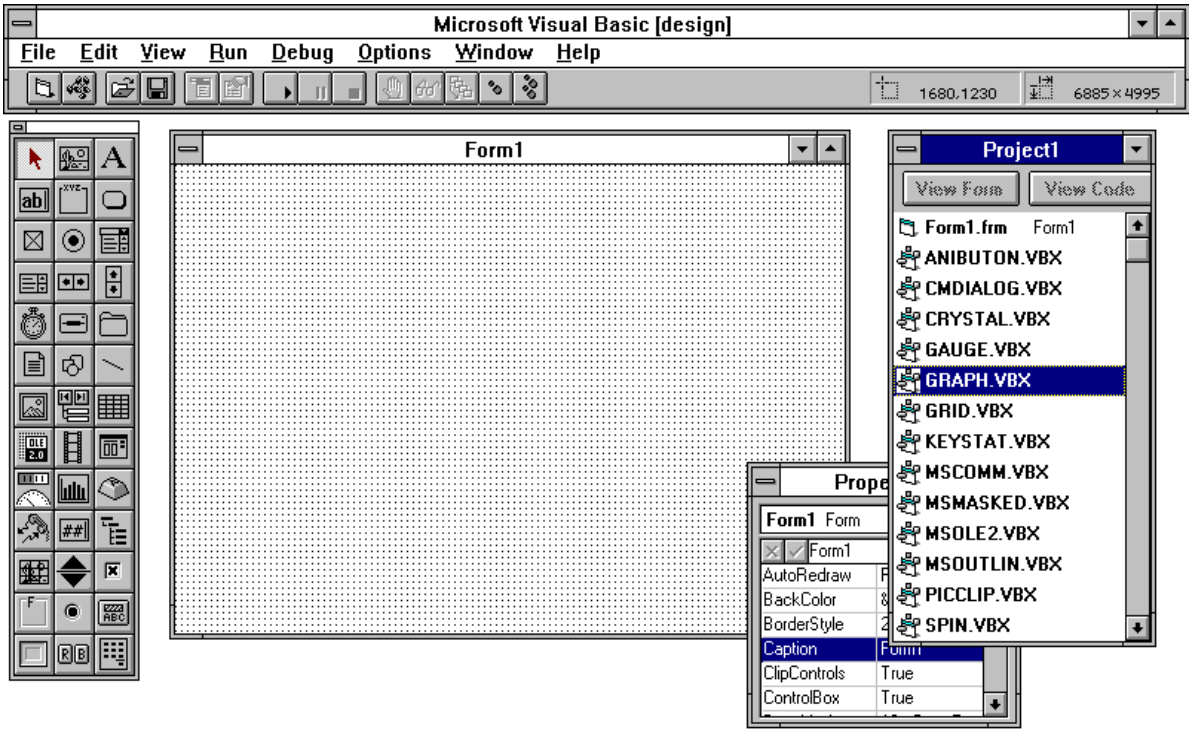
Apresentando Visual Basic

Objetivo do módulo

Apresentar os principais elementos do Visual Basic sem, no entanto, se aprofundar demais em nenhum tópico. Nos próximos capítulos, todos estes tópicos serão discutidos mais detalhadamente.

Formulário e as Janelas do Visual Basic



O **formulário** é o centro de uma aplicação gráfica. É com ele que o usuário interage de modo a executar suas tarefas. Nele, você define e posiciona os controles que apresentarão ao usuário as opções disponíveis

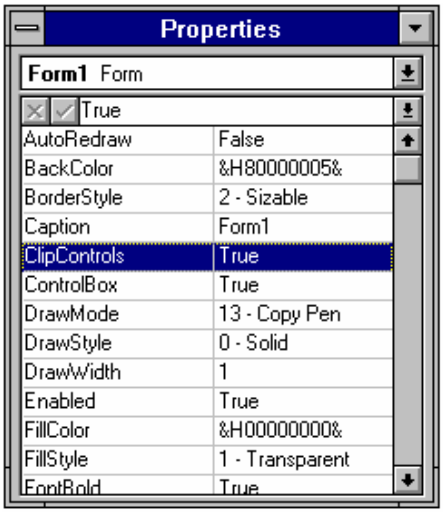
A **Janela de Projeto** (Project Window) é uma lista usada pelo Visual Basic para controlar que arquivos fazem parte da sua aplicação. Esta lista poderá ser composta por arquivos do tipo :

- ✓ .FRM (equivalente aos formulários da aplicação)
- ✓ .VBX (que representam controles adicionais)
- ✓ .BAS (que são blocos de código).

Caixa de Ferramentas (Toolbox)

A **Caixa de Ferramentas**, como o próprio nome sugere, é aonde você pega os elementos básicos que compoem qualquer aplicação desenvolvida em Visual Basic. Existem duas formas de colocar um controle em um formulário : Clicando duas vezes sobre o controle ou arrastando-o (drag) para dentro do formulário.

A Caixa de Propriedades



As propriedades definem as características de cada objeto/controle da aplicação. O conjunto de propriedades depende do controle selecionado. Por exemplo, um formulário tem propriedades diferentes de uma figura. As propriedades podem ser alteradas em tempo de construção ou de execução, sendo que algumas delas somente em tempo de execução.

Para alterar o valor de uma propriedade em tempo de construção, você deve:

1. Selecionar o controle cuja propriedade desejar alterar;
2. Rolar pela lista de propriedades até encontrar a propriedade desejada (apertando CTRL+SHIFT+<primeira letra do nome da propriedade> o VB se posiciona automaticamente na propriedade desejada .);
3. Digitar o novo valor;
4. Clicar o checkmark (√) ou pressionar ENTER para confirmar a alteração efetuada.

Barra de Ferramentas

A **Barra de Ferramentas** coloca à sua disposição os comandos e funções normalmente mais utilizados. Todos estes comandos e funções também se encontram nos menus do VB.

Os botões apresentados acima, da esquerda para a direita, executam as seguintes tarefas :

Barra de Ferramentas	Caminho por Menus	Tecla de acesso rápido
New Form	File New Form	n/a
New Module	File New Module	n/a
Open Project	File Open Project	n/a
Save Project	File Save Project	n/a
Menu Design Window	Window Menu Design	CTRL+M
Properties Window	Window Properties	F4
Start	Run Start	F5
Break	Run Break	CTRL+BREAK
End	Run End	n/a

Janela de Código

A janela de código é o lugar onde você escreve o código que a máquina deve executar para responder às ações do usuário. Para abrir uma janela de código, basta dar um click-duplo em cima do objeto do qual um evento deva ser tratado.

Exemplo - Codificando uma aplicação Visual Basic simples

➔ Para Inicializar o VB :

1. Dar duplo-click no ícone do Visual Basic.
   Visual Basic será inicializado com um formulário em branco na tela.
2. Redimensionar o formulário.
   Na Caixa de Propriedades, definir Height = 2700 e Width = 4065.

➔ Para projetar o formulário base :

1. Selecionar na caixa de propriedades, a propriedade Backcolor.
2. Clicar o botãoque possui os três pontinhos (...) e escolha um tom de verde.
3. Selecionar a propriedade Caption e digitar "Olá, Mundo !".
4. Selecionar a propriedade Name, e no TextBox digitar frmWorld.

➔ Para adicionar controles ao formulário :

1. Dar duplo-click no botão de comando da caixa de ferramentas.
2. Definir as seguintes propriedades :
Caption	Preencher
Name	cmdPreencher
Fontsize	14
3. Mover o botão "Preencher" para o canto superior direito do formulário.
4. Dar duplo-click no botão de comando da caixa de ferramentas.
5. Definir as seguintes propriedades :
Caption	Limpar
Name	cmdLimpar
Fontsize	14
6. Mover o botão "Limpar" para o canto inferior direito.
7. Dar duplo-click na ferramenta text box da caixa de ferramentas.
8. Definir as seguintes propriedades :
Name	txtBox1
Text	Apagar o texto existente nesta propriedade

➔ Para adicionar código aos controles :

1. Dar duplo-click sobre o botão "Preencher". (Uma janela de código será aberta.)

2. Adicionar o seguinte código :
   Txtbox1.text = "Olá, mundo !"
3. Na lista de objetos selecionar o botão "Limpar".
4. Escrever o seguinte código :
   Txtbox1.text = ""

➔ **Para salvar seu trabalho :**

1. No menu File, selecionar Save project.
2. Salvar o formulário como MUNDO.FRM e em seguida o projeto como MUNDO.MAK.

➔ **Para criar um arquivo executável :**

1. No menu File, selecionar Make Executable File.
2. Nomear o programa MUNDO.EXE.

➔ **Para executar a aplicação de dentro do Windows :**

1. Criar uma janela de grupo e, depois, um ícone de programa associado ao arquivo executável.
2. Dar duplo-click sobre o ícone de programa.

➔ **Para parar a execução da aplicação :**

1. Dar duplo-click sobre sua caixa de controle.

O Menu de comandos do Visual Basic

Menu File - Gerenciando Formulários e Projetos

Comando	Descrição
New Form	Adiciona um formulário novo ao projeto
Add File	Incorpora um arquivo existente ao projeto
Remove File	Destrói a ligação que um formulário tem com um projeto. Não apaga o formulário do disco.
Make EXE File	Cria o executável do projeto.
Print	A opção *code* imprime apenas o código de um formulário, enquanto a versão *Form* imprime tudo, inclusive a parte gráfica.
Save File As	Salva o formulário ou módulo com um nome diferente.
Save File	Salva o formulário com o mesmo nome.
Save Project	Salva o projeto inteiro.
Save Project As	Salva o projeto com um nome diferente.

Menu Edit - Editando o código Visual Basic

Comando	Descrição
Find	Procura um conjunto de caracteres pelo código.
Replace	Procura e substitui conjunto de caracteres por outro.
Cut, Copy e Paste	Efetuam corte, cópia e colagem de código.
Undo	Desfaz as mudanças.

Menu Run - Testando Aplicações

Comando	Descrição
Start	Inicia a execução da aplicação.
End	Finaliza a execução da aplicação.

Menu Window - Gerenciando as janelas do Visual Basic

Comando	Descrição
Properties	Mostra a janela de propriedades.
Toolbox	Mostra a caixa de ferramentas.
Project Window	Mostra a janela de projeto.
Color Palette	Mostra a paleta de cores.

Menu Help

Comando	Descrição
Contents	Mostra a tabela de conteúdo de Help.
Search	Procura por um tópico específico.
Product Support	Localiza informações do suporte da Microsoft.

Construindo aplicações Visual Basic

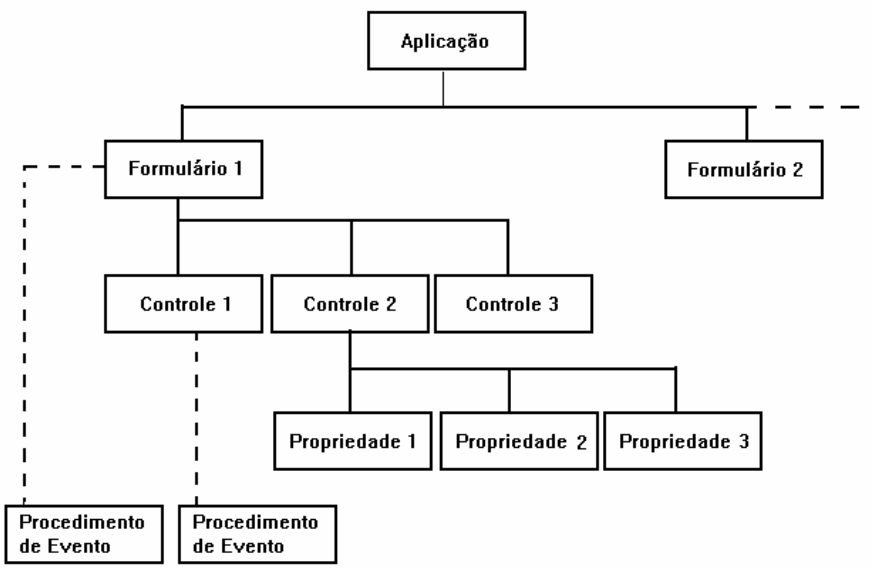
Objetivo do módulo

Apresentar vários conceitos interrelacionados que o ajudarão na transição do mundo procedural para o mundo orientado a eventos. Este módulo será a base lógica utilizada no resto do curso. Aqui, cria-se o contraste entre a programação voltada para ambiente Windows e aquela para DOS.

Aplicações Procedurais *versus* Aplicações Orientadas a Eventos

Procedural	Orientada a Eventos
Programação Linear	Eventos podem ser acionados pelo usuário ou pelo sistema.
Baseada em caracteres	Baseada em objetos gráficos
Mono-tarefa	Multi-tarefa
Programador tem controle do ambiente	Windows tem controle do ambiente

A Terminologia Visual Basic



Objetos - São ferramentas que o Visual Basic fornece com as quais você construirá aplicações. Um formulário é um tipo de objeto; controles dentro do formulário, como: botões, caixas de texto e figuras também são objetos. Cada objeto possui uma lista de **propriedades**; alterando-as você estará caracterizando, criando a identidade do seu objeto. A objetos você pode aplicar **métodos:** mostrar um formulário é o método mostrar aplicado a um formulário, adicionar um item a uma lista é o método adicionar aplicado a uma lista, etc.. A objetos também acontecem **eventos**. Eventos são percebidos pelo sistema, e você pode programar a sua aplicação para que ela reaja a estes eventos.

Vimos portanto vários conceitos importantes :

- Objetos - estrutura básica do VisualBasic. Podem ser : controles, formulários, impressora, etc.
- Propriedades - são as características que personalizam seu objeto. Cada objeto tem uma lista de propriedades própria.
- Métodos - Procedimentos que podem ser aplicados aos objetos. Cada objeto possui uma lista de métodos própria.
- Eventos - Acontecem aos objetos e são detectados pelo Windows. Podemos programá-los para que os objetos reajam aos eventos.

Passos para a criação de uma aplicação em Visual Basic

1. Abra um novo projeto.
2. Crie um formulário para cada janela que você conseguir visualizar dentro da aplicação.
3. Desenhe os controles nos formulários.
4. Crie uma barra de menus.
5. Defina as propriedades do formulário e dos controles.
6. Codifique os *event procedures* e *general procedures*.
7. Crie um arquivo executável.

Para distribuir o executável da aplicação

Para instalar o executável para os usuários ou clientes, você precisa levar também uma cópia da biblioteca do Visual Basic VBRUN300.DLL. Este arquivo faz parte dos arquivos de instalação do Visual Basic e pode ser distribuído livremente.

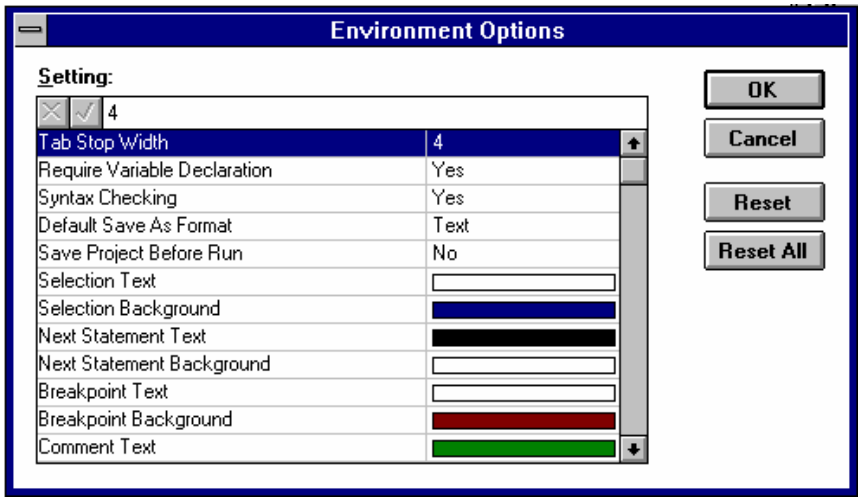
Projeto de interface com usuário

A interface com o usuário deve seguir um padrão. A interface atualmente adotada mundialmente para desenvolvimento de aplicações Windows, é a própria interface criada pela Microsoft.

Determinações Básicas :

- Projetar a interface para o usuário, não para o sistema. - Típico caso de criar telas diferentes para Inclusão, Alteração e Exclusão (quando, na maioria dos casos, estas três funções poderiam estar agrupadas em uma única tela).
- Ter em mente que agora é o usuário que mantém o controle da aplicação, e não mais o programa.
- Clareza - Ter certeza de que o propósito de cada tela está bem claro para o usuário.
- Estética
- *Feedback* - Dar sempre um retorno ao usuário do que está acontecendo no momento.
- Usar cores como um chamariz de atenção, porém cuidado para não exagerar !
- Preferir cores complementares, ao invés de cores inversas.
- A letra, normalmente utilizada em aplicações Windows, é qualquer uma *Sans Serif.* Ao usar tipos de letras diferentes, cuidado par não abusar da variedade.

Configuração do Ambiente



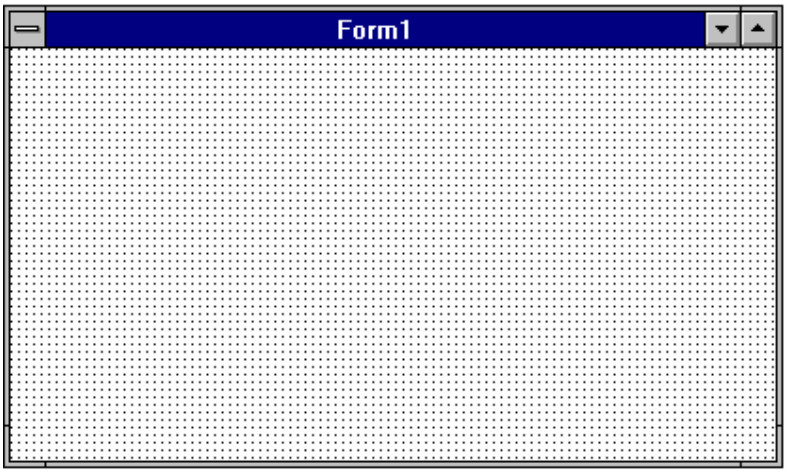
Para configurar o seu ambiente de trabalho, selecionar a opção Environment do menu Options. Nesta opção você pode alterar padrões, como por exemplo, cores. Dentre os ítens que podem ser alterados, as seguintes mudanças são sugeridas :

- Require Variable Declaration - Yes
- Default Save Format - Text
- Grid Width - 120 twips
- Grid Height - 120 twips

Trabalhando com Formulários

Objetivo do módulo

Apresentar a ferramenta principal para o programador : o Formulário. Apesar de ser apresentado de uma maneira simples, os conceitos e termos mostrados aqui serão necessários para um entendimento completo deste ambiente de programação.



rincipais propriedades

Propriedade	Default	Definição / Comentários
BorderStyle	2 - Sizeable	No padrão de interface Windows, o tipo da borda segue o seguinte padrão : • Fixed Single - para caixas de diálogo não modais. • Sizeable - borda padrão para qualquer tela que não seja uma caixa de diálogo. • Fixed Double - para caixas de diálogo modais.
Caption	Form1	Título da tela.
ControlBox	True	False inibe a aparição do controlbox na barra de título do programa.
FontSize	8.25	Tamanho da Letra.
FontName	Helv	Tipo da letra utilizada.
Name	Form1	Nome pelo qual o formulário será conhecido dentro da aplicação.
Height	4425 twips	Altura do formulário.
Icon		Ícone default para o formulário. Se este for o *Startup Form*, será o ícone default da aplicação.
Left		Distância da margem esquerda da tela.
MaxButton	True	False inibe a aparição do botão de maximizar na barra de título do programa.
MinButton	True	False inibe a aparição do botão de minimizar na barra de título do programa.
MousePointer	0 Default	Altera o formato do mouse. Quando algo estiver sendo executado, o cursor deve assumir o formato de ampulheta (11).
Top		Distância da margem superior da tela.
Visible	True	False torna o formulário invisível.
Width	7485 twips	Largura do formulário.

Comandos Relacionados a Formulários

Load

O comando LOAD é utilizado para carregar um formulário ou controle em memória. Normalmente aparece dentro de *event procedures* de outros formulários ou controles.

Sintaxe :

Load *objeto*

Importante O comando Load não mostra automaticamente o formulário; ele apenas carrega em memória. Para tornar o formulário visível, é necessário chamar o método Show.

Unload

O comando Unload é utilizado para retirar um formulário de memória. Note que ao retirar um formulário de memória apenas a parte gráfica da tela é descarregada, o código continua em memória.

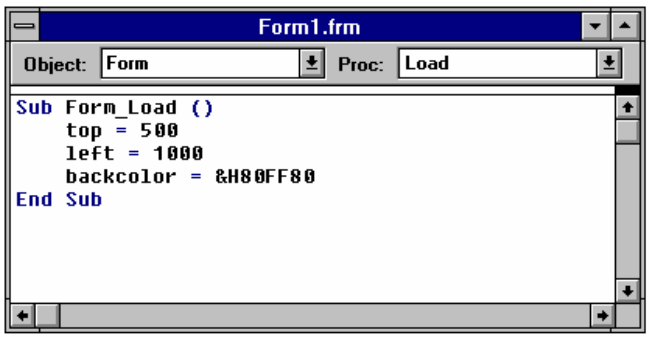
Sintaxe :

Unload *objeto*

Principais Eventos

Load

Acontece sempre antes que um formulário seja carregado em memória. É utilizado normalmente para inicializar os controles do formulário.



Unload

Este evento ocorre sempre momentos antes de um formulário ser descarregado. Nele devem ser tratados procedimentos de finalização de banco de dados, cálculos e etc.

Principais Métodos

Método Hide

O método Hide esconde um formulário. O formulário continua carregado em memória porém não mais visível. O metódo Hide atribui o valor **False** à propriedade Visible do formulário. Se você tentar esconder um formulário que não esteja carregado em memória, este é carregado porém fica escondido.

Sintaxe :

[formulário].**Hide**

Método Show

O método Show é utilizado para mostrar um formulário. Assim como o método Hide, o Show trabalha com a propriedade Visible do formulário, atribuindo-lhe o valor **True**. Caso o formulário não esteja carregado em memória, o método show executa automaticamente o comando Load.

Sintaxe :

[formulário].**Show** [estilo%]

Observação : O parâmetro *estilo%* é um valor inteiro que determina se um formulário será mostrado de forma modal ou não. Estes conceitos serão explicados mais adiante.

Exemplo - Abrindo e Fechando formulários

Para exemplificar a abertura e fechamento de formulários, construiremos uma aplicação composta por dois formulários. Ao clicar em qualquer lugar do formulário, o outro formulário será mostrado.

1. Inicializar o Visual Basic.

Criando o segundo formulário

2. Clicar com o mouse sobre o menu *File*. Selecionar *New Form*.

Programando o click do primeiro formulário

3. Clicar duas vezes sobre o primeiro formulário. Uma janela de código aparecerá.
4. Na lista de eventos, selecionar o evento *Click*.
5. Escrever as seguintes linhas de código :

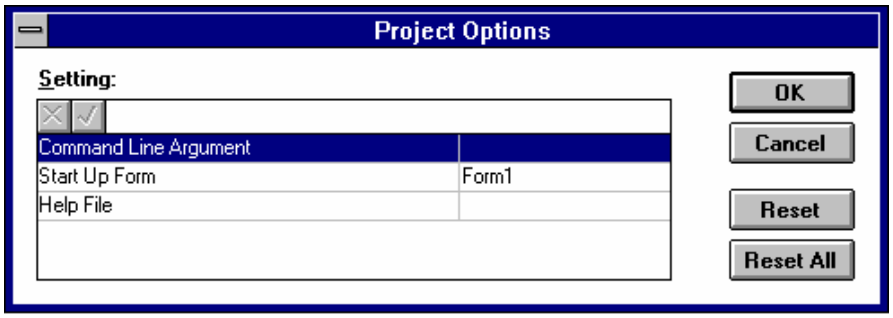
```
Form2.show
```

Programando o click do segundo formulário

6. Clicar duas vezes sobre o segundo formulário. Uma janela de código aparecerá.
7. Na lista de eventos, selecionar o evento *Click*.
8. Escrever as seguintes linhas de código :

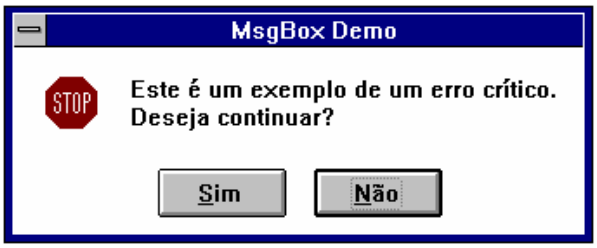
```
Form2.hide
```

Determinando o Formulário Inicial de uma Aplicação



Normalmente uma aplicação contém mais de um formulário. Se este for seu caso, você precisa definir o formulário inicial da aplicação (que por default é aquele que foi criado primeiro). Para fazer isto selecionar a opção *Project* do menu *Options.* O text box *Start Up Form* deve conter o nome (propriedade *Name*) do formulário inicial.

Message Box



Existem ocasiões em que o sistema deve enviar alguma mensagem para o usuário. Na maioria destas ocasiões, criar um formulário especialmente para isso, seria demorado demais. Para solucionar tais casos, o Visual Basic oferece uma ferramenta chamada Message Box.

Em uma Message Box, você pode definir a mensagem, o título e as respostas (botões) possíveis. Como retorno, a Message Boxdiz qual o botão utilizado.

```
Msg = "Este é um exemplo de um erro crítico." + Chr(13) + Chr(10)
Msg = Msg & " Deseja continuar?"
DgDef = MB_YESNO + MB_ICONSTOP + MB_DEFBUTTON2' Define botões utilizados.
Resp = MsgBox(Msg, DgDef,"MsgBox Demo")'Recebe resposta do usuário.
```

obs.: MB_YESNO, MB_ICONSTOP, MB_DEFBUTTON2 são constantes numéricas definidas pelo programador que indicam a quantidade de botões, o ícone mostrado e qual o botão default.

Caixas de Diálogo Modais e Amodais

Formulários ou caixas de diálogo podem ser modais ou amodais. Modais são aquelas que impedem o usuário de sair daquela caixa de diálogo enquanto ele não clicar sobre um dos botões da caixa. Caixas de diálogo amodais, permitem que o usuário saia dela a qualquer momento.

Para definir o tipo da caixa de diálogo, basta você passar um parâmetro no método Show do formulário.

Tipo	Parâmetro
Modal	1
Amodal	0

ex.: `Form1.show 1 'carrega um formulário modal`

Criando Menus

Objetivo do módulo

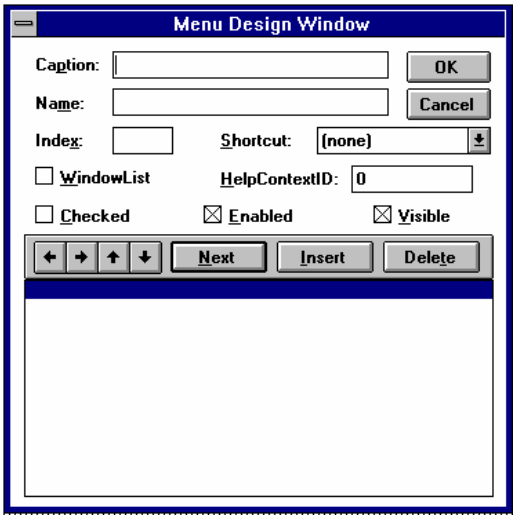
Mostrar como trabalhar com aplicação composta por mais de um formulário, montando menus. Nos módulos anteriores, trabalhamos com um único formulário, basicamente, para mostrar o funcionamento de uma aplicação Visual Basic.

Padrão de Menus Windows

A seguir destacamos alguns ítens que devem ser observados para que um menu esteja dentro do padrão Windows.

- A opção *Sair* deve ser o última opção do primeiro menu;
- Três pontos devem ser colocados ao final do comando para indicar a existência de uma caixa de diálogo, quando esta existir;
- Para indicar que uma opção está ativa, colocar um check-mark (√) ao lado da opção.
- Barras de separação devem ser utilizadas para separar, visualmente, ítens relacionados ou opção perigosas para o usuário.
- Teclas de acesso ou atalho devem ser definidas para todos os ítens do menu. (Pelo padrão Windows, as aplicações devem possibilitar seu uso, mesmo quando o usuário não tem um mouse).

A Implementação do Visual Basic para a Criação de Menus



Para criar um menu para uma aplicação, você deverá se posicionar no formulário ao qual o menu deverá estar vinculado e selecionar a opção *Menu Design* do menu *Window* ou então, clicar sobre o botão de *Menu Design* sobre a barra de ferramentas.

A janela de projeto de menus

A janela de projetos de menu está subdividida em duas partes : Ítens de menu e manipulação de layout, que serão apresentadas a seguir.

Ítens de Menu

Item	Descrição
Caption	Nome que aparece no menu.
Name	Nome usado em código.
Index	Usado na adição de menus dinamicamente.
ShortCut	Cria combinações de tecla para acesso rápido.
Checked	Indica a existência de uma check-mark ao lado do item.
Enabled	Indica se uma opção de menu está habilitada.
HelpContextId	Especifica um identificador para um item de menu em um sistema de Help.
Visible	Indica se uma opção de menu está visível ou não.
WindowList	Especifica se o menu conterá uma lista de formulários MDI abertos.

Na parte de manipulação de ítens, o você define a hierarquia dos ítens de menu, inclui novos ítens e apaga outros. Para qualquer uma das operações, o item deve estar selecionado.

Manipulação de Lay-Out

Item	Função
Seta para esquerda	Aumenta o nível hierárquico de um menu.
Seta para a direita	Diminui o nível hierárquico de um menu.
Seta para Cima	Move a posição do cursor para um item de menu acima.
Seta para Baixo	Move a posição do cursor para um item de menu abaixo.
Next	Move a seleção para o próximo item da lista.
Insert	Insere uma linha em branco acima do item onde você está posicionado.
Delete	Apaga um item de menu.

Exemplo - Construindo Menus

Para exemplificar a contrução de menus, contruíremos uma tela simples, onde apenas as funções de construção serão enfocadas.

1. Abrir o Visual Basic.
2. Do menu Window, selecionar a opção Menu Design.
3. Criar os seguintes ítens de menu :

Item de Menu	Caption	Name	Identação
Arquivo	&Arquivo	MNUArquivo	
Abrir	&Abrir	MNUAbrir	→
-	-	MNUSeparador	→
Sair	&Sair	MNUSair	→
Editar	&Editar	MNUEditar	
Copiar	&Copiar	MNUCopiar	→
Colar	Co&lar	MNUColar	→

4. Pressione F5 para executar o programa e verifique o encadeamento dos menus. Note que, para que os menus executem alguma ação é necessário que o evento Click de cada objeto menu seja codificado. No caso acima, deveríamos codificar o evento click dos seguintes objetos : MNUAbrir, MNUSair, MNUCopiar e MNUColar.

Utilizando Controles

Objetivo do módulo

Apresentar os controles, suas principais propriedades e eventos. Não é intenção do curso, que ao final deste módulo, você tenha domínio absoluto de todos os controles e propriedades existentes no Visual Basic.

Ao longo do módulo, uma série de pequenos exercícios e apresentações serão feitas de modo que você se familiarize com o uso de cada controle, e perceba como as propriedades os afetam.

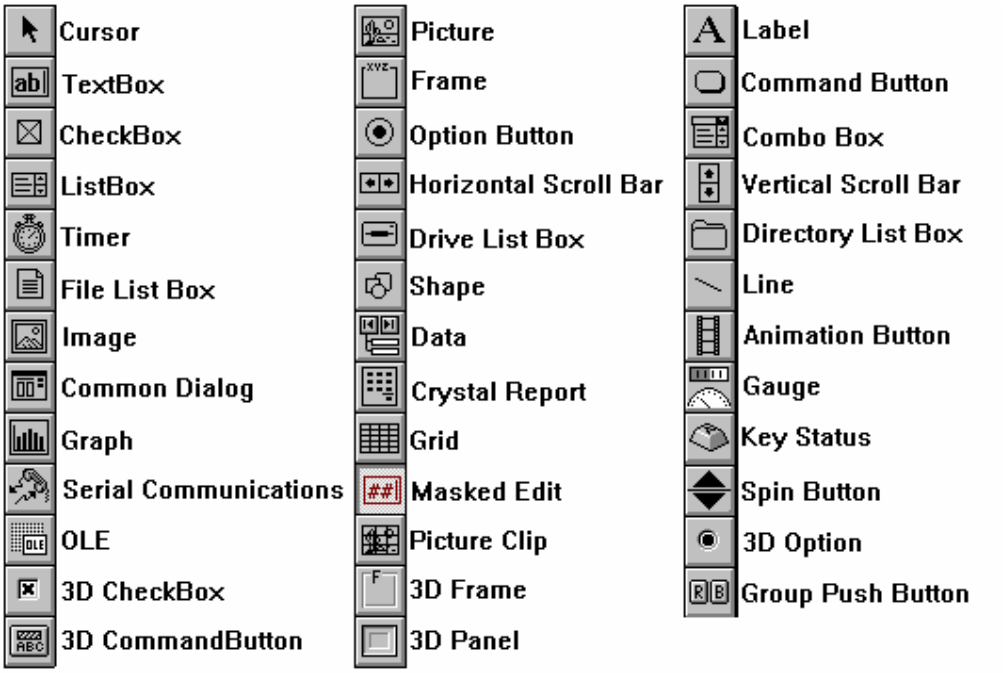
Os controles apresentados serão :

- Labels
- Text boxes
- Frames
- Command Buttons
- Check Boxes

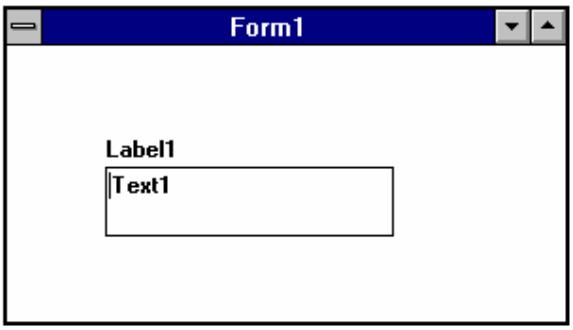
- Option Buttons
- Combo Boxes
- List Boxes
- Horizontal/Vertical Scroll Bars
- Timers
- Picture Boxes
- Grid
- Panel 3d
- Group Push Buttons
- File Browsers

Tipos de Controles

Caixa de Ferramentas



Labels e Text Boxes



Labels

Labels são normalmente utilizados para exibir textos que o usuário não deve modificar. Um exemplo clássico seria o título de campos em formulários.

Propriedade	Descrição
Alignment	Alinhamento do texto dentro do label.
Backcolor	Cor de fundo do label.
BorderStyle	Tipo de borda do label. Recomendável não utilizar borda.
Caption	Texto que aparece no label.
FontName	Tipo de letra utilizada.
FontSize	Tamanho da letra utilizada.
ForeColor	Cor da letra.
Height	Altura do label.
Left	Posição do extremo esquerdo do label em relação ao formulário.
Name	Nome do controle a ser utilizado no programa.
Tag	Propriedade sem finalidade definida.
Top	Posição do topo do label em relação ao formulário.

Text Boxes

- text box é um controle utilizado, basicamente, para a exibição de informação pelo sistema ou para a entrada de dados do usuário.

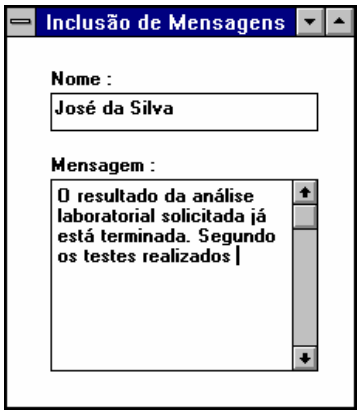
Propriedade	Descrição
Alignment	Alinhamento do texto dentro do text box.
FontName	Tipo de letra utilizada.
Height	Altura do text box.
Left	Indica margem esquerda do text box em relação ao formulário.
MaxLegth	Tamanho máximo do texto.
Multiline	Indica se um text box tem mais de uma linha.
Name	Nome do controle a ser utilizado pelo programa.
PasswordChar	Caracter utilizado para esconder palavras secretas. Funciona apenas quando a propriedades Multiline tem valor *False*. Ex.: senhas.
ScrollBars	Indica a existência de barras de rolagem.
Text	Texto do text box.
Top	Indica topo do text box em relação ao formulário.
Width	Largura da text box.

EVENTOS

Change Este evento acontece toda vez que o usuário altera o conteúdo de uma Text Box. Por exemplo, quando o usuário escreve a palavra "papel", o evento Change ocorre cinco (5) vezes.

Exemplo - Utilizando Labels e Text Boxes

Suponha que você tivesse que montar a seguinte tela para o envio de mensagens :



1. Inicialize o Visual Basic.
2. Com o formulário selecionado, pressione F4 para ter acesso à lista de propriedades.
3. Na propriedade *Caption*, escreva o título do formulário "Inclusão de Mensagens".

Criando os Label Nome e Mensagem

4. Clique duas vezes sobre a ferramenta Label da Caixa de Ferramentas.
5. Com o label selecionado, pressione F4 para ter acesso à lista de propriedades.
6. Na propriedade *Caption*, escreva o título do label "Nome".
7. Posicione o label de acordo com o desenho da tela acima.
8. Repita os passos de 4 até 7 para criar o label "Mensagem".

Criando o Text Box Nome

9. Clique duas vezes sobre a ferramenta Text Box da Caixa de Ferramentas.
10. Com o Text Box selecionado, pressione F4 para ter acesso à lista de propriedades.
11. Na propriedade *Text*, apagar o que estiver escrito.
11. Na propriedade *Name*, escrever TXBnome.
12. Posicione o textbox de acordo com o desenho da tela acima.

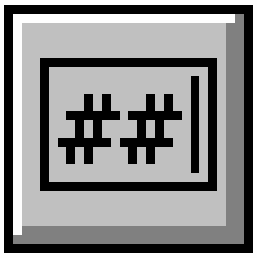
Criando o Text Box Mensagem

9. Clique duas vezes sobre a ferramenta Text Box da Caixa de Ferramentas.
10. Com o Text Box selecionado, pressione F4 para ter acesso à lista de propriedades.
11. Na propriedade *Text*, apagar o que estiver escrito.
11. Na propriedade *Name*, escrever TXBmensagem.
12. Na propriedade *ScrollBars*, selecionar *2-Vertical*.
13. Na propriedade *Multiline*, escrever *True*, sem esta propriedade o scroll bar vertical não funcionará.
14. Posicione o Text Box de acordo com o desenho da tela acima.

Executando a Aplicação

10. Pressione F5 ou selecione Start do menu Run.
11. Experimente dar tab entre controles; selecionar, cortar e colar parte do texto, etc.

Masked Edit



É um controle que se assemelha a um Text box. No entanto, ele permite restringir a entrada de dados do usuário e também formatar a saída dos dados.

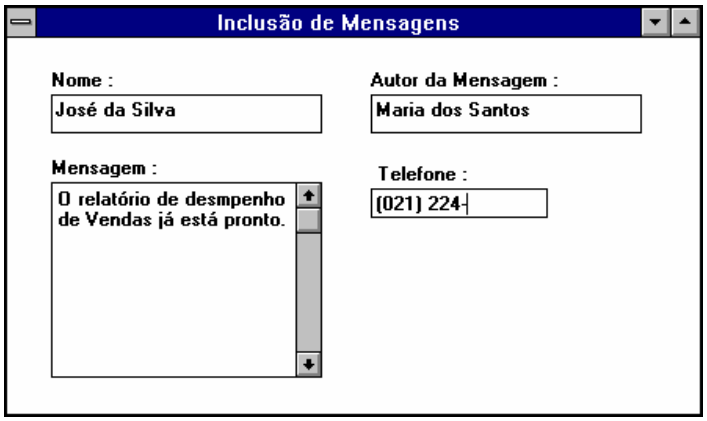
Propriedade	Descrição
ClipMode	Determina se os caracteres de máscara são incluídos ou não quando operações de Cut/Copy são executadas.
ClipText	Conteúdo do controle, sem os caractéres de máscara.
Format	Formato que números, datas, hora e strings serão mostrados na Masked Edit.
HideSelection	Indica se uma seleção em uma Masked Edit fica aperente quando o controle perde o foco.
Mask	Determina a máscara de entrada de dados.
MaxLength	Tamanho máximo do campo.
PromptChar	Caracter utilizado para pedir input ao usuário.
PromptInclude	Indica se os caracteres *PromptChar* serão incluídos na propriedade *Text*.
Text	Conteúdo do controle, com os caractéres de máscara.

Caracteres de Máscara	Descrição
.	Separador de casas decimais.
,	Separador de milhar.
:	Separador de hora.
/	Separador de data.
&	Reserva lugar para caracter.
?	Reserva lugar para letras.
A	Reserva lugar para alfanumérico.
#	Reserva lugar para dígito.

Obs.: O separador de casas decimais e o separador de milhar se regulam pelo definido na configuração do Windows. Desta forma, se a configuração do Windows for formato brasileiro ("**,**" para casas decimais e "**.**" para milhares), apesar de você colocar "**.**"para reservar o lugar do separador decimal, o que aparecerá na tela em tempo de execução será "**,**".

Exemplo - Utilizando Masked Edits

Continuando o exemplo mostrado anteriormente, queremos agora colocar o nome do autor da mensagem e seu telefone de contato. O campo de autor da mensagem só deverá conter letras e o telefone deverá ser colocado no formato (999) 999-9999. Para fazer isto usa-se *Masked Edits*.



1. Inicialize o Visual Basic.
2. Como demonstrado anteriormente, crie os labels de "Autor da Mensagem" e "Telefone".

Criando a Masked Edit Nome do Autor

3. Clique duas vezes sobre a ferramenta Masked Edit da Caixa de Ferramentas.
4. Com o Masked Edit selecionado pressione F4 para ter acesso à lista de propriedades.
5. Nomeie este Masked Edit "MEBNomeAutor".
6. Na propriedade *Mask* colocar ??. Repare o que acontece com a propriedade *MaxLength*.

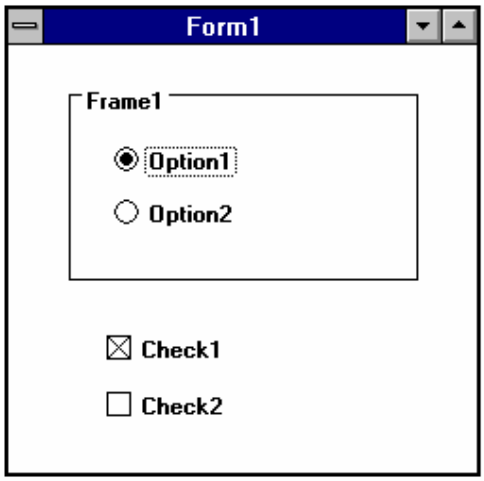
Criando a Masked Edit Telefone

3. Clique duas vezes sobre a ferramenta Masked Edit da Caixa de Ferramentas.
4. Com o Masked Edit selecionado pressione F4 para ter acesso à lista de propriedades.
5. Nomeie este Masked Edit "MEBTelefone".
6. Na propriedade *Mask* colocar (###) ###-####..

Executando a Aplicação

7. Pressione F5 ou selecione Start do menu Run.
8. Experimente digitar números no nome, letras no telefone e etc.

Frames, Option Buttons e Check Boxes Normais e Tridimensionais



Frames, check boxes e option Buttons possibilitam mostrar ao usuário conjuntos de opções entre as quais ele pode escolher. O arranjo na tela dos frames e option buttons definem grupos independentes de opções. A escolha em um grupo não afeta a escolha em outro.

Frames

Frames permitem agrupar, gráfica e funcionalmente, um grupo de controles. Um exemplo clássico do uso de Frames é o agrupamento de Option Buttons mutuamente exclusivos.

Propriedades	Descrição
Caption	O título do frame.
Name	Nome do controle usado no código.
Visible	Indica se um frame e seus controles estão visíveis ou não.

Observação : Para colocar controles dentro de um frame, você tem que criar primeiro o frame; selecioná-lo e, então, criar os controles dentro dele. Ou então, caso os controles tenham sido criados anteriormente, deve-se recortá-los, selecionar o frame e, dentro deste, colá-los (Cut e Paste).

Propriedades Adicionais de Frames Tridimensionais

Propriedade	Descrição
Alignment	Alinhamento do título do frame.
Font 3D	Tipo de letra tridimensional.
ShadowColor	Cor da sombra.
ShadowStyle	Tipo da sombra.

Check Boxes

Check Boxes são usados, normalmente, para permitir uma ou mais escolhas independentes, não exclusivas, em relação a outras já efetuadas.

Propriedades	**Descrição**
Caption	Título da opção na tela.
Enabled	Habilita e desabilita acesso do usuário.
Name	Nome do controle usado internamente.
Value	Indica se um check box está marcado ou não.
Visible	Mostra ou esconde o controle.

EVENTOS

Click O evento click indica que o usuário fez uma seleção. Ao ocorrer este evento , o Check Box é marcado/desmarcado conforme o seu valor anterior.

Option Buttons

Option Buttons são normalmente utilizados para criar conjuntos de escolhas mutuamente exclusivas dentro de um contexto, que pode ser a tela, um frame ou um frame dentro de outro.

Propriedades	**Descrição**
Caption	Valor da opção na tela.
Name	Nome interno do controle.
Enabled	Define se o controle está ativo ou não.
Value	Indica se um option button está marcado ou não.
Visible	Define se o controle está aparente ou não.

EVENTOS

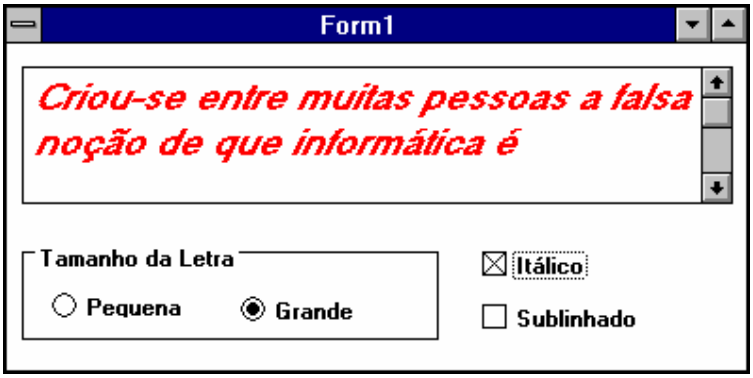
Click Quando o usuário clica sobre o option button, uma série de acontecimentos ocorrem : o option button selecionado é marcado, os outros Option Buttons do contexto são desmarcados e a propriedade *value,* para cada option button do contexto é atualizada com a nova seleção.

Propriedades Adicionais de Check Boxes e Option Buttons Tridimensionais

Propriedade	Descrição
Alignment	Posicionamento do título, em relação ao quadradinho.
Font 3D	Tipo de letra tridimensional.

Exemplo - Utilização de Frames, Check Boxes e Option Buttons

Suponha que você queira criar uma aplicação onde um texto escrito em um Text Box possa ser formatado por você :



1. Inicialize o Visual Basic.
2. Como demonstrado anteriormente, crie o Text Box.

Criando os Option Buttons

3. Clique duas vezes sobre a ferramenta Frame da Caixa de Ferramentas.
4. Com o Frame selecionado pressione F4 para ter acesso à lista de propriedades.
5. Nomeie este Frame "FRATamanho".
6. Com o frame selecionado, clique duas vezes sobre a ferramenta Option Button da Caixa de Ferramentas.
7. Com o Option Button selecionado pressione F4 para ter acesso à lista de propriedades.
8. Nomeie este Option Button "OPTPequena".
9. Atribua à propriedade *Caption* o valor "Pequeno"
10. Repita os passos 6 e 7 e nomeie o segundo botão "OPTGrande".
11. Atribua à propriedade *Caption* o valor "Grande"

Codificando os Option Buttons

12. Clique duas vezes sobre OPTPequena, uma janela de código deve aparecer.
13. Escreva os seguintes comandos :

```
Text1.FontSize = 10
```

14. Clique duas vezes sobre OPTGrande, uma janela de código deve aparecer.
15. Escreva os seguintes comandos :

```
Text1.FontSize = 14
```

Criando os Check Boxes

16. Clique duas vezes sobre a ferramenta Check Box da Caixa de Ferramentas.
17. Com o Check Box selecionado pressione F4 para ter acesso à lista de propriedades.
18. Nomeie este Check Box "CKBItalico".
19. Atribua à propriedade *Caption* o valor "Itálico"
20. Repita os passos 16, 17 e 18 e nomeie o segundo Check Box "CKBSublinhado".
21. Atribua à propriedade *Caption* o valor "Sublinhado"

Codificando os Check Boxes

22. Clique duas vezes sobre CKBItalico, uma janela de código deve aparecer.
23. Escreva os seguintes comandos :

```
If CKBItalico.Value = 1 Then
   Text1.FontItalic = True
Else
   Text1.FontItalic = False
End If
```

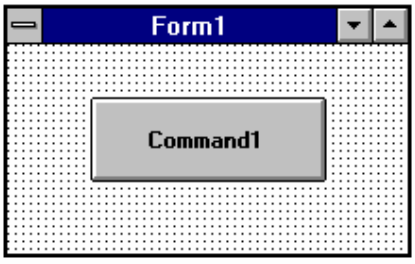
24. Clique duas vezes sobre CKBUnderline, uma janela de código deve aparecer.
25. Escreva os seguintes comandos :

```
If CKBSublinhado.Value = 1 Then
   Text1.FontUnderline = True
Else
   Text1.FontUnderline = False
End If
```

Executando a Aplicação

26. Pressione F5 ou selecione Start do menu Run.
27. Experimente apertar os botões e os Check boxes para ver o que acontece.

Command Buttons Normais e Tridimensionais



Propriedade	Descrição
Cancel	Ativa o botão quando o *Esc* for pressinado.
Caption	Valor do botão que aparece na tela.
Default	Ativa botão quando o *Enter* for pressionado.
Enabled	Permite, ou não, o acesso ao botão.
Height	Altura do botão.
Left	Posição da borda esquerda em relação ao formulário.
Name	Nome do controle usado internamente.
Top	Posição da borda superior em relação ao formulário.
Width	Largura do botão.
Value	Variável indica se o botão pode ser acessado ou não. Equivalente à propriedade *Enabled*.

EVENTOS

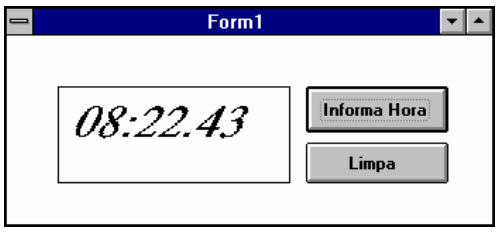
Click Normalmente, significa que o usuário deseja que alguma ação seja tomada. Neste caso, você deverá escrever o código correspondente.

Propriedades Adicionais de Command Buttons Tridimensionais

Propriedade	Descrição
Bevel Width	Especifica a largura da sombra ao redor do botão.
Name	Nome utilizado internamente.
Outline	Habilita e desabilita uma borda (linha) ao redor do botão.
Picture	Permite a colocação de uma figura ou ícone dentro do botão.

Exemplo - Utilização de Command Buttons

Suponha que você queira criar uma aplicação onde a hora será informada sempre que você desejar:



1. Inicialize o Visual Basic.
2. Como demonstrado anteriormente, crie o Text Box.
3. Nomeie o Text Box de TXBHora.
4. Com TXBHora selecionado, altere as propriedades FontSize, FontItalic e FontName.

Criando os Command Buttons

5. Clique duas vezes sobre a ferramenta Command Button da Caixa de Ferramentas.
6. Com o Command Button selecionado pressione F4 para ter acesso à lista de propriedades.
7. Nomeie este Command Button "CMDInforma".
8. Atribua o valor "Informa Hora" à propriedade *Caption*.
9. Repetir os passos 5 e 6 e nomeie este Command Button "CMDLimpa".
10. Atribua o valor "Limpa" à propriedade *Caption*.

Codificando os Command Buttons

11. Clicar duas vezes sobre CMDInforma, uma janela de código irá aparecer.
12. Digitar a seguinte linha de código:

```
TXBHora.Text = Format(Now, "hh:mm.ss")
```

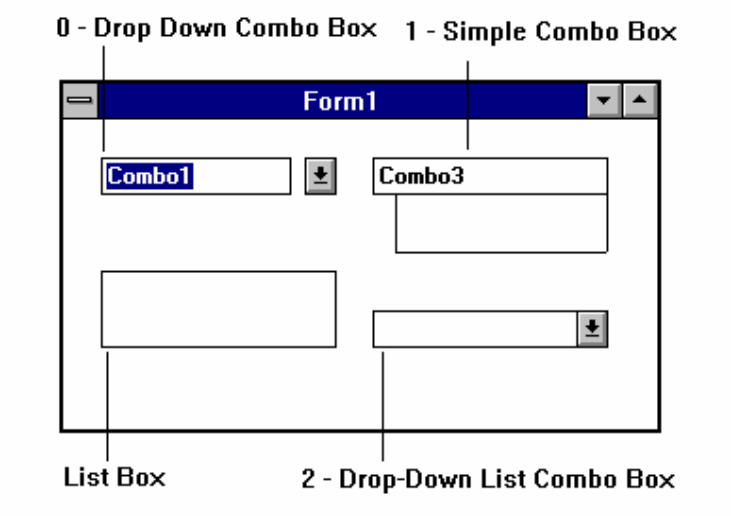
13. Clicar duas vezes sobre CMDLimpa. Uma janela de código irá aparecer.
14. Digitar a seguinte linha de código:

```
TXBHora.Text = ""
```

Executando a Aplicação

13. Pressione F5 ou selecione Start do menu Run.
14. Experimente apertar os botões.

Combo Boxes e List Boxes



Existem quatro tipos de controles de listas, cada um com uma função ligeiramente diferente. Ambas as listas que caem (Drop Down Combo Box e Drop Down List Combo Box) foram projetadas para economizar o espaço de tela utilizado pelo controle. Usuários somente podem adicionar ítens a uma lista do tipo *Drop-Down Combo Box* ou *Simple Combo Box*, pois os outros tipos permitem apenas escolha entre os ítens existentes.

Todas os List Boxes e combo boxes apresentarão automaticamente uma barra de rolagem, caso o tamanho da lista ou combo ultrapasse a sua própria altura (definida pela propriedade *Height*).

Nas páginas seguintes, serão apresentados Combo Boxes e List Boxes mais detalhadamente.

List Boxes

List Boxes mostram um conjunto de ítens entre os quais o usuário pode escolher um ou mais. Através de código, você pode adicionar ou remover ítens do list box.

Propriedade	Descrição
Columns	Permite a exibição de múltiplas colunas em uma única lista.
List	Vetor que contém a lista dos ítens da ListBox.
ListCount	Número de ítens da lista.
ListIndex	Índice do item ,da lista, selecionado.
MultiSelect	0 - Permite selecionar apenas um item por vez. 1 - Permite selecionar mais de um item da lista. 2 - Permite seleção do tipo File Manager do Windows, usando teclas CTRL e SHIFT.
Name	Nome interno do controle.
Sorted	Indica se a lista estará em ordem alfabética ou não.
Text	Retorna o item selecionado da lista.

O vetor boleano SELECTED() indica quais os ítens selecionados da lista, caso você tenha escolhido uma lista de seleção múltipla.

EVENTOS

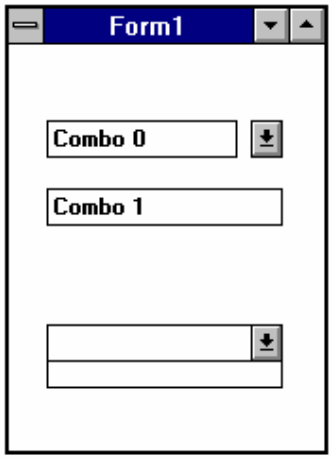
Click Seleciona um item da lista.

Double-Click Combina dois eventos : a seleção do item da lista e a inicialização de um processo associado àquele item.

MÉTODOS

Método	Descrição	Sintaxe
AddItem	Adiciona um item à lista. Normalmente executado no Form Load.	list1.additem "João da Silva"
RemoveItem	Remove um item específico da lista.	list1.removeitem (ListCount)
Clear	Remove todos os ítens da lista.	List1.Clear

Combo Boxes



Propriedade	Descrição
Name	Nome do controle internamente.
Style	Tipo da combo : 0 - Drop-Down Combo 1 - Simple Combo 2 - Drop-Down List
Text	Texto selecionado.
Height	Altura do Combo

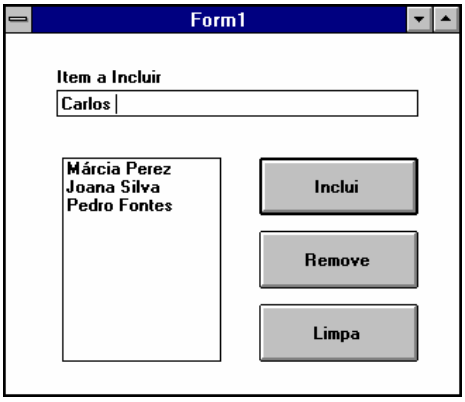
EVENTOS

Change O evento Change indica que o conteúdo do controle foi alterado. No caso de combo boxes, o evento change ocorre toda vez que, a parte editavel do combo á alterada.

Obs.: Os métodos *AddItem*, *RemoveItem* e *Clear*, demonstrados no item anterior sobre listas, também são válidos para Combo Boxes.

Exemplo - Utilização de ListBoxes

Suponha que você queira criar uma aplicação onde você possa incluir nomes em uma lista, removê-los e limpar a lista por completo :



1. Inicialize o Visual Basic.
2. Como demonstrado anteriormente, crie um Text Box.
3. Nomeie o Text Box de TXBNome.

Criando a Lista

4. Clique duas vezes sobre a ferramenta List Box da Caixa de Ferramentas.
5. Com o List Box selecionado pressione F4 para ter acesso à lista de propriedades.
6. Nomeie esta lista "LSBNomes".

Criando os Command Buttons

7. Crie três botões de nomeie-os "CMDInclui", "CMDRemove" e "CMDLimpa".
8. Com CMDInclui selecionado pressione F4 para ter acesso à lista de propriedades.
9. Colocar o valor TRUE na propriedade *Default*.

Codificando os Command Buttons

10. Clicar duas vezes sobre CMDInclui, uma janela de código irá aparecer. Certifique-se que o evento é *Click*.
11. Digitar as seguintes linhas de código:

```
If Trim(TXBNome.Text) <> "" Then
    LSBNomes.AddItem TXBNome.Text
End If
TXBNome.Text = ""
TXBNome.SetFocus
```

12. Clicar duas vezes sobre CMDRemove. Uma janela de código irá aparecer. Certifique-se que o evento é *Click*.

13. Digitar as seguintes linhas de código:

```
If LSBNomes.ListIndex <> -1 Then
    LSBNomes.RemoveItem LSBNomes.ListIndex
End If
TXBNome.SetFocus
```

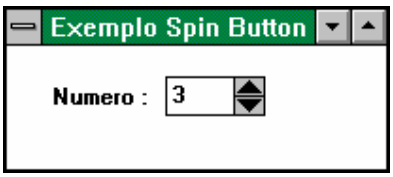
14. Clicar duas vezes sobre CMDLimpa. Uma janela de código irá aparecer. Certifique-se que o evento é *Click*.
15. Digitar as seguintes linhas de código:

```
LSBNomes.Clear
TXBNome.SetFocus
```

Executando a Aplicação

16. Pressione F5 ou selecione Start do menu Run.
17. Experimente adicionar e retirar nomes da lista.

Spin Buttons



Spin Buttons são utilizados para facilitar a entrada de dados em uma Text Box, quando se trabalha com números. Dependendo da seta pressionada, o conteúdo da Text Box será incrementado ou decrementado.

Propriedade	Descrição
Name	Nome do controle internamente.
Delay	Intervalo em milésimos de segundos entre cada evento SpinUp/SpinDown.
SpinOrientation	Direção das setas.

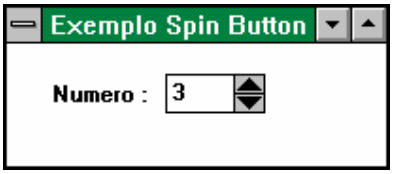
EVENTOS

SpinUp Este evento ocorre sempre que o usuário pressionar a parte superior do controle.

SpinDown Este evento ocorre sempre que o usuário pressionar a parte inferior do controle.

Exemplo - Utilização de Spin Buttons

Suponha que você queira criar uma aplicação onde seja necessário criar um campo numérico. Este campo deve ser incrementado apenas através do mouse. A tela do exemplo está abaixo.



1. Inicialize o Visual Basic.
2. Como demonstrado anteriormente, crie um Text Box.
3. Nomeie o Text Box de TXBNumero e sua propriedade Text = 0.
4. Como demonstrado anteriormente, crie um Label.

Criando o Spin Button

5. Clique duas vezes sobre a ferramenta Spin Button da Caixa de Ferramentas.
6. Com o Spin Button selecionado pressione F4 para ter acesso à lista de propriedades.
7. Nomeie esta lista "SPBNumero".

Associando código ao Spin Button

8. Clique duas vezes sobre o Spin Button. Uma janela de código será aberta.
9. No evento SpinUp, escrever a seguinte linha de código :

```
TXBNumero.text = TXBNumero.text + 1
```

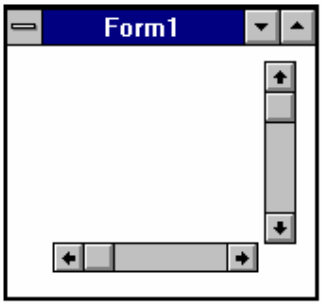
10. No evento SpinDown, escrever a seguinte linha de código :

```
TXBNumero.text = TXBNumero.text - 1
```

Executando a Aplicação

11. Pressione F5 ou selecione Start do menu Run.
12. Compostas

Scroll Bars



Scroll Bars são utilizados quando se deseja demonstrar a posição corrente em uma escala.

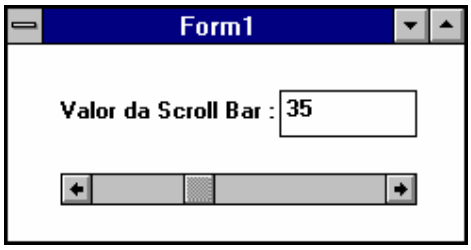
Propriedade	Descrição
LargeChange	Mudança de valor que ocorrerá quando o usuário clicar sobre o marcador de posição da barra.
Max	Valor máximo da barra.
Min	Valor mínimo da barra.
Name	Nome interno do controle.
SmallChange	Mudança de valor que ocorrerá quando o usuário clicar sobre as setas da barra.
Value	Valor correspondente à posição da barra.

EVENTOS

Change O evento Change indica que o conteúdo do controle foi alterado. No caso de scroll bars , ocorre quando o usuário rola a barra.

Exemplo - Utilização de ScrollBars

Para exemplificar o uso de um scrollbar, usaremos um text box. À medida que você mexer no Scrollbar, o valor do Text Box também alterará.



1. Inicialize o Visual Basic.
2. Como demonstrado anteriormente, crie o Text Box.
3. Nomeie o Text Box de TXBValor.

Criando o Scroll Bar

4. Clique duas vezes sobre a ferramenta Horizontal Scroll Bar da Caixa de Ferramentas.
5. Com o Scroll Bar selecionado pressione F4 para ter acesso à lista de propriedades.
6. Nomeie este Scroll Bar "HSBValor".
7. Coloque o valor 100 na propriedade *Max*.
8. Coloque o valor 0 na propriedade *Min*.
9. Coloque o valor 1 na propriedade *SmallChange*.
10. Coloque o valor 5 na propriedade *LargeChange*.

Codificando a Scroll Bar

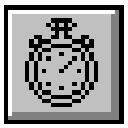
11. Clique duas vezes o scroll bar. A janela de código do evento Change aparecerá.
12. Digite a linha de código :

```
TXBValor.Text = HSBValor.Value
```

Executando a Aplicação

13. Pressione F5 ou selecione Start do menu Run.
14. Movimente-se pele scrollbar.
15. Experimente alterar o valor das propriedades *Large Change* e *SmallChange*.

Timers



Timers são utilizados para ativar eventos, periodicamente, em um prazo definido por você. Um exemplo seria sair de uma tela qualquer, quando o usuário deixasse o computador parado por um intervalo de tempo.

O controle timer somente é visível em *design time.*

Propriedade	Descrição
Enabled	Indica se o timer está ativo.
Interval	Intervalo de tempo que o controle será ativado.
Name	Nome interno do controle.

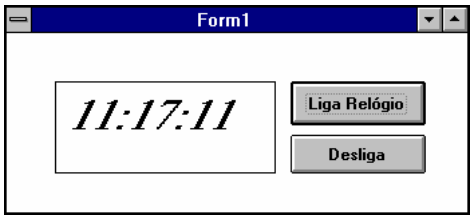
Obs.: Para um timer funcionar, a propriedade *Enabled* deve ter o valor *True* e a propriedade *Interval* deve ser diferente de 0.

EVENTOS

Timer Este evento ocorre a cada intervalo de tempo especificado na propriedade *Interval* do controle.

Exemplo - Utilizando um Timer

Relembrando a aplicação do relógio, agora vamos utilizar um timer para que a hora seja atualizada a cada segundo.



1. Inicialize o Visual Basic.
2. Como demonstrado anteriormente, crie o Text Box.
3. Nomeie o Text Box de TXBHora.

Criando os Command Buttons

4. Crie dois botões, nomeie-os "CMDLiga" e "CMDDesliga" e atribua, às suas propriedades *Caption* , os valores "Liga Relógio" e "Desliga", respectivamente.

Criando o Timer

5. Clique duas vezes sobre a ferramenta Timer da Caixa de Ferramentas.
6. Com o Timer selecionado pressione F4 para ter acesso à lista de propriedades.
7. Nomeie este timer "TIMsegundo".
8. Coloque o valor 1000 na propriedade *Interval*, e False em *Enabled*.

Codificando os Command Buttons

9. Clicar duas vezes sobre CMDLiga, uma janela de código irá aparecer. Certifique-se que o evento é *Click*.
10. Digitar a seguinte linha de código:

```
TIMsegundo.enabled = true
```

11. Clicar duas vezes sobre CMDDesliga. Uma janela de código irá aparecer. Certifique-se que o evento é *Click*.
12. Digitar a seguinte linha de código:

```
TIMsegundo.enabled = False
```

Codificando o Timer

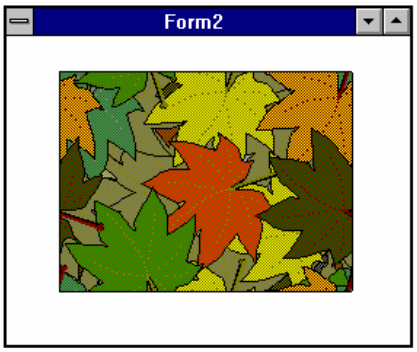
13. Clicar duas vezes sobre TIMSegundo, uma janela de código irá aparecer. Certifique-se que o evento é *Timer*.
14. Digitar a seguinte linha de código:

```
TXBHora.text = Format (Now, "hh:mm:ss")
```

Executando a Aplicação

15. Pressione F5 ou selecione Start do menu Run.
16. Ligue e desligue o relógio.
17. Experimente alterar o valor do *Interval* do timer.

Picture Box



Picture Boxes são basicamente utilizados para mostrar gráficos ou figuras. No entanto, picture boxes também podem conter controles. Desta forma, além de agrupar controles afins, você pode ter várias versões de uma parte de uma tela, em pictures diferentes , fazendo que apenas a versão (picture) que lhe convier no momento esteja aparente.

EVENTOS

Paint Acontece toda vez que um formulário ou controle é movimentado, mostrando parte de um picture box que inicialmente estava escondido. Este evento entretanto, só acontece quando a propriedade *AutoRedraw* do Picture Box está com valor *False*; caso contrário, a repintura do Picture Box é feita automaticamente, portanto o evento Paint não acontece.

Cuidado !! Em alguns casos não é aconselhável deixar a propriedade AutoRedraw de um picture Box com valor True; isto pode consumir muita memória do seu micro-computador.

Propriedade	Descrição
AutoRedraw	Indica se a pintura da Picture Box será feita automaticamente ou não
AutoSize	Se True, ajusta o tamanho da Picture Box de acordo o tamanho da figura.
Name	Nome interno do controle.
Picture	Exibe um Dialog Box para definir uma figura, no formato BMP, WMF ou ICO, com o qual o Picture Box deve ser preenchido.

CARREGANDO E DESCARREGANDO FIGURAS EM RUNTIME

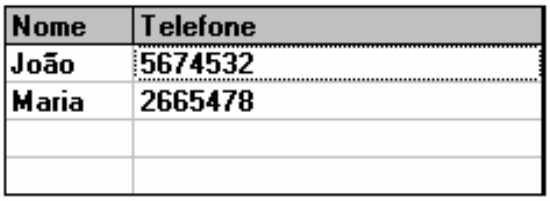
Para carregar ou descarregar figuras, em tempo de execução, utiliza-se a função *LoadPicture()*.

Exemplos :

```
Sub Command1_Click()       'Carrega uma figura na Picture Box
   Picture1.picture = LoadPicture ("c:\vb\icons\arrows\point12.ico")
End Sub

Sub Command2_Click()       'Limpa uma Picture Box
   Picture1.picture = LoadPicture ()
End Sub
```

Grid



Este controle trabalha com um conjunto de propriedades através das quais é possível selecionar linhas e colunas (como células em uma planilha), escrever dentro delas e copiar dados para elas. Um grid possui um número inicial de linhas e colunas que não pode ser menor que o seu número de linhas e colunas fixas. O dados que preenchem um grid são representados por um texto no qual separa-se colunas por *Tabs* e linhas por *CarriageReturn*.

Propriedade	Descrição
Rows, Cols	Indica o numero de linhas e colunas do grid.
Row, Col	Indica a linha e coluna corrente (apenas em *runtime*).
FixedRows, FixedCols	Número de linhas e colunas fixas do grid.
ColWidth	Vetor contendo a largura das colunas do grid em twips.
Clip	Retorna ou determina o conteúdo da(s) célula(s) selecionadas (apenas em *runtime*).
ColAlignment	Determina o alinhamento do texto dentro da célula.
GridLines	Indica se vai existir linhas de grade no grid
Text	Retorna ou determina o conteúdo da célula corrente.
Picture	Retorna ou determina um gráfico para a célula corrente.
SelStartRow, SelEndRow	Indica as linhas inicial e final de uma seleção.
SelStartCol, SelEndCol	Indica as colunas inicial e final de uma seleção.

Principais Métodos

AddItem - Adiciona uma linha ao grid na posição especificada e a preenche.

Sintaxe : nome-grid.additem conteúdo, índice

RemoveItem - Adiciona uma linha e seu conteúdo do grid. A linha a ser retirada é a especificada pelo índice.

Sintaxe : nome-grid.removeitem índice

Obs.: Índice em ambos os casos significa o número da linha a ser incluída ou retirada.

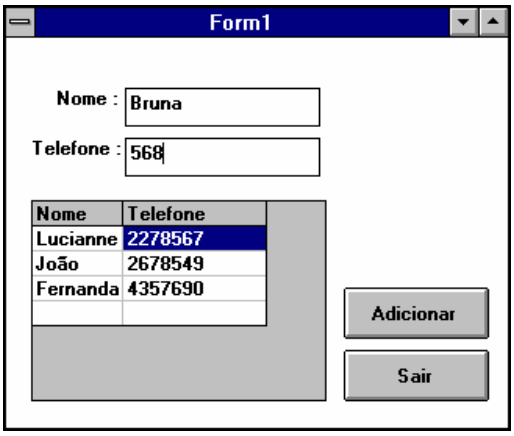
Exemplos :

```
Sub Command1_Click ()
   ' Determina a célula inicial e a preenche
   Grid1.col =0
   Grid1.Row =0
   Grid1.Text = "Nome"
End Sub


Sub Command2_Click ()
   ' Seleciona um grupo de células e as preenche
   Grid1.SelStartCol = 1
   Grid1.SelEndCol = 3
   Grid1.SelStartRow = 1
   Grid1.SelEndRow = 3
   tab = chr(9)
   carriage_return = chr(13)
   texto = "L1C1" + tab + "L1C2" + carriage_return
   texto = texto + "L2C1"+ tab + "L2C2"
   Grid1.clip = texto
End Sub
```

Exemplo - Utilização de Grid

Para exemplificar a utilização de um grid, construiremos uma pequena aplicação onde se possa cadastral nomes e telefones.



1. Inicialize o Visual Basic.
2. Como demonstrado anteriormente, crie dois Text Boxes.
3. Nomeie os Text Boxes de TXBNome e TXBTelefone.
4. Esvazie o conteúdo de suas propriedades *Text*.

Criando os Command Buttons

5. Crie dois botões de comando e nomeie-os "CMDAdicionar", "CMDSair", e atribua à sua propriedade Caption, os valores "Adicionar" e "Sair", respectivamente.

Criando o Grid

6. Clique duas vezes sobre a ferramenta Grid da Caixa de Ferramentas.
7. Com o Grid selecionado pressione F4 para ter acesso à lista de propriedades.
8. Nomeie este grid "GRDAgenda".
9. Coloque o valor 0 na propriedade *FixedCols*, e 1 em *FixedRows*.
10. Coloque o valor 2 na propriedade *Cols*, e 2 em *Rows*.

Codificando o Load do Formulário

11. Clique duas vezes sobre o formulário, a janela de código deve aparecer. Verificar se o evento selecionado é o *Load*.
12. Escrever as seguintes linhas de código :

```
GRDAgenda.Row = 0
GRDAgenda.Col = 0
GRDAgenda.Text = "Nome"
GRDAgenda.Col = 1
GRDAgenda.Text = "Telefone"
GRDAgenda.ColWidth(0) = 2500
GRDAgenda.ColWidth(1) = 1500
GRDAgenda.Width = 4300
```

Codificando os Command Buttons

13. Clicar duas vezes sobre CMDAdiciona, uma janela de código irá aparecer. Certifique-se que o evento é *Click*.
14. Digitar as seguintes linhas de código:

```
If Trim(TXBNome.Text) <> "" And Trim(TXBTelefone.Text) <> "" Then
    GRDAgenda.AddItem TXBNome.Text + Chr(9) + TXBTelefone.Text,
              GRDAgenda.Rows - 1
End If
```

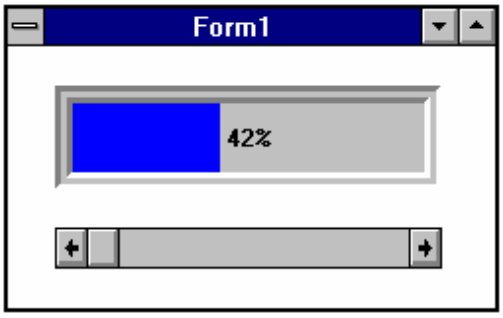
15. Clicar duas vezes sobre CMDSair. Uma janela de código irá aparecer. Certifique-se que o evento é *Click*.
16. Digitar as seguintes linhas de código:

```
End
```

Executando a Aplicação

17. Pressione F5 ou selecione Start do menu Run.

3D Panel

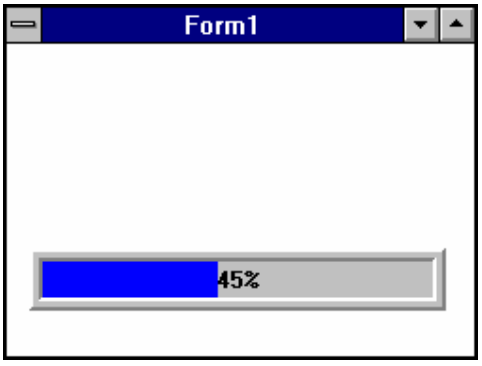


Este controle possui duas funções básicas: a primeira é dar ao formulário ou a um grupo de controles, uma qualidade tridimensional mais apurada, já que você pode colocar um painel 3D como fundo para um formulário inteiro ou para um grupo de controles (o painel 3D pode ser utilizado no lugar de um frame). A outra função, é utilizá-lo como indicador de andamento de uma atividade qualquer, uma vez que possui propriedades de preenchimento do seu interior.

Propriedade	Descrição
Alignment	Alinhamento do título do controle dentro do painel.
BevelInner	Determina o estilo do desnível interno do painel.
BevelOuter	Determina o estilo do desnível externo do painel.
BevelWidth	Determina a largura da borda do controle.
FloodColor	Determina a cor de preenchimento do controle.
FloodShowPct	Mostra, ou não, o percentual de preenchimento do controle.
FloodType	Indica como o preenchimento do controle irá ocorrer.

Exemplo - Painel 3D

Para demostrar o uso de um painel 3D, vamos marcar a passagem do tempo com o auxílio de um temporizador.



1. Inicialize o Visual Basic.

Criando o Timer

2. Crie o timer e nomeie-o "TIMSegundo".
3. Ajuste o intervalo para 250.

Criando o Painel

6. Clique duas vezes sobre a ferramenta Panel3D da Caixa de Ferramentas.
7. Com o Panel3D selecionado pressione F4 para ter acesso à lista de propriedades.
8. Nomeie este painel "SSPMarcador".
9. Coloque o valor 1 na propriedade *BevelInner*, e 2 em *BevelOuter*.
10. Coloque o valor 2 na propriedade *BevelWidth*, e 3 em *BorderWidth*.
11. Coloque o valor 1 na propriedade *FloodType*.

Criando variável a nível de formulário.

12. Clique duas vezes no formulário para abrir a janela de código.
13. Selecione o objeto "(general)".
14. Crie a variável com o seguinte comando:

```
Dim cont as single
```

Inicializando a Variável

15. Clique duas vezes sobre o formulário para abrir para abrir uma janela de código. Certifique-se que o evento selecionado é o *Load*.
16. Digitar o seguinte comando:

```
Cont = 0
```

Codificando o Timer

17. Clique duas vezes sobre o objeto Timer. Uma janela de código deverá aparecer.
18. Digite as seguintes linhas de código :

```
cont = cont + 0.25
If cont <= 10 Then
      SSPMarcador.FloodPercent  = cont * 10
Else
      TIMSegundo.Enabled = False
End If
```

Executando a Aplicação

19. Pressione F5 ou selecione Start do menu Run.

Group Push Buttons

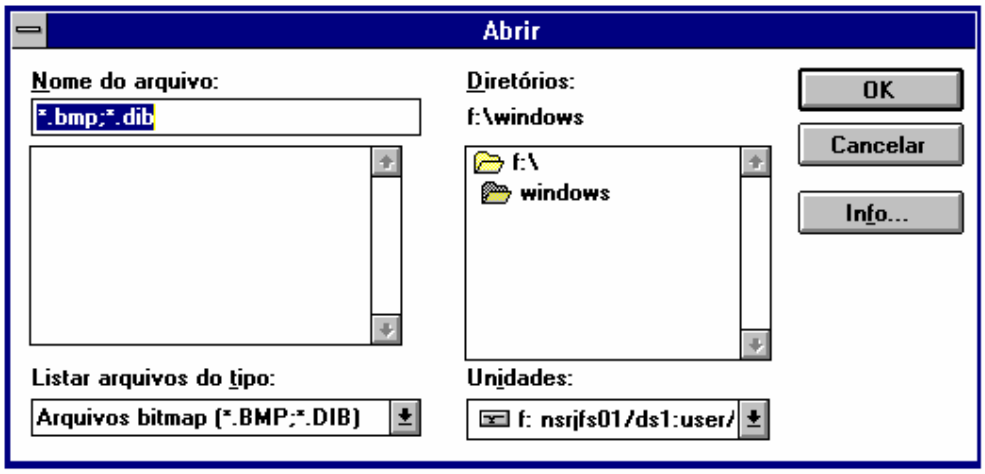
Group Push Buttons trabalham como se fossem uma combinação de command buttons e de option buttons. Trabalham como Command Buttons porque, quando pressionados, ações são executadas. Também trabalham como Option Buttons porque se você ajustar suas propriedades corretamente, eles funcionam como grupos de opções.

Propriedade	Descrição
GroupAllowAllUp	Determina se todos os botões de um grupo lógico podem estar não pressionados ao mesmo tempo.
GroupNumber	Determina grupos lógicos dentro de um toolbar.
Outline	Cria uma borda ao redor do botão.
PictureDisabled	Determina a figura que o botão deverá conter quando estiver desabilitado.
PictureDn	Determina a figura que o botão deverá conter quando estiver pressionado.
PictureDnChange	Determina de que modo a figura que o botão contém quando não está pressionado deverá ser alterada quando ele for pressionado.
PictureUp	Determina a figura que o botão deverá conter quando não estiver pressionado.

OBS.1: Para a criação de botões exclusivos entre si, deve-se obedecer a ordem de criação dos botões, ou seja, um grupo de botões somente será exclusivo em relação aos outros do grupo se eles tiverem sido criados um após o outro.

OBS.2: Para a criação de um toolbar os botões devem ser criados dentro de um *3DPanel*, que para se alinhar ao formulário deve ter a propriedade *Align* ajustada.

Common Dialog



Esta ferramenta do Visual Basic permite que você utilize as caixas de diálogo padrão do Windows. O tipo da caixa utilizada será definida pelo valor da propriedade *action* do controle.

Exemplo :

```
Sub Command1_Click
   CMDialog1.action = 2
End Sub
```

Valor da Propriedade Action	Tipo da caixa de diálogo
0	Não executa nenhuma ação.
1	Abrir Arquivo
2	Salvar Arquivo
3	Escolher Cores
4	Escolher Fonte
5	Configurar Impressora
6	Invocar o Ajuda do Windows

Drive List Box, Directory List Box e File List Box

Estes três controles são uma forma especial de List Boxes. Com eles, você pode criar telas de seleção de arquivos personalizadas. É importante visualisar a importância destes controles juntos, pois uma mudança de conteúdo de um controle refletirá no conteúdo dos outros.

O **File List Box** é uma lista que contém o nome de todos os arquivos pertencentes ao diretório atual. Além das propriedades normais de uma lista, ele apresenta uma série de propriedades inerentes à sua função.

Propriedade	Descrição
Archive	Mostra todos os arquivos do tipo ARCHIVE.
FileName	Retorna o nome do arquivo selecionado.
Hidden	Mostra todos os arquivos do tipo HIDDEN.
Normal	Mostra todos os arquivos do tipo NORMAL.
Path	Indica o caminho de procura dos arquivos. Acessível apenas em *runtime*.
Pattern	Indica o tipo de arquivos procurados. Acessível apenas em *runtime*.
ReadOnly	Mostra todos os arquivos do tipo READONLY.
System	Mostra todos os arquivos do tipo SYSTEM

O **Directory List Box** é um tipo combo box que mostra os diretórios na unidade de disco selecionada.

Propriedade	Descrição
Name	Nome interno do controle.
Path	Indica o diretório selecionado. (Acessível apenas em *runtime*).

EVENTO CHANGE

Sempre que o valor do Directory List Box é alterado, um evento *change* acontece. Neste evento, você deve codificar a sincronização deste controle com o File List Box.

Exemplo :

```
Sub Dir1_Change ()
   File1.Path = Dir1.path
End Sub
```

O **Drive List Box** é um tipo combo box que mostra os drives disponíveis no sistema.

Propriedade	Descrição
Name	Nome interno do controle.
Drive	Retorna o valor do Drive selecionado. (Acessível apenas em *runtime*).

EVENTO CHANGE

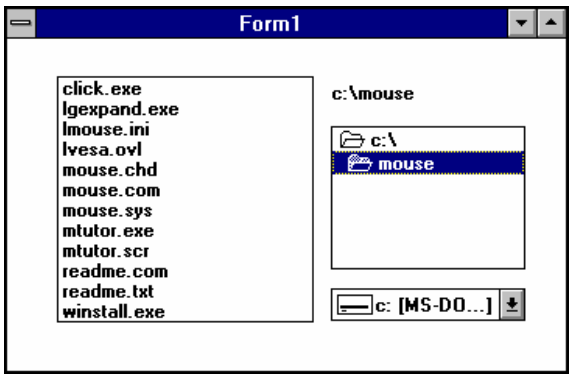
Sempre que o valor do Drive List Box é alterado, um evento change acontece. Neste evento, você deve codificar a sincronização deste controle com o Directory List Box.

Exemplo :

```
Sub Drive1_Change ()
   Dir1.Path = Drive1.drive
End Sub
```

Exemplo - Drive List Box, Directory List Box e File List Box

Para exemplificar estes três controles, criaremos uma tela de abertura de arquivos.



1. Inicialize o Visual Basic.

Criando o Drive List Box

2. Clique duas vezes sobre a ferramenta Drive List Box da Caixa de Ferramentas.
3. Com o Drive List Box selecionado pressione F4 para ter acesso à lista de propriedades.
4. Nomeie este Drive List Box "DVBdrive".

Criando o Directory List Box

5. Clique duas vezes sobre a ferramenta Directory List Box da Caixa de Ferramentas.
6. Com o Directory List Box selecionado pressione F4 para ter acesso à lista de propriedades.
7. Nomeie este Directory List Box "DLBDirectory".

Criando o File List Box

8. Clique duas vezes sobre a ferramenta File List Box da Caixa de Ferramentas.
9. Com o File List Box selecionado pressione F4 para ter acesso à lista de propriedades.
10. Nomeie este File List Box "FLBFile".

Codificando os eventos Change

11. Clique duas vezes no Drive List Box para abrir a janela de código.
12. Certifique-se que o evento selecionado é *Change*.
13. Escrever a seguinte linha de código :

```
DLBDirectory.Path = DVBDrive.Drive
```

14. Clique duas vezes no Directory List Box para abrir a janela de código.
15. Certifique-se que o evento selecionado é *Change*.
16. Escrever a seguinte linha de código :

```
FLBFile.Path = DLBDirectory.path
```

Executando a Aplicação

17. Pressione F5 ou selecione Start do menu Run.

Tipos de Dados

Objetivo do módulo

Introduzir os tipos de dados do Visual Basic e suas regras de escopo e durabilidade. Este é o primeiro, de um conjunto de capítulos, que ensina a programação em Visual Basic.

Esta página foi deixada em branco intencionalmente.

Tipos de Dados de Variáveis

Tipo		Descrição
Integer	%	Inteiro de 2 bytes
Long	&	Inteiro de 4 bytes
Single	!	Num. Ponto Flutuante de 4 bytes
Double	#	Num. Ponto Flutuante de 8 bytes
Currency	@	Num. Ponto Decimal Fixo de 8 bytes
String	$	String de caracteres
Variant		Data/Hora, string, num de ponto Flutuante

Obs.: Operações que utilizam o tipo de dado *Currency* são mais rápidas e exatas do que as que utilizam o tipo *Single* e *Double*.

Declaração de Variáveis

Você pode declarar um variável de duas maneiras : usando o comando *Dim* ou então, uma das duas palavras reservadas - *Global* ou *Static*. A declaração de variáveis no Visual Basic é extremamente importante, pois qualquer variável não declarada é considerada como sendo do tipo *Variant*.

Existem duas sintaxes de declaração explícita de variáveis :

Usando a palavra reservada AS	Usando caracter de tipo
`Dim I as Integer`	`Dim I%`
`Dim Total as Double`	`Dim Total#`
`Dim Nome as Sring`	`Dim Nome$`
`Dim Valor as Currency`	`Dim Valor@`

Para fazer declaração de variáveis compostas, usar :

```
Dim I as Integer, Total as Double
```

Cuidado !!! Em declarações de variáveis compostas, especificar o tipo para cada uma delas. No exemplo abaixo, apenas a última variável será do tipo Integer, as outras serão do tipo Variant.

```
Dim I, J, K as Integer
```

Declaração Obrigatória de Variáveis

Para que a declaração de variáveis no seu projeto seja obrigatória, colocar *Yes* no item *Require Variable Declaration* nas opções de configuração do ambiente.

Strings de Tamanho Fixo e de Tamanho Variável

Quando você não souber o tamanho de uma string, você poderá declará-la com tamanho variável. Caso contrário, você deverá declará-lo. Uma string de tamanho fixo jamais terá valor nulo (NULL).

Ex :

```
Dim palavra as string
Dim palavra2 as string * 50
```

Inicialização de Variáveis

O Visual Basic, automaticamente, inicializa todas as variáveis numéricas com zero (0) e todas as strings, de tamanho fixo, com brancos.

Nomenclatura de Variáveis

1. Nomes de variáveis podem ter, no máximo, 40 caracteres.
2. Nomes podem conter letras, números e *underscores* (_).
3. O primeiro caracter do nome deve ser uma letra.
4. Palavras reservadas do Visual Basic não poderão ser utilizadas.

Tipo de Dado Variant

Variant é o tipo de dado *default* do Visual Basic. Uma variável do tipo *variant* pode conter qualquer tipo de dado : números, letras ou datas. Não é necessário fazer qualquer tipo de conversão para atribuir valores destes tipos a um *variant*; o Visual Basic se encarrega de fazer a conversão automaticamente.

Para saber o tipo de dado que está dentro de um *variant*, você pode utilizar as seguintes funções boleanas do VB : `IsNumeric` e `IsDate`.

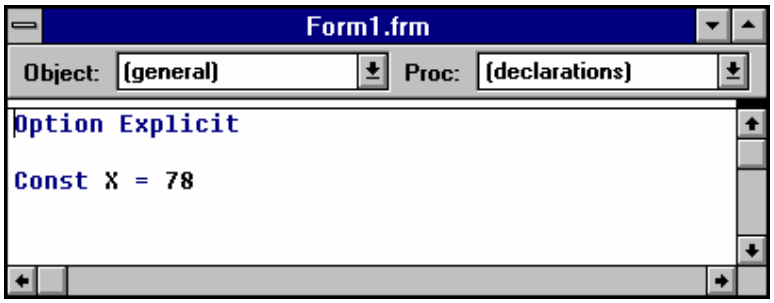
Para somar variáveis do tipo *variant*, as mesmas devem conter valores numéricos. Para concatenar variáveis do tipo *variant* utilizar o símbolo ***&*** para evitar ambigüidade.

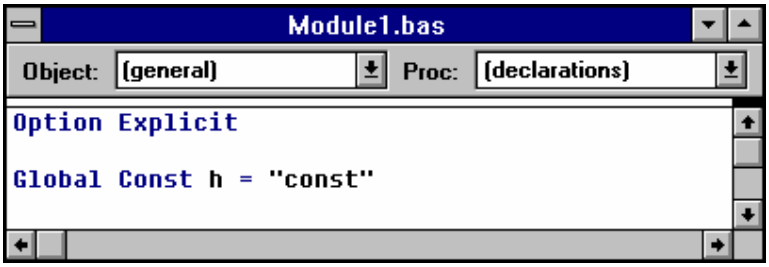
```
Sub Command1_Click ()
Dim a
Dim b
```

```
   a = 3
   b = 5
   Print (a & b)   'resulta em 35
   Print (a + b)   'resulta em 8
End Sub
```

Variant tem o valor *Empty* até que algum valor seja atribuído a ela. *Empty* é um valor especial diferente de Null, brancos ou zero. Para verificar se uma variável está Empty, utilizara função `IsEmpty` do Visual Basic.

Constantes



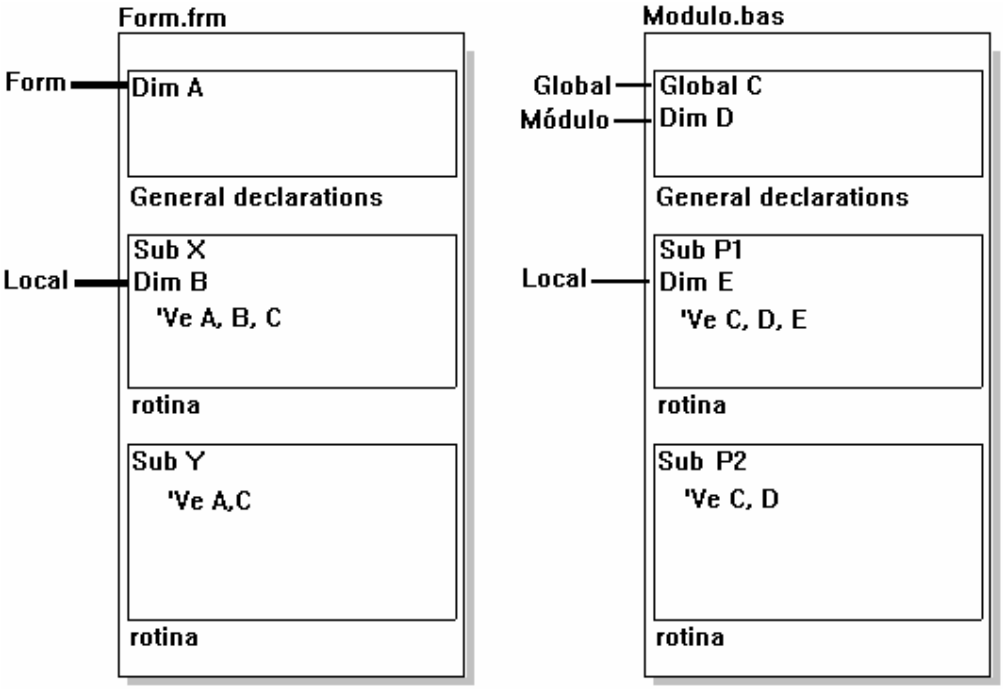


Constantes são entidades do programa cujo valor você necessita apenas saber, sem ter que atualizar. O Visual Basic mantém um arquivo, o CONSTANT.TXT que contém uma série de constantes predefinidas.

Este arquivo poderá ser adicionado ao seu projeto.

Para declarar uma constante, utilize a palavra reservada **Const** dentro do *general declarations* de qualquer formulário ou de um módulo de código (*.BAS).

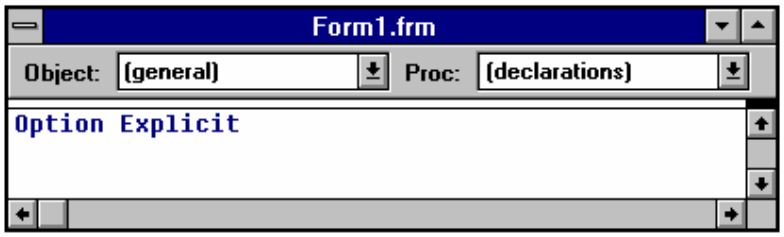
Escopo de Dados



Pode-se definir o escopo como sendo o nível de visibilidade de uma variável dentro de uma aplicação. Já vimos anteriormente que, uma aplicação Visual Basic é composta de formulários e módulos de procedimentos. Assim sendo, podemos ter variáveis visíveis, dentro de :

- ✓ Apenas uma rotina dentro de um formulário
- ✓ Dentro de todas as rotinas de um formulário
- ✓ Dentro de uma única rotina de um módulo de procedimentos
- ✓ Dentro de todas as rotinas de um módulo de procedimentos
- ✓ Dentro de todos os formulários e módulos de procedimentos da aplicação.

General Declarations



General Declarations

O "objeto" *general* de um formulário/módulo é a área onde você declara todas as variáveis e rotinas, que são comuns a todos os procedimentos daquele formulário/módulo. Na parte de *declarations*, você define suas variáveis (globais ou de módulo/formulário). Para definir uma rotina basta escrever *Sub nome_rotina* ou *Function nome_funcao (parametros)* que o Visual Basic abre uma janela para que o código seja escrito.

Escopo	Declaração	Visibilidade
Local	Dim, Static (dentro de um procedimento)	dentro do procedimento
Módulo ou Formulário	Dim (dentro da General Declarations de um módulo de procedimentos ou formulário)	dentro de todas as rotinas de um formulário ou módulo de procedimentos
Global	Global (dentro do General Declarations de um módulo de procedimentos)	em todos os pontos da aplicação

Static versus Dim

Ao usar a palavra reservada *Static* ao invés de *Dim* dentro de uma rotina de um formulário (.Frm) ou de um módulo de procedimento (.Bas), você estará trabalhando não com o escopo de uma variável, mas com sua durabilidade. Declarar uma variável como *static* dentro de uma rotina significa dizer que aquela variável não será reinicializada cada vez que a rotina for chamada (apenas no Load do formulário), no entanto ela só estará visível dentro daquela rotina. Ou seja, ela terá escopo Local e durabilidade enquanto o formulário estiver ativo.

Tipos de Dados Adicionais do Visual Basic

Tipos de Dados Definidos pelo Usuário

Além dos tipos de dados oferecidos pelo Visual Basic, você também pode criar estruturas de dados suas. Um exemplo típico é criar estruturas semelhantes a um registro do seu arquivo, ou criar variáveis que servirão de padrão para a aplicação (Ex.: Tipo nome é um string de 50 posições; qualquer nome do seu sistema será declarado como sendo do tipo nome, para que todas tenham o mesmo tamanho).

Sintaxe :

```
Type tipo-do-usuário
   elemento as string
   elemento as string
End Type
```

Exemplo :

```
Type Reg_cliente
   nome as string *50
   telefone as string *11
End Type
```

Em uma rotina utilizar :

```
Sub Le_Cliente
Dim Cliente as Reg_cliente
   Cliente.nome = txb_nome.text
   Cliente.telefone = txb_telefone.text
End Sub
```

Vetores

Assim como várias outras linguagens de programação, o Visual Basic também permite a criação de vetores. Vetores são grupos de variáveis de um mesmo tipo que compartilham um nome. Cada elemento do vetor é identificado por um índice único.

Sintaxe :

```
Dim nome_vetor(limite_superior) as tipo_de_dado
```

```
Dim nome_vetor( limite_inferior To limite_superior) as tipo_de_dado
```

Observação Por default o primeiro elemento de um vetor tem índice 0, portanto se você declarar um vetor com o comando *Dim vetor(10) as integer*, você terá um vetor de 11 elementos e não de 10. Para evitar problemas prefira utilizar a segunda sintaxe, declarando *Dim vetor(1 to 10) as integer*.

Vetores Multidimensionais ou Matrizes

O Visual Basic permite a criação de vetores com mais de uma dimensão. Desta forma, você pode criar vetores de até 60 dimensões (matrizes).

Exemplo :

```
Dim matriz(9, 9)  as single
Dim Matriz (1 to 10, 1 to 10) as single
```

Vetores Dinâmicos

Em alguns casos, você sentirá a necessidade de utilizar um vetor, mas o tamanho do vetor somente será definido em runtime. O VB permite que você crie vetores dinâmicos, ou de tamanho variável. Para isso, o vetor deve ser declarado sem tamanho dentro do *general declarations* de um formulário ou módulo.

Exemplo :

```
'Colocar dentro do general declarations de um formulário ou módulo
Dim Vetor() as string * 25
```

Feito isto, dentro da rotina onde vai ser definido o tamanho, redimensionar o vetor usando o comando *ReDim*.

Exemplo :

```
Sub Command1_click ()
   ReDim Vetor (List1.listcount)
End Sub
```

Importante Sempre que você usa o comando *ReDim*, todos os valores contidos no vetor são perdidos e o vetor é todo preenchido com o valor NULL. Para aumentar o tamanho de um vetor sem que seu conteúdo seja perdido, utilizar a palavra chave *Preserve*.

Exemplo :

```
Sub Command1_click ()
   ReDim Preserve Vetor (List1.listcount)
End Sub
```

Para limpar uma matriz e reduzir o seu tamanho fazer :

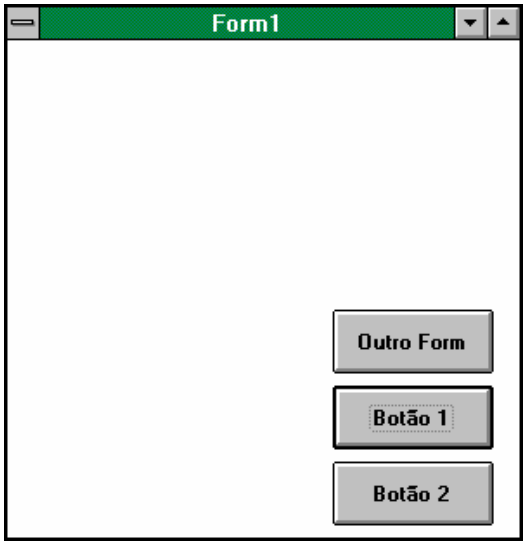
```
Redim vetor(0)
```

Exemplo - Escopo de Variáveis

Para demonstrar a declaração de variáveis, criaremos um exemplo onde quatro variáveis serão declaradas diferentemente :

- VAR_GLB - variável global.
- VAR_FRM - variável a nivel de formulário.
- VAR_TIM - variável estática local.
- VAR_LOC - variável estática local a dois procedimentos diferentes.

Teremos também dois formulários, onde um terá todas as funções de display e mudança de valores das variáveis e outro terá apenas a função de alterar o valor da variável de nível global.



1. Inicializar o Visual Basic.

Criação de Formulários

2. Nomear o formulário aberto de "form1".
3. Criar outro formulário. Nomeá-lo de "form2".
4. Criar um módulo de código, ou seja, um .BAS.

Criação de Controles

5. Em *form1*, criar três botões. Atribuir os seguintes valores às suas propriedades :

Caption	Name
Outro Form	CMDOutroForm
Botão 1	CMDBotao1
Botão 2	CMDBotao2

6. Em *form1*, criar um timer. Habilitá-lo e colocacar como intervalo de funcionamento o valor de 1 segundo (lembre-se que a unidade de VB é milésimo de segundo).

Criação de Variáveis

7. Declarar uma variável global do tipo inteiro com o nome de VAR_GL.
8. Declarar uma variável a nível de formulário no *Form1* do tipo inteiro com o nome de VAR_FRM.
9. Declarar uma variável estática local ao evento *Timer* do *Timer* no *Form1* do tipo single com o nome de VAR_TIM.
10. Declarar uma variável estática local ao evento *click* do *CMDBotao1* no *Form1* do tipo inteiro com o nome de VAR_LOC.

11. Declarar uma variável estática local ao evento *click* do *CMDBotao2* no *Form1* do tipo inteiro com o nome de VAR_LOC.

Escrevendo o código

12. Associar ao evento *Timer* do *Timer* no *Form1*, as seguintes linhas de código :

```
Var_Tim = Var_Tim + 1
If  Var_Tim MOD 2 = 0 Then
      Var_Frm = Var_Frm + 1
Endif
```

13. Associar ao evento *Click* do *CMDBotao1* no *Form1*, as seguintes linhas de código :

```
Var_Loc = Var_Loc + 1
Print Var_Loc, Var_Frm, Var_gl
Print
```

14. Associar ao evento *Click* do *CMDBotao2* no *Form1*, as seguintes linhas de código :

```
Var_Loc = Var_Loc - 1
Print Var_Loc, Var_Frm, Var_gl
Print
```

15. Associar ao evento *Click do CMDOutroForm no Form1*, a seguinte linha de código

```
Form2.Show
```

16. Associar ao evento *Load* do *Form2*, as seguintes linhas de código :

```
Var_gl = Var_gl + 1
```

Executando a Aplicação

17. Pressionar F5 ou selecione Start do menu Run.
18. Pressione várias vezes os botões e observe o comportamento das variáveis.

Codificando em Visual Basic

Objetivo do módulo

Fornecer as ferramentas necessárias para que você possa começar a escrever o código por trás das interfaces em Visual Basic.

Procedimentos

Procedimentos são conjuntos de comandos Visual Basic que podem ser chamados de unidades lógicas. Existem dois tipos de procedimentos : subrotinas e funções.

Subrotinas ➔ Conjunto de comandos que não obrigatoriamente retorna um valor ao procedimento que o chamou. Ao chegar ao final de sua execução retorna ao módulo que fez a chamada.

Sintaxe :

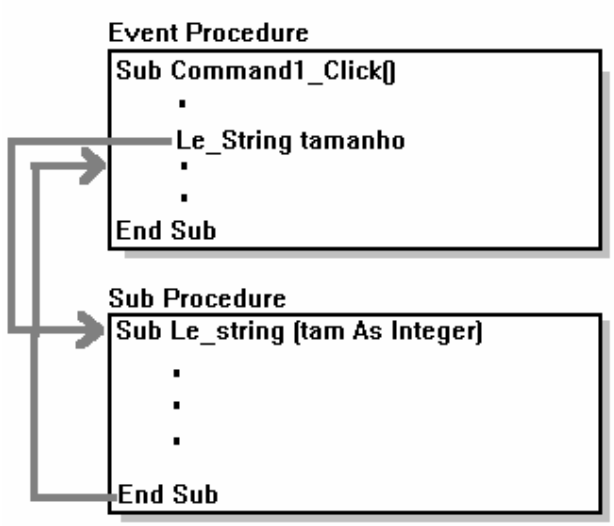
```
Sub Nome_da_subrotina()
      Bloco de comandos
End Sub
```

Funções ➔ Similar a uma rotina. No entanto, é de um tipo de dados, assim como variáveis. Ao final de sua execução, retorna um valor ao módulo que fez a chamada.

Sintaxe :

```
Function nome_função() As tipo_de_dados
      Blocos de comandos
      nome_função = Valor
End Function
```

Passando Parâmetros para uma Subrotina



Argumentos

Qualquer procedimento pode receber dados, se tiver sido declarado para tal. Cada argumento passado deve ter seu equivalente (do mesmo tipo de dado) na lista de parâmetros da subrotina ou função.

Sintaxe - declaração da subrotina :

```
Sub nome_subrotina ( parâmetro1 as Integer, parâmetro2 as Integer)
   Bloco de comandos
End Sub
```

Sintaxe - chamada da subrotina :

```
nome_subrotina argumento1, argument2
```

Sintaxe - declaração de função :

```
Function nome_função ( parâmetro1 as Integer,   parâmetro2 as
Integer) as Integer
   Bloco de comandos
   nome_função = valor
End Function
```

Sintaxe - chamada de função :

```
Dim resultado as Integer
Resultado = nome_função(argumento1, argumento2)
```

Passagem de Argumentos por Valor e por Referência

A passagem de argumentos para uma função ou subrotina pode ser feita por Valor ou por Referência. Na passagem por valor, a subrotina ou função recebe apenas uma cópia do argumento, sendo assim qualquer alteração no argumento dentro da subrotina/função não terá efeito no dado real.

Já na passagem de argumentos por referência, a subrotina/função recebe o endereço que realmente contém o dado. Assim sendo, qualquer alteração no argumento alterará o dado de verdade.

O Visual Basic, por default, passa argumentos por referência. Para passar argumentos por valor, utiliza-se a palavra chave *ByVal* na lista de parâmetros, ou então coloca-se o argumento entre parênteses na chamada da subrotina/função.

Observações :

1. Propriedades de objetos somente podem ser passadas por valor.

2. Se o seu argumento é do tipo variant, e a sua subrotina/função espera um parâmetro de um tipo de dado definido, a passagem do argumento deve ser feita por valor.

3. Você precisa apenas utilizar a palavra chave ByVal ou então os parênteses extra.

Exemplos :

-- declaração da subrotina --

```
Sub nome_subrotina ( ByVal parâmetro1 as Integer )
   Bloco de comandos
End Sub
```

-- chamada da subrotina --

```
nome_subrotina (argumento1)
```

-- declaração de função --

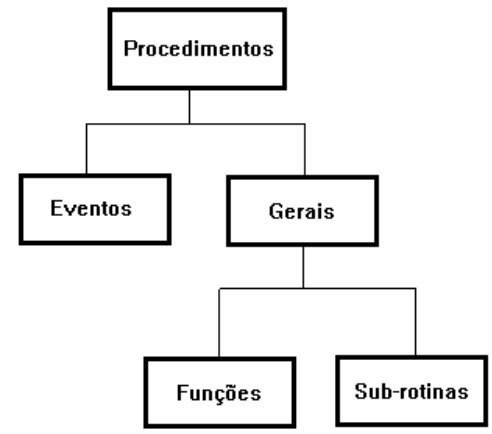
```
Function nome_função ( ByVal parâmetro1 as Integer) as Integer

   Bloco de comandos
   nome_função = valor
End Function
```

-- chamada de função --

```
Dim resultado as Integer
Resultado = nome_função((argument1))
```

Tipos de Procedimentos



O Visual Basic possui duas categorias de procedimentos : Eventos e Gerais. Procedimentos gerais, como vimos anteriormente podem ser funções ou subrotinas, e são ativados pelo próprio programador via código.

Event Procedures

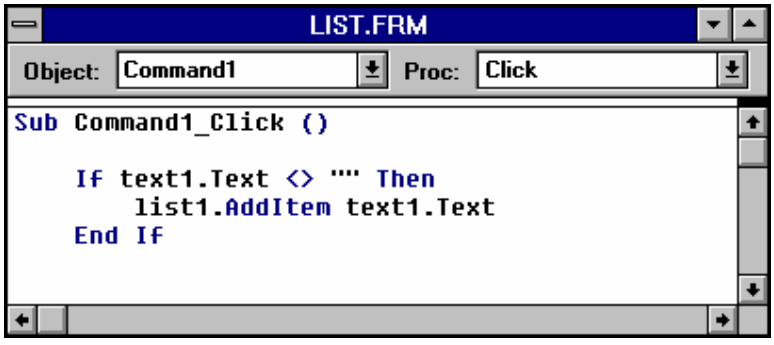
Procedimentos de Eventos, ou Event Procedures, são sempre ativados a partir de um evento acionado pelo usuário ou então pelo sistema (Windows).

Event Procedures estão sempre ligados a um formulário ou controle. O nome do procedimento indica o evento e a qual formulário/controle que o evento está associado.

Sintaxe : `Sub nome-objeto_nome-evento()`

Exemplo : Command1_Click, Form1_Load, Text1.Change, etc.

Criando uma Event Procedure



O Visual Basic já fornece as declarações de todos os procedimentos de eventos existentes para cada controle. Para inserir código de tratamento para qualquer evento, você deve abrir a janela de código, selecionar o objeto que você deseja tratar, selecionar o evento, e então digitar o código.

Ativando um Procedimento de Evento

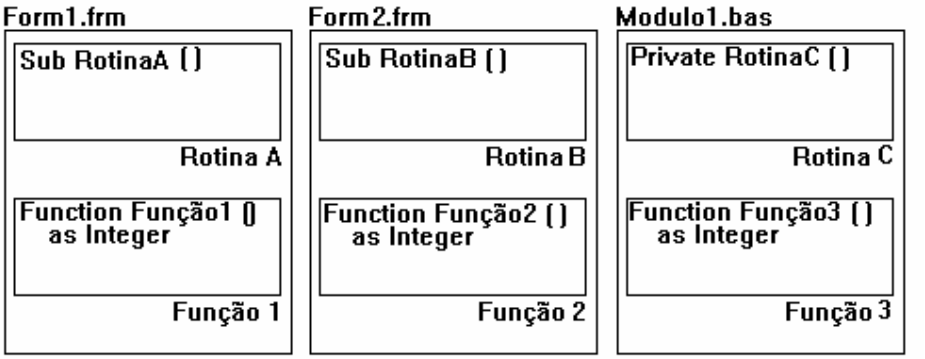
O Visual Basic reconhece automaticamente o acontecimento de um evento. Ao reconhecer o evento, ele executa imediatamente o código de tratamento para aquele evento.

Escopo de Procedimentos de Eventos

Procedimentos de Eventos somente estão disponíveis dentro do formulário onde eles foram definidos.

Escopo de Procedimentos Gerais

Assim como variáveis, procedimentos gerais também têm escopo. Veja no exemplo abaixo de onde as funções e subrotinas declaradas poderão ser vistas :



Fromulário/Módulo	Pode ver
Form1.frm	Rotina A, Função1, Função 3
Form2.frm	Rotina B, Função2, Função 3
Modulo.bas	Rotina C, Função 3

Como você pôde observar no exemplo acima, existem três escopos de rotinas: a nível de formulário, a nível global e a nível privado em módulos.

Escopo a nível de formulário

Declarada com as palavras reservadas Sub ou Function dentro de um formulário. Somente podem ser chamadas de dentro dele.

Escopo a nível global

Declarada com as palavras reservadas Sub ou Function dentro de um módulo. Podem ser chamadas de qualquer ponto da aplicação.

Escopo privado em módulos

Utilizar a palavra reservada Private antes de Sub ou Function dentro de um módulo. Somente podem ser chamadas de dentro dele.

Métodos

Métodos são um tipo especial de procedimentos que o Visual Basic fornece para você. Métodos sempre estão associados a objetos, mas é você que os ativa.

Características de um método :

- Você não pode criar um método, o VB já os cria para você. Métodos somente podem ser chamados.
- Não é possível ver ou alterar o código de um método.
- Os nomes de métodos são palavras reservadas do Visual Basic.

Sintaxe :

```
nome_controle.método
```

Exemplos :

```
Form1.Hide
List1.AddItem
GRDAgenda.RemoveItem
Picture1.Drag
```

Funções de Conversão de Expressões Numéricas e de Caracteres

Função	Descrição
Chr	Retorna um caracter para o código ANSI informado. Ex.: Chr(13) + Chr(10) , CarriageReturn e LineFeed.
Format	Rotina que retorna um número no formato que você quiser. Ex.: Format(Now, "dd-mm-yy"), traz a data de hoje no formato especificado.
LCase	Retorna o caracter minúsculo do caracter informado. Ex.: LCase("H"), retorna "h".
Left	Traz os *n* caracteres mais a esquerda. Ex.: Left("gravador",5), retorna "grava".
Len	Traz o tamanho de um string. Ex.: Len("Mar"), retorna 3.
LTrim	Retira espaços à esquerda em um string.
Mid	Retorna parte de uma string. Ex.: Mid("reflorestamento", 3, 8), retorna "floresta".
Right	Traz os *n* caracteres mais a direita. Ex.: Right("armário",3), retorna "rio".
Rtrim	Retira espaços à direita em um string.
Trim	Retira espaços à direita e à esquerda em um string.
Ucase	Retorna o caracter maiúsculo do caracter informado. Ex.: UCase("g"), retorna "G".
Val	Converte um string de dígitos em um número. Ex.: Val("100"), retorna 100; Val("1345,98") retorna 1345.

Função	Descrição
CCur	Converte uma expressão caracter para tipo currency.
Cdbl	Converte uma expressão caracter para tipo double.
CInt	Converte uma expressão caracter para tipo integer.
CLng	Converte uma expressão caracter para tipo long.
CSng	Converte uma expressão caracter para tipo single.
CStr	Converte uma expressão caracter para tipo string.
CVar	Converte uma expressão caracter para tipo variant.

OBS.: A função *CVAR* reconhece o indicador de decimal do Brasil (,). Para capturar um número de casas decimais, usá-la em conjunção com *CDBL*, *CINT*, *CLNG*, *CSNG*, dependendo do tipo de número que desejar armazenar.

Estruturas Lógicas e Condicionais

Objetivo

Sendo este um curso direcionado a profissionais de informática acostumados à utilização de estruturas de controle como condições e repetições, este módulo apresentará apenas as sintaxes das estruturas já conhecidas e também quaisquer novidades que estas estruturas no Visual Basic possuam.

If condição **Then** comando

If condição **Then**

bloco de comandos

End If

Existem dois tipos de estrutura condicional do tipo *If...Then...* . A primeira sintaxe, escrita em apenas uma linha, deve ser utilizada quando apenas um comando for ser executado como resultado de uma condição verdadeira. Quando, como resultado de uma condição, um bloco de comandos tiver que ser executado, torna-se necessária a utilização do *End If* como demarcador do final do bloco de comandos.

Os seguintes operadores podem ser usados na composição de condições : =, <>, <, >, <=, >=.

If condição1 **Then**

Bloco de comandos

ElseIf condição2 **Then**

Bloco de comandos

Else

Bloco de Comandos

End If

Nesta estrutura permite-se testar várias condições em um único bloco de *ifs*, e reagir diferentemente a cada uma das situações. Esta estrutura de comandos possui as seguintes características :

- ✓ Pode comportar um número ilimitado de *ElseIfs*
- ✓ Se nenhuma condição for verdadeira, o bloco de comandos seguindo o *Else* será executado.

Select Case variável a ser testada

Case expressão

Bloco de comandos

Case expressão

Bloco de comandos

Case Else

bloco de comandos

End Select

A estrutura de comandos *Select Case* funciona como uma estrutura *If...Then...ElseIf...Else...* , porém é mais eficiente. Nos exemplos abaixo,

demonstra-se o uso das palavras reservadas *TO* e *IS*, que diversificam ainda mais a utilização desta estrutura.

Exemplos :

```
Select case variável
   case 1,3,5,7 to 11
      comandos     'Entra se variável for igual a 1,3,5,7,8,9,10,11
   case 2, 4, 6, IS >=12
      comandos               'Entra   se   variável   for   igual   a
   2,4,6,12,13,14,15,...
End Select
```

```
Select case palavra
   Case "A" to "a"
      'Entra se pertencer a este intervalo. Ex.: "BRASIL" entra
      comandos
   Case "H" to "c"
      comandos
End Select
```

Do While condição

bloco de comandos

Loop

Do

bloco de comandos

Loop While

Ambas sintaxes acima tem a mesma funcionalidade : executar um bloco de comandos enquanto uma condição for verdadeira. A diferença entre elas é : a composição *do while/loop* testa a condição antes de executar o bloco de comandos. Já a

composição *do/loop while* executa o bloco de comandos uma vez e depois testa a condição.

Do Until condição

bloco de comandos

Loop

Do

bloco de comandos

Loop Until

Ambas sintaxes acima tem a mesma funcionalidade : executar um bloco de comandos até que uma condição se torne verdadeira. A diferença entre elas é : a composição *do until/loop* testa a condição antes de executar o bloco de comandos. Já a composição *do/loop until* executa o bloco de comandos uma vez e depois testa a condição.

For contador = valor_inicial **To** valor_final **{ Step** incremento }

bloco de comandos

{ **Exit For }**

bloco de comandos

Next contador

Esta estrutura de controle permite a execução de um bloco de comandos por um número fixo de vezes. O incremento pode ser positivo, negativo e em valores não inteiros. Cuidado para não criar loops infinitos!!!

O *Exit For* permite a finalização da execução do *For Next* a partir daquele ponto. Deve-se ter cuidado para não desestruturar o programa.

GoTo label

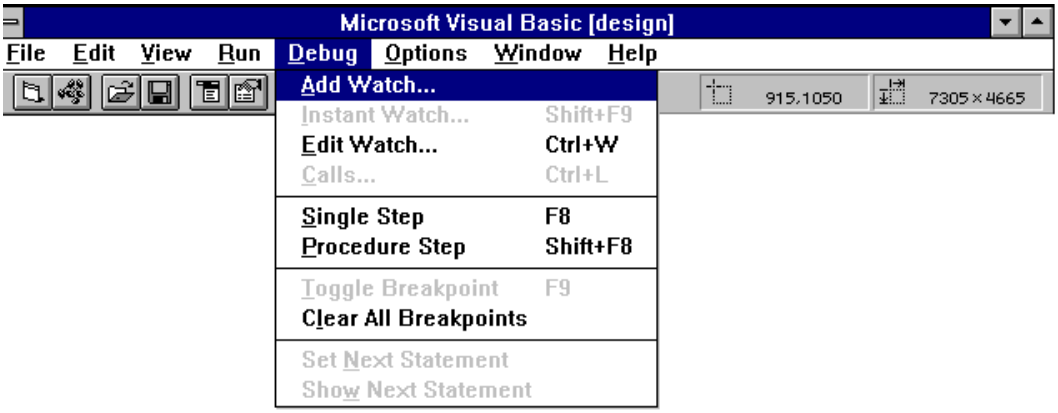
GoTo linha

Este comando faz com que a execução do programa pule para o label ou linha especificada. Deve ser usado basicamente em rotinas de tratamento de erro, pois sua utilização generalizada pode causar desestruturação do programa.

tilizando o Debug do Visual Basic

Objetivo

Neste módulo serão mostradas as ferramentas que o Visual Basic oferece para auxílio na identificação de erros de execução.

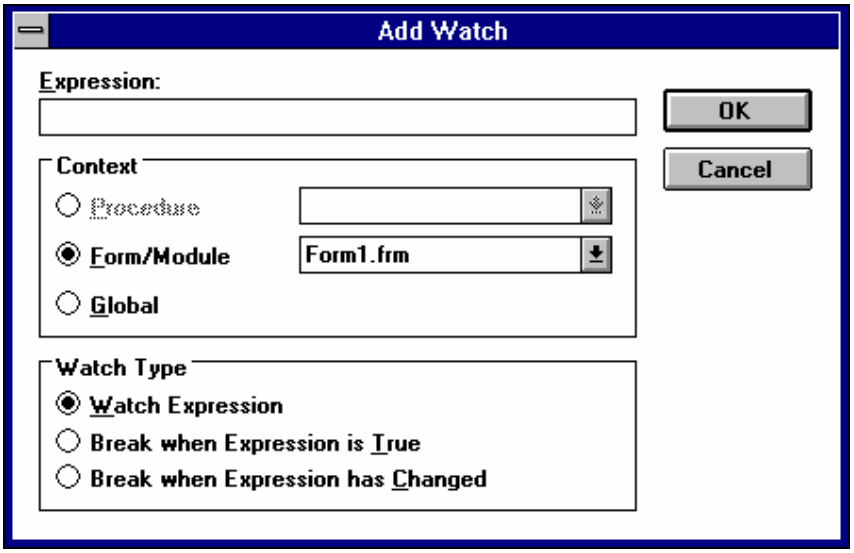

Microsoft Visual Basic [design]
File Edit View Run Debug Options Window Help
915,1050
7305 × 4665
Add Watch...
Instant Watch... Shift+F9
Edit Watch... Ctrl+W
Calls... Ctrl+L
Single Step F8
Procedure Step Shift+F8
Toggle Breakpoint F9
Clear All Breakpoints
Set Next Statement
Show Next Statement

Add Watch e Edit Watch

Permite a criação e a alteração de variáveis de watch. Estas variáveis exibem seu valor sempre que o programa entrar em break mode.

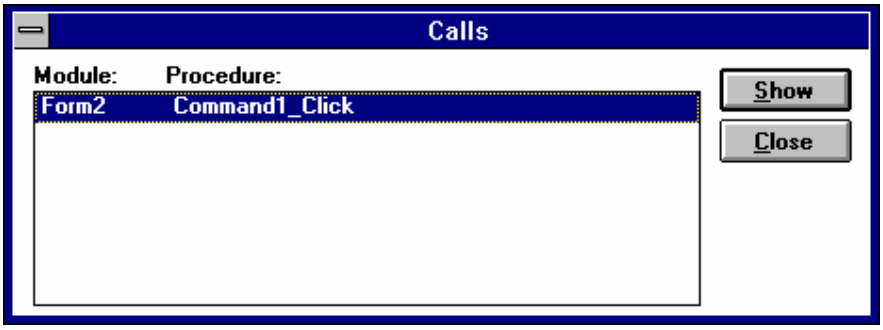
Uma variável de watch pode parar a execução de um programa, se assim for especificado na criação da variável. Existem duas opções : *Break when True,* que pára a execução do programa quando seu valor atingir verdadeiro, ou, *Break when Changed,* que para a execução do programa sempre que seu valor for alterado.

Sempre que um programa estiver em *Break Mode*, para voltar a execução do programa, basta escolher a opção *Start* do menu *Run* ou então apertar F5.



Calls - Árvore de Chamada

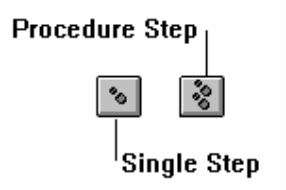
Disponível apenas em tempo de execução, estando o programa em Break Mode. Mostra a árvore de chamada naquele ponto, especificando quais as subrotinas que antecederam a atual em formato de pilha.



Single Step e Procedure Step

Permite a execução do programa comando a comando sem entrar em subrotinas (Single Step), ou comando a comando entrando em rotinas (Procedure Step). Para prosseguir com execução normal da aplicação, selecionar a opção *Start* do menu *Run* ou então pressionar F5.

As opções Single Step /Procedure Step estão no menu Debug, mas também podem ser encontradas no barra de ferramentas.



Show Next Statement

Frequentemente quando se trabalha com o Debug, você se perde no meio de tantas janelas de código. Para isso existe no Visual Basic um comando que posiciona o cursor na próxima linha a ser executada. O comando *Show Next Statement* fica no menu *Debug*.

Set Next Statement

O *Set Next Statement* funciona como um *GoTo* dinâmico. Ele desvia o fluxo do programa para a linha selecionada, e continua a partir dali. O comando S*et Next Statement* também fica no menu *Debug*.

Impressão

Objetivo

Apresentar o meio que o Visual Basic utiliza para fazer impressão em telas ou em impressoras. A impressão no Visual Basic não será muito detalhada, pois dentre os seu utilitários, o Visual Basic possui um gerador de relatórios, o Crystal Report.

O Objeto Printer

Coerentemente ao resto de sua estrutura, para realizar impressões o Visual Basic também utiliza o conceito de objetos. Assim sendo, da mesma forma que existe o objeto *Form*, *Picture*, existe também um objeto *Printer* ao qual aplicamos métodos de impressão.

Métodos de Impressão

Print

É utilizado para imprimir conjuntos de caracteres na impressora, em um formulário ou em um Picture.

Sintaxe : `objeto.Print expressão {,|;}`

, Posiciona o cursor para próxima impressão após 14 espaços.

; Posiciona o cursor para próxima impressão imediatamente após o último caracter impresso.

default Posiciona o cursor para próxima impressão na próxima linha.

PrintForm

Joga para impressora um bitmap do formulário corrente, ou daquele especificado pelo objeto. A saída na impressora, designada no Control Panel do Windows, é de formato gráfico.

Sintaxe : `nome-formulário.PrintForm`

Cls

Limpa o conteúdo de um formulário ou Picture, especificado pelo objeto.

Sintaxe : `Objeto.Cls`

Funções de Impressão

Spc

Pula o número de espaços especificados em uma impressão.

Sintaxe : `Printer.Print "Isto é um teste da "; Spc(10); "função Spc."`

Tab

Posiciona o cursor para próxima impressão no número da coluna especificada. Atenção!!! O Tab não pula um número *x* de colunas, mas se posiciona na coluna *x*.

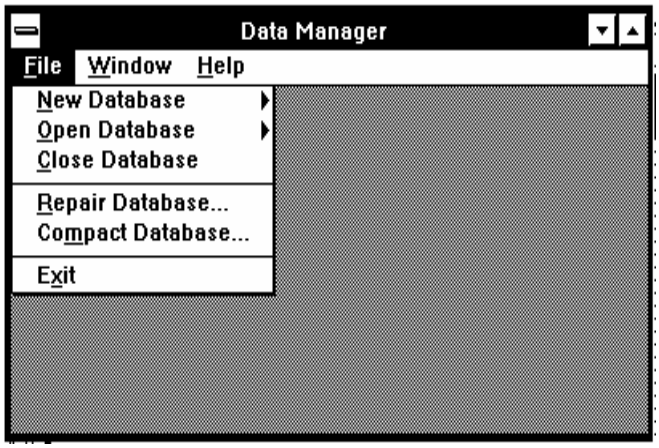
Sintaxe : `Printer.Print tab(10); "10"`

Manipulando Bases da Dados Access através do Visual Basic

Objetivo

Este módulo tem como objetivo mostrar a utilização e manipulação de uma base de dados qualquer de dentro de uma aplicação Visual Basic via ODBC (Open DataBase Connectivity). Deste ponto em diante, você estará realmente capacitado para o desenvolvimento de aplicações comerciais.

Data Manager



O Data Manager é uma ferramenta do Visual Basic que permite a criação de bancos de dado com estrutura Access. Ele permite também a criação de tabelas e a alteração da estrutura de tabelas. Além disso, ele também compacta e descompacta banco de dados, e repara banco de dados corrompidos.

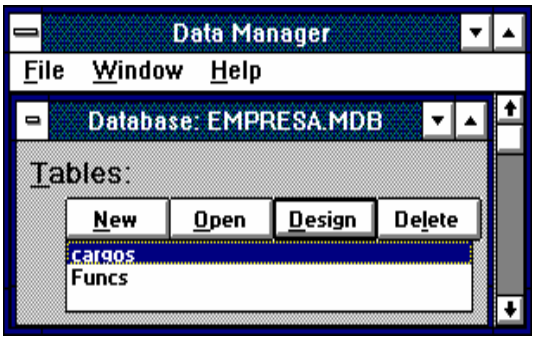
O Data Manager pode ser executado a partir do menu *Window*, selecionando-se a opção *Data Manager*. A seguir uma tabela das funções do Data Manager, por menu, e suas descrições.

Menu	Opção	Descrição
File	New DataBase	Cria um DataBase no formato Access.
	Open DataBase	Abre um DataBase em um dos seguintes formatos : Access, Paradox, DBase, BTrieve e FoxPro.
	Close DataBase	Fecha um banco de dados.
	Repair Database	Tenta recuperar banco de dados corrompido.
	Compact Database	Compacta/Descompacta DataBases.
	Exit	Sai do Data Manager.
Windo w	Cascade	Rearruma telas abertas.
Help	Contents	Aciona o Help do Data Manager.

ma vez com o um banco de dados aberto, você pode: ver os dados de qualquer tabela dentro dele, criar novas tabelas, adicionar campos a tabelas já existentes, alterar características de cada campo, criar índices e alterar características de índices já existentes.

Exemplo - Utilização do Data Manager

Para exemplificar a criação de uma base de dados através do Data Manager, criaremos uma base que serviria como apoio a um sistema de pessoal de uma empresa.



1. Inicialize o Visual Basic
2. Do menu *Window,* selecione a opção *Data Manager*.

Criando um banco de dados

3. Do menu *File*, selecionar a opção *New Database*.
4. Uma Caixa de Diálogo aparecerá. Especificar o nome "*Empresa.MDB"* dentro do diretório "*c:\Curso"*. Pressionar o botão *OK*.

Criando Tabelas e Índices

5. Uma janela, cujo título será o nome do banco de dados criado aparecerá. Nesta janela haverá uma lista de tabelas existentes no banco de dados, no momento, ainda vazia.

6. Pressionar o botão *New* para criar uma nova tabela. Esta tabela deverá ser denominada de *Cargos*.

7. Uma nova janela aparecerá. Esta janela possui duas regiões: a de campos e a de índices. Na parte de campos, pressionar o botão *Add*. Criar os seguintes campos :

Field Name	Field Type	Field Size
cd_cargo	integer	
nm_cargo	Text	20

8. Na parte de índices, pressionar o botão *Add*. Adicionar um índice de nome *ind_cargo* , que terá como chave o campo *cd_cargo* (ascendente). Marcar as opções de *Primary Index* e *Require Unique Index Values*.
9. Repetir os passos de 6 a 8, criando agora uma tabela de funcionários e um índice cuja chave será o código do funcionário. Os campos da tabela serão os seguintes :

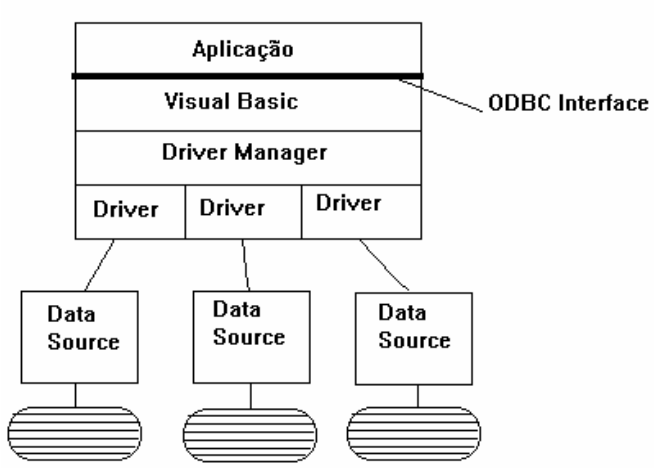
Field Name	Field Type	Field Size
cd_func	integer	
nm_func	Text	20
cd_cargo	integer	
endereco	Text	80

Open Database Connectivity (ODBC)

ODBC é um padrão de arquitetura de sistemas abertos da Microsoft que permite que uma aplicação qualquer em Visual Basic acesse dados armazenados em uma banco de dados, via SQL (Structured Query Language), independentemente do banco de dados utilizado.

Componentes da arquitetura ODBC do Visual Basic

- Aplicação - responsável por toda a definição dos dados e sua manipulação;
- Visual Basic - submete requerimentos de uma aplicação para que sejam processados pelo Gerenciador de Drivers;
- Gerenciador de Drivers - Carrega e direciona o requerimento recebido para o driver apropriado;
- Driver - Processa chamadas a funções ODBC, submete requerimentos SQL para uma fonte de dados e retorna o resultado;
- Fonte de Dados - É a base de dados propriamente dita..



Objetos relacionados a um banco de dados

Coerentemente a toda sua estrutura, o Visual Basic dispõe de um conjunto de objetos, que definidas as suas propriedades e aplicando métodos sobre eles, nos permitirão

manipular e controlar um banco de dados. Estes objetos são : database, dynaset, snapshot, field, fields collection, index, indexes, querydef, snapshot, table, tabledef e tabledefs collection.

Objeto DataBase

O objeto DataBase mantém informações sobre o banco de dados aberto, suas regras de acesso e sua estrutura (a propriedade TableDefs do objeto DataBase possui todas as informações sobre as tabelas de um banco de dados). Possui também funções de controle de transações. O Objeto DataBase é a representação lógica de um banco de dados no Visual Basic.

Assim como qualquer objeto, o *DataBase* possui propriedades e métodos, através dos quais você pode manipular seus dados.

Para abrir uma banco de dados, usa-se a função *OpenDataBase*. A sintaxe desta função depende do banco de dados que estiver sendo utilizado

Sintaxe : (abertura de banco de dados Access)

OpenDatabase (nome_bd [,exclusivo[,readonly [,string de conexão]]])

Exemplo :

```
Dim DB as database
Set DB = Open DataBase("C:\CURSO\EMPRESA.MDB", False, False)
```

Para fechar um banco de dados utiliza-se o método Close.

Exemplo :

```
Dim DB as database
Set DB = Open DataBase("C:\CURSO\EMPRESA.MDB", False, False)
....
DB.Close
```

Dentre suas principais propriedades, podemos ressaltar:

TableDefs Collection Coleção de objetos TableDef, que é correspondente a uma tabela do banco de dados.

TableDefs Collection

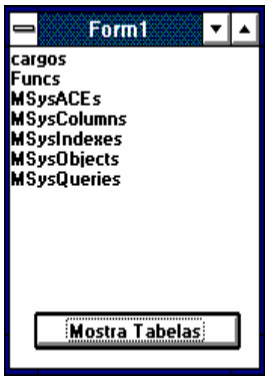
O objeto *TableDefs Collection*, que também é uma propriedade do objeto *DataBase*, é um conjunto de objetos *TableDef* , isto é, o conjunto de todas as tabelas do banco de dados.

Sua propriedade *Count* retorna o número de tabelas do banco de dados.

No exemplo abaixo, um botão foi criado, e ao click do botão, o seguinte código foi anexado. Veja o resultado que este produziu :

```
Sub Command1_Click ()
Dim db As database

    Set db = OpenDatabase("c:\curso\empresa.mdb")
    For i = 0 To db.TableDefs.Count - 1
        Print db.TableDefs(i).Name
    Next i
End Sub
```



O objeto *TableDef* de um banco de dados define a estrutura de uma tabela. O Objeto *TableDef* possui duas propriedades importantes : a *Fields Collection*, que possui a estrutura dos campos da tabela, e a *Indexes Collection*, que possui o conjunto de índices associados àquela tabela.

Se no exemplo anterior tivéssemos colocado os nomes de todas as tabelas de um banco de dados dentro de uma *ListBox* ao invés de ter impresso no próprio

formulário, poderíamos associar o seguinte código ao click da listbox, para ver os campos de uma tabela :

```
Sub List1_Click ()
Dim tabela as tabledef

    Set tabela = db.Tabledefs(list1.listindex)
    For i = 0 To Tabela.Fields.Count - 1
        List2.AddItem Tabela.Fields(i).Name
    Next i
End Sub
```

Table, Dynaset, Snapshot Objects

Tanto a table, como o dynaset, como o snapshot são *recordsets* que representam a tabela do banco de dados. No entanto eles possuem finalidades diferentes.

A **Table** é a representação lógica da tabela física do banco de dados. Através do objeto Table podemos acessar os dados que estão dentro dela, adicionar registros e deletá-los também.

Através do objeto **Dynaset**, podemos ler dados de uma ou mais tabelas, entretanto, estes dados somente poderão ser alterados se o dynaset estiver baseado em uma única tabela.

O **SnapShot** se assemelha a um retrato de uma tabela em um determinado momento. Através dele, podemos ver o conteúdo de uma tabela, porém não é possível fazer qualquer alteração nele, e também quaisquer alterações feitas sobre a tabela não serão refletidas no snapshot.

Abertura de *Tables*

Para a abertura de *tables*, utiliza-se o método OpenTable sobre o objeto database. É necessário criar anteriormente uma variável do tipo *Table*.

Exemplo :

```
Dim DB As Database
Dim Tabela As Table

Set DB = OpenDatabase("BIBLIO.MDB")
Set Tabela = DB.OpenTable("Publishers")
```

Criação de *Dynasets*

Para a abertura de *dynasets*, utiliza-se o método CreateDynaset sobre o objeto database. É necessário criar anteriormente uma variável do tipo *Dynaset*.

A abertura de um dynaset pode ser feita baseando-se em um comando SQL, ou diretamente em cima de uma tabela.

Exemplo :

```
Dim DB As Database
Dim DS As Dynaset

Set DB = OpenDatabase("BIBLIO.MDB")
Set DS = DB.CreateDynaset("Publishers")
```

Exemplo :

```
Dim DB As Database
Dim DS As Dynaset

Set DB = OpenDatabase("BIBLIO.MDB")
Set DS = DB.CreateDynaset("Select * from Publishers")
```

Criação de *SnapShots*

Para a abertura de *snapshot*, utiliza-se o método CreateSnapShot sobre o objeto database. É necessário criar anteriormente uma variável do tipo *SnapShot*.

Assim como *Dynasets*, A abertura de um *snapshot* pode ser feita baseando-se em um comando SQL, ou diretamente em cima de uma tabela.

Exemplo :

```
Dim DB As Database
Dim SS As Snapshot

Set DB = OpenDatabase("BIBLIO.MDB")
Set SS = DB.CreateSnapshot("Publishers")
```

Exemplo :

```
Dim DB As Database
Dim SS As Snapshot

Set DB = OpenDatabase("BIBLIO.MDB")
Set SS = DB.CreateSnapshot("Select * from Publishers")
```

Métodos de Tables, Dynasets e SnapShots

AddNew Limpa o buffer e prepara para a criação de um novo registro na tabela. A criação real do novo registro somente será efetivada pelo método Update (descrito a seguir). O método AddNew só é aplicável a *Tables* e *Dynasets*.

Sintaxe : [recordset].*AddNew*

Close Fecha um *Table, Dynaset* ou *SnapShot*.

Sintaxe : [recordset].*Close*

Delete Apaga o registro corrente da tabela. Método aplicável a Tables e Dynasets (que referenciem uma única tabela).

Sintaxe : [recordset].*Delete*

Edit Abre o registro corrente para alteração. A alteração efetiva somente será feita pelo método Update (descrito a seguir). O método Edit só é aplicável a *Tables* e *Dynasets*.

Sintaxe : [recordset].*Edit*

FindFirst, FindLast, FindNext, FindPrevious Acha o primeiro, último, próximo ou anterior registro que satisfaça a um critério estabelecido. Caso nenhum dos registros do recordset satisfaça à condição, a propriedade *NoMatch* do recordset receberá o valor *True*. Aplicável a Dynasets, Snapshots e Tables.

Sintaxe : [recordset].*FindFirst* Critério
Exemplo :

```
' Cria um Dynaset.
Set DSEditores = DB.CreateDynaset("Editores")
DSEditores.FindFirst "Estado = 'RJ'"
If Not DSEditores.NoMatch Then... '
```

MoveFirst, MoveLast, MoveNext, MovePrevious Faz com que o primeiro, último, próximo ou anterior registro seja o registro corrente. Caso você esteja tentando mover para antes do primeiro registro ou para depois do último registro, a propriedade BOF ou EOF do recordset receberá valor True. Aplicável a Dynasets, Snapshots e Tables.

Sintaxe : [recordset].*MoveFirst*

Seek Procura um registro em uma tabela indexada. Somente aplicável a tables.

Sintaxe : [Table]. seek operador, chave1, chave2...

Exemplo :

```
            Dim DB As Database, Tabela As Table
            Set DB = OpenDatabase("BIBLIO.MDB")
                  Set Tabela = DB.OpenTable("Publishers")
                        Tabela.Index  =  "Chave_Primária  'Define
índice corrente.
            Tabela.Seek "=", 3            'Procura registro.
         If Tabela.NoMatch Then...
```

Controle de Transações

Quando existe um grupo de transações que devem ser efetuadas obrigatoriamente em conjunto, ou todas ou nenhuma, deve-se definir uma transação. Transações são definidas através de métodos em cima do banco de dados. Eles são :

- BeginTrans - define o início de uma transação.

  Ex.: DB.BeginTrans

- CommitTrans - define o fim *com sucesso* de uma transação. Todas as ações serão realmente efetuadas no banco de dados.

  Ex.: DB.CommitTrans

- RollBack - define o fim *sem sucesso* de uma transação. Nenhuma ação será efetuada no banco de dados.

  Ex.: DB.RollBack

Obs. : O método *CommitTrans* sobre o objetos *DataBase* não dispensa o uso do método *Update* sobre qualquer *dynaset* ou *table* que tenha sido atualizada.

Passagem de Comandos SQL direto ao Banco de Dados

O Visual Basic permite a passagem de comandos diretamente para o banco de dados. Um comando deve ser passado por vez e deve ser uma *action query*, ou seja, uma query que inclua, altere ou exclua registros. Múltiplos registros podem ser afetados pela *action query* de uma única vez.

Para a execução de comandos SQL, o Visual Basic possui dois métodos :

- ExecuteSQL

  Exemplo :

```
Dim DB as database
Dim linhas as long
linhas = DB.ExecuteSQL "Delete * from funcs where cidade =
           'São Paulo'"
```

- Execute

  Exemplo :

```
Dim DB as database
DB.Execute "Delete * from funcs where cidade = São Paulo'"
```

A diferença entre Execute e ExecuteSQL é que :

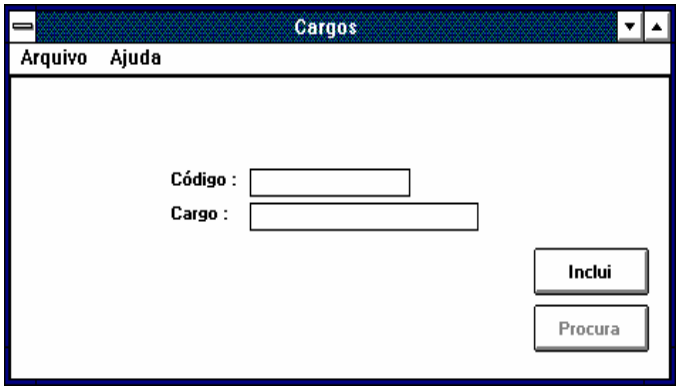
1. O *ExecuteSQL* somente pode ser executado sobre bases de dados remotas como *Oracle* ou *SQL Server*.

2. O *Execute* permite a colocação de opções sobre o funcionamento da *action query*, por exemplo, se a tabela deve ser "*Locked*" para outros usuários enquanto a query estiver sendo executada.

Exemplo - Trabalhando com Dynasets

Para exemplificar a utilização de Dynasets, você irá criar o começo de uma aplicação para preencher o banco de dados *empresa.mdb*, criado por você anteriormente.

A aplicação será composta por um menu, que permite acessar uma tela de cargos e outra de funcionários. Na tela de cargos, inicialmente você poderá incluir um cargo novo, ou procurar por um já existente. Caso você escolha por uma procura, a aplicação disponibilizará o registro procurado para alteração ou exclusão.



1. Inicialize o Visual Basic
2. Como aprendido anteriormente, montar um menu com a seguinte estrutura :

Name	Caption	Identação
MNUArquivo	Arquivo	0
MNUCargos	Cargos	1
MNUFuncs	Funcionários	1
MNUSeparador	-	1
MNUSair	Sair	1
MNUAjuda	Ajuda	0
MNUSobre	Sobre	1

Criando o Picture Box

3. Crie um Picture Box do tamanho do formulário.
4. Pressione F4 para ter acesso a sua tela de propriedades.
5. Nomeie o Picture Box PICCargos.
6. Atribua à sua propriedade Visible, o valor *False*.

Criando os controles dentro do Picture Box

7. Crie dois labels, cujos *caption* deverão ser "Código :" e "Cargo :".
8. Crie dois Text Boxes, de *Name* TXBCodigo e TXBCargo, e propriedade *Text* = "".
9. Crie cinco Botões, com as seguintes propriedades :

Caption	Name	Visible	Enabled

Incluir	CMDIncluir	True	True
Procurar	CMDProcurar	True	False
Alterar	CMDAlterar	False	True
Excluir	CMDExcluir	False	True
Cancelar	CMDCancelar	False	True

Obs.: Os botões devem estar nas seguintes posições:

- *Alterar*, logo abaixo *Incluir*, e mais abaixo, *Procurar*. Sobre o botão de *Incluir*, colocar o botão de *Excluir*. Sobre o botão de *Procurar*, colocar o botão de *Cancelar*.

Criando variáveis de nível de formulário

10. Criar as seguintes variáveis no General Declarations do BAS:

- DB , tipo database
- DS, tipo Dynaset

Inicializando as variáveis a nível de formulário :

11. Digitar no Form1_Load as seguintes linhas de código :

```
set db = OpenDatabase ("c:\cursos\empresa.mdb")
set ds = db.CreateDynaset("cargos")
```

Programando os menus

12. No Click de MNUCargos, digitar as seguintes linhas de código :

```
Set ds = db.CreateDynaset("Cargos")
```

13. No Click de MNUSair, digitar as seguintes linhas de código :

```
ds.close
db.close
End
```

Programando os botões

BOTÃO *INCLUIR*

14. Digitar as seguintes linhas de código :

```
If Trim(TXBCodigo.Text) <> "" And Trim(TXBCargo.Text) <> "" Then
    ds.AddNew
    ds("cd_cargo") = TXBCodigo.Text
    ds("nm_cargo") = TXBCargo.Text
    ds.Update
End If
TXBCodigo.Text = ""
TXBCargo.Text = ""
```

BOTÃO *PROCURAR*

15. Digitar as seguintes linhas de código :

```
Dim Criterio as string
criterio = "cd_cargo = " + TXBCodigo.Text + " or nm_cargo = '" +
             TXBCargo.Text + "'"
ds.FindFirst criterio

If Not ds.NoMatch Then
    TXBCodigo.Text = ds("cd_cargo")
    TXBCargo.Text = ds("nm_cargo")
    CMDExcluir.Visible = True
    CMDAlterar.Visible = True
    CMDCancelar.Visible = True
    CMDIncluir.Visible = False
    CMDProcurar.Visible = False
Else
    MsgBox "Registro Não Encontrado"
End If
```

BOTÃO *ALTERAR*

16. Digitar as seguintes linhas de código :

```
Dim Resp as integer
resp = MsgBox("Comfirma Alteração ?", 52)
If resp = 6 Then
    ds.Edit
    ds("cd_cargo") = TXBCodigo.Text
    ds("nm_cargo") = TXBCargo.Text
    ds.Update
End If
TXBCodigo.Text = ""
TXBCargo.Text = ""
```

```
    CMDExcluir.Visible = False
    CMDAlterar.Visible = False
    CMDCancelar.Visible = False
    CMDIncluir.Visible = True
    CMDProcurar.Visible = False
```

BOTÃO *EXCLUIR*

17. Digitar as seguintes linhas de código :

```
Dim resp as integer
    resp = MsgBox("Confirma Exclusão ?", 52)
    If resp = 6 Then
        ds.Delete
    End If
    TXBCodigo.Text = ""
    TXBCargo.Text = ""
    CMDExcluir.Visible = False
    CMDAlterar.Visible = False
    CMDCancelar.Visible = False
    CMDIncluir.Visible = True
    CMDProcurar.Visible = False
```

BOTÃO *CANCELAR*

18. Digitar as seguintes linhas de código :

```
    TXBCodigo.Text = ""
    TXBCargo.Text = ""
    CMDExcluir.Visible = False
    CMDAlterar.Visible = False
    CMDCancelar.Visible = False
    CMDIncluir.Visible = True
    CMDProcurar.Visible = False
```

Programando o Change da TextBox

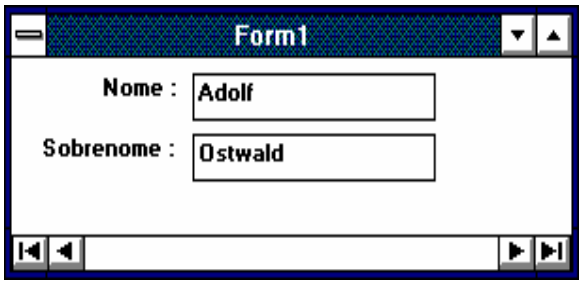
19. Digitar as seguintes linhas de código no Change de TXBCodigo:

```
If Trim(TXBCodigo) <> "" Then CMDProcurar.enabled = true
```

Executando a aplicação

20. Incluir, excluir e alterar cargos. Tente agora reescrever o código para inclusão, alteração e exclusão de funcionários.

Data Control



Este controle permite o acesso a dados armazenados em bancos de dados, movendo registro a registro e mostrando os dados de cada registro em controles do formulário vinculados ao banco de dados. A única forma de vincular controles do formulário, como Text Boxes e Combo Boxes, a um banco de dados é através do Data Control.

Usando o Data Control, não é necessário escrever quase nenhum código de programa para executar a maioria das operações de acesso a um banco de dados. Controles do formulário vinculados ao banco de dados, automaticamente mostram os dados do registro corrente. Quando alterações são efetuadas, eles também são automaticamente gravadas no banco de dados, ao se passar para o próximo registro.

Propriedades	Descrição
Connect	Determina a informação utilizada para abrir um Banco de Dados externo (ODBC ou Access remoto). Quando o BD for Access local, deixar em branco.
DataBase	Referencia o BD vinculado ao Data Control. Somente disponível em RunTime.
DataBaseName	Determina o nome do banco de dados ao qual o Data Control estará vinculado.
Exclusive	Determina se o BD estará aberto de modo exclusivo ou não.
Name	Nome interno do Data Control.
Options	Determina as características do dynaset.
ReadOnly	Determina se o BD estará aberto de modo apenas para leitura ou não.
RecordSet	É o dynaset definido pelos dados Connect, DataBaseName e RecordSource. Utilizando-se esta propriedade, pode-se movimentar pelo dynaset via programação.
RecordSource	Define a tabela ou expressão SQL que será fonte do Data Control.

Evento Reposition

Ocorre sempre depois que um registro se torne o registro corrente. Preenchimento de campos com dados de outras tabelas a partir de códigos contidos no registro deve ser feita neste momento.

Sintaxe :

```
Nome-do-DataControl_Reposition ( Index as Integer)
```

onde,

Index é o índice do controle, caso exista Control Array, que está causando o evento Reposition.

Evento Validate

Ocorre sempre antes que outro registro se torne o registro corrente. Qualquer validação de dados deve ser feita neste momento, pois pode-se cancelar a alteração do registro corrente.

Sintaxe :

```
Nome-do-DataControl_Validate ( Index as Integer, Action as
Integer,                                        Save as Integer)
```

onde,

Index é o índice do controle, caso exista Control Array, que está causando o evento Validate.

Action é o tipo de ação que está causando a mudança do registro corrente. Se, durante o evento Validate, você quiser cancelar a ação, à propriedade *Action* deve ser atribuído o valor zero (0).

Save é o parâmetro que indica se os controles vinculados foram alterados ou não. *Save* tem valor True quando qualquer um dos controles vinculados tem propriedade *DataChanged* com valor True.

Principais Métodos

Refresh Força uma atualização imediata do controle. Usado no Data Control, pode reabrir um Database, se uma das seguintes propriedades tiver sido alterada : DataBaseName, ReadOnly, Exclusive ou Connect. Também pode reconstruir um Dynaset se a propriedade RecordSet tiver sido alterada.

Exemplo :

```
Data1.Exclusive = True
Data1.Refresh
```

UpdateControls Pega os campos do registro corrente e os recoloca nos controles do formulário vinculados àquela RecordSet. Este método é bastante útil quando o usuário desiste de fazer uma alteração nos campos vinculados, porém não possui mais os valores antigos.

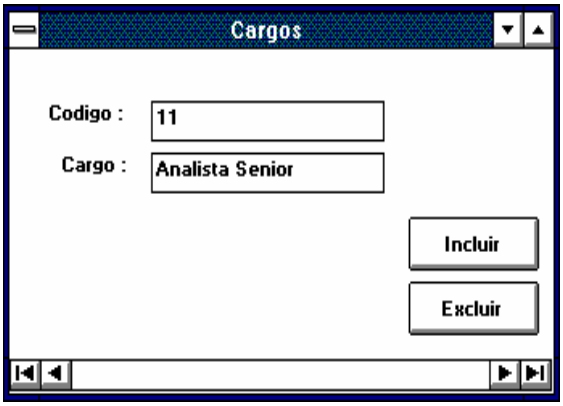
Exemplo :

```
Sub Form_KeyDown (KeyCode As Integer, Shift As Integer)
   Const KEY_ESCAPE = &H1B
   If KeyCode = KEY_ESCAPE Then
      Data1.UpdateControls
   End If
End Sub
```

UpdateRecord Este método deve ser utilizado dentro de procedimento de evento *Validate*, quando não se deseja que o evento *Validate* aconteça novamente.

Exemplo - Utilizando o Data Control

Para exemplificar o uso do Data Control, construiremos a mesma aplicação de cadastramento de cargos, feita anteriormente, de modo que você possa comparar as dificuldades, vantagens e desvantagens de usar uma método ou outro.



1. Inicializar o Visual Basic.

Criando Controles

2. Criar o Data Control. Atribuir os seguintes valores às suas propriedades :

Propriedade	**Valor**
Name	Data1
Caption	""
DatabaseName	"c:\curso\empresa.MDB"
RecordSource	"Cargos"

3. Criar dois labels, cujas propriedades *Caption* terão, respectivamente, os valores "Código" e "Cargo".
4. Criar dois TextBoxes. Atribuir os seguintes valores às suas propriedades :

	Text1	**Text2**
Name	TXBCodigo	TXBCargo
Text	""	""
DataSource	Data1	Data1
DataField	cd_cargo	nm_cargo

5. Criar dois CommandButtons. Nomeie-os CMDInclui e CMDExcluir, respectivamente. Atribua a suas propriedades Caption os valores "Inclui" e "Excluir", respectivamente.

Codificando os Eventos do Controles

6. No click botão CMDInclui, digite as seguintes linhas de código :

```
data1.recordset.addnew
```

7. No click botão CMDExclui, digite as seguintes linhas de código :

```
data1.recordset.delete
data1.recordset.movefirst
```

8. No evento Validate do Data Control, digite as seguintes linhas de código :

```
Dim resp as integer
   If action = 7 then
         resp = MsgBox("Confirma Exclusão ?", 4)
         if resp = 7 Then
               action = 0
         End If
   End If
```

Executando a Aplicação

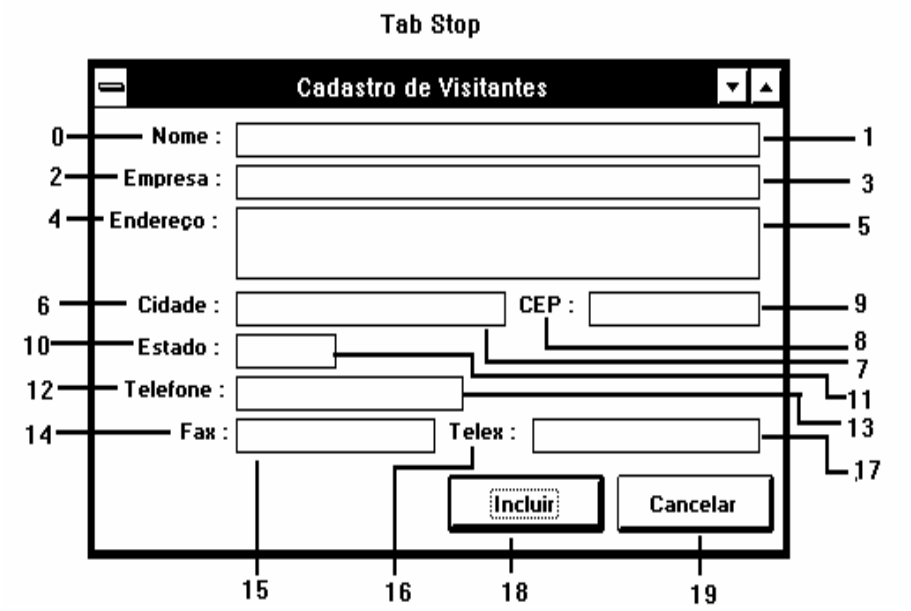
9. Inclua e exclua registros. Tente agora fazer o mesmo para Funcionários.

Entrada de Dados

Objetivo

Apresentar algumas técnicas para facilitar a entrada de dados para o usuário, além de introduzir conceitos de validação de dados, logo na entrada destes no sistema.

Controlando a Tabulação

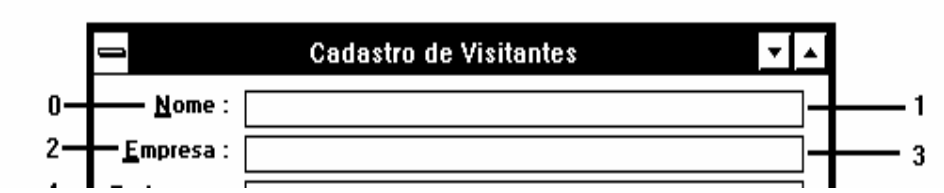


A ordem de tabulação entre os campos de um formulário é designada pela propriedade *Tab Index* dos controles do formulário. Esta propriedade vai sendo atribuída à medida que os controles vão sendo criados dentro do formulário. Portanto, a ordem de tabulação default é a ordem de criação dos controles. No entanto, para alterar esta ordem, basta alterar a propriedade *Tab Index* dos controles.

Tab Stop

Quando esta propriedade estiver com o valor *false*, o cursor não parará aqui quando o usuário estiver usando o Tab para se movimentar pelo formulário.

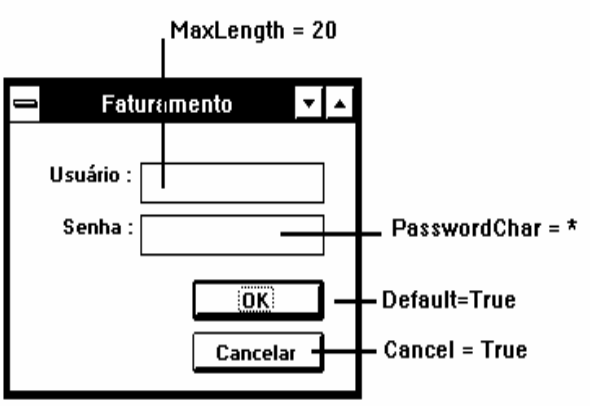
Designando Teclas de Acesso



Assim como podemos usar o & (E comercial) para designar teclas de acesso a menus e botões, podemos também utilizá-lo para acessar caixas de texto. Para isso, basta inserir um label na frente (na ordem de tabulação) de um TextBox, e utilizar o & para designar uma tecla de acesso para o label.

Como labels não recebem foco, o foco irá para o controle que tenha o TabIndex imediatamente maior, portanto o Text Box receberá o foco.

Propriedades que Restringem a Entrada de Dados



Maxlength de TextBoxes

Determina o número máximo de caracteres que uma caixa de texto pode conter. Se o usuário digitar mais que o permitido, o sistema emitirá um *beep*, e ignorará tudo que o usuário digitar que ultrapasse o limite.

PassWordChar de TextBoxes

Determina o caracter a ser mostrado na tela, independente do que o usuário estiver digitando. Normalmente é utilizado para que senhas e códigos secretos não

apareçam na tela. Normalmente o caracter utilizado em aplicações Windows é o asterísco (*).

Default de Command Buttons

Determina que, se o usuário pressionar a tecla de Enter, de qualquer ponto da tela, o evento Click do botão Default ocorrerá. Existe no máximo um botão default por tela.

Cancel de Command Buttons

Determina que se o usuário pressionar a tecla de Esc, de qualquer ponto da tela, o evento Click do botão, cuja propriedade Cancel estiver com valor True, ocorrerá. Existe no máximo um botão cancel por tela.

Key Press para restringir valores

O evento KeyPress ocorre sempre que o o usuário pressionar uma tecla que faça parte do padrão ASCII. Portanto podemos utilizar este evento para restringir e transformar as teclas digitadas pelo usuário.

Por exemplo, podemos impedir que o usuário digite qualquer coisa que não seja números em uma caixa de texto. Verifique o código a seguir :

```
Sub Text1_KeyPress (KeyAscII As Integer)
   If KeyAscii <48 or KeyAscII > 57 then
      KeyAscii = 0
   End If
End Sub
```

KeyUp e Key Down para restringir teclas

Ao contrário do KeyPress, os eventos KeyUp e KeyDown mapeiam qualquer tecla pressionada pelo usuário. Cada tecla é unicamente identificada por um código que pode ser encontrado no arquivo CONSTANT.TXT. Desta forma, pode-se mapear teclas de função (F1, F2, ...), o Delete, etc..

Os três eventos de tratamento de teclas, ocorrem na seguinte ordem :

1. KeyDown
2. KeyUp
3. KeyPress

Se o valor 0 (Null) for atribuído a KeyCode durante as *event-procedures* KeyDown e KeyUp, o evento KeyPress não acontece.

KeyPreview

KeyPreview é uma propriedade do formulário que, se tiver valor True (o default é False), dará ao formulário a primazia no tratamento de teclas em relação aos controles. Ou seja, ao pressionar qualquer tecla em um TextBox, ocorrerá no formulário os eventos KeyPress, KeyUp e KeyDown primeiro.

Outras Funções de Tratamento de String

Além das funções de tratamneo de string já apresentadas ao longo do curso, o Visual Basic possui outras bastante úteis. Eis mais algumas :

Função	Descrição
InStr	Retorna a posição onde um conjunto de caracteres começa dentro de uma string. Ex.: Pos = InStr ("flor", "reflorestamento"). O exemplo retorna valor 3.
String	Repete um caracter especificado por *n* vezes. Ex.: texto = string(5,"*"). O exemplo retorna "*****".

Foco

Em Windows, apenas um controle, formulário ou janela pode ter o foco de cada vez. Ter o foco significa que qualquer ação do usuário recai sobre aquele elemento, seja ele um controle ou uma janela. O foco pode ser transferido pelo usuário ou pela aplicação.

O Visual Basic possui alguns eventos e métodos para o tratamento de foco. A seguir detalharemos cada um deles.

EVENTOS

GotFocus Acontece sempre que um controle recebe o foco, ou porque o usuário pressionou Tab, ou porque ele clicou sobre o controle, ou através de aplicação. Note que o Formulário somente recebe o evento GotFocus quando não há nenhum controle visível dentro dele.

Lost Focus Acontece em um controle sempre que um outro controle recebe o foco, ou porque o usuário pressionou Tab, ou porque ele clicou sobre o outro controle, ou através de aplicação.

Activate Funciona como o GotFocus, mas para formulários. O Activate ocorre a um formulário sempre que este passa a ser o formulário ativo (com foco).

Deactivate Funciona como o LostFocus, mas para formulários. O Deactivate ocorre a um formulário sempre que este deixa de ser o formulário ativo.

MÉTODOS

SetFocus Este método, aplicado a qualquer controle, transforma aquele controle no controle ativo. O *SetFocus* em um controle, provoca o *LostFocus* do controle que possuía o foco anteriormente e o *GotFocus* do controle ao qual foi aplicado o método.

Sintaxe : controle.*Setfocus*

Cuidado !!!! Não utilize nunca o método *SetFocus* dentro de um evento *LostFocus*. Dependendo da lógica de programação, isto pode causar loop infinito !!

Obs.: No evento *Click* de botões cujas propriedades *Default* ou *Cancel* tenham valor *true*, deve existir um comado transferindo o foco para o próprio botão (nome_botão.setfocus), porque se o click do botão for ativado porque o usuário

pressionou ENTER ou CANCEL, o *LostFocus* do controle onde o foco estava anteriormente não ocorre.

O Evento Query Unload

O Evento QueryUnload de um formulário permite que você consiga detectar como o Unload do formulário foi iniciado. Este evento possui dois parâmetros : o *UnloadMode* e o *Cancel*. O UnloadMode indica como o Unload foi iniciado. O Cancel, que inicialmente possui valor *False*, se receber o valor True impede que o Unload do Formulário continue.

UnloadMode	Significado
0	O usuário iniciou o Unload a partir do Control Menu do formulário.
1	Pelo comando Unload, utilizado no código.
2	O Usuário está saindo do Windows.
3	O usuário selecionou Finalizar Tarefa a partir do Gerenciador de Tarefas.
4	Um formulário MDI-filho está sendo fechado porque o MDI-pai está sendo fechado.

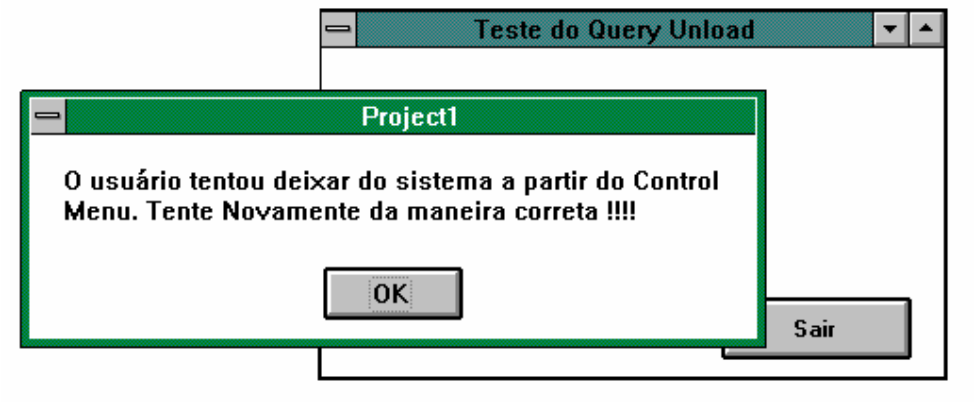
O evento QueryUnload é de grande importância para o programador de sistemas, pois impede o usuário de sair do sistema de uma forma incorreta, deixando cálculos incompletos, conexões com banco de dados pendentes, etc..

Exemplo :

```
Sub Form1_QueryUnload ( Cancel as Integer, UnloadMode as Integer )
   If UnloadMode <> 1 Then
      MsgBox "Utilize  o botão 'Sair' para finalizar a tarefa"
      Cancel = true
```

Exemplo - Testando o evento Query Unload

Para exemplificar a utilização do evento Query Unload, criaremos uma aplicação onde somente será possível finalizar a aplicação através do comando *End* a partir do código.



1. Inicializar o Visual Basic.
2. Criar um Botão em um formulário. Nomeá-lo CMD_Sair.
3. Associar ao CMD_Sair o comando *End*
4. Associar ao evento Query Unload do form o seguinte código :

```
 Select Case unloadmode
     Case 0
         MsgBox "O usuário tentou deixar do sistema a partir do
                    Control Menu. Tente Novamente da maneira
                    correta !!!!"
         cancel = True
     Case 2
         MsgBox "O usuário tentou deixar o Windows antes de sair
do                      sistema.  Tente  Novamente  da  maneira
correta                 !!!!"
         cancel = True
     Case 3
         MsgBox  "O usuário tentou deixar o sistema a partir da
                    Lista de Tarefas. Tente Novamente da maneira
                    correta !!!!"
         cancel = True
 End Select
```

Executando a Aplicação

3. Criar um executável, colocá-lo em uma janela de programas do Windows, e executr o programa. Tentar fechar o programa de todas as maneiras.