celta informática

Visual Basic 6

Sumário

PRIMEIROS PASSOS _____ 3

INÍCIO _____ 4

- BARRA DE MENU E FERRAMENTAS _____ 5
- ANATOMIA DE UMA JANELA _____ 10
- MEU PRIMEIRO PROGRAMA _____ 11
- Desenhar as janelas que se deseja usar _____ 11
- Adaptar as propriedades dos objetos _____ 12
- Escrever o código para os eventos associados _____ 16

EXEMPLO I - CALCULADORA _____ 22

- VARIÁVEIS _____ 26

EXEMPLO II - JOGO DA VELHA _____ 33

- DEPURAÇÃO _____ 39
- QUADROS DE MENSAGEM _____ 40

EXEMPLO III - BLOCO DE NOTAS _____ 43

- CRIANDO MENUS _____ 46
- SALVANDO E ABRINDO ARQUIVOS _____ 56

MÉTODOS GRÁFICOS _____ 63

- DESENHO DE PONTO _____ 63
- CORES _____ 66
- LINHA _____ 67

Círculos ____ 72
Carregando Figuras ____ 75

EXEMPLO IV - JOGO DA FORCA ____ 78
Evento do Teclado ____ 82
Eventos do Mouse ____ 83

EXEMPLO V - CATÁLOGO ____ 85
Variáveis Compostas e Array ____ 87
Caixas de Diálogo Padrão ____ 93
Comando de Impressão ____ 98

EXEMPLO VI - BANCO DE DADOS ____ 100
Controle Data ____ 100

LISTA DE EXERCÍCIOS ____ 109

MICROSOFT VISUAL BASIC 6.0

PRIMEIROS PASSOS

Vantagens: - Facilidade em alterações e implementações
- Melhor Estruturação do código

Linguagens: - Turbo Pascal, Quick Pascal, Turbo C++ e C/C++ (Baseadas em Objetos)

- Visual Basic (Orientada a Componentes - COM)

O Visual Basic é um padrão de linguagem de programação baseada em objetos que qualquer pessoa pode aprender e utilizar, desenvolvendo rapidamente aplicativos visuais com grande velocidade. Ele juntamente com o Delphi formam o que se chama de ferramenta RAD (rapid application development) que permitem desenvolver aplicativos visuais rapidamente com o uso de componentes gráficos pré-definidos.

Permite o uso de objetos e a criação de componentes, tendo a limitação de não podermos criar objetos a partir do VB.

Ele trabalha com eventos que dão início à alguma rotina de trabalho, ou seja, o programa fica parado até que um evento ocorra.

Um programa tradicional, feito para ser executado em DOS, é organizado em torno de estruturas de dados, com procedimentos e funções separadas para manipular os dados.

Um programa orientado a objetos e eventos é organizado em torno de um conjunto de objetos, que são estruturas combinando dados e rotinas em uma mesma entidade. Um Objeto possui dados internos, que não podem ser acessados de fora dele e dados externos, também chamados de **propriedades**, que podem ser acessados de fora deste objeto. De maneira semelhante, ele possui rotinas internas que são usadas apenas internamente e rotinas externas, também chamadas de **métodos**, que podem ser acessadas externamente.

Um método é uma rotina própria do objeto que o dá funcionalidade, isto é, torna-o "vivo", e as propriedades fazem o intercâmbio entre o objeto e o programa.

INÍCIO

Quando iniciamos o Visual Basic 6.0, ele nos apresenta a janela New Project, onde escolhemos qual o tipo de aplicativo iremos desenvolver.

Standard EXE - Aplicação Windows padrão.

ActiveX EXE - Biblioteca de objetos que funcionam fora da aplicação.

ActiveX DLL - Biblioteca de objetos que funcionam dentro da aplicação.

ActiveX Control - Cria extensões da caixa de ferramentas na própria linguagem.

VB Application Wizard - Assistente na criação de aplicações.

VB Wizard Manager - Ajuda na criação de custom wizards.

Data Project - Cria um projeto de banco de dados.

IIS Aplication - Cria um aplicativo para servidor de informações para internet.

Addin - Criação de Add-Ins, adiciona funções ao ambiente de desenvolvimento.

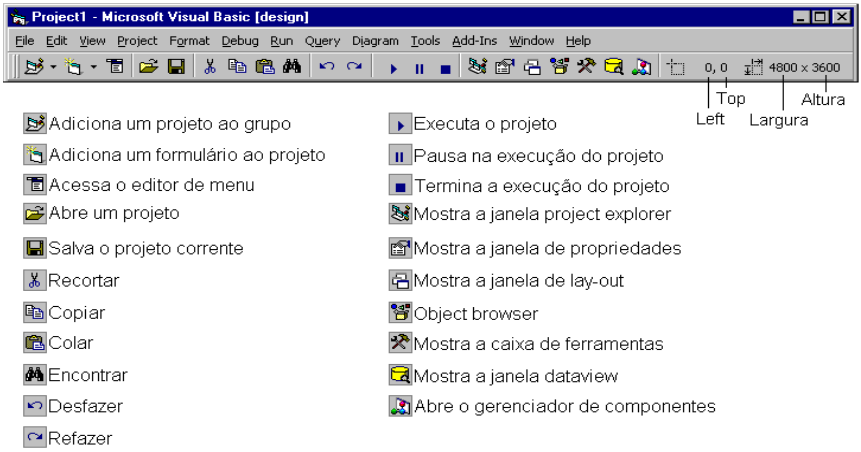
ActiveX Document DLL - Objetos para aplicativos Internet/Intranets.

ActiveX Document EXE - Aplicação que requer um browser (Internet/Intranets) para operar.

DHTML Application - Cria uma aplicacão DHTML composta por um projeto ActiveX DLL que automaticamente seleciona as referências necessárias para desenhar uma página HTML dinâmica.

VB Enterprise Edition Controls - Aplicação Windows padrão, com recursos adicionais.

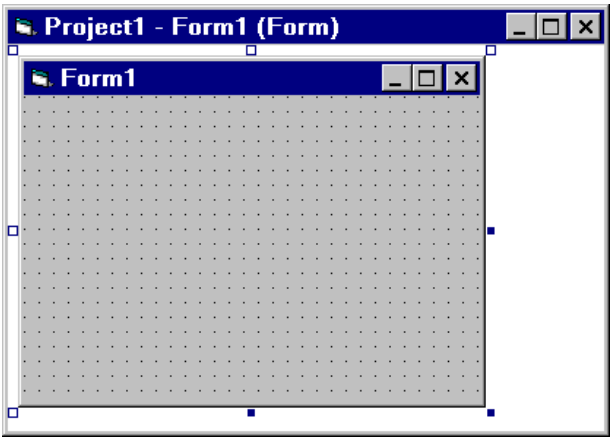
BARRA DE MENU E FERRAMENTAS



Janelas do Visual Basic

Formulário

Entre os vários tipos de objetos no VB, os formulários e controles são os mais comuns. O Formulário é a janela que aparece no centro da tela do Visual Basic, formando a unidade básica de um aplicativo, onde o usuário interagirá enquanto trabalha com o aplicativo desenvolvido.



Toolbox - Controles

Controles são todos os objetos que podemos trabalhar, inserindo-os em um formulário e/ou controlando os seus métodos e propriedades. Um controle é qualquer objeto que o usuário possa manipular, desde que não seja uma janela (formulário).

A **Caixa de Ferramentas** (Toolbox), possui todos os controles que iremos precisar para desenharmos nossa janela - formulário - como um programa de desenho livre. Para incluir um controle ao formulário, existem dois métodos:

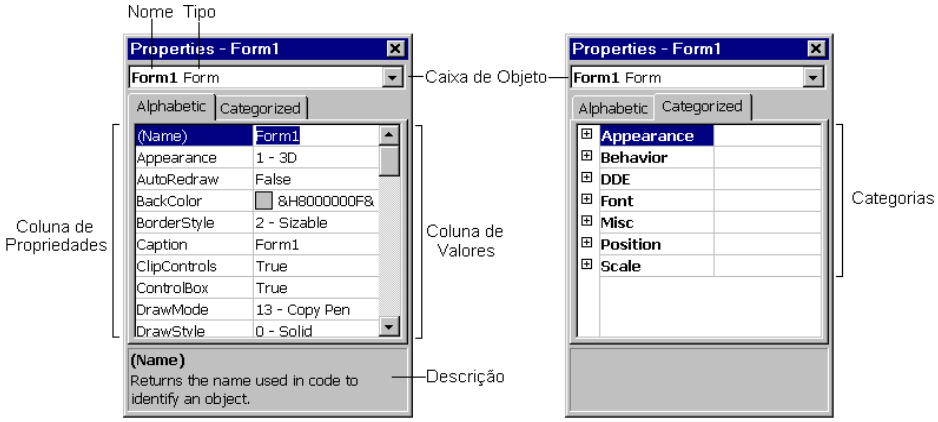
1. Click Duplo no ícone do controle, na caixa de ferramentas. Que fará com que o controle seja inserido no centro do formulário com um tamanho padrão.
2. Selecionar o ícone na caixa de ferramentas, e depois dimensioná-lo no formulário, arrastando e soltando o mouse na área do cliente, no formulário.

Podemos dimensionar estes controles depois de inseridos a qualquer momento durante o desenvolvimento. Primeiro, selecionamos o controle dando um clique nele, em seguida, o dimensionamos arrastando um dos oito dimensionadores que circundam este objeto.

Properties - Propriedades

Nesta janela definimos as características de cada objeto do aplicativo (botões de comando, quadros de texto, formulários, e outros), escolhendo como eles serão apresentados. Cada um desses objetos possui um conjunto específico de propriedades que podem ser associadas a eles. Ao trabalharmos com diferentes objetos, a janela de propriedades nos permitirá mudar as propriedades do objeto ou objetos atualmente selecionados.

Existem propriedades que podemos mudar enquanto construímos nosso projeto, ou seja, em tempo de projeto, e outras propriedades que só podemos mudar durante a execução do projeto, neste caso, em tempo de execução.



Na janela de propriedades acima, temos algumas das propriedades do formulário inicial do VB. Ela possui duas guias, uma lista as propriedades por ordem alfabética e a outra por ordem de categorias. Na pasta por categoria observamos um sinal de + , usado para expandir a lista, ou um sinal de - para diminuí-la.

Nome - Contém o nome do objeto atualmente selecionado, este nome está na propriedade Name.

Tipo - Nesta posição encontraremos qual é o tipo do objeto selecionado, se ele é um Form (formulário), Command Button (botão de comando), Label (legenda), ou então um Text Box (quadro de texto).

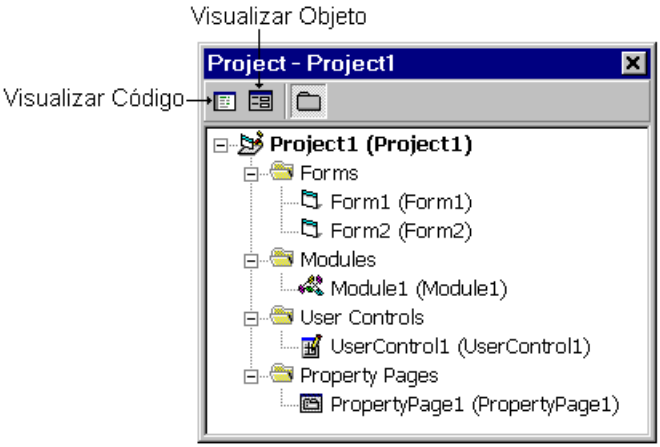
Caixa de Objeto - Esta caixa mostra o objeto atualmente selecionado, através dela também podemos selecionar o objeto que queremos mudar as suas propriedades, basta dar um clique na seta que um menu de cortina se abrirá, onde poderemos selecionar o objeto a trabalhar.

Coluna de Propriedades - Exibe todas as propriedades que podemos modificar em tempo de projeto do objeto selecionado.

Coluna de Valores - Exibe o valor da propriedade correspondente.

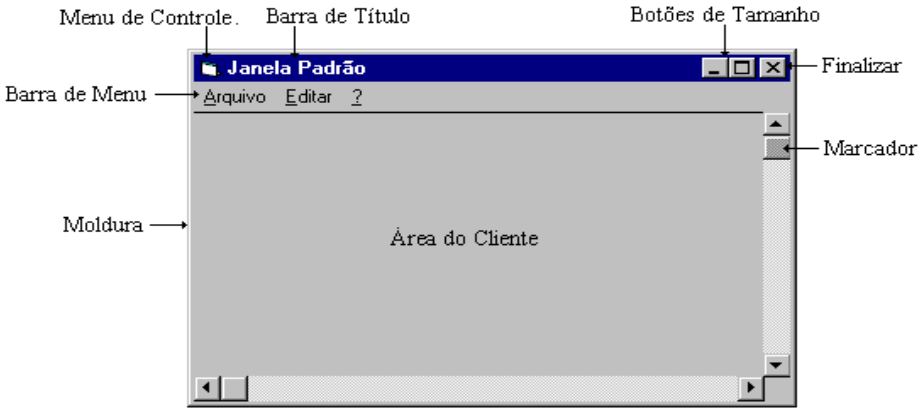
Descrição - Apresenta uma breve descrição da propriedade.

Project Explorer



Esta janela mostra-nos todos os componentes do nosso projeto, de uma forma hierárquica semelhante ao Explorer do Windows, nela podemos gerenciar todos os arquivos do aplicativo e vários projetos simultaneamente, quando trabalhados em grupos.

ANATOMIA DE UMA JANELA



Moldura - Os quatro lados da janela, que definem seu tamanho.

Barra de Título - Abaixo da moldura superior com nome da janela e documento corrente.

Menu de Controle - A esquerda da Barra de Título. Um botão com um ícone que representa o programa.

Botões de Tamanho - A direita da Barra de Título. São dois botões, um com um traço e o outro com duas janelinhas ou uma janela desenhada. Se forem duas janelinhas, mostra que a janela está maximizada e se for uma janela um pouco maior, mostra que a janela está em seu tamanho normal e pode ser maximizada. O botão com um traço serve para minimizar a janela.

Barra de Menu - Está abaixo da barra de título e contém as opções de controle do aplicativo.

Área do Cliente - É a parte interna da janela, também chamada de área do documento. No VB, é o espaço que temos para inserir os controles da nossa aplicação.

Marcador - botão deslizante para rolar a tela.

Janela - Uma Janela é plena quando podemos dimensioná-la (mini, maxi e restaurá-la) e movê-la.

MEU PRIMEIRO PROGRAMA

Para iniciar, vamos construir um programa que, quando for dado um clique num botão, será mostrada uma mensagem.

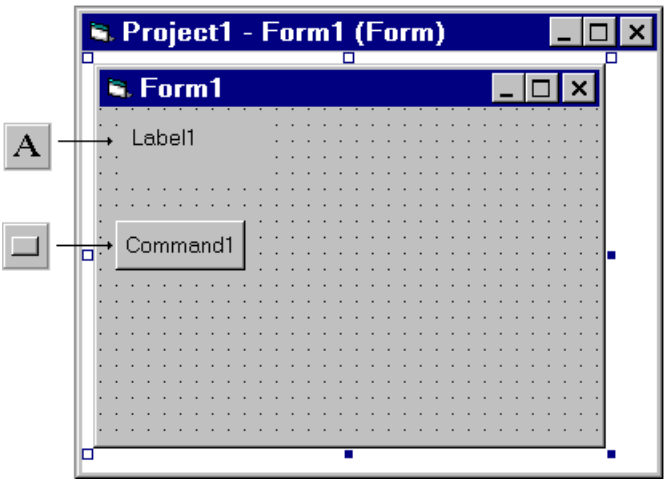
Existem três passos principais, para a criação de uma aplicação no Visual Basic, que iremos seguir:

- **Desenhar as janelas que se deseja usar**
  Inserir no formulário os controles que serão necessários

- **Adaptar as propriedades dos objetos**
  Alterar as propriedades dos controles às necessidades da aplicação

- **Escrever o código para os eventos associados**
  Esta é a parte mais complexa do desenvolvimento, é ela que dá a funcionalidade ao programa, são as rotinas que começam a ser executadas a partir de um evento.

Desenhar as janelas que se deseja usar



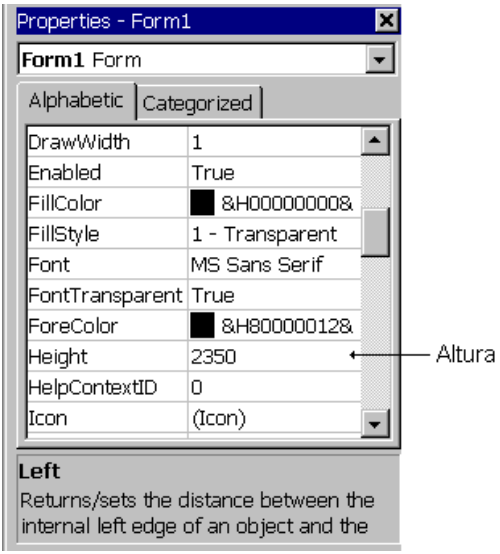
1. Começamos inserindo um Label (Legenda) e um Botão de Comando no Formulário, de uma das duas maneiras indicadas anteriormente.
2. Observe que, quando o controle estiver selecionado, podemos arrastá-lo para qualquer lugar no formulário.

Adaptar as propriedades dos objetos

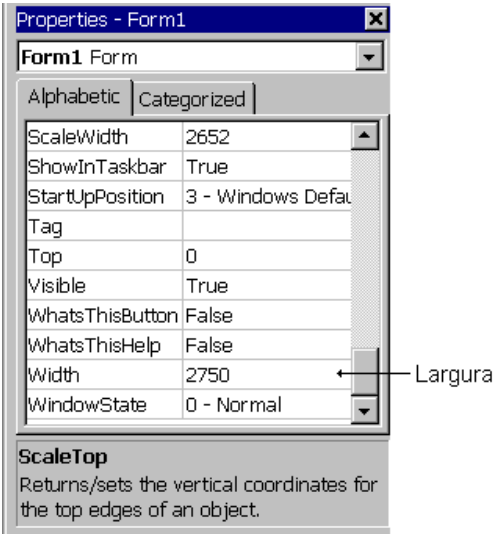
Para alterar a propriedade de um objeto, ele tem que estar selecionado (os oito pontos visíveis), em seguida, procurar o nome da propriedade a ser alterada e selecionar (no caso de valores padrão) o seu valor, ou então, escrever um valor. Caso a Janela de Propriedades esteja oculta, pressione F4 ou dê um clique no botão Properties Window () na barra de ferramentas, para visualizá-la.

1 - Dimensione o formulário da seguinte maneira:

Selecione a propriedade **Height** (altura), e entre com o valor 2350.

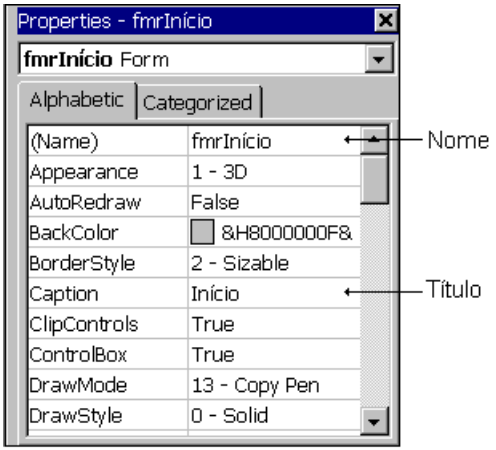


Selecione a propriedade **Width** (largura), e entre com o valor 2750.



Estes números correspondem a **Twips**, esta unidade foi criada para que houvesse uma independência do VB em relação aos dispositivos de entrada e saída de dados (impressoras, monitores e scanner) e que fosse mais precisa que estes aparelhos. Um Twip corresponde a 1/1440 de polegada.

Altere também as propriedades **Name** e **Caption**. A propriedade Name será a identificação do Objeto quando construirmos o código da aplicação. E a propriedade Caption, é a palavra que aparecerá como título da janela.

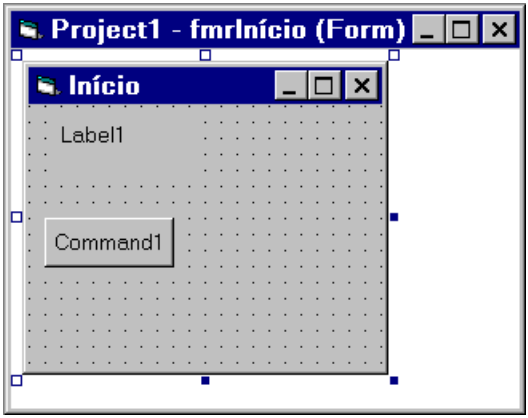


Para a propriedade Name há uma convenção indicada no manual do VB que iremos seguir neste curso; as três primeiras letras indicam o tipo do objeto, e as seguintes, um conjunto de caracteres qualquer que identifique o objeto. De preferência, com a primeira letra maiúscula para facilitar a leitura. A propriedade Name deve começar com uma letra e ter no máximo 40 caracteres e não pode conter espaços ou pontuação.

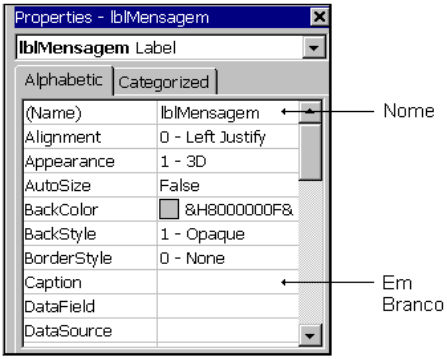
Convenções de prefixos para nomes de objetos no Visual Basic

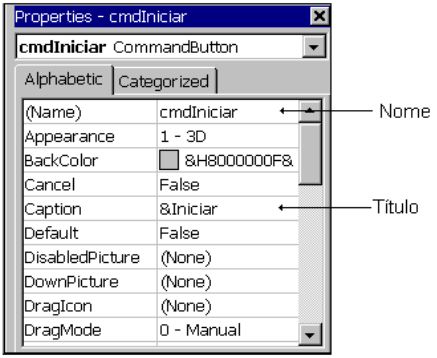
Objeto	**Prefixo**	**Exemplo**
Check box	chk	chkVerificar
Combo box	cbo	cboLivros
Command button	cmd	cmdCancelar
Data	dat	datBiblio
Directory list box	dir	dirDiretório
Drive list box	drv	drvDiscos
File list box	fil	filArquivos
Form	frm	frmInício
Frame	fra	fraOpções
Grid	grd	grdPlanilha
Horizontal scroll bar	hsb	hsbVolume
Image	img	imgFigura
Label	lbl	lblNome
Line	lin	linSeparar
List box	lsl	lslCódigos
Menu	mnu	mnuEditar
OLE	ole	oleObjeto1
Option button	opt	optGramas
Picture box	pic	picQuadro
Shape	shp	shpRetângulo
Text box	txt	txtCliente
Vertical scroll bar	vsb	vsbVolume

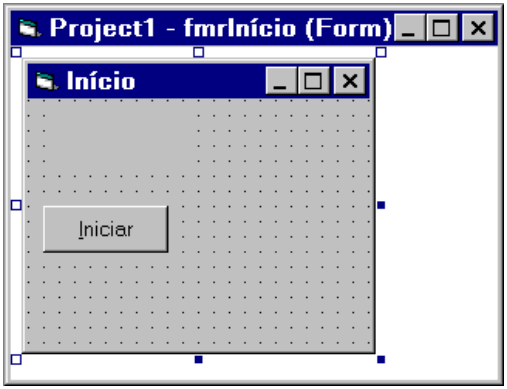
Após você alterar estas quatro propriedades (Caption, Height, Name e Width) do formulário, ele estará assim:



Agora, altere as propriedades do **Label** e do **Botão de Comando** seguindo as figuras.



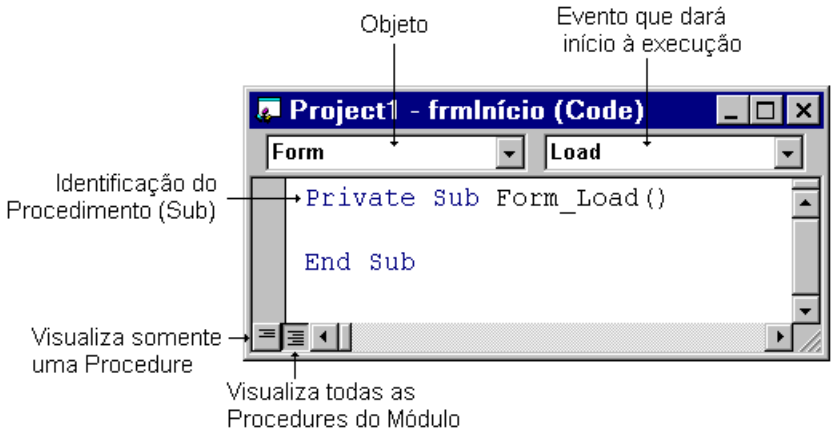




Escrever o código para os eventos associados

O código é escrito na janela de código, para acessá-la, damos um duplo clique em qualquer objeto do projeto ou na janela **Project Explorer** selecionamos View Code ( ).

Janela de Código



Nesta janela notamos 3 elementos importantes:

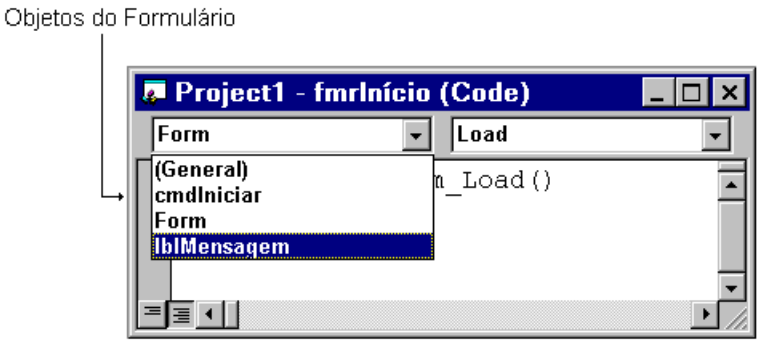
1. Nome do Objeto associado ao procedimento.
2. Nome do Evento que quando ocorrer, dará início ao procedimento.
3. Procedimento que conterá os códigos

Todo procedimento inicia tendo na primeira linha o seu nome e, na última, a declaração **End Sub** (final).

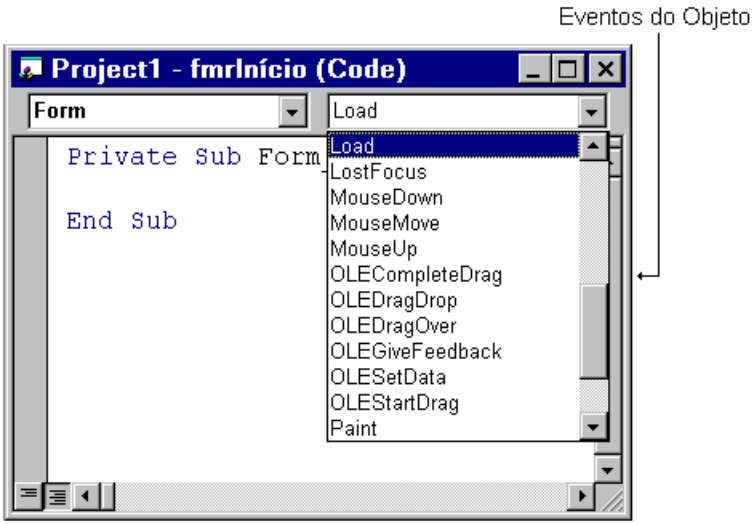
A primeira linha segue o padrão; **Private Sub** nome do objeto + **_** + evento + **( )**

Cada objeto tem um evento que é mais comumente utilizado, e é com este evento que o VB inicia a Janela de Código, não impedindo que utilizemos outro ou mais de um evento.

Se for dado um Clique na seta do quadro de lista dos objetos, serão mostrados todos os objetos neste formulário, e poderemos escolher em qual destes iremos trabalhar o código.



Da mesma forma, se dermos um Clique na seta do quadro de lista dos eventos, serão mostrados todos os eventos do Objeto escolhido, permitindo a seleção do evento para o qual queremos criar um procedimento.



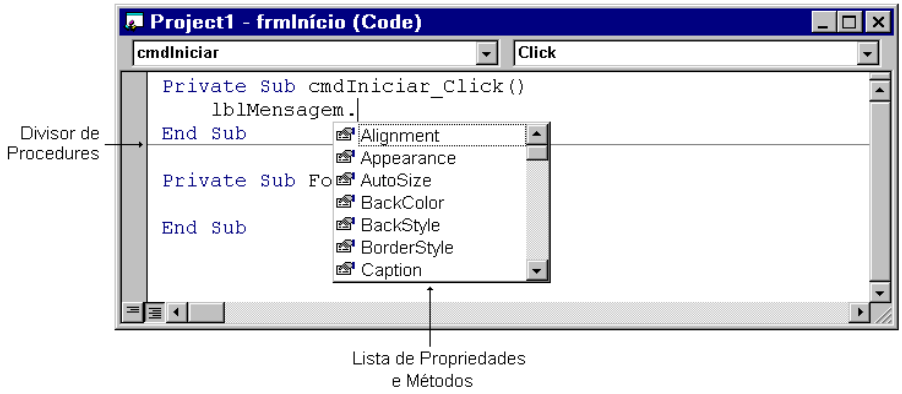
O nosso projeto **Início**, mostrará uma mensagem no Label (objeto) com um Clique (evento) no Botão "Iniciar" (objeto). Ou seja, iremos alterar a propriedade Caption de lblMensagem, esta propriedade contém o que será mostrado ao usuário.

Atribuímos um valor a uma propriedade de um objeto seguindo o padrão:

objeto + . + propriedade + = + valor da propriedade

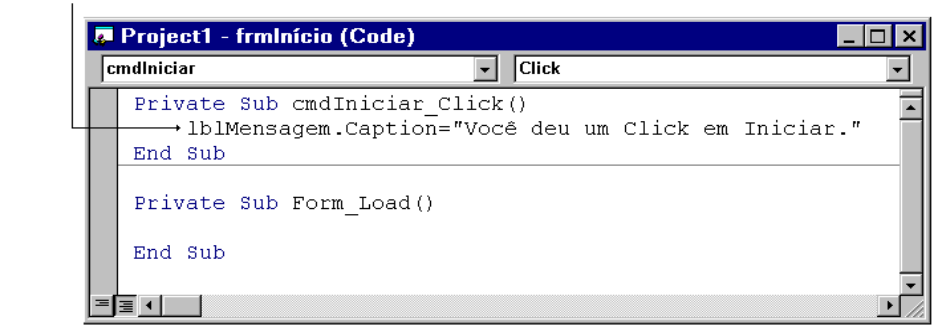
Quando for dado um clique em Iniciar será mostrada a mensagem “Você deu um Click em Iniciar.”. Abra a Janela de Código para o botão de comando, e digite o código conforme a figura a seguir. Observe que após a digitação de **lblMensagem.** é mostrado um quadro de lista com todas as propriedades e métodos associados ao objeto (lblMensagem - Label), quando parte da palavra digitada representar uma propriedade ou método inequívoco, basta pressionar a barra de espaço que o VB completará a sentença. Os métodos e propriedades são representados pelos seguintes ícones:

- Propriedade
- Método

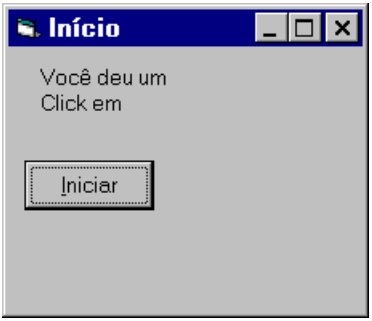


No caso da propriedade Caption, basta digitar a letra C e o sinal de = .

Dar um TAB para edentação

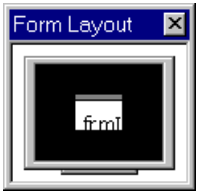


Em seguida, clique sobre o botão **Start** da barra de ferramentas ( ), logo após, dê um clique no botão Iniciar para ver o resultado.



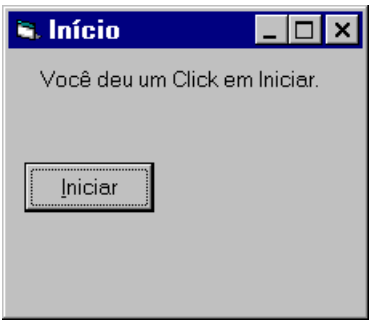
Observe a posição do formulário na tela, ele não está na mesma posição quando você o estava projetando. Finalize a execução através do botão **End** ( ).

Para posicionar o formulário na posição real em tempo de execução, utilizamos a janela de layout (**Form Layout Window**), para acessá-la dê um clique no botão Form Layout Window ( ) da barra de ferramentas.

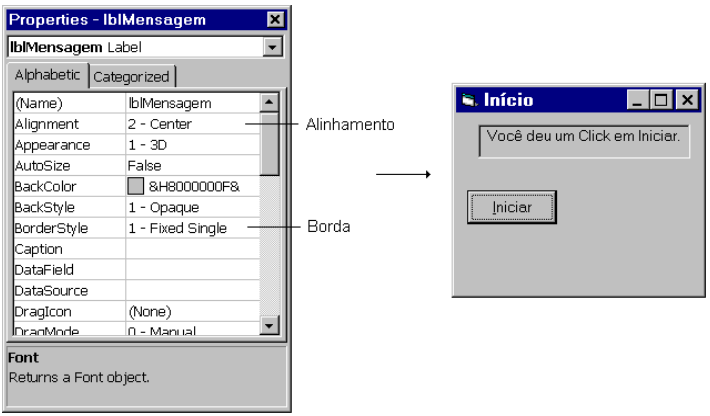


Clique sobre o desenho do formulário e arraste-o para a posição desejada.

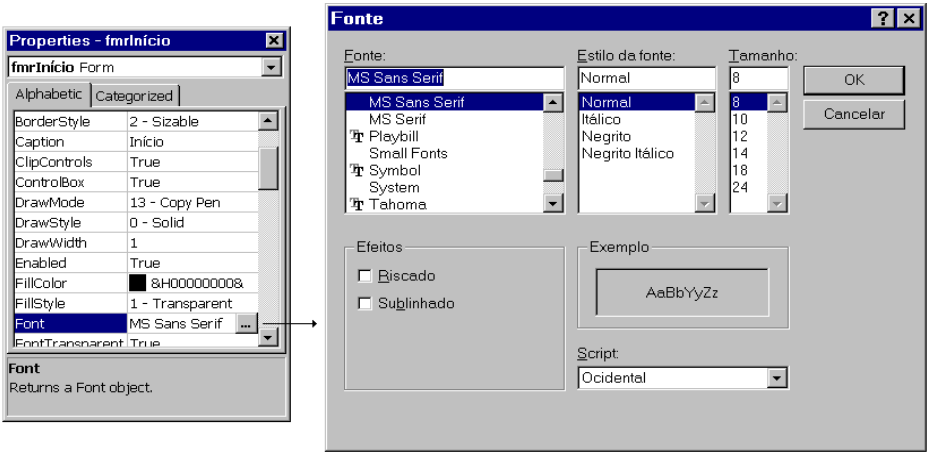
Altere a dimensão do objeto lblMensagem para que toda a mensagem caiba na mesma linha. Execute e observe a mudança.



Pare a execução, e altere as propriedades **Aligment** e **BorderStyle** de lblMensagem.



Existem propriedades que possuem vários valores, quando escolhemos **Aligment** e damos um clique na seta, aparecem os tipos de alinhamento para o texto, mas existem propriedades que possuem inúmeras escolhas, neste caso, ao invés de uma seta, encontraremos três pontos, é o caso da propriedade **Font**.



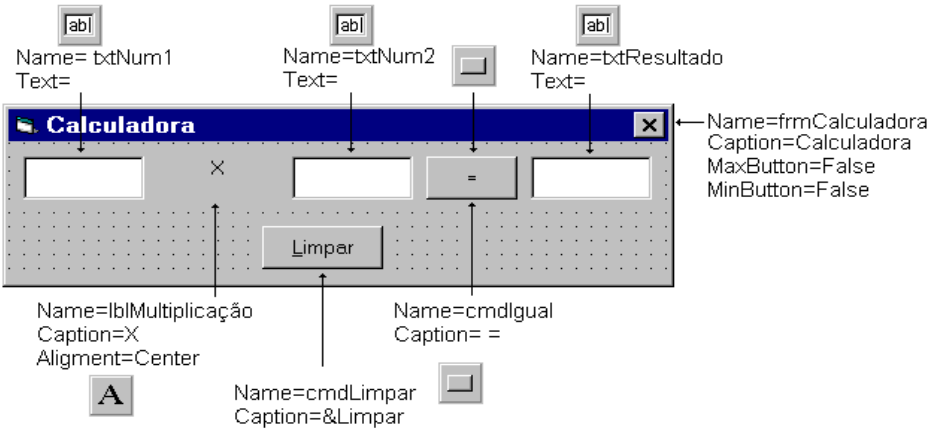
Quando selecionamos os três pontos, aparece um Quadro de Diálogo para escolhermos o formato da fonte para a exibição da Mensagem.

No projeto Início, teste as alterações de fonte observando as mudanças.

EXEMPLO I - CALCULADORA

Para iniciar um novo projeto, escolha a opção New Project... do menu File, ou pressione Ctrl+N, abrindo um programa do tipo **Standard EXE**

Dimensione e insira os controles, utilizando a Janela de Ferramentas (Toolbox) no formulário, como o exemplo abaixo. Caso a Toolbox não esteja visível, selecione a opção Toolbox do menu View. Dimensionamos o formulário no VB da mesma forma que no Windows dimensionamos as janelas. Siga a figura para alterar as propriedades assinaladas dos Objetos:



As propriedades MaxButton = False e MinButton = False, desabilitam os botões de maximizar e minimizar da janela.

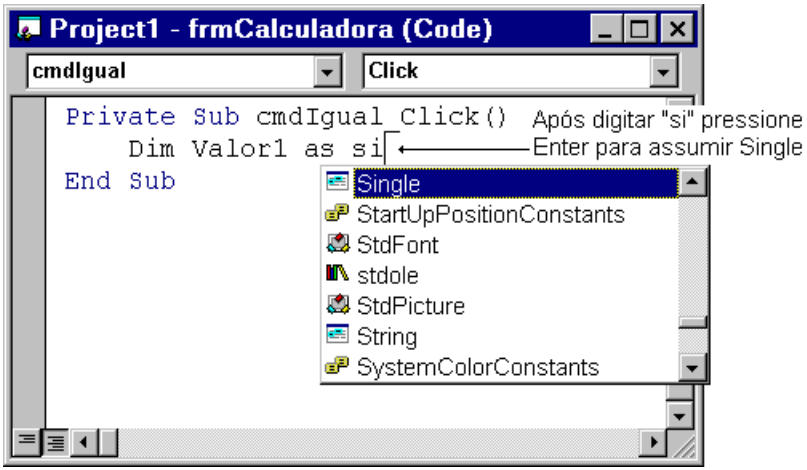
Neste exemplo de projeto, digitaremos um número em txtNum1, outro em txtNum2 e quando dermos um clique em cmdIgual, o resultado da multiplicação aparecerá em txtResultado. Para limpar os quadros de texto, clique em cmdLimpar.

O projeto irá trabalhar basicamente com dois eventos :

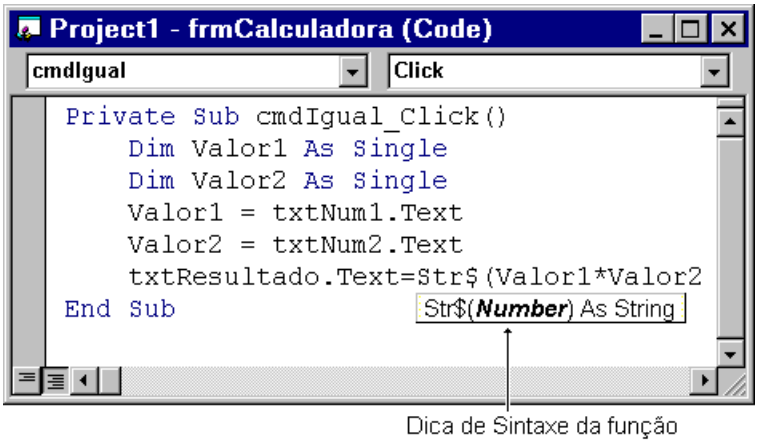
Clique em cmdIgual (=)

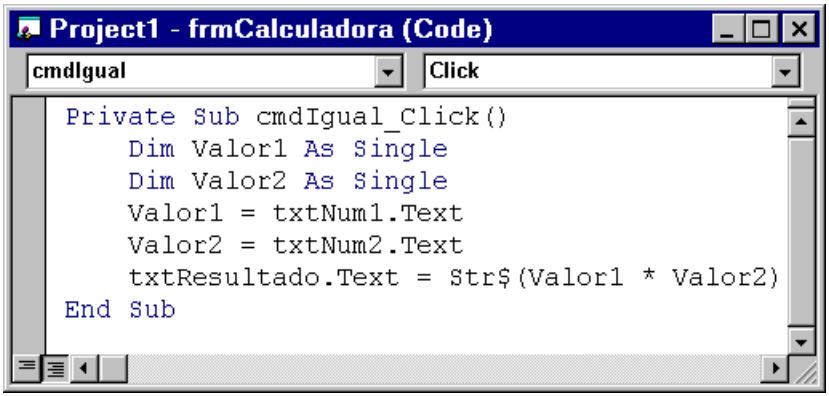
Clique em cmdLimpar (Limpar)

Então, para escrevermos o código, damos um Duplo Clique no Botão cmdIgual, e a janela de código será mostrada. Entre com o código conforme a figura a seguir:

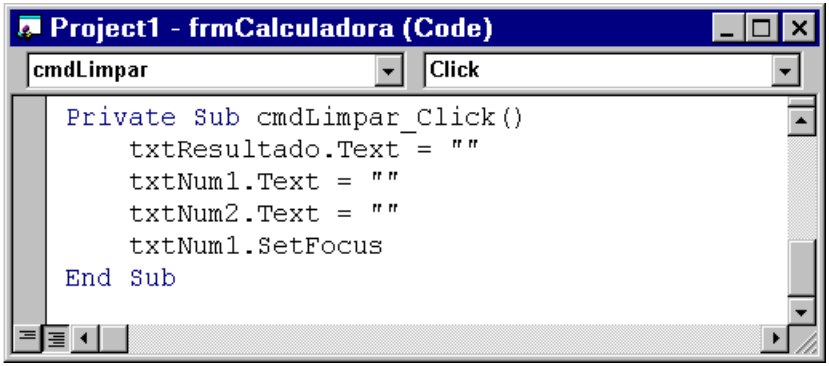


O editor de código do Visual Basic 6, fornece dicas de uso das funções e procedures da linguagem, sempre que uma parte reconhecível for digitada ele nos fornecerá uma dica de utilização.





Alterne para o procedimento **cmdLimpar_Click**. E entre com os comandos a seguir:

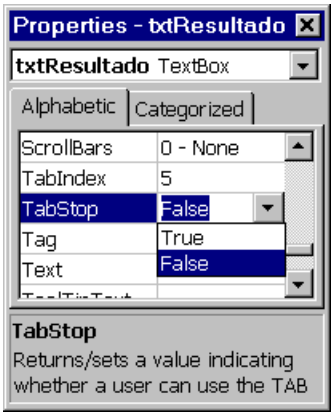


Execute o projeto. Para utilizar, entre com um número em txtNum1, outro em txtNum2 e dê um Click em "=", que o resultado aparecerá em txtResultado.

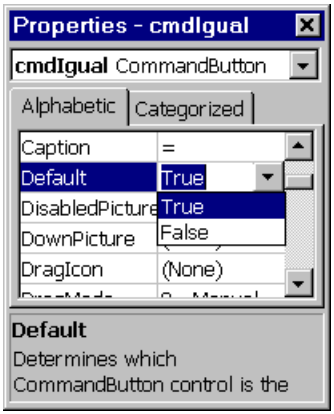
Note que alternamos os campos com a tecla Tab, a ordem de tabulação corresponde à ordem em que os controles foram colocados no formulário. Esta ordem é determinada pela propriedade TabIndex dos controles, caso o seu projeto não esteja, coloque-o na seguinte ordem:

Objeto	TabIndex
txtNum1	1
txtNum2	2
cmdIgual	3
cmdLimpar	4
txtResultado	5
lblMultiplicação	6

Para alterar esta propriedade, basta selecionar o controle, Janela de Propriedades, procurar TabIndex e alterar o valor, o VB não aceita controles com TabIndex de mesmo valor.



Execute o projeto e observe a alteração. Note que podemos alterar o valor de txtResultado mesmo após a operação ter sido efetuada. Para evitar isso, defina as propriedades **TabStop** = **False** e **Locked** = **True** do txtResultado, e verá que o usuário não terá mais acesso com a tecla Tab ao txtResultado e nem poderá editá-lo.



Existem, nas aplicações para Windows, botões de comando que são acionados também com a tecla Enter, chamados botões default. No nosso projeto, este botão será o cmdIgual, para isso, defina a propriedade **Default** = **True** para esse objeto aparecendo um contorno mais espesso ao seu redor, dando a indicação que se a tecla **Enter** for acionada, o comando será executado.

VARIÁVEIS

Variável é um local onde podem ser guardados dados com possibilidade de alteração em tempo de execução. O nome de uma variável pode ter até 255 caracteres, tem que começar com uma letra e tem que ser única. O nome pode conter números e sublinhados e não pode ser uma palavra reservada.

Existem vários tipos de variáveis, dependendo do tipo de dados que queremos que ela armazene.

Tipo	Número de Bytes	Caracter	Faixa
Byte	1		0 a 255
Boolean	2		True (-1) ou False (0)
Date	8		1/Jan/100 a 31/Dez/9999
Integer	2	%	-32.768 a 32.767
Long	4	&	-2.147.483.647 a 2.147.483.647
Single	4	!	-3,402823E38 a -1,401298E-45 1,401298E-45 a 3,402823E38
Double	8	#	-1,79769313486232E308 a -4,94065645841247E-324 4,94065645841247E-324 a 1,79769313486232E308
Currency	8	@	-922,337,203,685,477.5808 a 922,337,203,685,477.5807
String	variável	$	Não se aplica

Podemos usar certos caracteres para indicar o tipo da variável desejada, quando usá-la inicialmente. Por exemplo: i%, trôco@. Como tipo básico, o VB usa o tipo Single, portanto se tivermos uma variável com o nome de Valor!, será o mesmo que deixar como Valor.

Formas de Declarar uma Variável

1. Usar a variável onde desejar. Na linha de código onde for necessária usando um dos caracteres que identificam o tipo.
2. Usar as declarações **Dim**, **Private**, **Public** ou **Static**, alocando um espaço na memória para a variável.

Dim NomeVariável **As** tipo

Private NomeVariável **As** tipo

Public NomeVariável **As** tipo

Static NomeVariável **As** tipo

Escopo e Tempo de Vida das Variáveis

Escopo são os pontos da aplicação de onde podemos acessar a variável. Existem 4 locais diferentes para declarar variáveis.

Local, a variável será usada apenas pelo procedimento onde ela foi declarada.

Em nível de **Formulário**, a variável poderá ser acessada por todos os procedimentos do formulário quando for declarada na seção geral (general) .

Em nível de **Módulo**, a variável poderá ser acessada por todos os procedimentos do módulo.

Em nível **Publico**, toda aplicação poderá usar esta variável.

Toda vez que executamos um procedimento, suas variáveis locais são reinicializadas. Para que a variável retenha o seu valor, usamos a declaração **Static**. Mais adiante, veremos um exemplo desta declaração e das outras.

FORMATAÇÃO DE TEXTO

A função **Str$**, transforma um número em texto, mas não padroniza a sua apresentação. Caso você precise formatar um dado a ser exibido, use a função;

Format$(expressão [,formato])

Onde:

expressão = expressão numérica ou string a ser formatado.
formato = a maneira como deverá ser mostrada a expressão.

Formatando números:

Formato	**5 positivo**	**5 negativo**	**5 decimal**
0	5	-5	1
0,00	5,00	-5,00	0,50
#.##0	5	-5	1
#.##0,0	5,0	-5,0	0,5
$#.##0;($#.##0)	$5	($5)	$1
$#.##0,00;($#.##0,00)	$5,00	($5,00)	$0,50
0%	500%	-500%	50%
0,00E+00	5,00E+00	-5,00E+00	5,00E-1

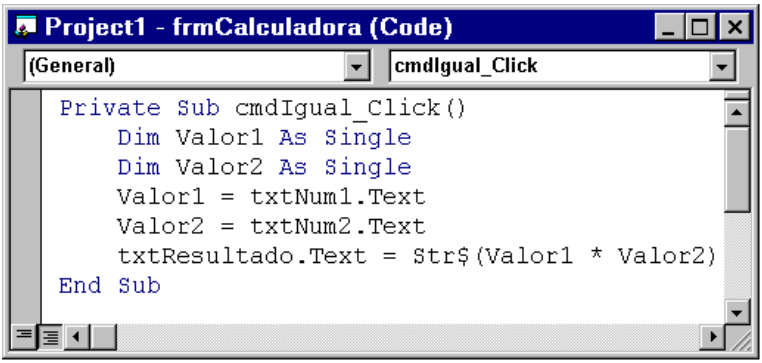
Em "formato" o número 0 será mostrado ou trocado pelo caractere em sua posição, já o nirus (#) não será mostrado. Podemos inserir símbolos na função Format, como no exemplo: $, % e E .
Formatando Data e Hora:

Formato	**Exibido**
d/m/yy	10/7/96
dd-mm-yyyy	01-Jun-1996
dd-ddd	02-dom
hh:mm AM/PM	08:50 AM
h:mm:ss a/p	8:50:20 a
d/m/yy h:mm	03/12/95 9:30
General Date	06/09/96 9:40:18
Long Date	Sexta, 9 de setembro de 1996
Medium Date	09-set-96
Short Date	09/09/96
Long Time	9:40:19
Medium Time (12 horas)	09:40 AM
Short Time (24 horas)	09:40

MODIFICANDO A CALCULADORA

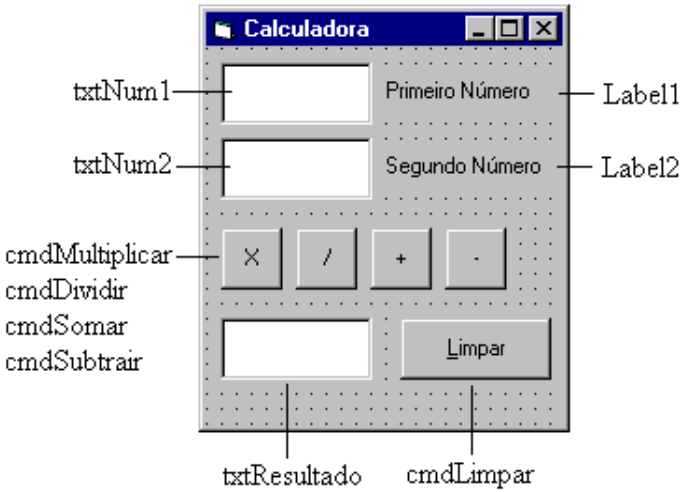
No formulário da calculadora, selecione o botão de comando cmdIgual e pressione a tecla Delete. Lembre-se que tínhamos um código associado a este objeto e agora que ele sumiu, para onde foi o código?

Chame a janela de código, indo até a janela Project Explorer, e clique em ViewCode. No quadro de objetos (object) selecione a seção General, o código o cmdIgual estará lá como um procedimento geral.

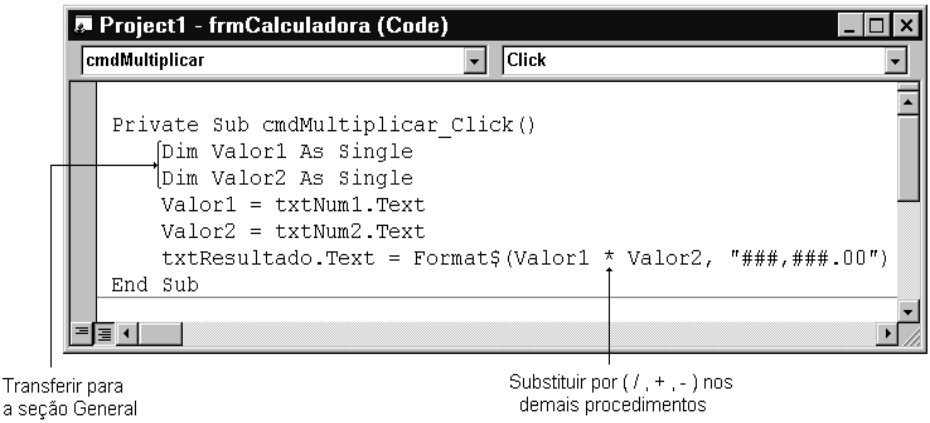


Procedimento geral é aquele que pode ser chamado por qualquer outro procedimento do formulário, funciona como uma sub-rotina. Ele não é como um procedimento associado a um objeto executado a partir de um evento, e sim, quando chamado. Agora, deixe o formulário frmCalculadora como o exemplo a seguir:

Na figura aparecem as propriedades Name de cada objeto



Chame a janela de código dando um clique duplo no formulário ou na janela Project Explorer - View Code. Aparecerá então o procedimento Form_Load, vá até o quadro Object e selecione (General), no quadro de Eventos procure o código do antigo cmdIgual (cmdIgual_Click) e altere o cabeçalho como no exemplo a seguir, observe que ao dar Enter, o procedimento que antes era geral, passou a ser associado a um novo objeto (cmdMultiplicar).



Na última linha já estamos usando a função Format$, para formatar o número a ser apresentado.

Selecione todo o texto, menos o cabeçalho e End Sub, e copie (Ctrl + C). Chame o procedimento para outro botão de operação, cole o texto e altere o operador correspondente.

Teste os vários formatos de apresentação dos números, alterando a forma de apresentação da função Format$.

Um aplicativo construído em VB trabalha com vários arquivos. Um arquivo para cada formulário ou módulo, um arquivo para o projeto e outros opcionais. Suas descrições são as seguintes:

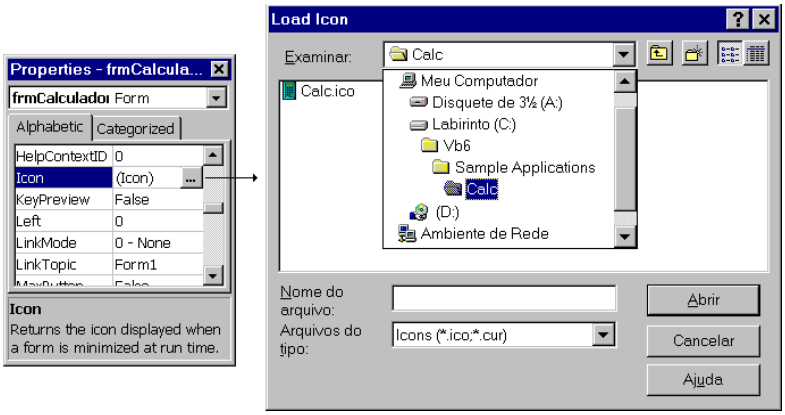
- O arquivo de projeto contém a localização de todos os componentes (.vbp).
- Um arquivo para cada formulário (.frm).
- Um arquivo de dados binários para cada formulário que contém os valores das propriedades dos controles no formulário (.frx).

- Um arquivo para cada módulo de classe (.cls).
- Um arquivo para cada módulo standard (.bas).
- Um ou mais arquivos que contêm controles ActiveX (.ocx).
- Um único arquivo de recurso (.res).

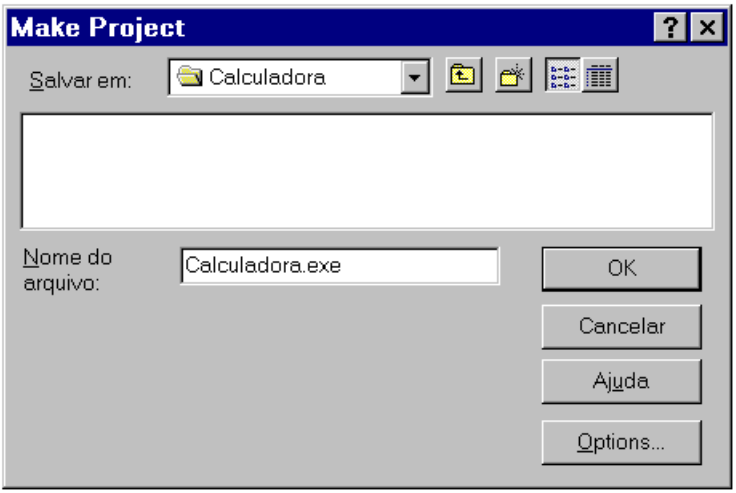
Vamos salvar o nosso projeto de calculadora.

No menu **File**, selecione **Save Project**, aparecerá o quadro de diálogo de Salvar do Windows pedindo para dar um nome ao arquivo de formulário, extensão *.frm*, dê o nome de calculadora.frm e clique em Salvar. A seguir, aparecerá o mesmo quadro pedindo para dar um nome ao arquivo de projeto, extensão *.vbp*, dê o nome de Calculadora.vbp e clique em Salvar. O projeto estará salvo.

Antes de fazer do projeto um arquivo executável, vamos escolher um ícone para o nosso projeto ser representado no Windows. Selecione a propriedade **Icon** do formulário e clique no botão com reticências tendo acesso ao quadro de diálogo **Load Icon**, escolhendo um ícone que será associado ao formulário;



Escolha um ícone e clique em Abrir. Quando o seu projeto aparecer no Windows, ele será representado por este ícone. Agora, vamos fazê-lo executável fora do VB, no Menu File, escolha Make EXE File..., aparecendo o quadro de diálogo para escolher o nome do arquivo executável:



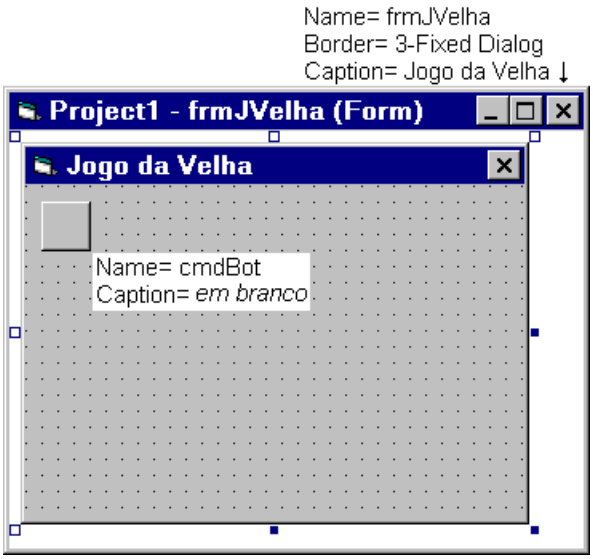
Escolhido o nome do arquivo executável, clique em Ok. Agora, você tem um programa executável em qualquer microcomputador que possua o sistema Windows 95, sem necessariamente ter o VISUAL BASIC instalado.

EXEMPLO II - JOGO DA VELHA

Para iniciar um novo projeto, selecione New Project... do menu File, ou Ctrl+N.
Caso você ainda não tenha salvo o seu projeto corrente, o VB abrirá as janelas para salvar o projeto. E só então iniciará o novo projeto.

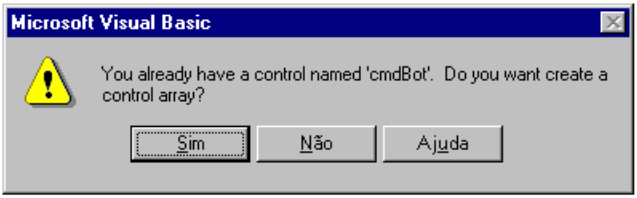
Vamos iniciar um projeto de Jogo da Velha, onde o usuário irá jogar contra o computador que não "pensa" as suas jogadas, trabalhando com números aleatórios - e ao final da partida será mostrado um quadro de mensagem informando o ganhador. O objetivo é conhecermos as estruturas condicionais e de repetição, tão utilizadas nos programas.

Insira um botão de comando no formulário dimensionando-o como um quadrado, e altere suas propriedades como mostra a figura:

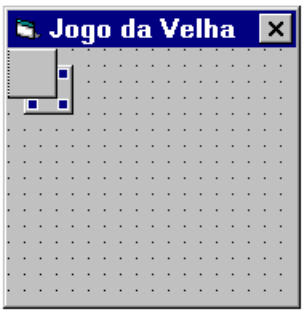


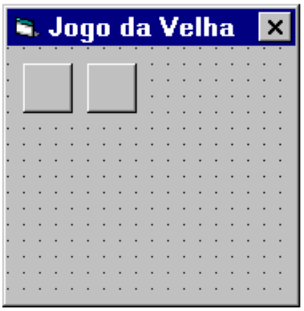
O nosso Jogo da Velha possui 9 botões iguais, todos irão executar a mesma rotina quando o usuário der um clique em um deles. Para economizar recursos, vamos criar um **Array** de controle (ordem de controles), onde todos os botões tem o mesmo nome com índices diferentes.

Para criar um Array de controle, selecione o botão e copie-o (Ctrl + C) e depois cole-o (Ctrl + V). Quando for dada a ordem de colar, o VB abrirá um quadro de mensagem indicando que já existe um controle com o nome de cmdBot, e se você quer criar ou não um Array de controle, responda Sim.

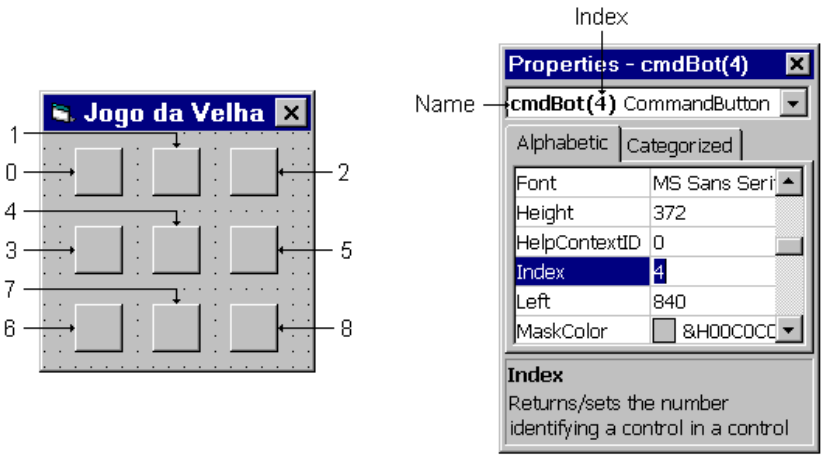


O botão de comando irá aparecer no canto superior esquerdo do formulário, depois é só arrastá-lo para a posição desejada.

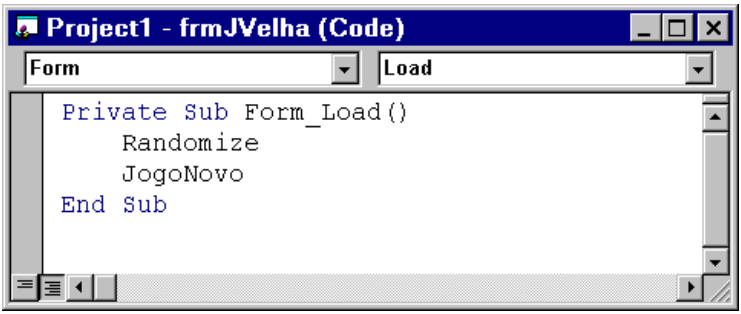




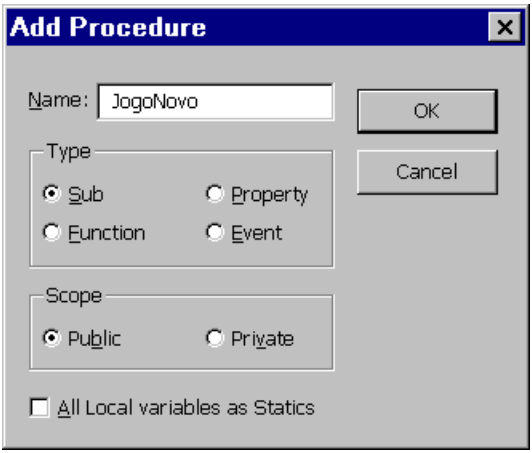
Para inserir os próximos botões, basta ir colando e o VB não perguntará mais sobre o Array de Controle. Posicione os botões da esquerda para a direita e de cima para baixo, pois assim o índice deles coincidirá com o código na hora da verificação. Se você observar a janela de propriedades (properties - cmdBot( )) dos botões, notará que além do nome, eles possuem um índice, este índice está na propriedade **Index** de cada botão, caso você tenha errado as posições, basta corrigir alterando essa propriedade.



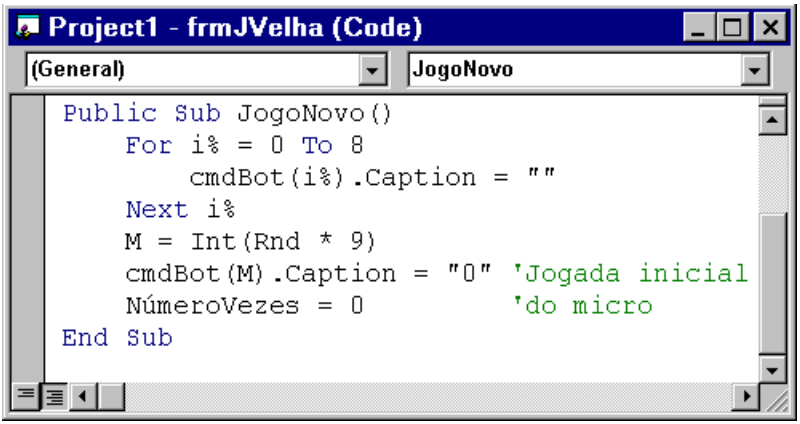
Quando o projeto iniciar, o formulário será carregado na memória, neste fato ocorre o evento **Load**. Criamos para este evento um procedimento de inicio do jogo. Selecione o botão View Code para abrir a janela de Códigos e entre com as declarações **Randomize** e **JogoNovo**, responsáveis respectivamente pela inicialização de números aleatórios e o carregamento de um procedimento chamado **JogoNovo**.



No formulário temos a seção General, nesta seção colocamos os procedimentos e as variáveis que serão solicitados por todos os procedimentos do formulário. No nosso projeto, na seção General teremos o procedimento JogoNovo que dará início a um novo jogo, e a variável NúmeroVezes, que servirá para armazenar o número de vezes que jogamos - indicará se houve empate ou não. Para criar este procedimento, vá até o Menu Tools e escolha a opção Add Procedure..., aparecerá então o seguinte quadro de diálogo:



Selecione as opções **Sub** e **Public** e entre com o nome do procedimento, em seguida, clique em OK. Digite o código conforme figura a seguir:



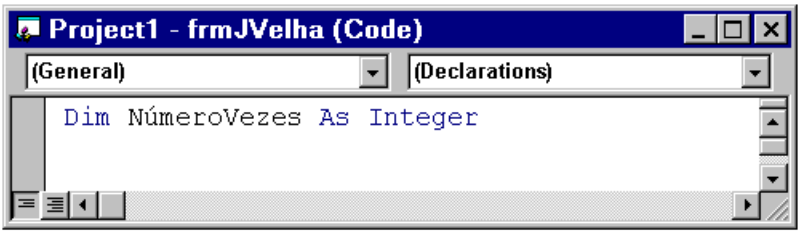
Todo bom programador inclui em seu código linhas de comentário que são muito úteis quando for necessária uma manutenção do programa. Estas linhas contém descrições a respeito de como o programa funciona, servindo tanto para o programador inicial quanto para outros que irão realizar a manutenção.

Colocamos uma linha do comentário no VB utilizando o caracter (‘) antes de cada comentário. Esta linha ficará destacada em verde no VB.

Ex: ‘Jogada inicial do micro

A estrutura de repetição **For**... **Next** é utilizada aqui para apagar o conteúdo dos botões. A função **Int** e **Rnd** retornam, respectivamente, a porção inteira de um número, e um número aleatório entre 0 (inclusive) e 1.

Declaramos a variável NúmeroVezes na seção General - Declarações, porque ela será utilizada em dois procedimentos distintos - JogoNovo e cmdBot_Click. O seu valor permanecerá até que o formulário seja retirado da memória ou o programa finalizado.



O próximo passo é fazer o código dos botões, dê um duplo clique em qualquer um deles para exibir a janela de código, entrando com o código conforme texto abaixo. A seguir, encontraremos o uso da estrutura condicional **If**... **End If** e a estrutura de repetição **Do Until**... **Loop**. Procure entender as estruturas e tire todas as dúvidas.

```
Private Sub cmdBot_Click(Index As Integer)
  'O parâmetro Index informa o index do botão acionado
    NúmeroVezes = NúmeroVezes + 1
   cmdBot(Index).Caption = "X"
    If cmdBot(0).Caption = "X" And cmdBot(1).Caption = "X" And
cmdBot(2).Caption = "X" Then GoTo MensX
    If cmdBot(3).Caption = "X" And cmdBot(4).Caption = "X" And
cmdBot(5).Caption = "X" Then GoTo MensX
    If cmdBot(6).Caption = "X" And cmdBot(7).Caption = "X" And
cmdBot(8).Caption = "X" Then GoTo MensX
    If cmdBot(0).Caption = "X" And cmdBot(3).Caption = "X" And
cmdBot(6).Caption = "X" Then GoTo MensX
    If cmdBot(1).Caption = "X" And cmdBot(4).Caption = "X" And
cmdBot(7).Caption = "X" Then GoTo MensX
    If cmdBot(2).Caption = "X" And cmdBot(5).Caption = "X" And
cmdBot(8).Caption = "X" Then GoTo MensX
    If cmdBot(0).Caption = "X" And cmdBot(4).Caption = "X" And
cmdBot(8).Caption = "X" Then GoTo MensX
```

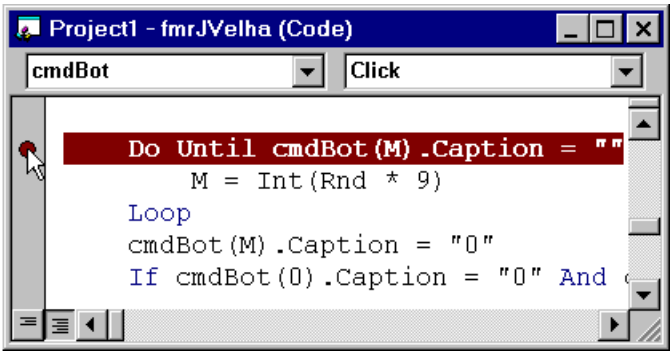
```
    If cmdBot(2).Caption = "X" And cmdBot(4).Caption = "X" And
cmdBot(6).Caption = "X" Then GoTo MensX

   Do Until cmdBot(M).Caption = ""
      M = Int(Rnd * 9)
   Loop
   cmdBot(M).Caption = "0"
    If cmdBot(0).Caption = "0" And cmdBot(1).Caption = "0" And
cmdBot(2).Caption = "0" Then GoTo Mens0
    If cmdBot(3).Caption = "0" And cmdBot(4).Caption = "0" And
cmdBot(5).Caption = "0" Then GoTo Mens0
    If cmdBot(6).Caption = "0" And cmdBot(7).Caption = "0" And
cmdBot(8).Caption = "0" Then GoTo Mens0
    If cmdBot(0).Caption = "0" And cmdBot(3).Caption = "0" And
cmdBot(6).Caption = "0" Then GoTo Mens0
    If cmdBot(1).Caption = "0" And cmdBot(4).Caption = "0" And
cmdBot(7).Caption = "0" Then GoTo Mens0
    If cmdBot(2).Caption = "0" And cmdBot(5).Caption = "0" And
cmdBot(8).Caption = "0" Then GoTo Mens0
    If cmdBot(0).Caption = "0" And cmdBot(4).Caption = "0" And
cmdBot(8).Caption = "0" Then GoTo Mens0
    If cmdBot(2).Caption = "0" And cmdBot(4).Caption = "0" And
cmdBot(6).Caption = "0" Then GoTo Mens0
   If NúmeroVezes = 4 Then
         MsgBox "Partida Empatada", 64, "Empate"
              JogoNovo
         Exit Sub
    End If
   Exit Sub

MensX:
   MsgBox "Você Ganhou", 64, "Vencedor"
   JogoNovo
   Exit Sub

Mens0:
   MsgBox "Eu Ganhei", 64, "Vencedor"
   JogoNovo
   Exit Sub
End Sub
```

Terminando de digitar este procedimento, salve o formulário e o projeto. Mas antes de executá-lo vamos ver um pouco de depuração.

DEPURAÇÃO

Usamos a depuração para encontrar erros lógicos ou de digitação responsáveis pelo mau funcionamento do programa. Alguns erros o VB nos indica durante a digitação e outros durante a compilação, mas exixtem erros que o próprio programador terá que descobrir utilizando as ferramentas de debug do VB.

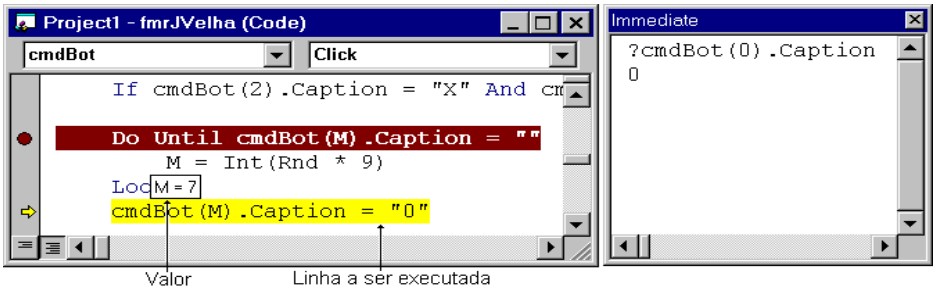
Para incluir um ponto de parada no programa basta dar um clique na barra cinza da esquerda na direção da linha que se deseja como ponto de parada. Dê um clique na linha mostrada na figura a seguir.



Execute o programa pressionando **F5**, ou clique sobre o botão . Após a jogada do usuário, o programa irá parar na linha marcada, sendo possível examinar os valores das variáveis.

Pressione a tecla **F8** para executar o programa passo-a-passo até a linha com a instrução **Loop** quando veremos os valores de algumas variáveis.

Existem duas formas de sabermos o valor de uma variável ou propriedade de objeto. Uma delas e posicionando o ponteiro do mouse sobre ela e esperar alguns instantes, que o seu valor será mostrado pelo VB.



A segunda maneira, é escrever o nome da variável ou da propriedade do objeto na janela **Immediate** precedido pelo ponto de interrogação. Após pressionar a tecla **Enter** o valor será mostrado pelo VB.

Continue a execução do programa pressionando F5, para retirar o ponto de parada basta dar um clique no círculo vermelho desta linha.

QUADROS DE MENSAGEM

O Windows possui quadros padronizados de mensagem que servem para emitir avisos e recolher opções de tratamento dessas mensagens.

Estes quadros são fáceis de criar no VB com a declaração ou função **MsgBox**. **MsgBox** será uma declaração, quando não tratamos a resposta do usuário, e será uma função, quando esta resposta for tratada.

Para construir um Quadro de Mensagem, use o seguinte padrão:

Declaração - MsgBox mensagem, tipo, título

Função - MsgBox (mensagem, tipo, título)

Onde:**mensagem** - expressão mostrada dentro do quadro de diálogo.

tipo - somatória de números, conforme o que queremos que seja exibido no Quadro de Mensagem, seguindo a tabela a seguir.

título - título do Quadro de Mensagem (barra de título).

Argumento **tipo** para a Declaração/Função MsgBox

Valor	Significado
0	Somente o botão de OK
1	Botões de OK e Cancelar
2	Botões Anular, Repetir e Ignorar
3	Botões Sim, Não, Cancelar
4	Botões Sim, Não
5	Botões Repetir e Cancelar
16	Sinal de Stop
32	Sinal de Pesquisa
48	Sinal de Aviso
64	Ícone de Informação
0	Primeiro botão com foco
256	Segundo botão com foco
512	Terceiro botão com foco
768	Quarto botão com foco

Teste o projeto alterando o valor de tipo para MsgBox, faça a sua soma escolhendo um item de cada seção.

Agora, vamos alterar o nosso projeto para que ele nos pergunte, ao final da partida, se queremos jogar novamente ou finalizar o programa. Para isso usaremos MsgBox como função, o que nos retornará o valor do botão acionado pelo usuário. Altere o procedimento cmdBot_Click conforme texto a seguir:

```
MensX:
   Resposta$ = MsgBox("Você Ganhou, Deseja Jogar Novamente?", 36, "Vencedor")
   If Resposta$ = 6 Then
      JogoNovo
    Else
       End
   End If
Exit Sub
```

```
Mens0:
   Resposta$ = MsgBox("Eu Ganhei, Deseja Jogar Novamente?", 36, "Vencedor")
   If Resposta$ = 6 Then
      JogoNovo
    Else
      End
   End If
   Exit Sub
End Sub
```

A variável Resposta$ (declarada implicitamente como String), conterá a resposta do usuário que segue o padrão da tabela abaixo;

Valor	Significado
1	Botão OK foi pressionado
2	Botão Cancelar foi pressionado
3	Botão Anular foi pressionado
4	Botão Repetir foi pressionado
5	Botão Ignorar foi pressionado
6	Botão Sim foi pressionado
7	Botão Não foi pressionado

No nosso caso, o programa verificará se o botão Sim foi pressionado, em caso afirmativo, iniciará novo jogo, senão finalizará.

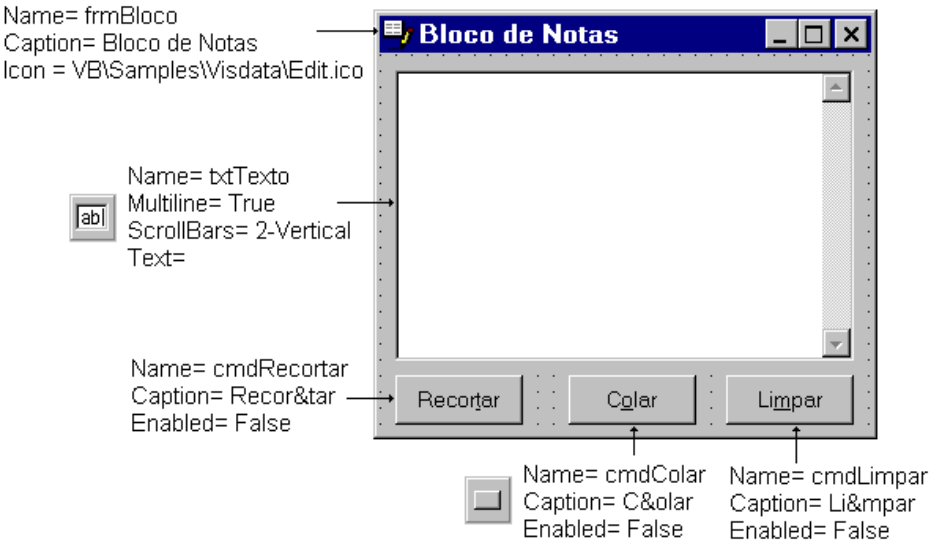
A mensagem aparece em uma única linha no quadro de mensagem. Se quisermos que ela apareça dividida em duas ou mais linhas deveremos utilizar o caracter Chr(13) - Carriage Return, separando as linhas da mensagem. Experimente a alteração abaixo:

```
Resposta$ = MsgBox("Eu Ganhei, Deseja" + Chr(13) + "Jogar Novamente?", 36, "Vencedor")
```

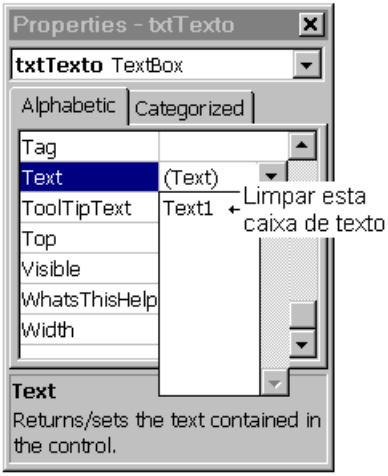
EXEMPLO III - BLOCO DE NOTAS

O nosso próximo projeto será um editor de texto simples do tipo caractere, com ele poderemos alterar o tipo e tamanho da fonte utilizada em todo o texto, recortar, colar e copiar partes selecionadas, e salvar e abrir nosso texto em um arquivo de acesso seqüencial.

Monte o formulário conforme o exemplo, alterando as propriedades dos 5 objetos:



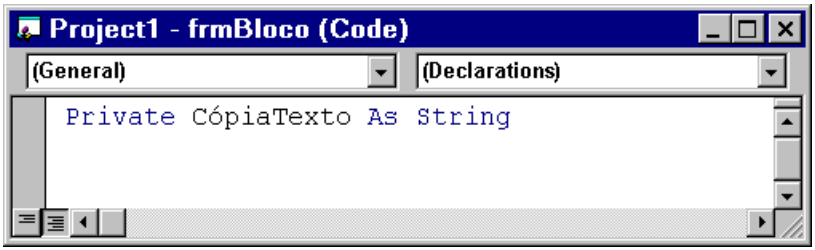
No quadro de texto (txtTexto), a propriedade **Text**, possui uma caixa de texto onde digitamos o texto inicial deste objeto, temos também a propriedade **Multiline** = **True** para permitir que este quadro tenha várias linhas, e a propriedade **ScrollBars** = **Vertical**, para possibilitar a paginação destas linhas quando ultrapassarem a área do quadro.



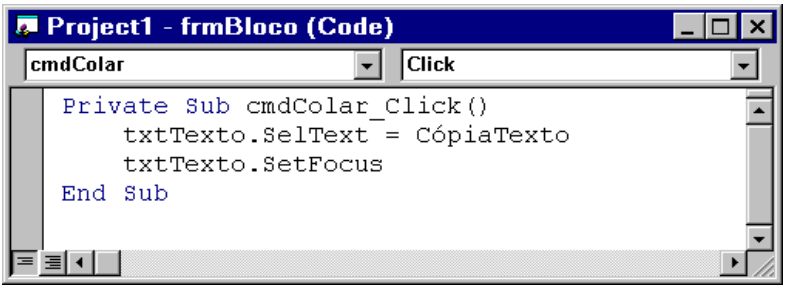
Os botões de comando tem a propriedade **Enabled** = **False** para tornar o botão desabilitado (cinza claro), não permitindo a acesso a eles pelo usuário. Esta propriedade será mudada em tempo de execução quando tivermos algum texto a ser Recortado, Colado ou Limpo.

Vamos ao Código:

Declare a variável (CópiaTexto) que conterá o texto que foi Recortado ou Copiado.

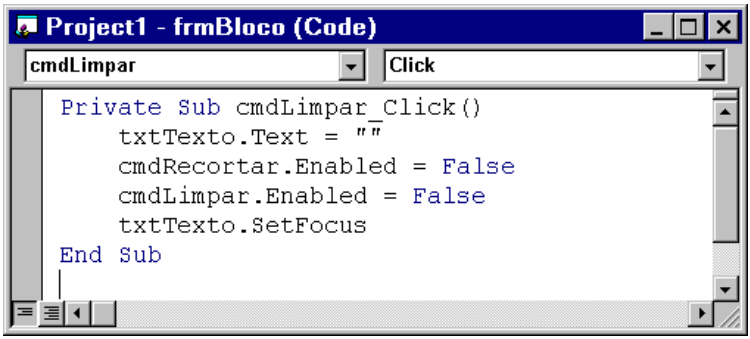


cmdColar
Copia a variável CópiaTexto para o quadro de texto no local do cursor ou área selecionada e devolve o foco para o quadro de texto, se o foco não fosse devolvido ele ficaria com o botão que foi acionado.



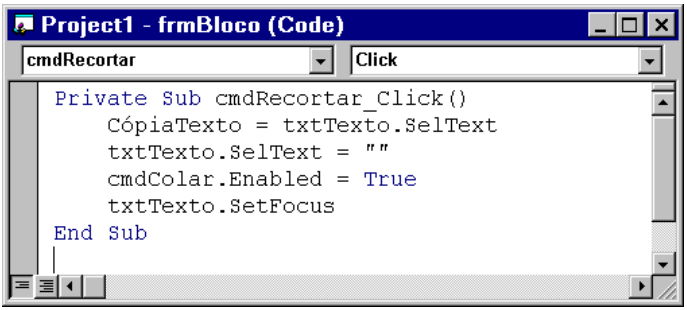
cmdLimpar

Limpa o quadro de texto, limpando a propriedade Text e desabilita os botões cmdRecortar e cmdLimpar. O botão cmdColar não é desabilitado porque ainda existe conteúdo na variável CópiaTexto, que poderá ser colado.



cmdRecortar

Atribui o texto selecionado, propriedade SelText, à variável CópiaTexto, limpa o texto selecionado e habilita o botão cmdColar.

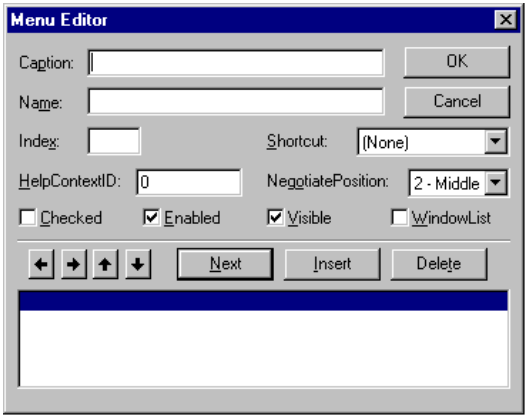


Salve o Formulário, o Projeto, e depois execute testando o funcionamento.

Em nosso Bloco de Notas, as opções de edição de texto estão na forma de botões, mas no Windows essas opções estão na forma de Menu. Neste caso, vamos agora trabalhar com menu e transferir o código dos botões para as opções do menu que iremos construir. Cada item de menu também é um objeto e portanto, também possui propriedades e responde a eventos.

CRIANDO MENUS

Selecione o formulário (frmBloco) e escolha a opção Menu Editor... do menu Tools ou Ctrl+E, ou clique no botão Menu Editor ( ) da barra de ferramentas, aparecendo o quadro de diálogo Menu Editor para construirmos nosso menu.



O Quadro Menu Editor possui as seguintes partes:

Caption. O texto que aparecerá escrito no menu ou item de menu. Para criar uma barra separadora em seu menu, basta digitar um hífen (-). Para o acesso por teclado usamos o *e* comercial (&) antes da letra que queremos que seja o atalho, como nos botões.

Name. Contém o nome que será dado ao objeto que o identificará nas linhas de código.

Index. Número atribuído ao objeto para identificá-lo caso seja usado como um control array. Anteriormente usamos control array em botões e o Index foi determinado automaticamente, mas para itens de menu, teremos que determiná-los manualmente.

Shortcut. Uma lista drop-down de onde poderemos escolher a tecla de atalho para o item. Exemplo: Colar = Ctrl + V.

HelpContextID. Contém um valor numérico único que será usado para encontrar uma referência do objeto no arquivo de help.

NegotiatePosition. Determina a posição em que o menu irá aparecer quando objetos de outras aplicações estiverem ativos no formulário.

Checked. Seleciona se você quer que apareça uma marca de check antes do item de menu.

Enabled. Seleciona se você quer que o item de menu responda aos eventos ou não. Desabilita ou habilita o item de menu.

Visible. Seleciona se você quer que o item de menu esteja visível ou não.

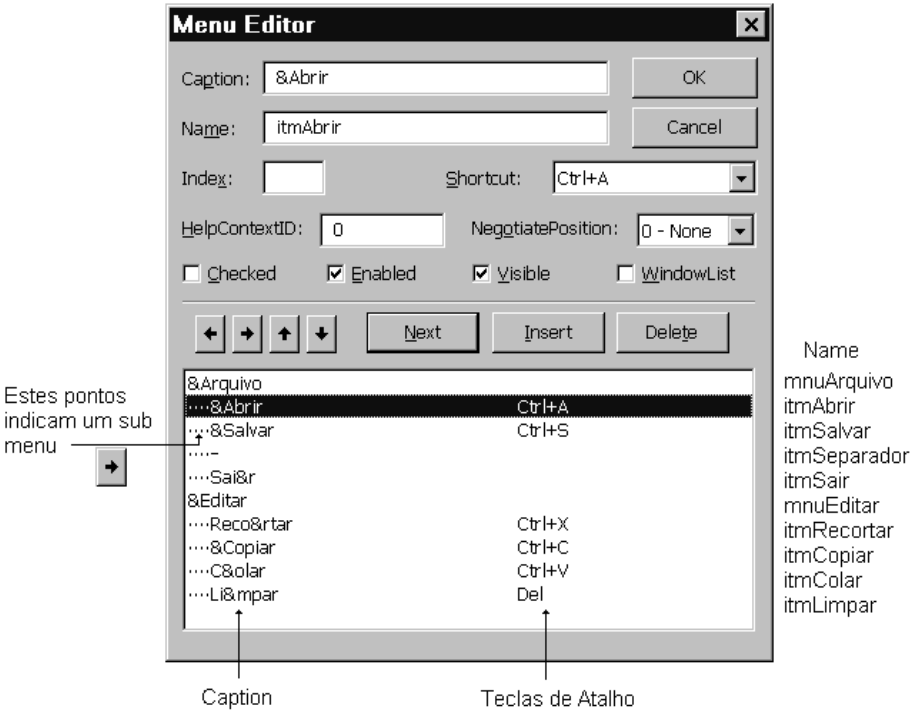
 Use esses botões para mudar o nível do item no menu, criando submenus. Podem ser criados até quatro níveis de submenus.

 Use esses botões para mudar a posição do item de menu.

Botões:

Next Move a seleção para a próxima linha.
Insert Insere uma linha acima da linha atualmente selecionada.
OK Fecha a Menu Editor e aplica todas as mudanças efetuadas.
Cancel Fecha a Menu Editor e cancela as mudanças efetuadas.

O nosso menu deverá ficar da forma mostrada abaixo.

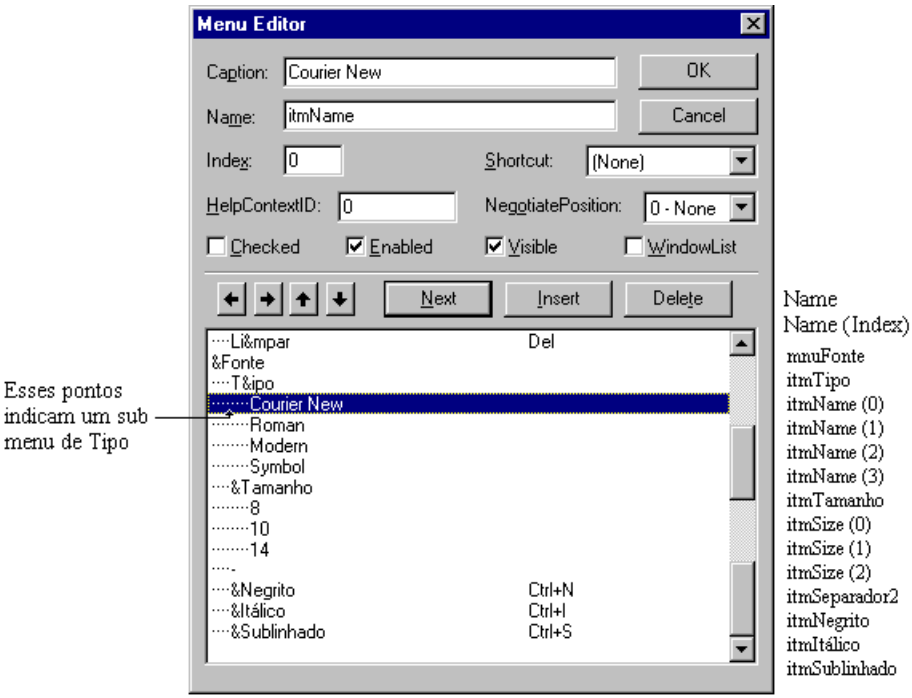


Para os itens: itmRecortar, itmCopiar, itmColar e itmLimpar, deixe a propriedade Enabled desabilitada. Clique em OK, e verifique o formulário, se você der um clique em alguma opção de menu, aparecerão os itens, e se der um clique em algum item de menu, aparecerá a janela de código deste item.

O nosso Bloco de Notas terá a opção de alterar o nome da fonte, o tamanho e a aparência das letras através de menu.

No menu, cada item de nome da fonte ou tamanho, terá a mesma função, ou seja, mudar as propriedades do txtTexto. Neste caso, criaremos um mesmo procedimento para vários objetos utilizando para isso o recurso de Control Array. No projeto de jogo da velha usamos Control Array para os botões, e automaticamente o Index foi incrementado, para o menu, o incremento do Index não é automático, então teremos que fazê-lo manualmente.

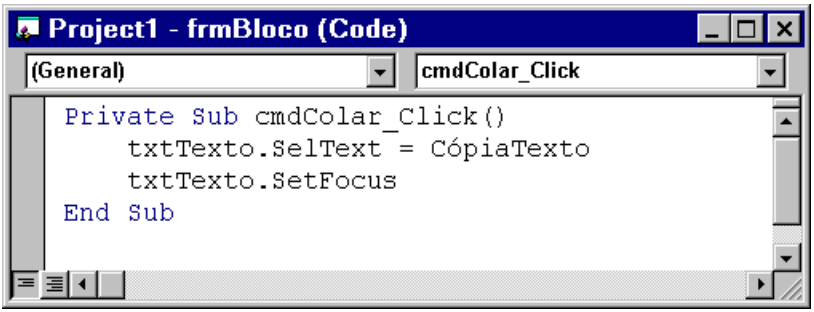
Vamos acessar novamente a Menu Editor... e complementá-la.



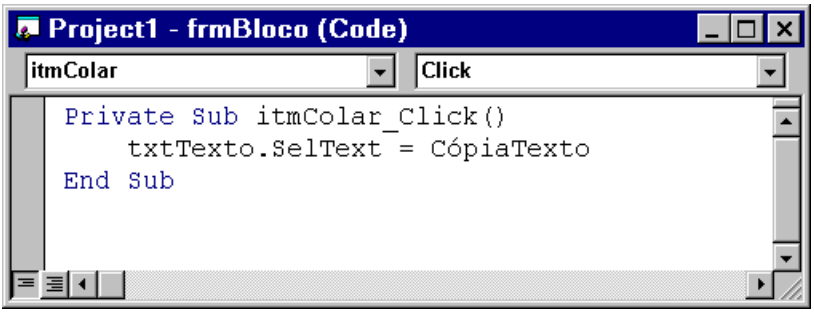
Agora, nosso menu está pronto, clique em OK e verifique no formulário a apresentação. Note que nos itens Tipo e Tamanho aparece uma seta para a direita indicando outro menu de opções.
Nós tínhamos rotinas para Copiar, Recortar e Limpar mas na forma de botões, agora que temos na forma de menu, não precisamos mais dos botões. Vá até o formulário, selecione cada um desses botões e delete-o, ao fazer isso, o código associado irá para a seção general do formulário.

Selecione cada uma destas rotinas e altere o seu nome para que elas estejam associadas ao menu.

Nós não precisaremos mais devolver o foco para o txtTexto, porque ao fazer uma seleção no menu, ele desaparece, então o foco volta para o “único” objeto do formulário - txtTexto.



Após alterar o nome do procedimento, tecle a seta de direção para baixo e o VB irá alterar o nome do objeto e do Evento.



Faça o mesmo para os outros dois “ex-botões”, alterando o código aonde são feitas referências a estes botões.

Note que teremos que acrescentar mais uma linha para itmCopiar.

Private Sub itmLimpar_Click( )
    txtTexto.Text = “”
    itmRecortar.Enabled = **False**
    itmLimpar.Enabled = **False**
    itmCopiar.Enabled = **False**
End Sub

Private Sub itmRecortar_Click( )
    CópiaTexto = txtTexto.SelText
    txtTexto.SelText= “”
    itmColar.Enabled = **True**
End Sub

```
Private Sub txtTexto_Change ( )
        itmRecortar.Enabled = True
        itmLimpar.Enabled = True
        itmCopiar.Enabled = True
End Sub
```

Para construir o procedimento itmCopiar_Click, utilize o recurso de copiar (Ctrl+C), o procedimento itmRecortar_Click, e colar (Ctrl+V). Logo após, delete a linha que apaga o texto em txtTexto.

```
Private Sub itmCopiar_Click ( )
        CópiaTexto = txtTexto.SelText
        itmColar.Enabled = True
end Sub
```

Salve e execute o projeto, verifique se os itens do menu Editar, os únicos que funcionam, ficam habilitados e desabilitados.

Faremos agora o código para os outros objetos.

```
Private Sub itmNome_Click (Index As Integer)
        Select Case index
                Case 0
                        txtTexto.Font.Name = "CourierNew"
                Case 1
                        txtTexto.Font.Name = "Roman"
                Case 2
                        txtTexto.Font.Name = "Modern"
                Else
                        txtTexto.Font.Name = "Symbol"
        End Select
        For i% = 1 To 3
                itmNome(i%).Checked = False
        Next i%
        itmNome(Index).Checked = True
End Sub
```

Quando damos um clique num item de Nome da Fonte, é iniciado o procedimento itmName_Click, que recebe o Index do item acionado, esse Index é armazenado em uma variável de nome Index - (Index As Integer). Todo texto é exibido no novo formato e a marca de verificação (Checked) aparece ao lado do nome da fonte selecionada.

A declaração **Select Case** executa diferentes blocos de declarações dependendo do valor do Index.

Estrutura:

```
Select Case palavra teste
    Case lista de palavras 1
        declarações 1
    Case lista de palavras 2
        declarações 2
End Select
```

A declaração **For**...**Next** retira o Check do item anteriormente selecionado, é mais fácil retirar de todos, a ter que procurar qual o item que tem o Check e depois retirá-lo. E depois colocamos Check no item que foi clicado - itmName(Index).Checked = True.

```
Private Sub itmSize_Click (Index As Integer)
    txtTexto.Font.Size = Val(itmSize(Index).Caption)
    For i% = 0 To 2
        itmSize(i%).Checked = False
    Next i%
    itmSize(Index).Checked = True
End Sub
```

Para o procedimento itmSize_Click podíamos também usar a declaração Select Case, mas no exemplo é utilizada a propriedade Caption do item selecionado para alterar o tamanho da fonte.

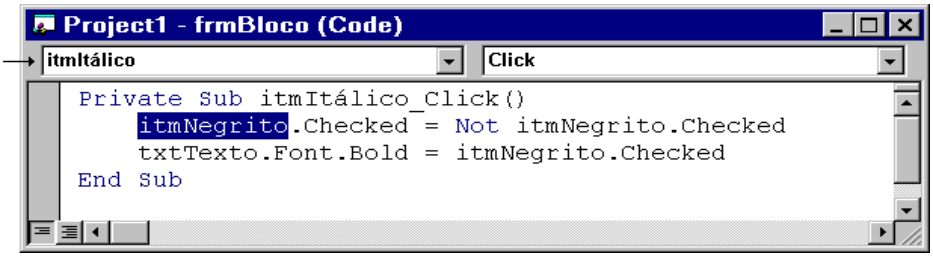
Os itens Negrito, Sublinhado e Itálico quando estiverem selecionados, deverão apresentar um Check ao seu lado. Esta propriedade será utilizada para alterar ela mesma, e depois alterar a apresentação da fonte.

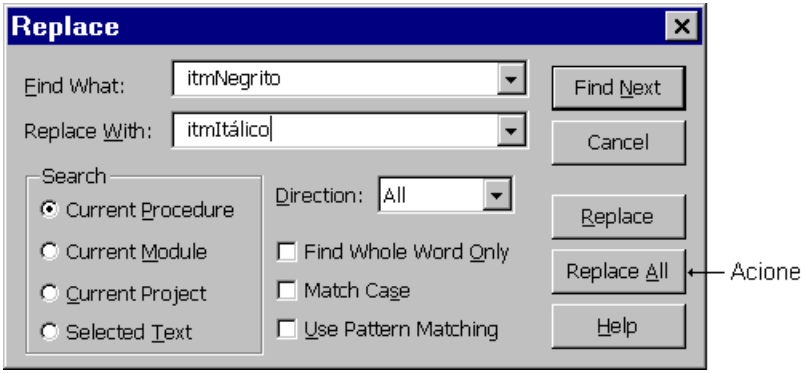
```
Private Sub itmNegrito_Click( )
        itmNegrito.Checked = Not itmNegrito.Checked
        txtTexto.Font.Bold = itmNegrito.Checked
End Sub
```

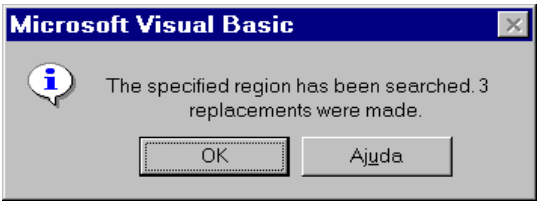
No início itmNegrito.Checked = False, quando o usuário der um clique em itmNegrito, o valor da propriedade será o inverso do que era - itmNegrito.Checked = True, e vice-versa, utilizando-se o operador lógico **Not**.

O mesmo procedimento é utilizado para Itálico e Sublinhado.

Copie o código da procedure itmNegrito_Click() e cole na procedure do itmItálico e itmSublinhado, feito isto utilize o recurso de substituir (**Replace**), do menu Edit.








```
Private Sub itmItálico_Click ( )
        itmItálico.Checked = Not itmItálico.Checked
        txtTexto.Font.Italic = itmItálico.Checked
End Sub

Private Sub itmSublinhado_Click ( )
        itmSublinhado.Checked = Not itmSublinhado.Checked
        txtTexto.Font.Underline = itmSublinhado.Checked
End Sub
```

Finalize a execução utilizando o procedimento de evento itmSair_Click.

```
Private Sub itmSair_Click ( )
        End
End Sub
```

Salve e execute o projeto testando todos os itens, somente o menu Arquivo com as opções Abrir e Salvar, ainda não estará ativos.

Existem objetos que não são incorporados ao formulário, mas possuem propriedades e métodos associados, o **ClipBoard** é um desses objetos. Nós utilizaremos o **ClipBoard** para armazenar o texto que foi Recortado ou Copiado substituindo a variável **CópiaTexto** utilizada anteriormente.

O **ClipBoard** é a própria área de transferência do Windows, e possui os seguintes métodos:

Clear - Limpa o conteúdo do ClipBoar
GetData - Retorna um gráfico do ClipBoard
GetFormat - Retorna um valor indicando qual o tipo de dado do ClipBoard
GetText - Retorna um texto do ClipBoard
SetData - Grava no ClipBoard um elemento gráfico
SetText - Grava no ClipBoard um texto

Para trabalhar com o ClipBoard no projeto Bloco de Notas, altere os seguintes procedimentos:

```
Private Sub itmColar_Click ( )
        if Clipboard.GetFormat (1) Then
                txtTexto.SelText = Clipboard. GetText (1)
        Else
                MsgBox "Não há texto no ClipBoard", 64, "Erro"
        End If
End Sub
```

O procedimento itmColar_Click verifica se há realmente um texto no ClipBoard antes de colar no quadro de texto txtTexto, caso não tenha um texto, é exibida uma mensagem de erro.

```
Private Sub itmRecortar_Click ( )
        Clipboard.SetText txtTexto.SelText
        txtTexto.SelText = ""
        itmColer.Enabled = True
End Sub

Private Sub itmCopiar_Click ( )
        Clipboard.SetText txtTexto.SelText
        itmColar.Enabled = True
End Sub
```

Para obter um melhor resultado, acrescente no procedimento mnuEditar_Click a verificação da existência ou não de texto no ClipBoard.

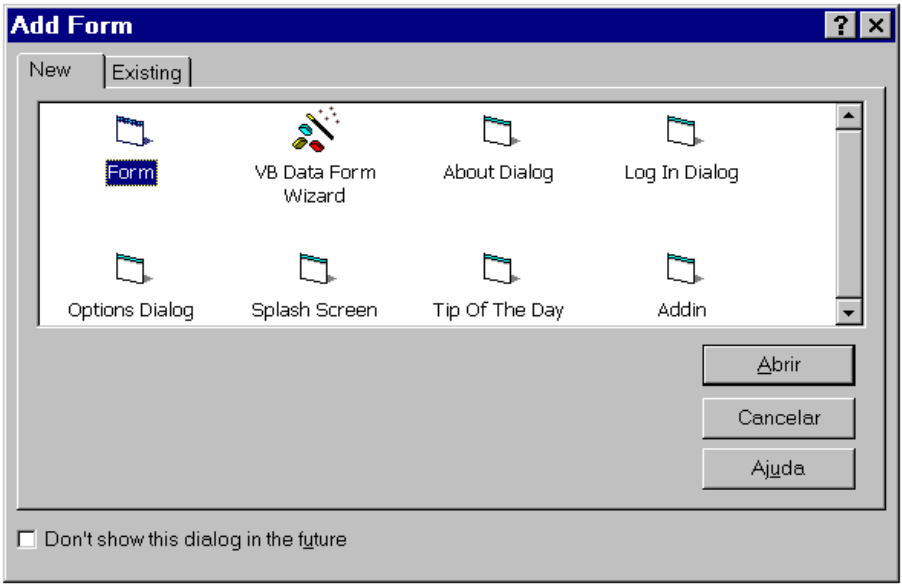
```
Private Sub mnuEditar_Click ( )
        If Clipboard.GetFormat (1) Then
                itmColar.Enabled = True
        End If
End Sub
```

Salve e Execute o projeto. Para testar o funcionamento do ClipBoard, abra um editor de texto enquanto executa o projeto, copiando e colando textos entre eles. Faça um desenho no Paint do Windows, Copie e tente colar no Bloco de Notas, e verá que a mensagem "Não há texto no ClipBoard" aparecerá.

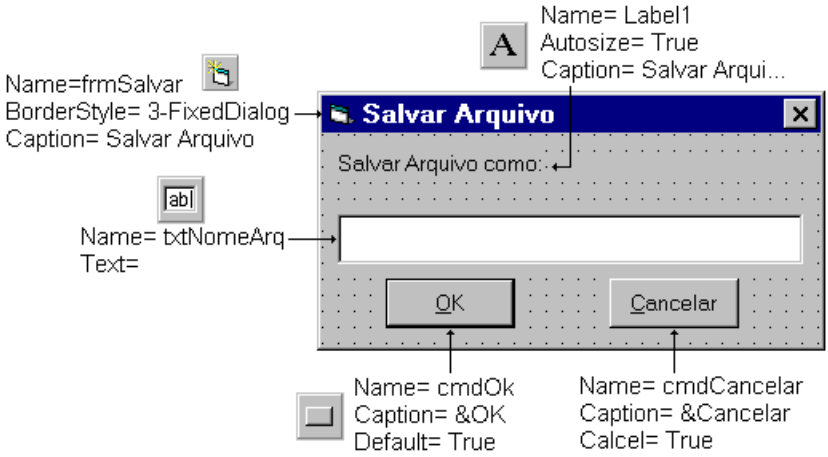
SALVANDO E ABRINDO ARQUIVOS

Os itens **Abrir** e **Salvar** do menu **Arquivo** quando selecionados em aplicações para Windows, abrem outras janelas. Em nosso projeto de Bloco de Notas estas opções também irão abrir outras janelas ou formulários.

Primeiro vamos criar a janela de **Salvar Arquivo**. Por enquanto nosso projeto possui apenas um formulário chamado frmBloco, para criar mais um formulário - frmSalvar, escolha a opção Add Form do menu Project ou dê um clique no botão da barra de ferramentas, para inserir um novo formulário ao projeto. Aparecerá então o quadro de diálogo Add Form para escolhermos o tipo de formulário a ser adicionado.



Escolha o tipo Form, e adicione neste novo formulário os objetos como mostrado abaixo. Altere as propriedades dos objetos:



Digite o código para os botões OK e Cancelar:

```
Private Sub cmdOk_Click ( )
        On Error GoTo Erro
        Open txtNomeArq.Text For Output As #1
        Print #1, frmBloco.txtTexto.Text
        Close #1
        frmSalvar.Hide
        Exit Sub
Erro:
        MasBox “Erro de Arquivo”, 48, “Bloco de Notas”
        Close #1
End Sub
```

A declaração **Open txtNomeArq.Text For Output As #1**, é utilizada para abrirmos um arquivo do tipo seqüencial. Sua sintaxe é a seguinte:

Open arquivo **For** modo **As** #número

Onde:

Arquivo - nome do arquivo a ser aberto
Modo - a maneira como o arquivo será aberto. Que pode ser:

Append (Adicionar): Adiciona mais conteúdo no final de um arquivo do tipo seqüencial.
Input (Entrada): Abre um arquivo do tipo seqüencial para leitura.

Output (Saída): Abre um arquivo do tipo seqüencial para escrita.
Random (Aleatório): Abre um arquivo do tipo de acesso aleatório, para leitura ou gravação

Número - Associa um número ao arquivo como referência para a aplicação. Pode variar de 1 até 511. Ou seja, podemos ter até 511 arquivos abertos ao mesmo tempo.

A declaração **Print #1, frmBloco.txtTexto.Text**, escreve o conteúdo do quadro de texto txtTexto do formulário frmBloco no arquivo que foi aberto como número 1. Sua sintaxe é a seguinte:

Print # número**,** expressão

Número - número com o qual o arquivo que queremos escrever nele, foi aberto na declaração Open.

Expressão - cadeia de caracteres, números ou não, que serão escritos no arquivo.

Com a declaração **Close #1**, fechamos nosso arquivo após a gravação do dados. Caso não fosse fornecido o número do arquivo a ser fechado, a declaração Close fecharia todos os arquivo abertos.

O método **Hide**, esconde um formulário mas não o descarrega da memória. Para que o formulário saia da memória e desapareça, usamos a declaração Unload. Ex: Unload frmSalvar

A declaração **On Error** desvia a rotina do programa para um tratamento do erro. Caso esta declaração não exista, e ocorra um erro no momento de salvar o arquivo, o VB gera uma mensagem de erro e pára a execução do programa, e isto é muito desagradável para o programador frente ao usuário. No nosso projeto, caso ocorra um erro, será mostrada uma mensagem e encerrado o procedimento.

```
Private Sub cmdCancelar_Click ( )
    frmSalvar.Hide
End Sub
```

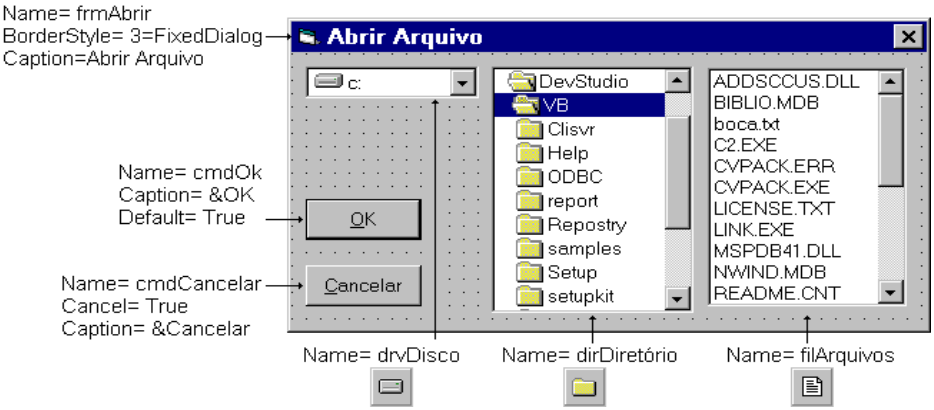
Para que o formulário frmSalvar apareça, devemos digitar o procedimento abaixo associado ao objeto itmSalvar do frmBloco.

```
Private Sub itmSalvar_Click ( )
        frmSalvar.Show 1
End Sub
```

O método **Show** mostra um formulário. O número **1** após Show, indica que o formulário a ser mostrado será do tipo **Modal**, ou seja, não poderemos alternar entre janelas antes de fechá-lo. Caso queiramos que seja alternado, basta informar o número 0 após o método Show.

O método Show antes de mostrar o formulário, carrega-o na memória. Para que o formulário seja carregado na memória e não apareça usamos a declaração Load.
Ex: Load frmSalvar.

Agora, vamos construir o formulário para abrir um arquivo - **frmAbrir**. Primeiro insira um novo formulário, em seguida, coloque os objetos alterando suas propriedades conforme figura a seguir.



Agora, nosso projeto consta de três formulários.

Selecione o formulário frmBloco e dê um clique na opção Abrir do menu Arquivo, veremos a janela de código para o procedimento de evento **itmAbrir_Click**. Que exibirá o formulário para a escolha de um arquivo a ser aberto, de acordo com o exemplo a seguir.

```
Private Sub itmAbrir_Click ( )
        frmAbrir.Show 1
End Sub
```

As caixas de Lista de Unidade, Diretório e Arquivos ainda não estão integradas, ou seja, caso alteremos o diretório, o conteúdo da caixa de arquivos não se altera. Experimente executar o projeto e escolha a opção Abrir no meu Arquivo para verificar.

Para que as três caixas fiquem em sincronismo, nós devemos alterar a propriedade **Path** das caixas de diretório e de arquivo, quando houver mudança de escolha em alguma delas. Esta propriedade determina o caminho absoluto atual incluindo o nome do drive. Ex: "C:\VB6\BITMAPS", indica que o diretório atual é o subdiretório BITMAPS do diretório VB6 na unidade C: .

```
Private Sub drvDisco_Change ( )
        dirDiretório.Path = drvDisco.Drive
End Sub
```

No código acima, toda vez que houver alteração (evento **Change**) na unidade atual, será chamado o procedimento que altera a propriedade Path de dirDiretório que será o diretório raiz da unidade selecionada na caixa de unidades, pois na propriedade Drive teremos a unidade atualmente selecionada.

Feito isto, as caixas de lista de unidades e diretório estarão sincronizadas, falta a caixa de lista de arquivo. Para sincronizá-la, é só transferir a propriedade Path da dirDiretório para a propriedade da filArquivos, como a seguir:

```
Private Sub dirDiretório_Change ( )
        filArquivos.Path = dirDiretório.Path
End Sub
```

As três caixas estão agora sincronizadas, execute o projeto e alterne entre unidades e diretórios.

Nas aplicações para Windows, quando queremos abrir um arquivo de uma janela do tipo Abrir, podemos fazê-lo de duas formas:

1. Dando um clique no arquivo e outro no botão Ok.
2. Dando um duplo clique no arquivo selecionado.

No VB não precisamos de dois procedimentos para estas duas funções. Um evento pode dar início a outro evento.

No projeto de Bloco de Notas, quando for dado um duplo clique no nome do arquivo, este evento dará partida ao seu procedimento (a seguir), que chamará o evento cmdOk_Click.

```
Private Sub filArquivos_DblClick ( )
    cmdOk_Click
End Sub
```

Entre com o seguinte código para o botão cmdOk:

```
Private sub cmdOk_click ( )
      On Error GoTo Erro
      If Right$(dirDiretório, 1) = "\" Then
            Arquivo$ = dirDiretório.Path + filArquivos.filename  Else
            Arquivo$ = dirDiretório.Path + "\" +
            filArquivos.filename
      End If
      Open Arquivo$ For Input As #1
      frmBloco.txtTexto.Text = Input$ (LOF(l) #1)
      Close  #1
      frmAbrir.Hide
Exit Sub

Erro:
      MsgBox  "Erro de Arquivo", 49,  "Erro"
      Close  #1
End Sub
```

No código, a função **Right$ - Right$(dirDiretório.Path,1)**, verifica qual é o último caractére do caminho do diretório selecionado. Ela retorna **n** caracteres da direita para a esquerda de um string, e sua sintaxe é a seguinte:

Right$(expressão, n**)**

Onde: **expressão** - é uma cadeia de caracteres numéricos ou não.

n - são quantos caracteres se quer retornar.

Ex: txtTexto.Text = Right$(impressão, 3),
Será exibido no quadro de texto as letras **são**.

O código verifica se o caminho atual possui "\" no seu final, caso não exista, ele adiciona à variável Arquivo$ que contém o caminho e nome do arquivo a ser aberto com a declaração Open.
A linha **frmBloco.txtTexto.Text = Input$(LOF(1), #1)**, lê o conteúdo do arquivo que foi aberto com o número 1, transferindo este conteúdo a uma variável ou objeto, no nosso caso para o objeto - txtTexto.

Input$ (n, #número**)**

Onde: **n** - indica o número de caracteres a serem lidos do arquivo.

#número - é o número do arquivo a ser lido.

A função LOF, **Input$(LOF(1),#1)**, retorna o tamanho em bytes de um arquivo aberto.

LOF (número)

Onde: **número** - é o número do arquivo a ser lido.

Falta o botão cancelar, que esconderá o formulário - frmAbrir, sem executar nada.

```
Private Sub cmdCancelar_Click ( )
    frmAbrir.Hide
End Sub
```

Salve e execute o projeto testando todas as opções apresentadas.

MÉTODOS GRÁFICOS

Embora o uso dos métodos gráficos - que permitem desenhos de linhas, círculos e animações sejam complexos, poderá ser divertido e útil para quem deseja sofisticar seus programas. A seguir, conheceremos tais recursos através de exemplos simples.

O sistema de coordenadas do Visual Basic possui o seu ponto de início (0,0) no canto superior esquerdo, ao contrário do que nós estamos acostumados.

Existem no VB, diversos tipos de escala, são elas:

1. Twip = 1440 twips por polegada; 567 twips por centímetro. (Default)
2. Point =72 points por polegada.
3. Pixel = Unidade do monitor ou resolução da impressora.
4. Character = horizontal - 120 twips por unid.; vertical - 240 twips por unid.
5. Polegada
6. Milímetro
7. Centímetro

Estas escalas são definidas para cada objeto gráfico (formulário, picture box e impressora), na propriedade **ScaleMode**.

DESENHO DE PONTO

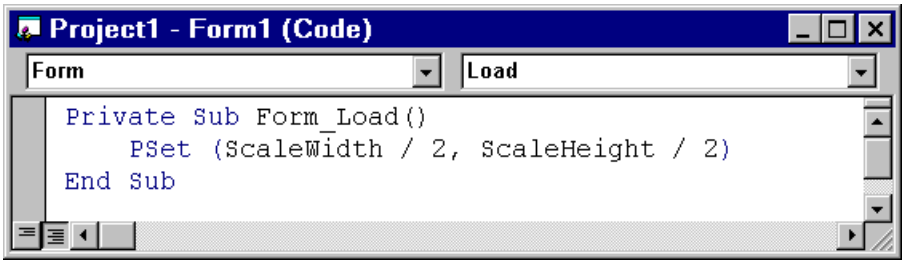
Para desenharmos um ponto utilizamos o método Pset;

objeto.**Pset** [Step] (x,y) [,cor]

Onde:

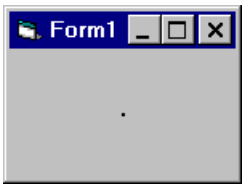
Objeto - objeto onde o ponto será desenhado, se for omitido o ponto será desenhado no formulário corrente.
Step - especifica que as coordenadas serão relativas à posição corrente, propriedades **CurrentX** e **CurrentY**.
x,y - coordenadas horizontal e vertical
Cor - especifica uma cor para o ponto, se for omitida, será usada a cor da propriedade **ForeColor** do objeto.

Inicie um novo projeto, e digite o código a seguir para o evento **Load** do Formulário.

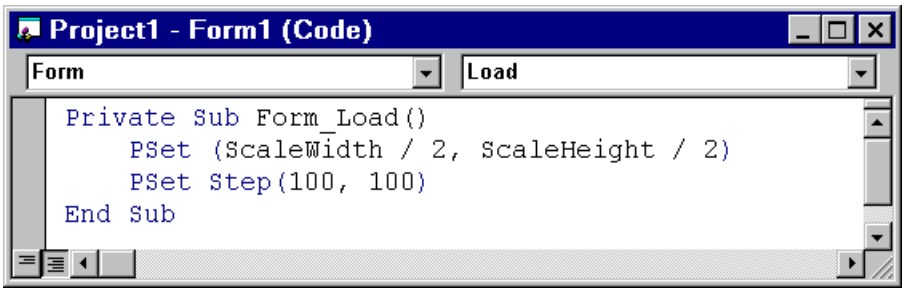


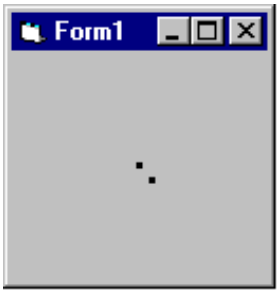
Aperte a tecla F5 para rodar o projeto e observe se algum ponto aparece no centro do formulário.

O ponto não aparece, porque teremos que alterar a propriedade do formulário **AutoRedraw** = **True**. Esta propriedade redesenha um gráfico ou formulário automaticamente quando houver alguma alteração nele ou, encoberto por outra janela. Quando usamos o evento Load para desenhar gráficos, temos que deixar AutoRedraw = True. Altere a propriedade e execute o projeto novamente.

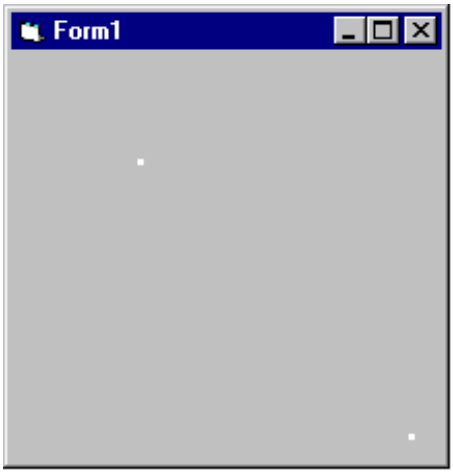


Para desenharmos outro ponto distante do primeiro em 100 Twips, usamos a palavra **Step** (figura abaixo), e para mudarmos a dimensão do ponto, basta alterar o valor da propriedade **DrawWidth** do formulário.





Mude a propriedade do formulário **ScaleMode** = **Point**, e execute novamente o projeto. Caso o ponto não apareça no formulário, basta redimensionar o formulário durante a execução, e observe o aumento da distância entre os pontos com a mudança na escala.



Teste os outros tipos de escala, alterando a propriedade **ScaleMode** do formulário.

CORES

Para determinarmos a cor de um objeto gráfico, temos 3 formas diferentes, são elas:

1. RGB (NRed, NGreen, NBlue), onde NRed, NGreen e NBlue podem variar de 0 a 255.
   Ponto vermelho: Pset (100,100), RGB(255,0,0)

2. Definir a propriedade ForeColor do objeto antes de desenhar o gráfico, não alterando o que já estava desenhado.

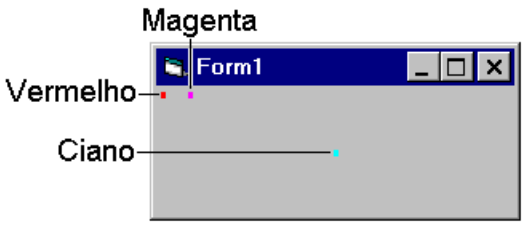
   Ponto Magenta: ForeColor = RGB(226, 0, 127)
   Pset (100,100)

3. Usando a função QBColor, que possui valores inteiros para as 16 cores mais utilizadas.

   Ponto Ciano: PSet (ScaleWidth / 2, ScaleHeight / 2), QBColor(11)

Vamos fazer um exemplo utilizando os três métodos para cor, mas antes certifique-se que a propriedade **DrawMode** esteja com o valor 3-Copy Pen.

```
Private Sub Form_Load ( )
        PSet (50, 50), RGB(255,0,0)              'Vermelho
        ForeColor = RGB (255,0,255)
        PSet Step (200,0)                        'Magenta
        PSet (ScaleWidth/2,ScaleHeight/2), QBColor (11)        'Ciano
End Sub
```



LINHA

Para desenhar linhas, use o método **Line**:

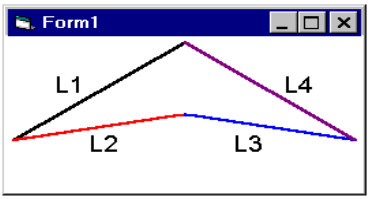
[objeto.] **Line** [[Step$_a$] (x1,y1)] - [Step$_b$] (x2,y2) [, [cor], [B], [F]]

Onde:

Objeto - objeto onde será desenhada a linha (formulário, picture box ou impressora).
Step$_a$ - especifica que as coordenadas são relativas à posição corrente, propriedades CurrentX e CurrentY.
(x1,y1) - coordenadas do ponto inicial.
Step$_b$ - especifica que as coordenadas do ponto final são relativas às coordenadas do ponto inicial.
(x2,y2) - coordenadas do ponto final.
Cor - especifica a cor que a linha será desenhada.
B - opção que desenha um retângulo usando as coordenadas de cantos opostos.
F - especifica que o retângulo será preenchido com a mesma cor usada para desenhar a borda do retângulo. Se não for usada, o retângulo será preenchido com o valor da propriedade FillColor do objeto.

Digite o código abaixo, não esquecendo de alterar as propriedades do formulário: **AutoRedraw** = **True** e **BackColor** = **branco**.

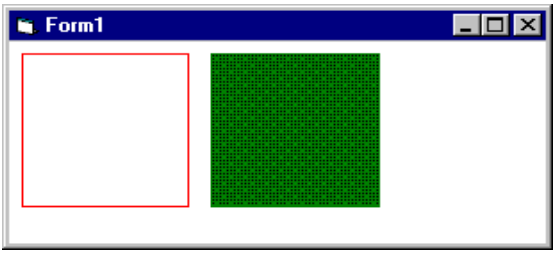
```
Private Sub Form_Load()
    Dim SW As Integer, SH As Integer
    SW = ScaleWidth              'Largura do formulário
    SH = ScaleHeight             'Altura do formulário
    Line (SW/2, 50)-(50, 2*SH/3)                          'L1
    Line Step(0,0)-(SW/2, SH/2), RGB(255,0, 0)            'L2
    Line Step(0,0)-Step(SW/2-50,SH/6), RGB(0,0,255)       'L3
    Line (SW-50, 2*SH/3)-(SW/2,50), RGB(226,0,127)        'L4
End Sub
```



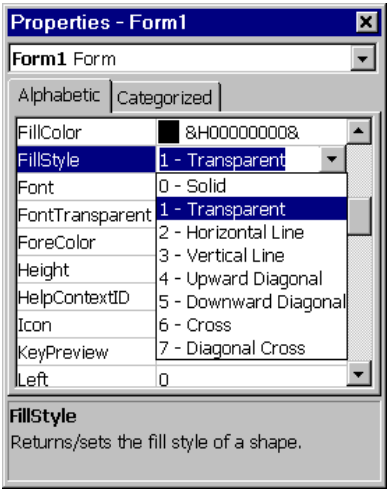
Note que a coordenada do ponto final da segunda linha (SW/2, SH/2) é interpretado como um valor absoluto, enquanto a coordenada de ponto inicial (0,0) usa como referência o CurrentX e CurrentY - (50, 2*SH/3).

Para desenhar um retângulo, usaremos o método Line com a opção B, preenchendo este retângulo, usaremos também a opção F.

```
Private Sub Form_Load ( )
        Line (100,100) - (1500,1500), RGB (255,0,0), B
        Line (1700,100)-Step(1400,1400), RGB(0,100,0), BF
End Sub
```

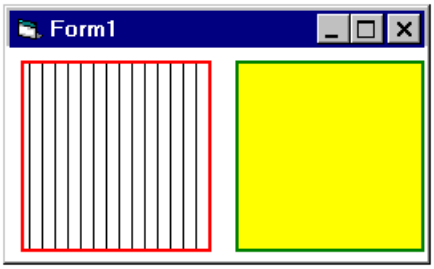


Vamos alterar agora as propriedades de preenchimento do formulário, **FillColor**=RGB(255,255,0) e **FillStyle**=3 - Vertical Line. Estas propriedades determinam a cor e o padrão de preenchimento e podem ser modificadas tanto em tempo de projeto, quanto em tempo de execução.



Entre com o seguinte código:

```
Private Sub Form_Load ( )
        Line (100,100) - (1500,1500), RGB (255,0,0), B
        FillColor = RGB (255,255,0)
        FillStyle = 0
        Line (1700,100)-Step(1400,1400), RGB(0,100,0), B
End Sub
```

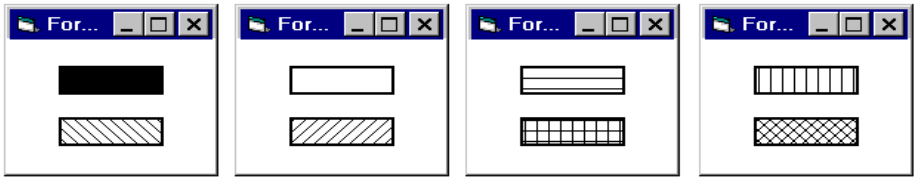


Faça o exemplo abaixo para visualizar todos os tipos de preenchimento;

```
Private Sub Form_Click ( )
        Static I As Integer
        Dim SW As Integer, SH As Integer
        SW = ScaleWidth              'Largura do formulário
        SH = ScaleHeight             'Altura do formulário
        If I > 3 Then End
        Cls                          'Limpa o formulário
        FillStyle = I
        Line (SW/4, SH/5)-(3*SW/4, 2*SH/5),,B
        FillStyle = I + 4
        Line (SW/4, 3*SH/5)-(3*SW/4, 4*SH/5),,B
        I = I + 1
End Sub
```

Neste procedimento, quando damos um clique no formulário, aparecerão dois tipos de preenchimento, até o último, quando o programa será finalizado.

A declaração **Static I As Integer**, declara a variável **I** como um inteiro e o seu valor não é reinicializado junto com o procedimento, o VB armazena o último valor de **I** para ser utilizado na próximo vez que o evento Click ocorrer.



A figura acima ilustra todos os tipos de preenchimento (FillStyle).

Existem ainda as propriedades **DrawStyle** e **DrawWidth**, que veremos a seguir.

DrawStyle

Define o estilo da linha a ser desenhada no objeto gráfico. Pode ser definida tanto em tempo de projeto, quanto em tempo de execução.

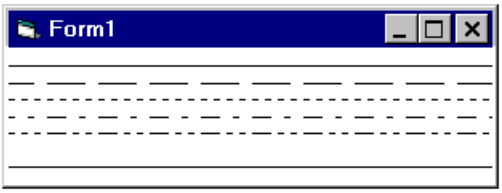
[objeto.] **DrawStyle** = estilo

Estilos: 0 - sólida
1 - tracejada
2 - pontilhada
3 - traço-ponto
4 - traço-ponto-ponto
5 - transparente
6 - interna

Normalmente o VB usa como referência o centro da linha, usando a propriedade DrawStyle=6, ele passa a usar a parte externa da linha como referência.

Entre com o código a seguir para visualizar as opções de **DrawStyle**:

```
Private Sub Form_Click ( )
        Dim SW As Integer, SH As Integer
        SW = ScaleWidth
      SH = ScaleHeight
      For I% = 1 To 7
            DrawStyle = I% - 1
            Line (0, I%*SH/8)-(SW, I%*SH/8)
      Next I%
End Sub
```



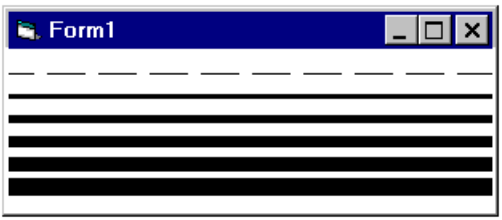
DrawWidth

Define a espessura da linha em pixel, a ser desenha no objeto gráfico e pode variar de 1 até 32.767. Se a propriedade **DrawStyle** estiver entre 1 e 4 e a **DrawWidth** for maior que 1, o estilo passará para sólido (**DrawStyle**=0). Ou seja, não conseguiremos desenhar linhas pontilhadas e tracejadas (modificando a propriedade **DrawStyle**) com espessura maior que 1 pixel.

[objeto.] **DrawWidth** = tamanho

Execute o exemplo abaixo para visualizar as opções:

```
Private Sub Form_Load()
        Dim SW As Integer, SH As Integer
        Dim I As Integer, J As Integer
        SW = ScaleWidth
        SH = ScaleHeight
        DrawStyle = 1
        For I = 1 To 11 Step 2
                DrawWidth = I
                J = Int(I / 2) + 1
                Line (0, J * SH / 7)-(SW, J * SH / 7)
        Next I
End Sub
```



CÍRCULOS

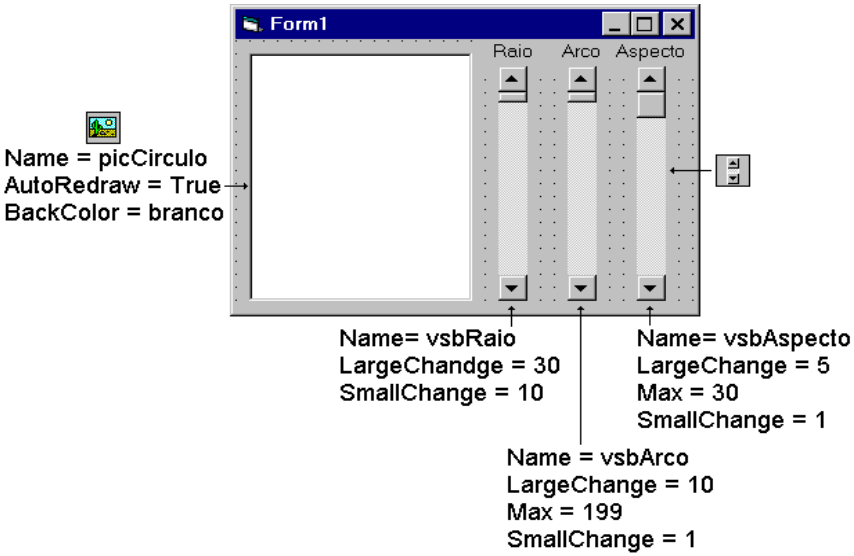
Para desenhar círculos, usamos o método **Circle**. Que desenha círculos, elipses ou arcos em um objeto gráfico.

[objeto,] **Circle** [Step] (x,y), radius , [cor], [start], [end], [aspect]

Onde:

Step - palavra que indica que a coordenada de centro será relativa à posição corrente.
(x,y) - coordenadas de centro do círculo, elipse ou arco.
Radius - raio do círculo, elipse ou arco.
Cor - especifica a cor para o círculo
Start/End - valores que especificam o início e fim do arco a ser desenhado. Valores expressos em radianos, o default é 0 radianos para Start e 2p radianos para End.
Aspecto - indica a proporção entre os raios, sendo na elipse diferente de 1. No círculo perfeito o aspecto é igual a 1.

Faça o projeto Círculo conforme exemplo a seguir, para visualizar o método Circle.



As três **Vertical Scroll Bar** servirão para modificar os parâmetros do método **Circle**. Ou seja, poderemos desenhar um círculo, elipse ou arco dentro do Picture Box. Existem cinco propriedades primárias das barras de paginação em que estamos interessados: **Min**, **Max**, **Value**, **LargeChange** e **SmallChange**.

Onde:

Min - valor numérico atribuído ao lado superior ou esquerdo da barra de paginação.
Max - valor numérico atribuído ao lado inferior ou direito da barra de paginação.
Value - valor correspondente à posição do marcador na barra. Que está entre Min e Max.
LargeChange - indica a quantidade que a propriedade Value deverá variar toda vez que o usuário acionar a barra acima ou abaixo do marcador.
SmallChange - indica a quantidade que a propriedade Value deverá variar quando o usuário acionar as setas da barra de paginação.

O procedimento Form_Load desenhará um círculo no centro Picture Box e depois atribuirá valores às propriedades Value das barras de paginação.

```
Private Sub Form_Load()
        Dim SW As Integer, SH As Integer
        SW = picCirculo.ScaleWidth
        SH = picCirculo.ScaleHeight
        picCirculo.Circle (SW/2, SH/2), SW/10, RGB(255, 0, 0)
        vsbRaio.Max = SW/2
        vsbRaio.Value = SW/10
        vsbArco.Value = 199
        vsbAspecto.Value = 10
End Sub
```

O evento mais utilizado com as barras de paginação, é o evento **Change**, que ocorre toda vez que o usuário move o marcador. E como este evento irá redesenhar o círculo com os novos padrões, devemos criar um procedimento geral de formulário.

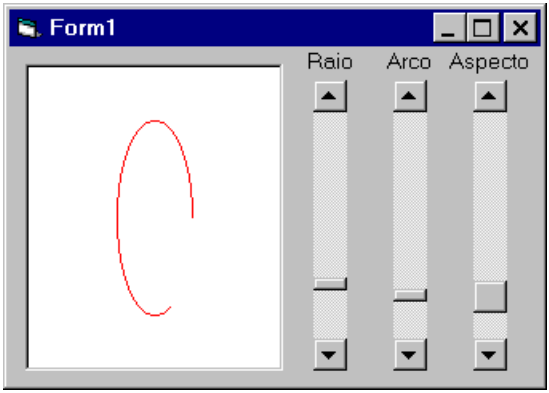
```
Private Sub vsbArco_Change()
   RedesenhaCirculo
End Sub
```

```
Private Sub vsbAspecto_Change()
   RedesenhaCirculo
End Sub

Private Sub vsbRaio_Change()
   RedesenhaCirculo
End Sub
```

Digite o código do procedimento geral de formulário RedesenhaCirculo.

```
Public Sub RedesenhaCirculo( )
        Dim SW As Integer, SH As Integer
        SW = picCirculo.ScaleWidth
        SH = picCirculo.ScaleHeight
        picCirculo.Cls
        Raio = vsbRaio.Value
        Fim = vsbArco.Value / 100 * 3.1415
        Aspecto = vsbAspecto.Value / 10
        picCirculo.Circle(SW/2,SH/2),Raio,RGB(255,0,0),0,Fim, Aspecto
End Sub
```

O procedimento acima irá redesenhar o círculo toda vez que o usuário mover o marcador de qualquer uma das barras de paginação. Para redesenhar o círculo, ele utilizará a propriedade Value das barras como parâmetros.



CARREGANDO FIGURAS

Podemos carregar seis tipos de arquivos de figura:

BMP	(mapa de bits)
GIF	(graphical interchange format)
ICO	(ícones)
JPEG	(joint photographics experts group)
RLE	(run-length encoded)
WMF	(windows metafiles)

Inserimos figuras em, formulários, quadros de figura e controle imagem. Existem diferenças entre o controle imagem e o quadro de figura:

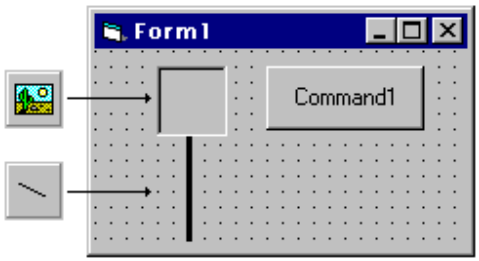
Controle Imagem - Não faz linhas nem círculos.
Muda automaticamente de tamanho de acordo com a figura.
Ajusta a figura de acordo com o seu tamanho se a propriedade **Strech** (esticar) = **True**.
Permite redimensionar a figura depois de carregada.
Requer menos recursos do sistema, mostrando figura mais rapidamente

Para carregar uma figura em um objeto, use a função **LoadPicture** atribuindo o nome do desenho na propriedade **Picture** do objeto.

objeto.Picture = **LoadPicture** (“caminho e nome da figura”)

A propriedade Picture pode ser modificada tanto em tempo de projeto quanto em tempo de execução.

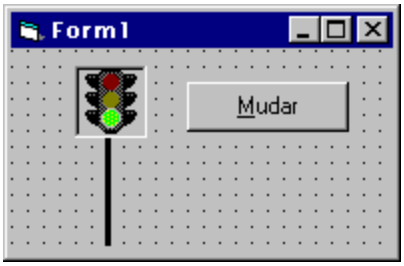
Como exemplo, vamos construir o projeto semáforo.



Altere a propriedade **Autosize** = **True** do Pictutre Box, para que ele ajuste o seu tamanho ao da figura que será carregada.

Selecione a propriedade **Picture** do Pictutre Box e escolha a figura "C:\...\TrafficA.ico" no quadro de diálogo Load Picture.

Ajuste as coordenadas (propriedades X1, X2, Y1 eY2) e a largura do objeto Line, para um acerto com o Quadro de Figura.



Quando o usuário der um clique no botão Mudar, o semáforo mudará de estado. Vamos agora construir o código para este evento.

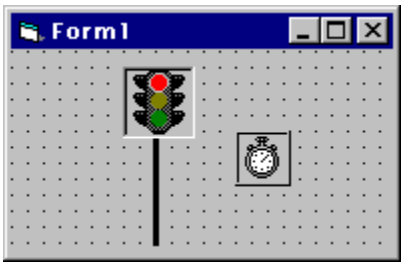
```
Private Sub cmdMudar_Click()
    Static Sinal As Integer
    If Sinal = 0 Then
        Picture1.Picture = LoadPicture("C:\...\TrafficB.ico")
        Sinal = 1
    ElseIf Sinal = 1 Then
        Picture1.Picture = LoadPicture("C:\...\TrafficC.ico")
        Sinal = 2
    ElseIf Sinal = 2 Then
        Picture1.Picture = LoadPicture("C:\...\TrafficA.ico")
        Sinal = 0
    End If
End Sub
```

No exemplo anterior, toda vez que o usuário selecionar o botão Mudar, uma nova figura será carregada no Picture1. Mas poderemos também colocar no formulário três Picture Box, cada um com uma figura diferente e ir modificando a propriedade **Visible** desses quadros.

No nosso projeto de semáforo, insira mais dois quadros de figura e posicione-os no mesmo local do primeiro, um em cima do outro definindo as propriedades **Picture** e **Visible**, como segue:

```
Picture1.Picture = " C:\...\TrafficA.ico"
Picture1.Visible = True
Picture2.Picture = " C:\...\TrafficB.ico"
Picture2.Visible = False
Picture3.Picture = " C:\...\TrafficC.ico"
Picture3.Visible = False
```

E também vamos eliminar o botão de comando, inserindo em seu lugar o controle **Timer** - . Este controle gera o evento **Timer** a intervalos de tempo determinados pela propriedade **Interval**. Esta propriedade é expressa em milisegundos, se quisermos um intervalo de 2 segundos: Interval = 2000.



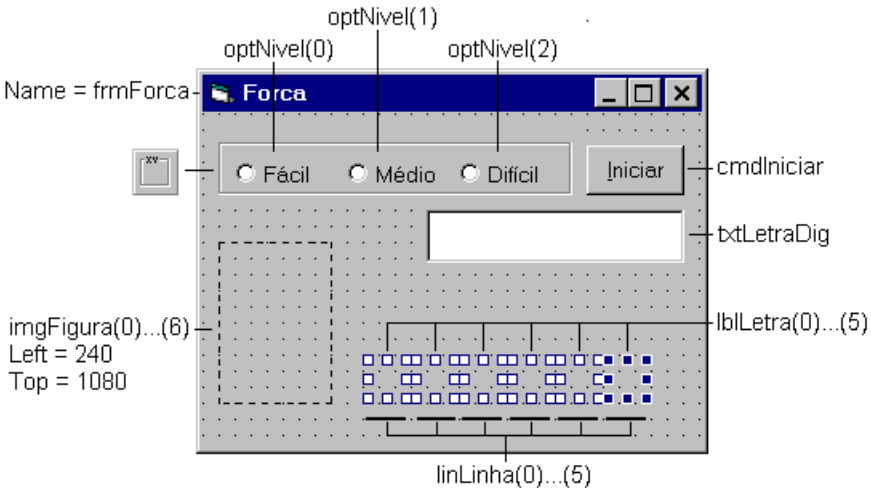

```
Private Sub Timer1_Timer()
    Static Sinal As Integer
    If Sinal = 0 Then
        Picture1.Visible = True
        Picture3.Visible = False
        Sinal = 1
    ElseIf Sinal = 1 Then
        Picture2.Visible = True
        Picture1.Visible = False
        Sinal = 2
    ElseIf Sinal = 2 Then
        Picture3.Visible = True
        Picture2.Visible = False
        Sinal = 0
    End If
End Sub
```

Execute o projeto alterando a propriedade Interval do Timer1, observando as mudança

EXEMPLO IV - JOGO DA FORCA

Neste exemplo você aprenderá a trabalhar com botões de opção, controle Image, evento de teclado e evento de mouse, enquanto constrói um divertido e simples jogo.

Construa o fomulário seguindo a figura abaixo. Tenha o cuidado de inserir primeiro o controle **Frame** e só depois os botões de opção dentro dele.



No Windows usamos botões de opção quando temos que escolher entre opções mutuamente excludentes, ou seja, apenas uma das opções pode estar selecionada. Em um formulário podemos ter somente um botão de opção selecionado de cada vez. Caso o projeto necessite de mais de um grupo de seleção, utilizamos o objeto Frame para separar os diferentes grupos.

O jogo funciona da seguinte forma:

1. O usuário escolhe o nível de dificuldade que deseja.
2. Clica no botão Iniciar.
3. O programa escolhe uma palavra de sua matriz.
4. O usuário digita uma letra qualquer.

5. Esta letra é mostrada na caixa de texto.
6. Se a letra digita estiver na palavra escolhida pelo programa, ela é mostrada no Label correspondente, senão aparece uma parte da figura no controle Image.
7. Se o usuário acertou a palavra antes de seis tentativas, exibe mensagem que ganhou o jogo, senão a figura se completa e perde a cabeça após o usuário fechar uma caixa de mensagem.

· Caso o usuário queira uma dica, basta pressionar o botão do mouse junto com a tecla Ctrl sobre uma Label vazia, que será mostrada a letra desta posição.

Carregue as figura Fig0.gif até Fig6.gif nos controles Image seguindo o index de cada controle. Estas figuras GIF possuem transparências o que facilita a montagem da imagem em partes separadas e sobrepostas.

Começamos criando os procedimentos gerais do formulário:

```
Public Sub ApagarFigura()
'Apaga a figura a move a "cabeça" para a posição original
    For I = 0 To 6
        imgFigura(I).Visible = False
    Next I
    imgFigura(6).Move 240, 1080
    txtLetraDig.Text = ""
End Sub
```

Para mover um objeto usamos o método Move que possui a seguinte sintaxe;

objeto.**Move** left[,top[,width[,height]]]

Onde:

left - coordenada horizontal
top - coordenado vertical
width - nova largura
height - nova altura

```
Public Sub MoverCabeça()
'Move a "cabeça" para baixo, com uma curva exponencial
    Dim X As Single, Y As Single
   For X = 0 To 840 Step 30
      Y = (X ^ 2 * 0.002) + 1080
      imgFigura(6).Move (X + 240), Y
    Next X
End Sub

Public Sub ApagarLetraLinha()
'Apaga as letras e as linhas
   For I = 0 To 5
      lblLetra(I).Caption = ""
      linLinha(I).Visible = False
    Next I
   txtLetraDig.Text = ""
End Sub
```

Declare as variáveis gerais para o formulário:

```
Private Palavras(2, 4) As String      'Matriz com as palavras
Private Palavra As String             'Palavra selecionada
Private Nivel As Integer                    'Nível escolhido
Private Letras As Integer     'Número de letras da palavra
Private Jogada As Integer             'Número da jogada
Private AchouQuantas As Integer   'Número de letras corretas
```

Quando o programa é inicializado ocorre o evento **Load** para o formulário. É neste procedimento de evento que serão carregados os valores da matriz Palavras(2,4), onde as colunas representarão o nível de dificuldade e as linhas terão as palavras de cada nível.

Também neste procedimento é feita a inicialização dos controles **Label**, **Image** e **Option Button**.

```
Private Sub Form_Load()
'Fácil
      Palavras(0, 0) = "SAL"
      Palavras(0, 1) = "BACIA"
```

```
        Palavras(0, 2) = "CABRA"
        Palavras(0, 3) = "COLHER"
        Palavras(0, 4) = "ABRIR"
'Médio
        Palavras(1, 0) = "BACURI"
        Palavras(1, 1) = "BELEZA"
        Palavras(1, 2) = "CIDRA"
        Palavras(1, 3) = "LAGO"
        Palavras(1, 4) = "RIFA"
'Difícil
        Palavras(2, 0) = "ACEIRO"
        Palavras(2, 1) = "CHOUPO"
        Palavras(2, 2) = "REBOAR"
        Palavras(2, 3) = "MITRA"
        Palavras(2, 4) = "HERTZ"
        Randomize           'Inicia gerador de números aleatórios
        Nivel = 0                   'Seleciona nível Fácil
        optNivel(0).Value = True  'Seleciona opção do nível Fácil
        ApagarFigura
        ApagarLetraLinha
End Sub
```

Após a inicialização do programa o usuário terá que escolher o nível de dificuldade, dando um clique em um dos botões de opção, armazenando o valor do nível na variável Nivel.

```
Private Sub optNivel_Click(Index As Integer)
'O parâmetro Index contém o valor da propriedade
'Index do botão escolhido
    Nivel = Index
End Sub
```

Quando o usuário der um clique no botão Iniciar, o programa gera um número aleatório para a escolha da linha na matriz Palavras ( ). A linha escolhida junto com o nível escolhido representarão a palavra escolhida, depois disso o programa mostrará somente as linhas que terão letras sobre elas.

```
Private Sub cmdIniciar_Click()
    Dim NPal As Integer, I As Integer
    NPal = Int(Rnd * 5)
```

```
    Palavra = Palavras(Nivel, NPal)
    'A função Len retorna a quantidade de caracteres
    Letras = Len(Palavra)
    For I = 0 To Letras - 1
        linLinha(I).Visible = True
    Next I
    txtLetraDig.SetFocus
    Jogada = 0
    AchouQuantas = 0
    ApagarFigura
End Sub
```

EVENTO DO TECLADO

O próximo passo será o tratamento das letras digitas pelo usuário. Utilizamos o evento de teclado **KeyPress** que ocorre toda vez que uma tecla é pressionada e depois liberada.

Este evento estará ligado à caixa de texto txtLetraDig e será a alma do programa. Ele recebe a letra digitada, verifica se existe na palavra escolhida, se existir mostra a letra, se não mostra uma parte da figura e aumenta a jogada. Quando terminadas as chances move a cabeça para fora do corpo - na figura.

A função **Mid$**, retorna um String que é parte de outro String.

Mid$ (string, começo [,comprimento])

Onde:

string - é a cadeia de caracteres original
começo - especifica a partir de qual caractere começará a extração.
comprimento - indica quantos caracteres se deseja extrair.

Ex: Palavra$ = Mid$ ("Constituição",2,4)
equivale a: Palavra$ = "onst"

```
Private Sub txtLetraDig_KeyPress(KeyAscii As Integer)
'O parâmetro KeyAscii envia o valor da tecla pressionada
Dim Letra As String      'Armazena a letra a verificar
```

```
Dim AscLetra As Integer      'Contém o código ASCII da letra
Dim Achou As Boolean'Pode ser True ou False
    Achou = False
    For I = 1 To Letras
        Letra = Mid$(Palavra, I, 1)
        'Asc retorna o valor na tabela ASCII da Letra
        AscLetra = Asc(Letra)
        If (KeyAscii=AscLetra) Or (KeyAscii-32=AscLetra) Then
            lblLetra(I - 1).Caption = Letra
            Achou = True
            AchouQuantas = AchouQuantas + 1
            If AchouQuantas = Letras Then
                MsgBox "Você acertou"
                ApagarLetraLinha
            End If
        End If
    Next I
    If Not Achou Then
        imgFigura(Jogada).Visible = True
        If Jogada = 5 Then
            imgFigura(0).Visible = False
            imgFigura(6).Visible = True
            MsgBox "Você errou"
            MoverCabeça
            ApagarLetraLinha
        End If
        Jogada = Jogada + 1
    End If
End Sub
```

EVENTOS DO MOUSE

Para finalizar a construção deste exemplo estão faltando os procedimentos de evento para os eventos do mouse que funcionarão como dica para o usuário.

Os eventos utilizados são o **MouseDown** que ocorre quando um botão é pressionado, e o evento **MouseUp** que ocorre quando um botão é liberado. Ambos eventos enviam para o procedimento, qual botão foi pressionado, se alguma tecla de controle está pressionada (Alt, Ctrl, Shift) junto com o botão e a posição do mouse.

Private Sub objeto.MouseDown (Button **As Integer**, Shift **As Integer**, X **As Single**, Y **As Single**)

Onde: **Button** - é o botão pressionado, sendo
Esquerdo = 0 ; Direito = 2 ; Central = 4
Shift - indica a tecla de controle, podendo ser um somatório, sendo
Shift = 1 ; Ctrl = 2 ; Alt = 4
X e **Y** - indicam a posição

Private Sub lblLetra_MouseDown(Index **As Integer**, Button **As Integer**, Shift **As Integer**, X **As Single**, Y **As Single**)

‘*O parâmetro Index contém o index do lblLetra*
Dim I **As Integer**
If Shift = 2 **Then**
I = Index + 1
lblLetra(Index).Caption = Mid$(Palavra, I, 1)
End If
End Sub

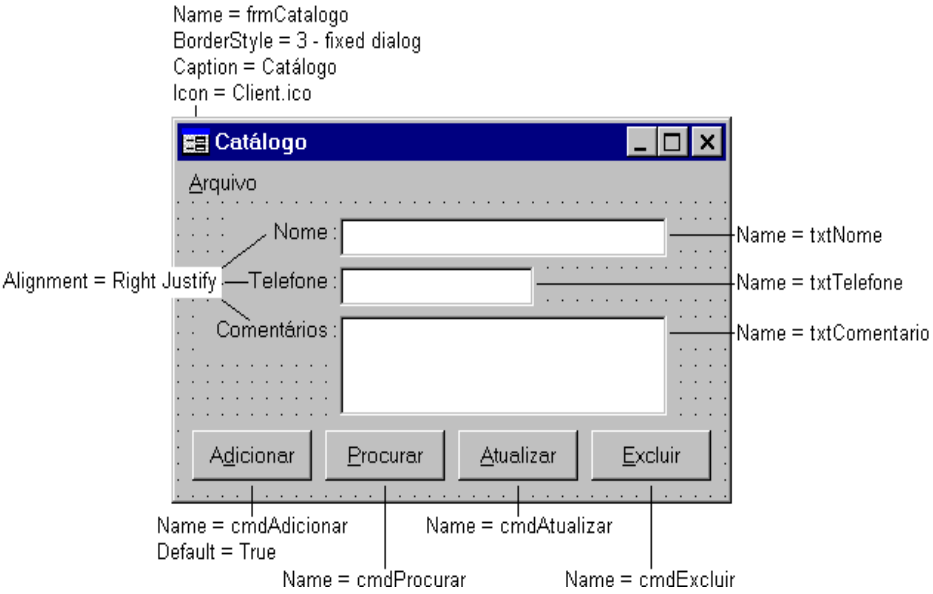
Private Sub lblLetra_MouseUp(Index **As Integer**, Button **As Integer**, Shift **As Integer**, X **As Single**, Y **As Single**)

lblLetra(Index).Caption = “”
End Sub

EXEMPLO V - CATÁLOGO

Neste projeto, conheceremos a importância das variáveis compostas e array, o uso do comando de impressão, e trabalharemos com acesso a arquivos do tipo aleatório, todas essas informações serão apresentadas através de um catálogo de nomes e telefones.

Inicie um novo projeto e crie o formulário como a figura abaixo.

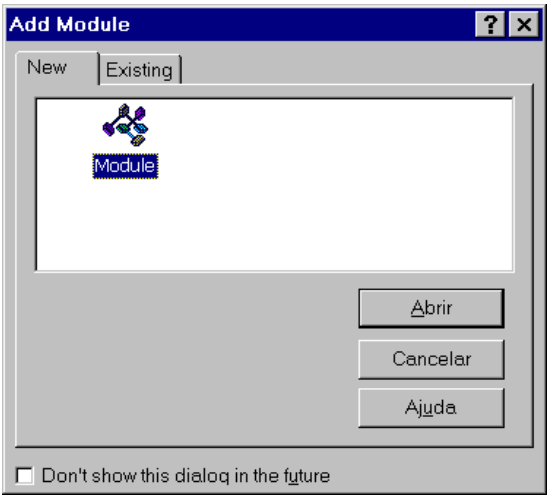


Vamos construir um menu para o nosso catálogo. Selecione o formulário Catálogo e escolha a opção Menu Editor... do menu Tool, e monte-o conforme a tabela a seguir.

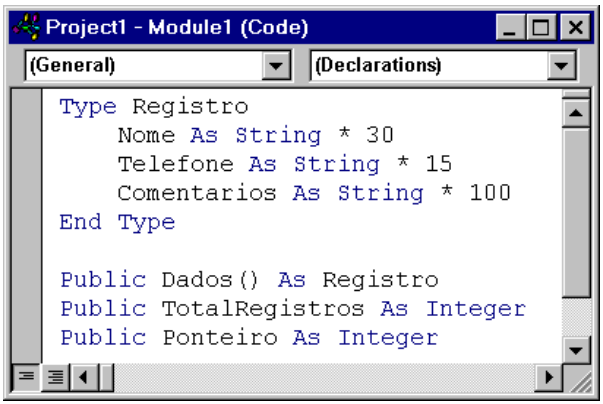
Caption	**Shortcut**	**Name**
&Arquivo		mnuArquivo
····&Abrir	Ctrl+A	itmAbrir
····&Salvar	Ctrl+S	itmSalvar
····-		itmSep1
····&Imprimir	Ctrl+P	itmImprimir
····-		itmSep2
····Sai&r		itmSair

Um módulo contém as variáveis e procedimentos que poderão ser acessados por toda a aplicação, ou seja, qualquer outro procedimento de qualquer formulário poderá chamar um outro procedimento do módulo. Ou então, pode ser dado um valor a uma variável em um formulário e este valor ser lido por outro formulário, quando esta variável é declarada no módulo.

Para inserir um módulo ao projeto, basta escolher a opção Add Module do menu Project ou selecione o botão Add Module ( ) na barra de ferramentas. Abrindo o quadro de diálogo Add Module mostrado a seguir, escolha Module e Abrir.



Logo após, digite o código no editor de código do módulo, mostrado na figura abaixo.



VARIÁVEIS COMPOSTAS E ARRAY

No código do módulo construído anteriormente, estamos declarando uma **variável composta** (Registro) contendo os campos Nome, Telefone e Comentarios como String de tamanho fixo. Uma variável do tipo String, não precisa ter o seu tamanho definido, a menos que estejamos preparando registros de arquivos.

Declaramos também um **Array Dinâmico** de variável (Dados()). Quando declaramos um **Array Dinâmico** estamos maximizando o uso da memória pois poderemos dimensioná-lo para o tamanho ideal a qualquer momento. Para redimensionar um **Array Dinâmico** utilize a declaração **ReDim** informando a dimensão desejada.

A variável Dados() é declarada como sendo do tipo Registro, declarado anteriormente, desta forma teremos dentro da variável Dados() os campos Nome, Telefone e Comentarios, que poderão ser acessados usando o operador (.) ponto:

```
Dados(5).Nome = “Carlos”
Dados(5).Telefone = “981-0000”
Dados(5).Comentarios = “Bom pagador”
```

Neste Módulo também estamos declarando a variável **TotalRegistros** que conterá o número total de registros armazenados e a variável **Ponteiro** indicando o número do registro atual.

Digite o código para cada objeto e evento do formulário catálogo (frmCatálogo), como os exemplos a seguir.

Adicionar

O procedimento de evento cmdAdicionar_Click, executará instruções de acordo com a seguinte situação:

- Quando a legenda do botão for Novo, apenas limpa as caixas de texto e muda a legenda para Adicionar.

· Quando Adicionar, é gravado o conteúdo das caixas de texto na variável Dados() com o índice do último registro existente mais 1. E utilizamos o método AddItem para adicionar um novo nome ao quadro de lista do formulário frmProcurar, que ainda será criado. O método AddItem serve para adicionar um novo item a um Quadro de Lista ou Quadro Combo:

controle.**AddItem** item [,index]

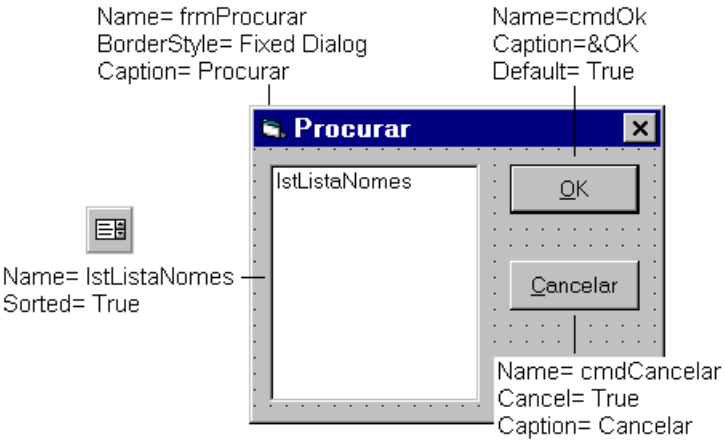
```
Private Sub cmdAdicionar_Click()
    If cmdAdicionar.Caption = "&Novo" Then
        txtNome.Text = ""
        txtTelefone.Text = ""
        txtComentario.Text = ""
        cmdAdicionar.Caption = "A&dicionar"
    Else
        TotalRegistros = TotalRegistros + 1
        Dados(TotalRegistros - 1).Nome = txtNome.Text
        Dados(TotalRegistros - 1).Telefone = txtTelefone.Text
        Dados(TotalRegistros-1).Comentarios=txtComentarios.Text
        frmProcurar.lstListaNomes.AddItem txtNome.Text
        cmdAdicionar.Caption = "&Novo"
        Ponteiro = TotalRegistros
    End If
        txtNome.SetFocus
End Sub
```

Procurar

O botão Procurar irá abrir uma segunda janela para pesquisa dos dados desejados

```
Private Sub cmdProcurar_Click()
   frmProcurar.Show 1
End Sub
```

Insira um novo formulário, e deixe-o como a figura a seguir.



Com a janela de código do formulário frmProcurar exibida, selecione a opção Add Procedure... do menu Tools, para inserir um procedimento geral de formulário. Dê o nome de **ObterItem** para este procedimento.

Digite o código a seguir:

```
Public Sub ObterItem( )
      Dim As integer
      For I = 0 To TotalRegistros - 1
      If RTrim$(Dados(I).Nome) = RTrim$(lstListaNomes.Text) Then Exit
For
    Next I
    frmCatálogo.txtNome.Text = Dados(I).Nome
    frmCatálogo.txtTelefone.Text = Dados(I).Telefone
    frmCatálogo.txtEndereço.Text = Dados(I).Endereço
    frmProcurar.Hide
End Sub
```

A Estrutura de Repetição **For**...**Next** irá procurar em todos os campos Nome da variável Dados( ) um Nome igual ao nome selecionado no quadro de lista e, quando achar, sairá do loop (mantendo o valor de **I**), atribuindo os valores dos campos aos quadros de texto do frmCatálogo.

A função **RTrim$** (expressão), elimina os espaços existentes após o String. Esta função tem outras semelhantes, **LTrim$** (expressão) que elimina os espaços antes e, **Trim$** (expressão) que elimina os espaços antes e depois do String. No procedimento acima usamos RTrim$, porque foi reservado um espaço de trinta (30) caracteres para o campo Nome e na comparação entre expressões, o número de espaços também é comparado.

A seguir, encontraremos os códigos para os botões **Cancelar** e **Ok**, e a chamada para a procedure **ObterItem** após um duplo clique sobre um dos nomes da lista.

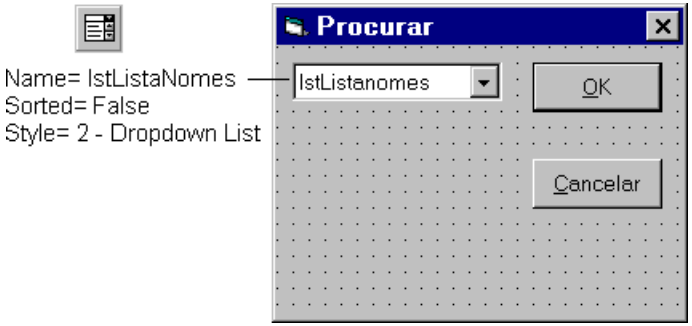
```
Private Sub cmdCancelar_Click()
   frmProcurar.Hide
End Sub

Private Sub cmdOk_Click()
    ObterItem
End Sub

Private Sub lstListaNomes_DblClick()
    ObterItem
End Sub
```

Toda vez que o formulário frmProcurar for mostrado é interessante que ele mostre uma lista atual de nomes. Para atualizar esta lista usamos o evento **Activate**, que ocorre toda vez que o formulário se torna a janela ativa.

Neste caso não usaremos o evento **Load**, pois este evento só ocorrerá na primeira vez que esta janela for mostrada pois para escondê-la usamos o método **Hide** que não remove o formulário da memória.

```
Private Sub Form_Activate()
     lstListaNomes.Clear
     For i = 0 To TotalRegistros - 1
          If Dados(i).Nome <> "" Then
               lstListaNomes.AddItem Dados(i).Nome
          End If
     Next i
End Sub
```

No formulário frmProcurar, podemos substituir o quadro de lista por um quadro combo, porque ambos trabalham de forma semelhante. Para isto, selecione o quadro de lista e escolha a opção Delete no menu Edit para retirar este controle do formulário. Insira no mesmo local um quadro Combo, como segue:



O quadro Combo possui o mesmo nome do quadro de lista para não termos que alterar o código. A propriedade **Style**, determina qual tipo de quadro combo iremos trabalhar, podendo ter os seguintes estilos:

0 - Dropdown Combo; inclui uma lista de queda e um quadro de texto. O usuário pode selecionar um item da lista ou digitar no quadro de texto.

1 - Simple Combo; inclui uma lista sempre visível e um quadro de texto.

2 - Dropdown List; Mostra somente uma lista de queda onde o usuário escolherá o seu item.

Atualizar

O botão Atualizar atualiza o conteúdo do registro atual (Ponteiro) no formulário frmCatalogo e no frmProcurar, removendo e depois adicionando o item relativo à Ponteiro.

```
Private Sub cmdAtualizar_Click()
    Dados(Ponteiro).Nome = txtNome.Text
    Dados(Ponteiro).Telefone = txtTelefone.Text
    Dados(Ponteiro).Comentarios = txtComentarios.Text
    O método RemoveItem remove o item indicado em Ponteiro.
```

```
    'O método RemoveItem remove o item indicado
    frmProcurar.lstListaNomes.RemoveItem Ponteiro
    frmProcurar.lstListaNomes.AddItem txtNome.Text, Ponteiro
    txtNome.SetFocus
End Sub
```

Excluir

O botão Excluir limpa os valores da variável Dados () e substitui o registro vazio pelo registro subseqüente. Nesta substituição é utilizada a estrutura de repetição condicional **While** *condição ... instruções ...* **Wend**, que executa as instruções enquanto a condição for verdadeira.

```
Private Sub cmdExcluir_Click()
    Dados(Ponteiro).Nome = ""
    Dados(Ponteiro).Telefone = ""
    Dados(Ponteiro).Comentarios = ""
    txtNome.Text = ""
    txtTelefone.Text = ""
    txtComentarios.Text = ""
    txtNome.SetFocus
    TotalRegistros = TotalRegistros - 1
    While Ponteiro + 1 <= TotalRegistros
        Dados(Ponteiro) = Dados(Ponteiro + 1)
        Ponteiro = Ponteiro + 1
    Wend
End Sub
```

Load e Resize

Se você reparar, abaixo na barra de menu dos aplicativos Windows existem duas linhas (cinza e branca) separando o menu da área do cliente. Estas duas linhas serão construídas quando o formulário for carregado na memória e serão redesenhadas quando este mesmo formulário for redimensionado pelo cliente.

Inclua o procedimento FazerBarra listado a seguir, no módulo.

```
Public Sub FazerBarra()
    Dim SX As Integer
    SX = ScaleWidth
    frmCatalogo.Line (0, 0)-(SX, 0), RGB(100, 100, 100)
    frmCatalogo.Line (0, 1)-(SX, 1), RGB(255, 255, 255)
End Sub
```

Digite o seguinte procedimento para o evento Load do formulário frmCatalogo.

```
Private Sub Form_Load()
    ReDim Dados(100)        'Redimensiona a variável Dados()
    FazerBarra
    TotalRegistros = 0
End Sub
```

Quando o usuário redimensiona uma janela o evento Resize ocorre, redesenhando as linhas.

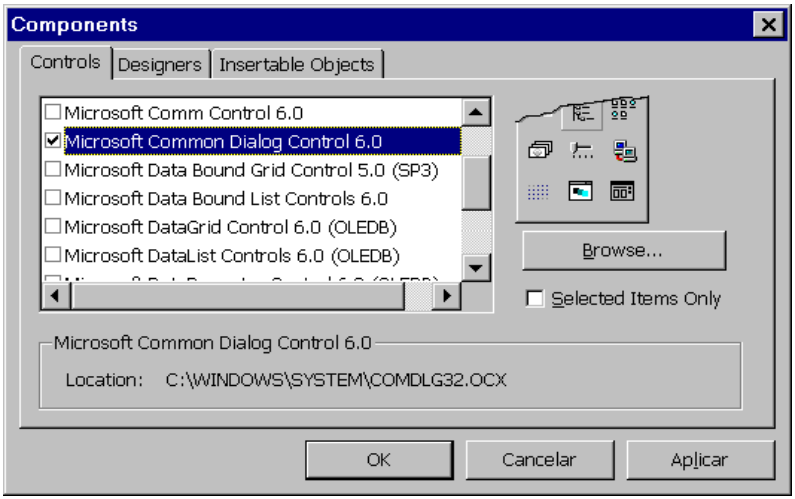
```
Private Sub Form_Resize()
    FazerBarra
End Sub
```

Teste o programa inserindo alguns nomes e depois fazendo uma pesquisa no formulário Procurar, e depois exclua alguns deles.

Caixas de Diálogo Padrão

No exemplo do Bloco de Notas, construímos um quadro de diálogo para Salvar o arquivo e outro para Abrir um arquivo existente, neste projeto utilizaremos o controle **Common Dialog** que permite a utilização das janelas padrão do Windows.

Antes de utilizar este componente devemos adicioná-lo ao projeto utilizando o menu **Projects** opção **Components...**, selecionando a opção indicada na figura a seguir.



Insira o controle CommonDialog ( ) no formulário, não se preocupando com sua posição pois ele não é visível em tempo de execução. Usaremos este controle basicamente para abrir um arquivo e obter o nome de um arquivo a ser salvo.

Defina as seguintes propriedades: Name= dlgArquivo
CancelError = True
Filter = Arquivo (*.dat)|*.dat|Texto (*.txt)|*.txt

A propriedade **Filter** especifica os tipos de arquivos que serão mostrados na combo box de tipos de arquivo no diálogo.

O controle CommonDialog pode mostrar as seguintes caixas de diálogo usando métodos específicos.

Métodos	**Caixas de diálogo**
ShowOpen	Show Open Dialog Box
ShowSave	Show Save As Dialog Box
ShowColor	Show Color Dialog Box
ShowFont	Show Font Dialog Box
ShowPrinter	Show Print or Print Options Dialog Box
ShowHelp	Invokes the Windows Help Engine

Iniciamos a construção do processo de salvar e abrir arquivos digitando alguns procedimentos no módulo.

Abrir arquivo

Este procedimento abre uma caixa de diálogo para o usuário escolher um arquivo para abrir. E depois abre este arquivo no modo randômico, carregando seus registros na variável Dados ( ).

A leitura em um arquivo de acesso randômico é feita com a declaração **Get**.

Get #númeroarquivo, númeroregistro, variável

Onde: **variável** - indica onde será guardado o registro lido.
númeroregistro - é opcional e indica o registro que queremos ler, no exemplo, a declaração Get incrementa automaticamente o númeroregistro.

```
Sub AbrirArquivo()
    On Error Resume Next  'Se algum erro ocorrer, executa
                          'a linha seguinte
    Dim NomeArquivo As String
    Dim NumeroRegistros As Integer      'Contém total de registros
    Dim Reg As Registro                 'Usado apenas como referencia
    frmCatalogo.dlgArquivo.FileName = ""
    frmCatalogo.dlgArquivo.ShowOpen
    If Err <> 32755 Then        'Verifica se Cancel foi pressionado
        NomeArquivo = frmCatalogo.dlgArquivo.FileName
    End If
    Open NomeArquivo For Random As #1 Len = Len(Reg)
    NumeroRegistros = LOF(1) / Len(Reg)
    ReDim Dados(NumeroRegistros)
    Screen.MousePointer = 11        'Muda ponteiro p/ ampulheta
    For I = 1 To TotalRegistros
        frmProcurar.lstListaNomes.RemoveItem 0
    Next I
    For I = 1 To NumeroRegistros
        Get #1, , Dados(I - 1)
        frmProcurar.lstListaNomes.AddItem Dados(I - 1).Nome
    Next I
```

```
    Close #1
    TotalRegistros = NumeroRegistros
    frmCatalogo.txtNome.Text = RTrim$(Dados(0).Nome)
    frmCatalogo.txtTelefone.Text = RTrim$(Dados(0).Telefone)
    frmCatalogo.txtComentarios.Text =       RTrim$(Dados(0).Comentarios)
    Ponteiro = 1
    Screen.MousePointer = 0          'Volta ponteiro ao normal
End Sub
```

Obter nome do arquivo

Antes de salvar um arquivo, é necessário escolher um nome para ele. Mas para o sistema operacional o nome consta de todo o caminho para se chegar até o arquivo desejado.

Por isso utilizaremos uma Common Dialog para escolher além do nome, o local onde deverá ser armazenado o arquivo de catálogo. Construiremos uma função que abrirá a caixa de diálogo para salvar retornando o nome escolhido pelo usuário.

Toda função retorna um valor após ser executada, e este valor geralmente está associado à uma variável com o mesmo nome desta função.

```
Function ObterNomeArquivo(NomeArquivo As Variant)
    On Error Resume Next
    frmCatalogo.dlgArquivo.FileName = "*.dat"
    frmCatalogo.dlgArquivo.ShowSave
    If Err <> 32755 Then
        'Se não Cancel, retorna o nome do arquivo
        ObterNomeArquivo = frmCatalogo.dlgArquivo.FileName
    Else
        ObterNomeArquivo = ""
    End If
End Function
```

Salvar arquivo

Para finalizar as rotinas com arquivos temos o procedimento SalvarArquivo (NomeArquivo As String) que salvará os registros contidos na variável Dados ( ) em um arquivo randômico utilizando a declaração **Put**.

Put #númeroarquivo, numeroregistro, variável

Onde: **númeroregistro** - é o número do registro que queremos gravar, caso não seja informado, será gravado o próximo registro do anteriormente gravado.
variável - é o local onde encontram-se os dados a serem gravados no arquivo.

```
Sub SalvarArquivo(NomeArquivo As String)
    On Error GoTo Erro_SalvarArquivo
    'Se houver erro desvia para a linha Erro_SalvarArquivo

    Open NomeArquivo For Random As #1 Len = Len(Dados(1))
    Screen.MousePointer = 11
    For I = 0 To TotalRegistros -1
        If Dados(I).Nome <> "" Then
            Put #1, , Dados(I)
        End If
    Next I
    Close #1
    Screen.MousePointer = 0
    Exit Sub

Erro_SalvarArquivo:
    MsgBox "Erro de arquivo", 48, "Catálogo"

End Sub
```

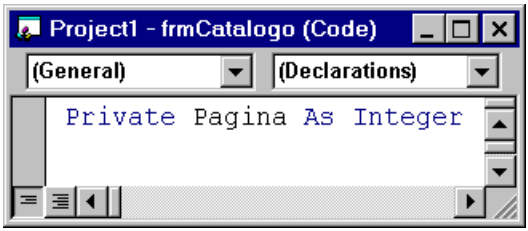
Voltando ao código do formulário frmCatalogo teremos que implementar os procedimentos de evento para o menu com chamadas às rotinas escritas no módulo.

```
Private Sub itmAbrir_Click()
    AbrirArquivo
End Sub
```

```
Private Sub itmSalvar_Click()
    Dim strNomeArquivo As String
    strNomeArquivo = ObterNomeArquivo(NomeArquivo)
    If strNomeArquivo <> "" Then
        'Envia o nome do arquivo para ser salvo
        SalvarArquivo strNomeArquivo
    End If
End Sub

Private Sub itmSair_Click ()
    End
End Sub
```

Comando de Impressão

Agora, vamos incluir um comando para impressão. Inicie declarando a variável **Pagina** como geral do formulário frmCatalogo.



Procedimento a seguir irá construir um cabeçalho no início de cada página de impressão.

```
Public Sub Cabeçalho()
    Pagina = Pagina + 1
    Printer.Print Format$(Now, "dd/mm/yyyy"); Tab(70); "Pág.:"; Pagina
    Printer.Print
    Printer.FontSize = 12
    Printer.Print Tab(13); "Nome"; Tab(35); "Telefone"; Tab(60); "Comentários"
    Printer.Line (0, Printer.CurrentY)-(Printer.ScaleWidth,
Printer.CurrentY)
    Printer.CurrentX = 0
```

```
    Printer.CurrentY = 800
    Printer.FontSize = 10
End Sub
```

O procedimento acima começa imprimindo a data atual e o número da página, usando o método **Print** para o objeto **Printer**.

O método **Print** também se aplica para Formulários e PictureBox da mesma forma que para a impressora. A sintaxe deste método é a seguinte:

[objeto,]**Print** [Tab(n)] [expressão] [; | ,]

Onde: **objeto** - objeto onde desejamos imprimir uma expressão, que pode ser form, picture box ou printer.
tab(n) - número opcional para indicar o número de colunas antes de começar a impressão.
expressão - número ou cadeia de caracteres que se deseja imprimir.
(; | ,) - caracteres que determinam o local do cursor para a próxima impressão. O (;) faz a impressão ser imediata, sem espaços, e a (,) faz o cursor passar para a próxima zona de tabulação antes de imprimir.

Digite o código para o item de menu imprimir:

```
Private Sub itmImprimir_Click()
    Dim TamanhoPapel As Integer
    Pagina = 0
    TamanhoPapel = Printer.ScaleHeight - 2880
    Cabeçalho
    For i = 1 To TotalRegistros
        If Printer.CurrentY >= TamanhoPapel Then
            Printer.NewPage
            Cabeçalho
        End If
        Printer.Print RTrim$(Dados(i).Nome)
        Printer.Print Tab(36); RTrim$(Dados(i).Telefone)
        Printer.Print Tab(60); RTrim$(Dados(i).Comentarios)
        Printer.Print
    Next i
    Printer.EndDoc
End Sub
```

A variável **TamanhoPapel** contém a área de impressão, ou seja, o comprimento do papel (ScaleHeight) menos duas polegadas (2880Twips).

O método **NewPage** faz a impressora avançar para a próxima página quando o limite de impressão for atingido. E o método **EndDoc** finaliza o documento liberando a impressão.

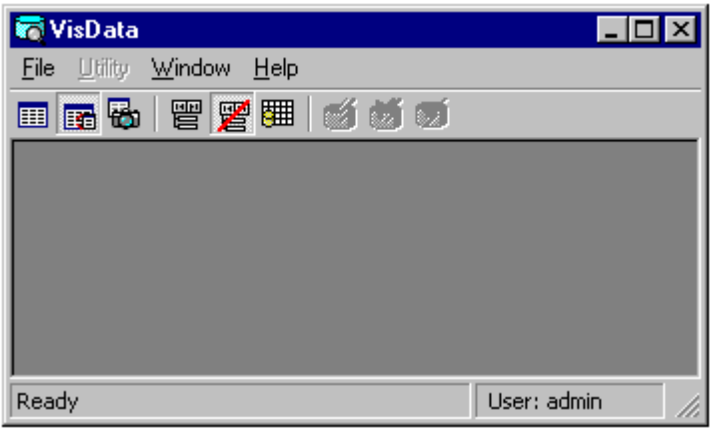
EXEMPLO VI - BANCO DE DADOS

CONTROLE DATA

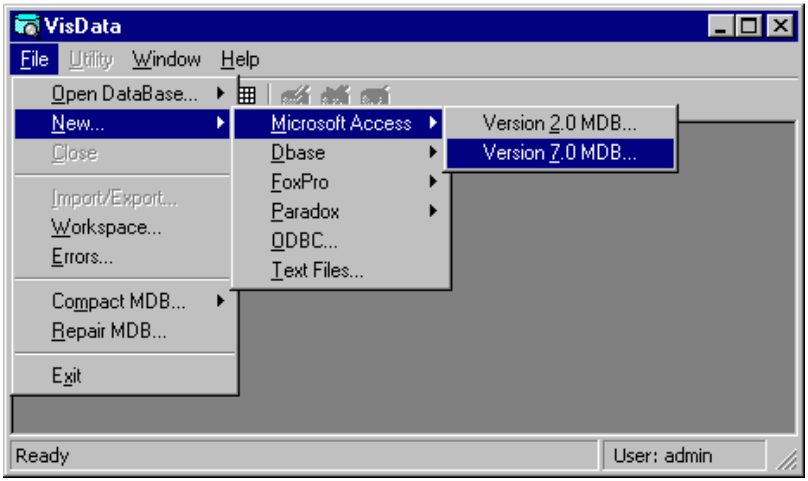
O Controle Data permite que você crie programas que acessem banco de dados como a Access, FoxPro, dBase, Paradox. Com o uso deste controle podemos economizar muitas linhas de código. Ele permite ao usuário visualizar registro por registro, ir para o último ou o primeiro registro de uma tabela, mostrando o conteúdo dos campos através de outros controles.

Os controles que podem exibir informações do Controle de Dados são: CheckBox, Label, TextBox, Image Control e PictureBox. Estes controles são ligados ao Controle Data através da sua propriedade DataSource, como veremos adiante.

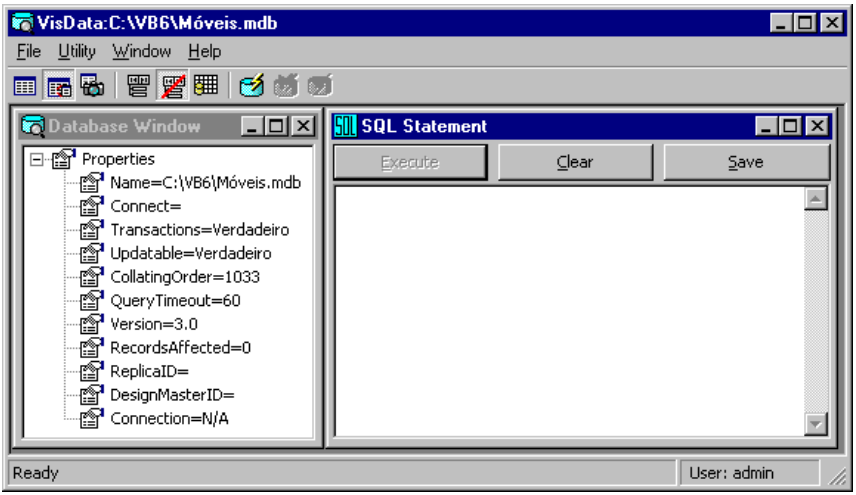
O Visual Basic possui o utilitário Visual Data Manager, por onde poderemos criar um banco de dados para ser usado em nosso projeto exemplo. Para iniciar o Visual Data Manager, escolha a opção Visual Data Manager... do menu Add-Ins, aparecendo a janela mostrada abaixo.



Escolha a opção New... do menu File, e siga a figura abaixo para abrir a janela Select Microsoft Acess DataBase to Create.

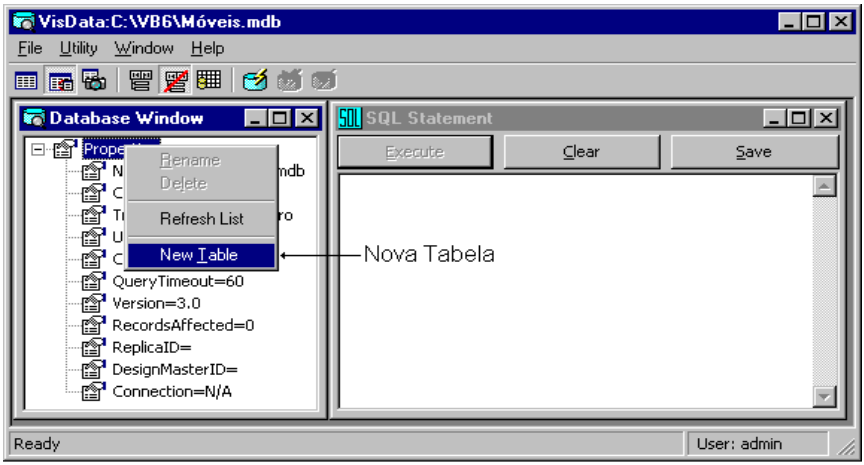


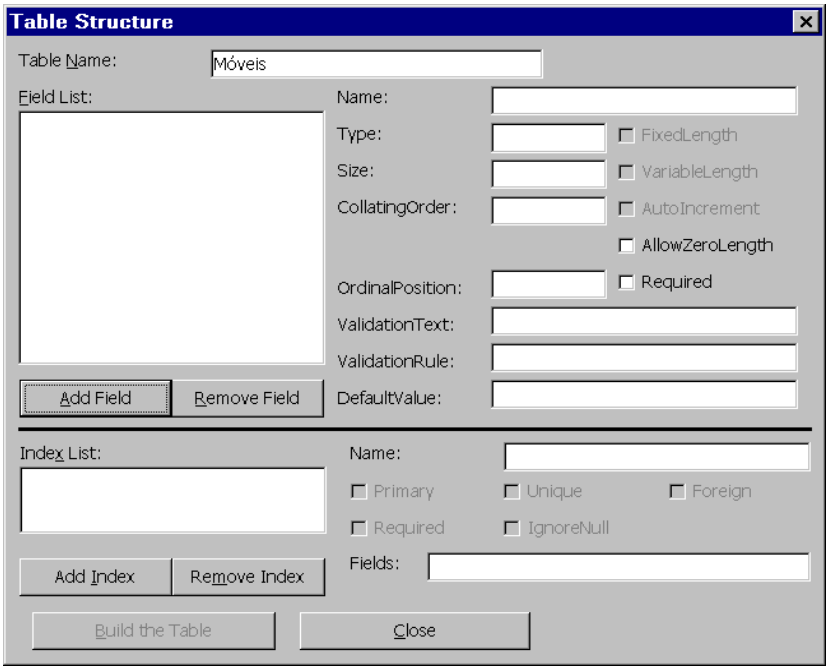
Dê o nome de Móveis.mdb e escolha Salvar, aparecendo a janela de construção do banco de dados - Database: C:\Móveis.mdb.



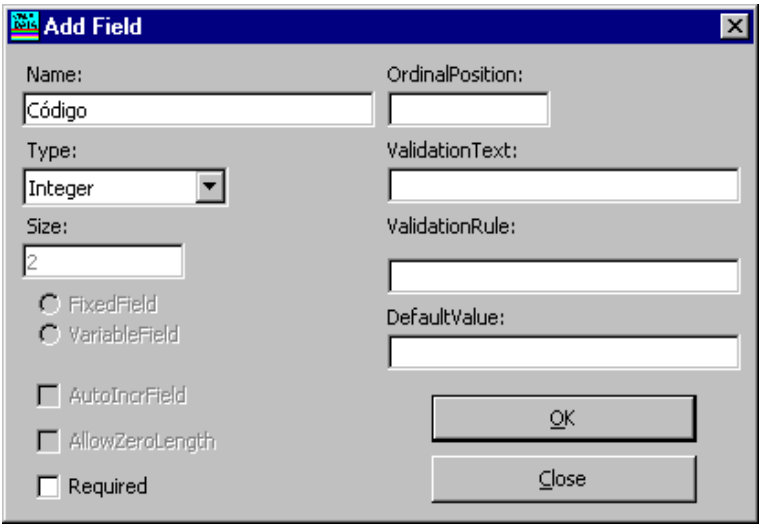
A partir desta janela adicionaremos nossa tabela. Uma tabela é composta de linhas (registros) e colunas (campos). No momento que estivermos criando uma tabela nova, definiremos quais os campos que farão parte dela e quais os tipos de dados que podem ser armazenados em cada campo.

Clique com o botão direito do mouse em cima da janela DataBase Window, para podermos construir a nossa tabela. Selecione a opção New Table, para abrir a janela Table Structure.





Entre com o nome da tabela - Móveis. E com os campos, acionando o botão Add Field, aparecendo o quadro de diálogo Add Field, como mostra a tabela abaixo:



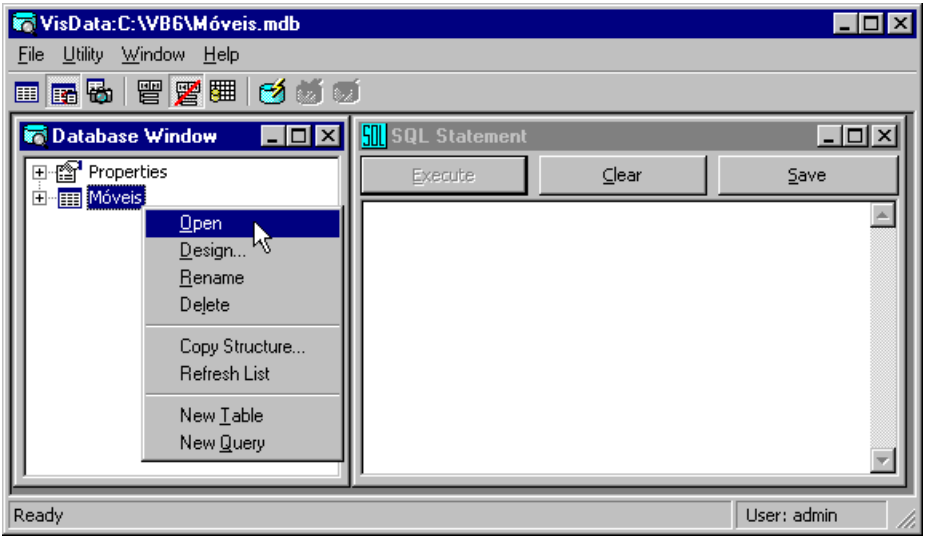
Siga a tabela abaixo, para a definição dos campos; após definir um campo selecione o botão OK, e após entrar com todos os campos, feche o quadro de diálogo.

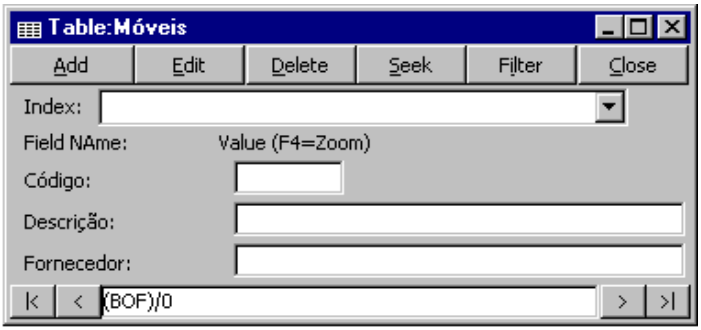
Nome do Campo	Tipo de Dado	Tamanho
Código	Integer	2
Descrição	Text	20
Fornecedor	Text	20

Após entrar com todos os campos, dê um clique em Close, retornando à janela Table Structure.

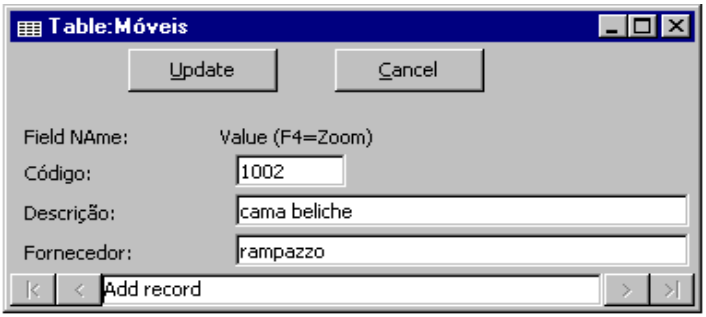
Após conferir todos os campos, construa a tabela (Build the Table), e feche (Close).

Ao retornar, selecione a tabela Móveis e clique com o botão direito do mouse escolhendo opção Open (abrir), para inserirmos dados em nossa tabela, através da janela Table: Móveis.





Digite um registro. Logo após, dê um clique no botão **Update** e depois em **Add**, e assim sucessivamente até completar todos os registros mostrados na tabela abaixo:



Código	Descrição	Fornecedor
1001	cama casal	rampazzo
1002	cama beliche	rampazzo
1003	cama solteiro	rampazzo
1004	cama beliche c/ gaveta	rampazzo
2001	mesa quadrada	santos andirá
2002	mesa redonda	santos andirá
2004	mesa oval	santos andirá
3001	cadeira giratória	santos andirá
3006	cadeira gir. c/ roda	sutran
4005	sofá	sutran

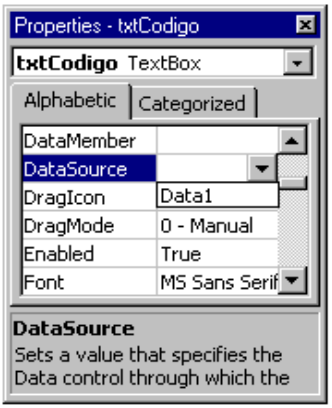
Após digitar todos os registro acima feche a tabela (Close), e também o Visual Data Manager escolhendo a opção Exit no menu File.

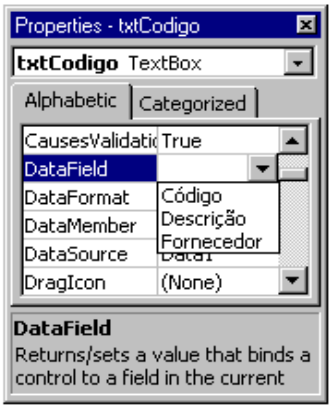
Agora, construa o formulário mostrado abaixo:

A propriedade **DatabaseName** do controle **Data** faz a ligação entre o controle e o banco de dados existente. Quando selecionamos a propriedade DatabaseName surge um quadro de diálogo para escolhermos qual o banco de dados será vinculado a este controle, no nosso projeto escolheremos o banco que acabamos de criar, ou seja, Móveis.mdb.

A propriedade **RecordSouce** define qual tabela do banco de dados selecionado será utilizada pelo controle Data, no nosso casso só existe a tabela móveis, caso existissem mais tabelas, elas seriam mostras na lista da propriedade.

Os quadros de texto possuem as propriedades **DataSource** e **DataField** para fazerem o vínculo com o controle Data. A propriedade DataSource define com qual controle data o quadro de texto estará relacionado, neste projeto existe apenas um controle Data, e a propriedade **DataField** determina qual o campo da tabela o quadro de texto estará vinculado.





Construa o código:

```
Private Sub cmdSair_Click()
        End
End Sub

Private Sub cmdPesquisar_Click()
        Data1.RecordSource = "Select * from Móveis where Código > 1000
And Código < 2000"
        Data1.Refresh
End Sub
```

O programa pode utilizar instruções **SQL** (Structured Query Language) para selecionar alguns registros a serem exibidos. Esta linguagem foi criada para ser uma linguagem padrão de consulta, atualização e manipulação de banco de dados, no nosso projeto usaremos a linguagem SQL para extrair registros selecionados. Lembre-se que a propriedade **RecordSouce** original do controle Data é toda a tabela Móveis, através do comando Select (SQL) recuperamos todos os registros que satisfaçam a seguinte condição: 1000 < Código < 2000.

O método **Refresh** atualiza o conteúdo do controle Data após a pesquisa.

```
Private Sub cmdAdicionar_Click()
    Data1.Recordset.AddNew
   txtCódigo.SetFocus
End Sub
```

A propriedade **RecordSet** contém os registros atuais que o programa pode acessar. Como esta propriedade representa os registros do nosso banco de dados, e nós queremos inserir um novo registro, devemos adicionar este registro a esta propriedade e para isso usamos o método **AddNew**.

Quando o usuário der um clique no botão Adicionar, todos os quadros de texto ficarão em branco aguardando uma entrada de dados, para que estes dados "entrem" no banco de dados, basta alternar algum registro.

```
Private Sub cmdExcluir_Click()
    Data1.Recordset.Delete
    Data1.Recordset.MoveNext
End Sub
```

O procedimento acima apaga - **Delete** - o registro atualmente selecionado e move - **MoveNext** - para o próximo registro da tabela.

```
Private Sub txtCódigo_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then txtDescrição.SetFocus
End Sub
```

Grande parte dos usuários está acostumada a pressionar a tecla Enter quando finaliza um campo, mas no Windows para alternarmos entre objetos utilizamos a tecla Tab. Para contornar este problema, deveremos criar o procedimento acima que parte do evento KeyPress (ato de pressionar e soltar uma tecla no teclado) que retorna o código da tecla pressionada (KeyAscii), o procedimento verifica se foi pressionada a tecla Enter (KeyAscii=13) e, se foi, muda o foco para o próximo quadro de texto.

```
Private Sub txtDescrição_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then txtFornecedor.SetFocus
End Sub

Private Sub txtFornecedor_KeyPress(KeyAscii As Integer)
   If KeyAscii = 13 Then cmdAdicionar.SetFocus
End Sub
```

Salve e execute o projeto verificando o seu funcionamento.

LISTA DE EXERCÍCIOS

1 - Inclua, no primeiro exemplo, um botão de comando para sair do programa. Utilize a declaração End na procedure deste botão.

2 - Na calculadora, troque o quadro de texto txtResultado, por um Label. E inclua um botão de saída.

3 - Ainda na calculadora, coloque um Label onde serão exibidas a data e hora do sistema, quando o usuário calcular o resultado. Use as funções Time e Date.

4 - Crie um quadro de mensagem para que Jogo da Velha questione quem começa a jogada.

5 - No Bloco de Notas, inclua mais um quadro de texto e um botão para transferir o texto selecionado de um quadro para outro.

6 - Crie uma seqüência de pontos utilizando a estrutura de repetição For...Next.

7 - A partir da seqüência construída no item 5, altere as cores dos pontos de uma forma aleatória.

8 - Construa um triângulo equilátero e outro retângulo, utilizando o método Line.

9 - Altere o tipo de traço e cor, para o triângulo construído anteriormente

10 - Utilizando como base o programa para desenhar círculos, altere-o para que faça retângulos (lados opostos com mesma medida).

11 - O tempo de alteração do semáforo de rua é diferente do que foi construído, o vermelho e verde duram mais que o amarelo. Insira outro controle Timer para controlar esta nova mudança.

12 - Inclua no jogo da forca mais um grupo de opções para escolha do idioma. Contruindo uma matriz com palavras deste segundo idioma.

13 - Faça com que o evento MouseDown mostre a letra com uma cor diferente da normal.

14 - Faça com que a cabeça de figura da forca suba e desça numa trajetória retilínea, alterando a fórmula do Y.

15 - No Despertador, coloque mais um botão de opção e outro quadro de texto para a função soneca. No quadro de texto deverá estar o intervalo de soneca.

16 - Para o exercício Catálogo, crie mais um campo para a cidade e desenvolva um formulário de procura por telefones.

17 - No exercício Controle de Dados, crie mais uma tabela para os clientes com Código, Nome, Endereço e Telefone.

18 - Desenvolva o formulário para trabalhar com esta tabela.

19 - Na instrução SQL, procure clientes.