique sobre o componente no form
- Dando um clique sobre o nome do componente no Object Selector, na parte superior do Object Inspector
- Pressionando tab ate o componente no form

Você pode selecionar vários componentes das seguintes maneiras:

- Mantendo pressionada a tecla Shift conforme for dando cliques sobre os componentes
- Dando um clique em uma área em branco do form e arrastando o ponteiro do mouse pelo componente para inclui-los. Um retângulo aparecera conforme for arrastando o ponteiro do mouse.

### Definindo Propriedades para Vários Componentes

No início do curso você aprendeu como definir uma propriedade para um componente. A maioria dos componentes possuem propriedades em comum, tais como as propriedades Height e Visible. Você pode definir propriedades em comum em um form sem ter que selecionar e alterar cada componente individualmente.

### Passos para Definir Propriedades Compartilhadas

Siga os passos a seguir para definir propriedades em comum para vários componentes:

Passo	Ação
1	Selecione todos os componentes para os quais você queira definir propriedades em comum, dando um clique sobre o primeiro componente e, mantendo a tecla Shift pressionada, dando cliques em cada componente adicional. A página Properties do Object Inspector exibe somente as propriedades que todos os componentes possuem em comum.
2	Defina as propriedades que você queira compartilhar. Observe que todos os componentes adquirem a mesma definição da propriedade. Alterações visíveis no design são refletidas em cada componente.

### Reajustando um Componente

Você pode reajustar componentes tanto quando os insere como após inseri-los. Quando você seleciona um componente, pequenos quadrados chamados manipuladores de reajuste aparecem em torno da borda do componente. Quando passar com o ponteiro do mouse sobre um manipulador, o ponteiro do mouse alterna para uma seta de duas pontas (). Quando o ponteiro do mouse estiver neste formato, você pode mover os manipuladores para tornar o componente maior ou menor. Se você reajustar o tamanho do componente conforme for adicionando, você não precisa utilizar os manipuladores.

### Passos para Reajustar um Componente na Inserção

Execute os passos a seguir para reajustar um componente conforme for inserindo-o:

Passo	Ação
1	Selecione o componente na Component Palette.
2	Posicione o ponteiro do mouse no form onde queira que o componente apareça e arraste o mouse na direção desejada. Conforme for arrastado, um retângulo aparece para indicar o tamanho e posição do componente.
3	Quando o retângulo tiver o tamanho desejado, solte o botão do mouse. O componente aparece no mesmo tamanho que o retângulo.

### Passos para Adicionar um Componente e Reajustá-lo

Execute os passos a seguir para adicionar um componente e reajustá-lo.



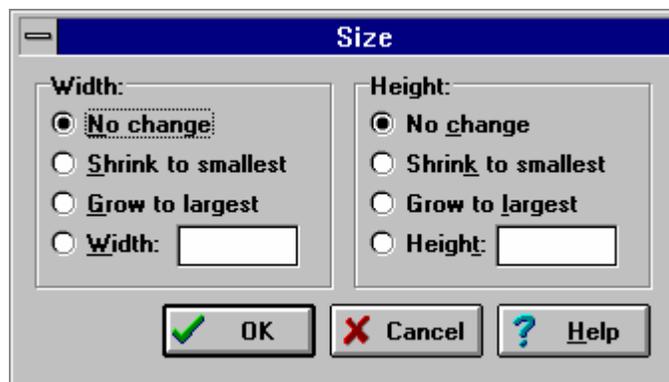
Para um reajuste em pixels, você pode utilizar **Size** no menu **Edit**.

Passo	Ação
1	Selecione um componente na Component Palette.
2	Aponte para a área do form onde queira que o componente apareça e de um clique com o botão esquerdo do mouse. O componente aparece no form.
3	Passa com o ponteiro do mouse por sobre os manipuladores. Quando o ponteiro do mouse alternar para uma seta de duas pontas, arraste os manipuladores para reajustar o tamanho.

### Passos para Reajustar Vários Componentes

Execute os passos a seguir para reajustar vários componentes ao mesmo tempo.

Passo	Ação
1	Selecione o primeiro componente.
2	Mantenha pressionada a tecla Shift e de um clique sobre o resto dos componentes que queira reajustar. Todos os componentes selecionados aparecem realçados.
3	No menu Edit, selecione Size. O quadro de dialogo aparece, como segue:
4	Selecione as opções de reajuste desejadas. Para um reajuste preciso, você pode digitar um número em pixels nos campos <b>Width</b> e <b>Height</b> .
5	Dê um clique em <b>Ok</b> .



### Adicionando Várias Cópias de Um Componente

Você pode adicionar várias cópias do mesmo tipo de componente pressionando a tecla Shift quando selecionar o componente na Component Palette. Quando você pressiona a tecla Shift e da um clique sobre o componente para a primeira cópia, um retângulo aparece ao redor do componente, como mostrado no exemplo com o componente Edit:

Quando der um clique em uma área do form, a primeira cópia do componente aparecerá. Cada clique no form, após isto, adiciona uma cópia ao form. Dando um clique sobre a ferramenta de seleção cursor (o primeiro botão da Component Palette) termina o modo de múltiplas cópias.



### Alinhando Componentes

Você pode alinhar componentes através da barra de menus de três maneiras:

- No menu **Views**, selecione **Alignment Palette**
- No menu **Edit**, selecione **Align**
- No menu **Options**, selecione **Environment**



**Environment** no menu **Options** permite alterar opções default no ambiente do Delphi.

Você também pode alinhar componentes utilizando a grade do form. A grade é uma característica default e aparece como linhas de pontos do form.

### Passos para Alinhar Componentes Utilizando a View/Alignment Palette

Execute os passos a seguir para alinhar componentes através do menu View selecionando Alignment Palette.

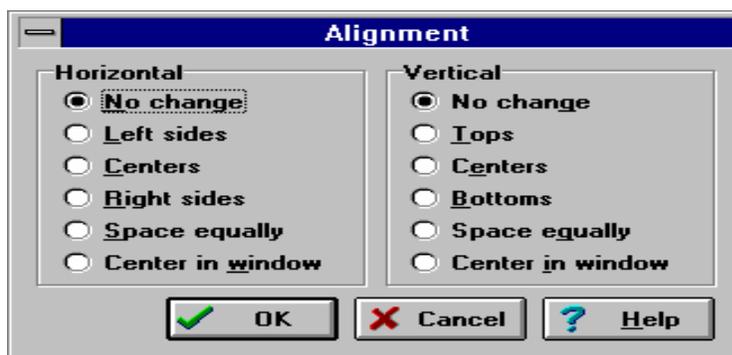
Passo	Ação
1	Selecione os componentes que queira alinhar.
2	No menu View, selecione Alignment Palette. A <b>Align Palette</b> aparece, como segue:
3	Passa com o ponteiro do mouse sobre cada botão para exibir seu significado.
4	Dê um clique em um botão de alinhamento.



### Passos para Alinhar Componentes Utilizando Edit/Align

Execute os passos a seguir para alinhar componentes utilizando o menu Edit e selecionando Align.

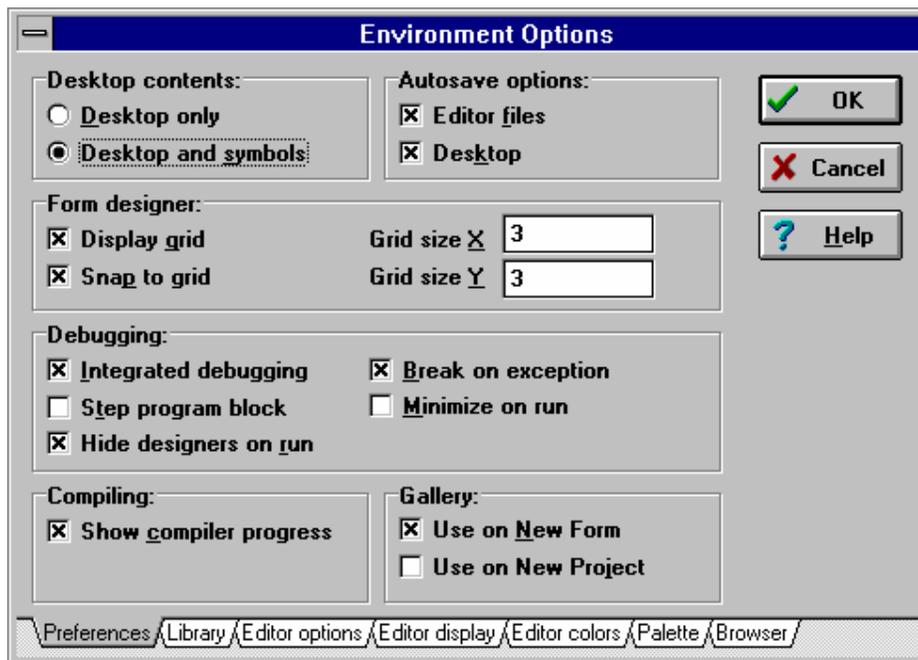
Passo	Ação
1	Selecione os componentes que queira alinhar.
2	No menu Edit, selecione Align. O quadro de dialogo Alignment aparece como segue:
3	Dê um clique sobre uma opção de alinhamento e de um clique em OK.



### Passos para Alinhar Componentes Utilizando Options/Environment

Execute os passos a seguir para alinhar componente utilizando o menu Options e selecionando Environment.

Passo	Ação
1	No menu <b>Options</b> , selecione <b>Environment</b> . O quadro de dialogo Environment Options aparece, como segue, com a página Preferences na frente:
2	No GroupBox Form designer, certifique-se de que os seguintes estão habilitados: <ul style="list-style-type: none"> <li>• <b>Display grid</b> Torna as linhas de pontos visíveis no form, para alinhamento dos componentes</li> <li>• <b>Snap to grid</b> Faz com que o canto superior esquerdo dos componentes sejam alinhados com a marca de grade mais próxima</li> </ul>
3	Para alterar a granularidade ( a distancia dos pontos da grade), digite novos valores no lugar dos exibidos. O valor default é 8 pixels para o espaçamento X (horizontal) e Y (vertical). Um número maior que 8 aumenta a distancia dos pontos, é um número menor que 8 aproxima os pontos.
4	De um clique em <b>OK</b>
5	Posicione os componentes no form alinhando-os com a grade.



## Resumo do capítulo

### Pontos Chave

Após completar este capítulo, você aprendeu que:

- Componentes visuais geralmente aparecem durante a execução da mesma forma que aparece durante o design.
- Componentes não visuais aparecem como pequenas figuras no form, mas não aparecem durante a execução a menos que exibam um quadro de dialogo.
- Um componente possui três aspectos: propriedades, eventos e métodos.
- Uma propriedade é um atributo de um componente. As propriedades controlam a forma como um componente é exibido e se comporta em uma aplicação.

- Um evento é uma ação do usuário ou uma ocorrência do sistema que sua aplicação pode reconhecer, tal como um clique de mouse.
- Um método é uma procedure ou função declarada como parte de um objeto.
- Manipulação e personalização de componentes envolvem várias operações, tais como definir suas propriedades ou reajustar os componentes.

### **Termos e Definições**

A tabela a seguir é uma referência rápida aos termos mostrados neste capítulo:

<b>Termo</b>	<b>Definição</b>
<b>Event handler</b>	Uma procedure associada a um evento do componente que indica como manipular o evento
<b>Evento</b>	Uma ação do usuário ou ocorrência do sistema que um componente pode reconhecer
<b>Função</b>	Uma subrotina que retorna um valor
<b>Grade</b>	As linhas e colunas de pontos que servem como guias no alinhamento de componentes em um form
<b>Instrução</b>	A unit mais simples em um programa. Instruções são separadas por um ponto-e-virgula (;).
<b>Método</b>	Uma procedure ou função declarada dentro do objeto, componente ou controle.
<b>Procedure</b>	Um subprograma
<b>Propriedades Compartilhadas</b>	Propriedades comuns a todos os componentes selecionados
<b>Rotina</b>	Uma procedure ou função