

VISUAL BASIC 5.0

Autora:

Daniela Cristina Maestro

Universidade Estadual de Campinas

Centro de Computação

Versão: 2 – Novembro/98

Requisitos para instalação da Linguagem **(Versão Enterprise)**

Microsoft Windows NT 3.51 ou superior, ou Microsoft Windows 95 ou superior.

Mínimo 486 microprocessador.

Mínimo resolução VGA.

35 Mb para instalação mínima e 345 Mb para instalação completa.

8 MB de RAM para aplicações. (Isto pode variar, dependendo das especificações que você utilizar para sua aplicação. Tipos de DDL, projetos) e 16 MB de RAM para o ambiente de desenvolvimento do VB.

As versões do Visual Basic 5.0

O VB possui 3 versões diferentes. Cada uma delas foi desenvolvida para atender as diferentes necessidades dos usuários:

Standard (Learning Edition): É a versão mais simples do Visual Basic, seus itens são:

Visual Basic development environment

Controles Padrões

Exemplos

Setup Wizard

Setup Kit

Imagens de Ícones

Arquivos de Help

Curso Interativo Learn VB Now

Professional: Esta versão possui todos os itens anteriores (exceto o curso interativo Learn VB Now), e inclui também:

Controles adicionais e Help

Arquivos de Imagens, ícones:Metafiles and bitmaps

Compilador Microsoft Windows Help

Crystal Reports

Books Online (Livros de Help)

Referência Online Windows 32-bit API e DLL Declare statement for Visual Basic

Arquivos necessários para criação de outros Controles.

O Data control (para acesso a banco de dados):

Ambas as versões: Learning e Professional Edition incluem o Data control. Porém com a Professional Edition, você pode também utilizar o Data control para fazer acesso a Open Database Connectivity (ODBC).

Data access objects (DAO)

Editor de Imagens, Resource Compiler, Code Profiler, and Ferramentas para Help Workshop.

Enterprise: Esta versão possui todas as características das anteriores e:

Microsoft Visual SourceSafe: controle de desenvolvimento em grupo.

Suporte para remote OLE Automation e DCOM

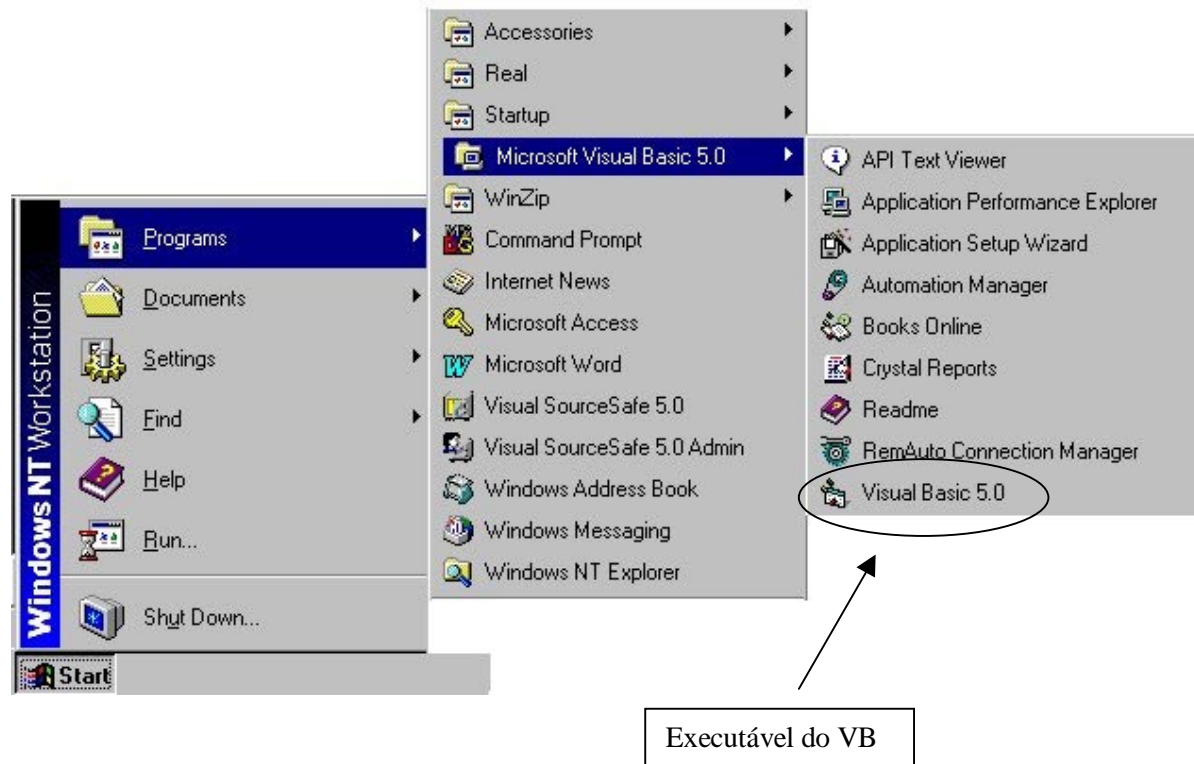
Ferramentas para acesso remoto a base de dados

Gerenciador de Automação.

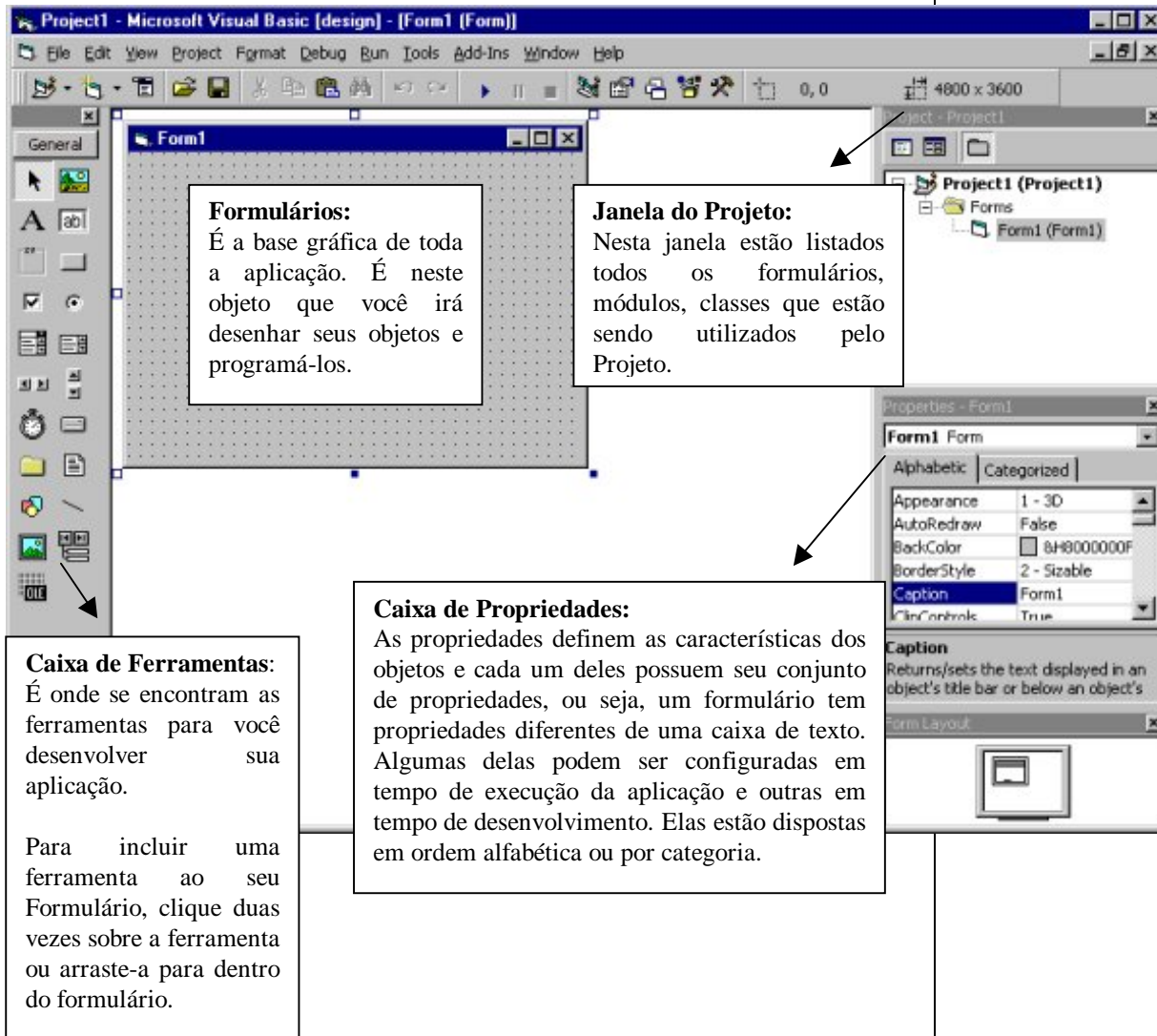
Gerenciador de Componentes.

Visual Basic 5.0

Inicializando o VB:



Tela Principal do Visual Basic 5.0



The screenshot displays the Microsoft Visual Basic 5.0 development environment. The main window is titled "Project1 - Microsoft Visual Basic [design] - [Form1 (Form)]". It features a menu bar (File, Edit, View, Project, Format, Debug, Run, Tools, Add-Ins, Window, Help), a toolbar, and a status bar. The central area is a design grid for "Form1". To the left is the "Caixa de Ferramentas" (Toolbox) with various controls. To the right are the "Janela do Projeto" (Project Window) and the "Caixa de Propriedades" (Properties Window). The Properties Window shows settings for "Form1 Form", including Appearance, AutoRedraw, BackColor, BorderStyle, Caption, and WinControls. A "Form Layout" window is also visible at the bottom right.

Formulários:
É a base gráfica de toda a aplicação. É neste objeto que você irá desenhar seus objetos e programá-los.

Janela do Projeto:
Nesta janela estão listados todos os formulários, módulos, classes que estão sendo utilizados pelo Projeto.

Caixa de Propriedades:
As propriedades definem as características dos objetos e cada um deles possuem seu conjunto de propriedades, ou seja, um formulário tem propriedades diferentes de uma caixa de texto. Algumas delas podem ser configuradas em tempo de execução da aplicação e outras em tempo de desenvolvimento. Elas estão dispostas em ordem alfabética ou por categoria.

Caixa de Ferramentas:
É onde se encontram as ferramentas para você desenvolver sua aplicação.

Para incluir uma ferramenta ao seu Formulário, clique duas vezes sobre a ferramenta ou arraste-a para dentro do formulário.

Características do Visual Basic 5.0

Muitos programadores já estão familiarizados com a linguagem Basic, na qual o VB é baseado. O Visual Basic tem toda a base para a linguagem de programação usada por todas as aplicações Microsoft Office, Microsoft Visual Basic for Applications (VBA).

O Visual Basic possui uma série de características que facilitam a criação de aplicações:

Múltiplas plataformas Windows. Com a implementação da tecnologia ActiveX, é possível migrar suas aplicações para documentos ActiveX que rodam em browser (IE) em máquinas UNIX e Macintosh.

Objetos OLE, porém nesta nova versão, foi implementada a tecnologia ActiveX que é muito mais rápida que OLE e pode ser utilizada em aplicações Internet/Intranet.

Rapid Application Development (RAD) – Desenvolvimento Rápido de Aplicações (Wizards).

Ambiente de desenvolvimento muito mais amigável que as versões anteriores, além de permitir que este ambiente seja personalizado de acordo com que o desenvolvedor necessita.

Compilação para código nativo que utiliza a tecnologia de compilação do C++.

IntelliSense: editor de código que mostra as sintaxe das funções, parâmetros, constantes.

Nesta versão, gera apenas aplicações 32 bits.

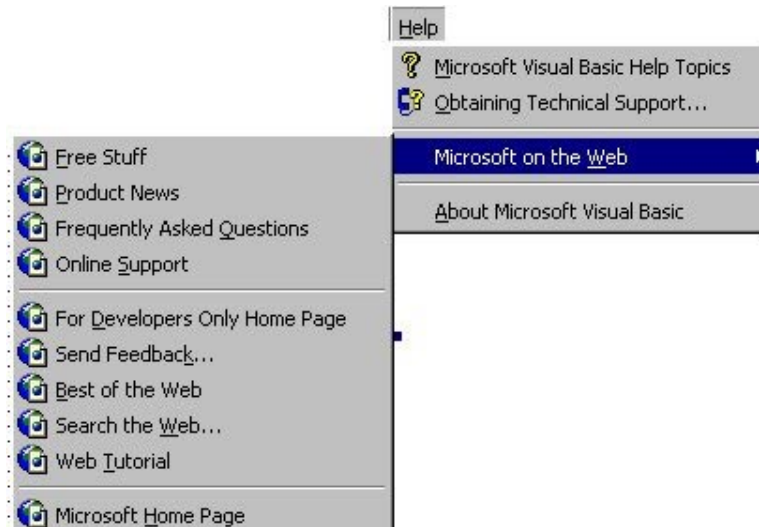
Não é uma linguagem completamente orientada a objetos, mas possui recursos para gerar objetos e atribuir-lhes propriedades e métodos. Seus objetos são compatíveis com a tecnologia COM/DCOM.

NOTAS:

Utilizando o HELP do VB

O Help do VB é uma das melhores documentações sobre a linguagem. Para ativá-lo, siga os passos:

1. A partir do menu Help, escolha a opção que atenderá a sua necessidade.



Se você tem ligação com a Internet, poderá visualizar as últimas informações e novidades diretamente do site da Microsoft, a partir da opção Microsoft on the Web, os tópicos disponíveis estão listados na figura acima do lado esquerdo.

Se você precisar ter informações sobre um determinado objeto, selecione-o (clique sobre o objeto desenhado no formulário) e pressione a tecla de função F1. Será aberto o help completo para o objeto em questão: todas as suas propriedades, seus métodos, seus eventos, exemplos.

NOTAS:

Criando uma Aplicação em Visual Basic

Para criar uma aplicação no Visual Basic é muito fácil. Porém, você precisa desenvolver um software que, para o usuário final, seja funcional, seguro e prático.

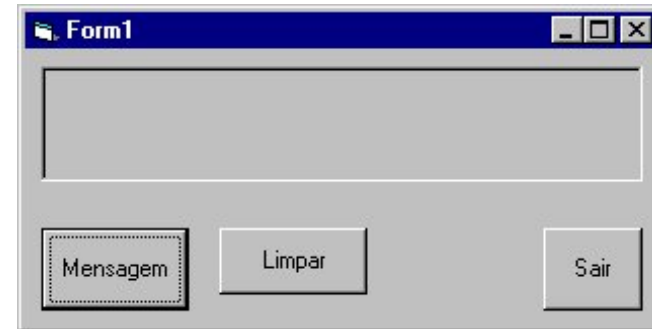
Os passos são:

1. Criar uma interface, ou seja, dispor os objetos de forma amigável na tela.
2. Ajustar as propriedades dos objetos. Como: nome, tamanho, fonte, etc.
3. Escrever o código necessário. Isso inclui: definir constantes, declarar variáveis, criar procedimentos e funções.

O Visual Basic é uma linguagem que possui muitas propriedades e definindo-as corretamente, praticamente você consegue criar sua aplicação. Porém, alguns cálculos, operações precisam ser programadas. É neste ponto em que você precisa programar os eventos dos objetos. O que você precisa observar, é qual o evento a ser programado para executar a ação.

Para saber quais eventos devem ser codificados, pensar em tudo o que o usuário pode fazer e como seu programa vai responder a essas ações.

NOTAS:



Interface Inimiga



Interface Amiga

Barra de Ferramentas



É na barra de ferramentas que estão os comandos e funções mais utilizados no VB.

Descrição dos botões:



Adiciona um novo Formulário, Módulo, Módulo de Classe, etc. ao seu projeto.



Exibe a Janela de Construção de Menus.



Abre e Salva um Projeto, respectivamente.



Recorta, copia, cola e procura por objetos e códigos do seu projeto.



Desfaz e refaz ações.



Executa, pausa (break) e para a aplicação, respectivamente.



Exibe a janela do Projeto.



Exibe a Janela de propriedades.



Exibe a Janela de LayOut do Formulário.



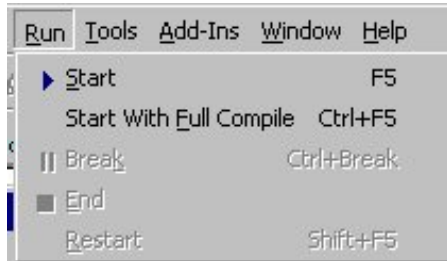
Exibe a janela Object Browser. Esta janela exibe as classes, propriedades, módulos e métodos disponíveis das bibliotecas e os módulos e procedimentos em seu Project. Você pode usar esta janela para procurar e usar os objetos que você criou.



Exibe a Caixa de Ferramentas.

Executando um Programa no Visual Basic

A partir do menu Run, escolha a opção Start ou pressione a tecla de função F5. Para interromper a execução do programa em qualquer ponto, tecla



CTRL+Break.

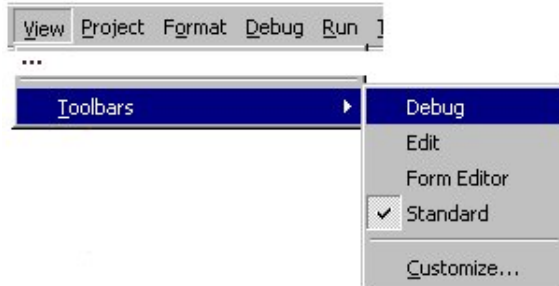
Utilizando Ferramentas de Debug:

F8 – para execução o programa passo-a-passo.

F9 – para definir BreakPoints, ou seja, determinar paradas no seu programa durante a sua execução para encontrar possíveis erros de programação ou lógica.

Para exibir a Barra de Debug do VB:

1. A partir do menu View, item ToolBars e selecione a opção Debug.



A janela Locals Window é utilizada para mostrar o valor corrente da expressão selecionada. Quando o programa é pausado, a janela Calls mostra uma lista dos procedimentos ativos que ainda não foram finalizados.

Você pode utilizar o Step Into para executar linha por linha de seu programa. Se um procedimento é chamado, é executado linha por linha também. O Step Over é similar ao Step Into. A diferença acontece quando há uma chamada de procedimento. Pois o Step Over não executa o procedimento linha a linha, mas como um bloco, de uma vez. O Step Out executa as linhas restantes de uma função na qual o breakpoint se encontra.

NOTAS:

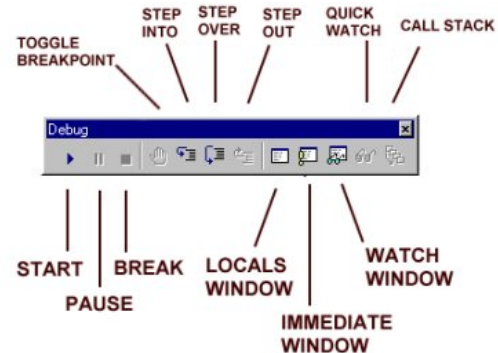
Salvando um Project:

Na barra de Menus, selecione File e escolha a opção Save Project. Na primeira vez, o VB solicitará um nome para o Project e para o(s) Form(s), Module(s).

Para Salvar apenas um dos objetos e não o projeto inteiro, utilize a opção Save <NomeObjeto>.

Criando um executável:

Na barra de Menus, selecione File e escolha a opção Make <NomeProject>.exe.



Project

Uma aplicação desenvolvida em VB, é baseada em Projects (em português, Projetos).

O que é um *Project*?

É uma coleção de arquivos utilizada para criar sua aplicação. É no Project que estão listados os outros arquivos do VB, como: formulários, classes, módulos, etc. As opções de ambiente que você configura também se encontram neste arquivo. Estas informações são atualizadas toda vez que você salva o projeto. Todos esses arquivos e objetos podem ser compartilhados por outros projetos.

Você utiliza os Projects para gerenciar todos os diferentes arquivos que você cria em seu sistema. O Project é constituído por:

- Arquivos de formulários (.FRM). O formato .FRM é um formato texto.
- Arquivos que contém imagens em formato binário (.FRX).
- Arquivos de Class Module (.CLS)
- Arquivos de Standard Module (.BAS)
- Arquivos contendo os Custom Controls (.OCX)
- Um arquivo de projeto contendo todos os componentes (.VBP)

Criando um novo Project no Visual Basic

A partir do menu File, escolha a opção New Project.

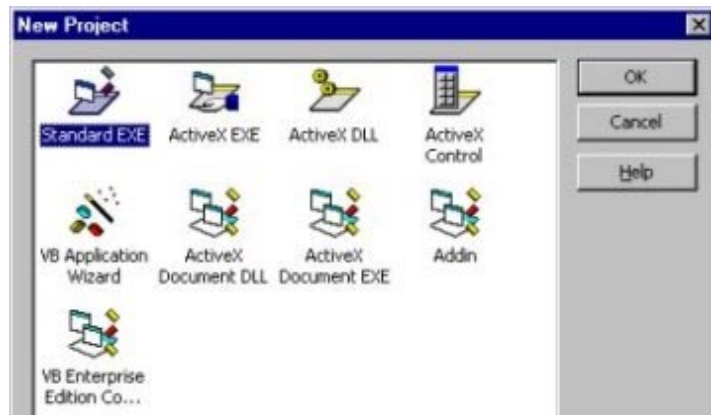
Para visualizar sua janela, a partir do menu View, escolha Project Explorer ou tecle CTRL + R.

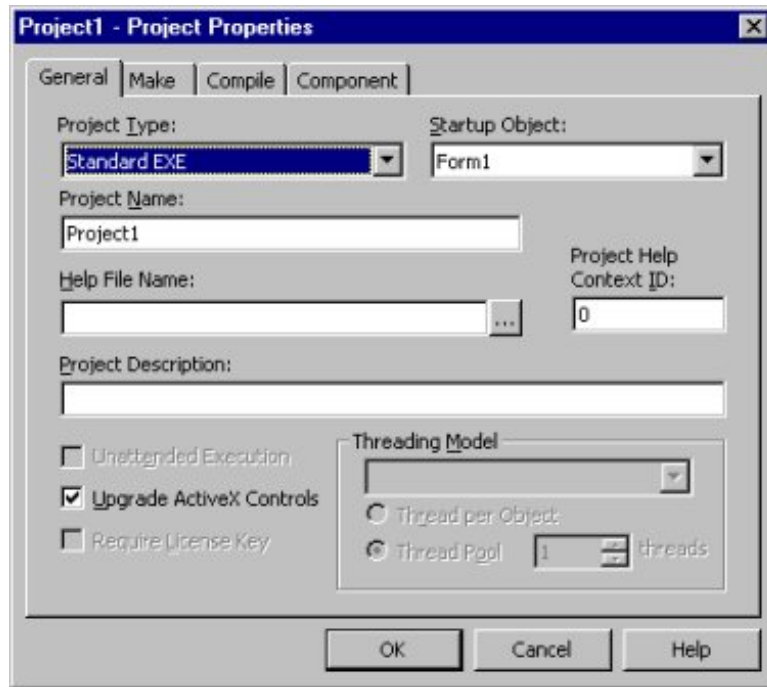
NOTAS:

Tela apresentada para escolha de um novo projeto.

No VB 5.0, é possível criar vários tipos de Projetos. Os tipos de Project são:

1. Standard EXE — Cria um executavel padrão do VB.
2. ActiveX EXE — Cria um arquivo ActiveX executavel.
3. ActiveX DLL — Cria um controle ActiveX no formato DLL
Obs.: projetos do tipo ActiveX DLL/EXE não possuem interface com o usuário. São muito utilizados para criarem regras de negócios.
4. ActiveX Control — Cria um controle ActiveX.
5. ActiveX Document DLL/EXE: Cria um projeto que pode ser levado para a Internet. Possui interface com o usuário.





NOTAS:

Alterando as propriedades de seu projeto

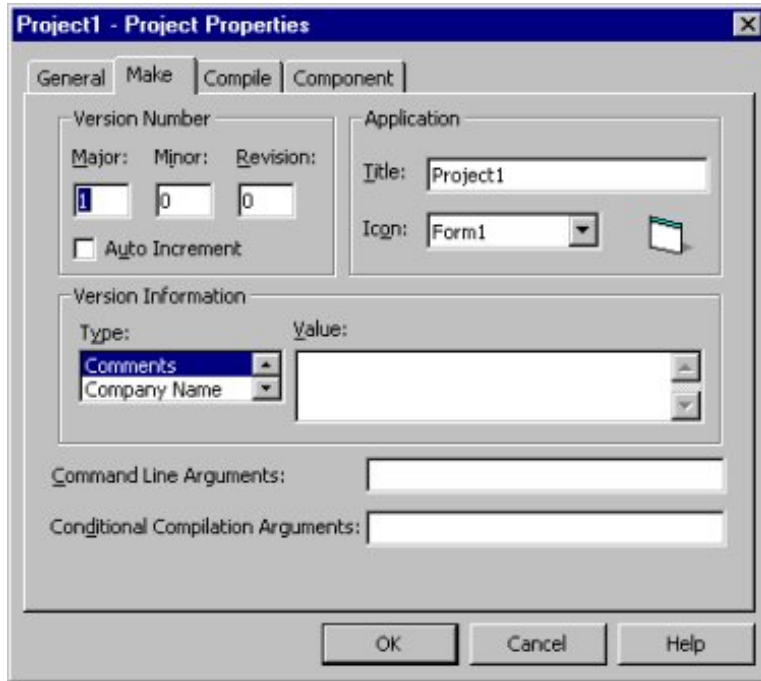
1. A partir do menu Project, escolha a opção <Nome_projeto> Properties.

Tela de propriedades do Projeto: General.

Na tag General, serão configuradas as propriedades básicas do projeto. Seus itens são:

1. **Project Type:** define o tipo de projeto que está sendo utilizado.
2. **Startup Object:** define qual o objeto que será exibido assim que o seu projeto for inicializado.
3. **Project Name:** Nome do Projeto.
4. **Help File Name:** define o arquivo de HELP pertencente ao projeto.
5. **Project description:** descrição rápida sobre o projeto.

Alterando as propriedades de seu projeto (cont.)



Tela de propriedades do Projeto: Make.

Version Number Cria o número da versão do projeto.

Major/ Minor — números da versão do projeto; 0 – 9999.

Revision — Versão da Revisão do projeto; 0–9999.

Auto Increment — Se selecionado, automaticamente incrementa o número de toda vez que é gerado um executável da aplicação.

Application – permite que um nome e um ícone sejam associados ao projeto:

Title — Nome da Aplicação.

Icon — Ícone da Aplicação.

Version Information – define informações específicas sobre a versão atual do projeto aberto.

Type — São as informações que podem ser configuradas, como por exemplo: nome da sua companhia, descrição do arquivo, informações de Copyright, etc.

Value — É o valor para o tipo de informação selecionada na caixa Type.

NOTAS:

Custom Controls (Controles)

O que são “Custom Controls”?

São os arquivos de controle (ferramentas) do VB. Um Custom Control é uma extensão para a Caixa de Ferramentas. Quando você adiciona um controle ao seu programa, ele passa a fazer parte do ambiente de desenvolvimento e run-time, promovendo novas funcionalidades para sua aplicação.

Cada Controle tem suas próprias características (propriedades), procedimentos pré-definidos (métodos) e suas próprias ações (eventos) que podem ser programados e configurados de acordo com a necessidade do sistema. Você pode visualizar essas propriedades na Properties Window, seus métodos na Object Browser e seus eventos na Code Window.

Instalação dos Custom Controls:

As versões Professional e Interprise instalam os custom controls no Windows, no diretório SYSTEM ou SYSTEM32.

Adicionando um custom control (controle) a Caixa de Ferramentas:

1. A partir do menu Project, escolha Components ou Ctrl+T.
2. Para adicionar um controle (.OCX) à Caixa de Ferramenta, selecione a Check Box ao lado do nome do controle.
3. Escolha OK para fechar a caixa de diálogo do Components. Todos os controles selecionados aparecerão na Caixa de Ferramenta. Para não confirmar a inserção dos controles à caixa de ferramentas, clique no botão Cancelar.

NOTAS:

Certas propriedades são comum a vários controles, da mesma forma como existem propriedades individuais a cada controle.



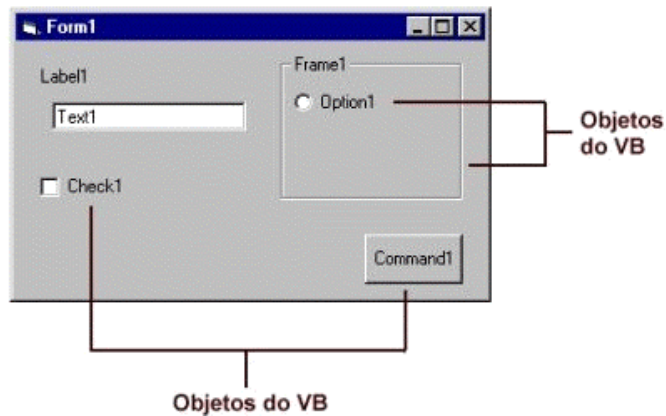
Custom Control (Controle)

Caixa de Ferramentas.

Custom Controls (Controles – cont.)

O que é um Objeto?

É um elemento que será utilizado como interface para o usuário que é criado em um formulário. Serão utilizados os controles para criarem estes objetos.



NOTAS:

Exemplos de Controles:

Label: textos informativos que não podem ser alterados pelo usuário.

Text Box: é um campo onde o usuário pode digitar suas informações.

Command Button: executa uma série de ações que foram escritas nos Modules. São acionados por um evento, por exemplo: um click ou barra de espaço.

Option Button: é usado para escolher somente uma opção dentro de um grupo

Data Control: é a forma mais fácil de acessar suas tabelas de um banco de dados.

Os controles no VB têm o que é conhecido por Funcionalidade Inerente, ou seja, eles sabem como operar e responder a certas situações por eles próprios.

Propriedades dos Objetos

Cada controle no VB possui suas próprias características, sendo que algumas podem ser iguais para mais de um.

Essas propriedades podem ser definidas para determinar a sua aparência e o seu funcionamento. Com isso, você configura o controle de acordo com a sua necessidade.

Algumas propriedades podem ser ajustadas em tempo de execução. Para isso, use a sintaxe: nomeobjeto.propriedade = valor.

Por exemplo: txtDataIni.Text = "26/02/1976"
txtDataIni.BackColor = Blue

BackColor = propriedade para alterar a cor de fundo do objeto.

Text = propriedade para inserir um texto no objeto "Caixa de Texto".

NOTAS:

Visualizando as Propriedades de um determinado objeto:

Existem dois modos de visualizar as propriedades de um objeto na Janela de Propriedade: por ordem alfabética ou por categoria.

Para visualizar a Janela de propriedades de um objeto:



Selecione o objeto com um clique e tecla "F4".

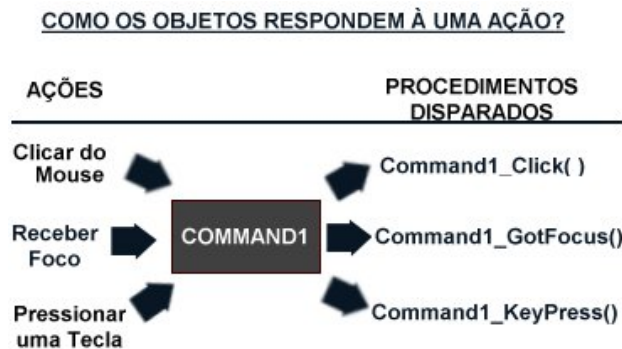
Eventos

O que são eventos?

São ações pré-definidas que podem ocorrer com cada objeto. Cada objeto possui seus próprios eventos, sendo que alguns deles são iguais para mais de um objeto. Essas ações precisam ser programadas. Sem programação elas não funcionam, mas continuam existindo.

Exemplos de alguns eventos existentes:

- Click (um click do botão do mouse)
- KeyPress (qualquer tecla pressionada)
- MouseMove (um movimento do mouse)



NOTAS:

Cada objeto no VB possui um conjunto pré-definido de eventos. aos quais ele pode responder. Esses eventos são listados para cada objeto na caixa de listagem drop-down Proc (Procedure) na janela de código.

Pode-se acionar um evento dentro de um outro evento que está sendo executado pois ele nada mais é que um procedimento.

O nome dos eventos é definido pelo nome do objeto, underscore e o nome do evento:

nomeobjeto_nomeevento()

Exemplo:

txtDataIni_KeyDown()

Para visualizar os eventos, clique 2 vezes sobre o objeto.

Métodos

São procedimentos pré-definidos que cada objeto possui. Não é possível mudar suas definições tentando reprogramá-los.

Os métodos são comandos que desempenham funções para os objetos aos quais estão associados.

A sintaxe para sua utilização:

<nome_do_objeto>.<método>

Por exemplo: txtdata_inic.SetFocus

Forms (Formulários)

O que é um Formulário?

É o objeto onde você define as telas do seu programa (projeto).
Com os Forms você cria a interface da sua aplicação para o usuário.

Um Formulário é composto por:

- janela do formulário;
- janela de Código.

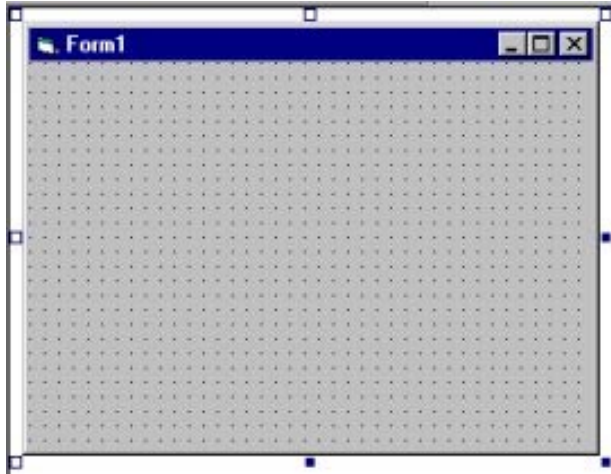
Quando um novo Project é criado, o VB automaticamente cria um formulário padrão: Form1. Este novo formulário aparece com uma grade padrão (pontos uniformemente espaçados) para você ajustar e alinhar seus objetos na tela. Para dimensionar um Form, você pode utilizar o mouse.

É na janela do Formulário que você desenha os objetos da barra de ferramentas. O Formulário é também considerado um objeto.

Na sua janela de código, você pode escrever a programação para esses objetos.

Cada objeto num Formulário possui uma janela de código correspondente.

A extensão desses arquivos é: <nome_do_form>.FRM



Exemplo de uma tela de Formulário (Form) em tempo de Design.

NOTAS:

Acrescentando um Form a um Project já existente

A partir do menu Project, escolha Add Form.

Visualizando um Formulário

A partir da janela do Project, selecione o form desejado, clique no ícone de View Form ou duplamente sobre o mesmo.

Visualizando a Janela de Código do Formulário

Para visualizar apenas o código de um formulário, selecione-o e clique no ícone correspondente ao View Code.

Forms (Formulários – cont.)

Propriedades mais utilizadas nos Forms

Appearance: determina a aparência do Form. Pode ser definido como Flat (modo normal) e 3D.

BackColor: determina a cor de fundo do Form.

BorderStyle: define a borda do Form.

Caption: esta propriedade funciona como se fosse um título para o Form.

ControlBox: define se a caixa de controle (para fechar, minimizar, maximizar, etc) do lado esquerdo do Form deve estar disponível ou não. Possui dois valores: True/False.

Enabled: define se o form estará ou não disponível para uso. Valores: True/False.

Font: define um novo tipo de letra para o Form.

Icon: define um ícone para identificação do Form

MaxButton: determina se o botão para Maximizar deve ou não ser exibido no Form (do lado direito).

MinButton: determina se o botão para Minimizar deve ou não ser exibido no Form (do lado direito).

Name: utilizada para definir um nome para o Form. (para a programação)

Visible: determina se um Form deve ser visível ou não. Valores: True/False

Eventos mais utilizados nos Formulários

Load: antes do form ser carregado na memória, será executada a função/procedimento que estiver descrita neste evento. Muito utilizado para inicializar os objetos do formulário, por exemplo, preencher uma caixa combo.

Unload: este evento é executado antes do formulário ser descarregado da memória. Utilize este evento para finalizar banco de dados, tabelas, arquivos abertos.

QueryUnload: este evento detecta como o evento Unload foi ativado. Possui dois parâmetros:

1. UnloadMode: indica como o evento Unload foi ativado. Retorna os seguintes valores:

Constante	Valor	Descrição
vbFormControlMenu	0	O usuário escolheu o comando Fechar do menu de Controle do formulário.
vbFormCode	1	O evento Unload foi iniciado a partir de
vbAppWindows	2	A sessão corrente do ambiente operacional do Windows foi terminada.
vbAppTaskManager	3	O Gerenciador de Tarefas do Windows terminou a sessão.
vbFormMDIForm	4	Um formulário MDI child está sendo fechado porque um formulário MDI foi fechado.

2. Cancel: determina se o evento Unload deve ou não continuar. Inicialmente, seu valor é False, se receber True, o Unload do formulário é interrompido.

NOTAS:

Teste!

Crie um novo Project:

A partir do menu File, opção New Project, tipo de Project Standard.

Junto com o Project, o VB criará um form automaticamente. Abra este form a partir da janela do Project e visualize suas propriedades (tecle F4).

Altere suas propriedades de acordo com as suas necessidades.

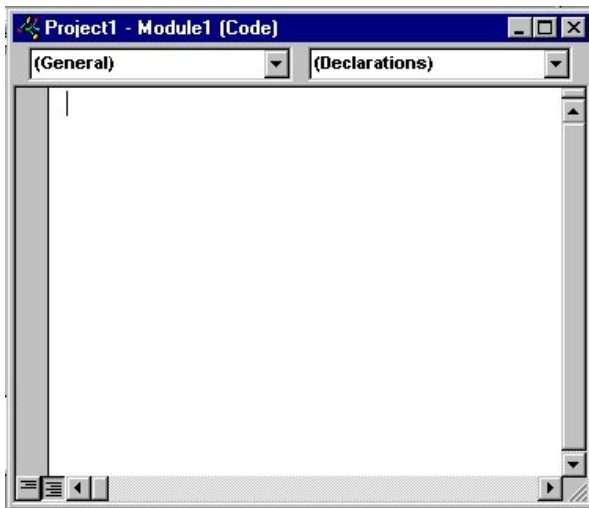
Módulos

As funções e procedimentos que servem para mais de um formulário (uso geral do sistema), as variáveis globais e as constantes devem ser escritas em Módulos.

Os Módulos podem conter:

- Declarações. Você pode declarar Constantes, tipos, variáveis.
- Procedimentos. Uma Sub-rotina, uma função e Propriedades que contém códigos.

A extensão desses arquivos é: <nome_arquivo>.BAS



Janela de Código do Módulo.

NOTAS:

Acrescentando um novo Módulo ao projeto:

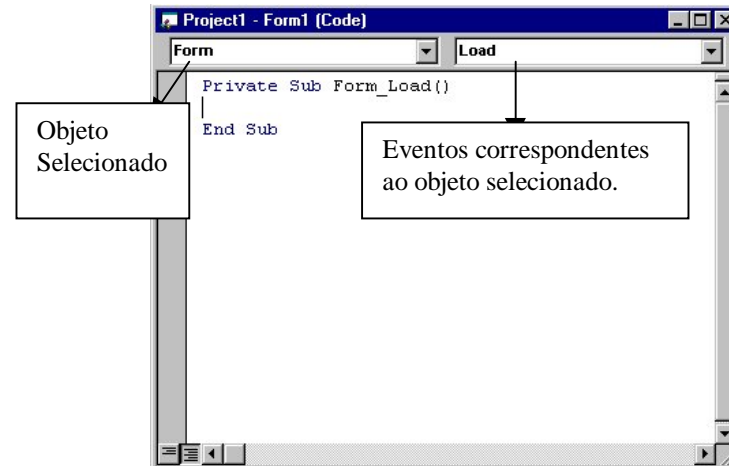
A partir do menu Project, escolha Add Module.

Janela de Código

É nesta janela onde você deve escrever seu código em VB. Cada objeto criado no Project tem sua própria Janela de Código.

Para visualizá-la:

Para exibí-la, dê um duplo clique sobre o objeto do qual o evento será tratado.





Controle Label

São textos informativos (legendas) em nossos forms. Não são editados pelos usuários.

Algumas propriedades:

Alignment: determina o alinhamento do texto.

AutoSize: determina se o tamanho do Label deve ser o mesmo tamanho que o seu texto.

BackColor: define a cor do fundo do label.

BackStyle: define se deve ser transparente ou opaco.

BorderStyle: define o tipo de borda.

Caption: determina o texto da sua legenda.

WordWrap: determina se o AutoSize deve expandir vertical ou horizontalmente.

NOTAS:



Controle Botão (Command Button)

Com este controle, você pode criar botões em sua aplicação. Quando você clicar sobre este botão, uma ação ocorrerá.

Algumas propriedades:

Caption: é a legenda do botão.

Cancel: ativa o botão quando o ESC for pressionado.

Font: esta propriedade controla a aparência do Caption. Você pode deixá-lo itálico, negrito, mudar o tipo de fonte.

Left, Top: coordenadas da extremidade esquerda superior do botão.

Height, Width: é a altura e a largura do botão. Você também pode alterá-las apenas redimensionando-as com o mouse.

Name: nome para o objeto.

Default: ativa o botão que o ENTER for pressionado.

Enabled: habilita ou desabilita um botão. Configure essa propriedade para True - para habilitá-lo ou para False, para desabilitá-lo.

ToolTipText: define o texto explicativo sobre a função do objeto.

Visible: define se um objeto deve estar visível (True) ou invisível (False) na tela.

Eventos:

Click: o que estiver escrito neste evento ocorrerá quando apertamos o botão.

NOTAS:

Criando teclas de atalho (ALT + <tecla>) :

Para criar essas teclas de atalho, quando for configurar a propriedade Caption basta colocar na frente da letra desejada como atalho um & (e comercial). Por exemplo:



No caso do botão de comando ao lado, quando o usuário teclar ALT+S, a ação programada no evento Click será executada.

Caixas de Diálogo



As Caixas de Diálogo permitem avisar ao usuário sobre algum problema, pedir algum parâmetro ou tomar alguma decisão. O VB possui duas funções para a construção de Caixas de Diálogo:

Função MsgBox

MsgBox(mensagem[, botões de diálogo][, título da caixa])

Parâmetros da função:

Mensagem: é a expressão que é mostrada na caixa de diálogo.

Botões de diálogo: número dos ícones que devem ser mostrados, o estilo que deve ser usado. Se for omitido, o valor padrão para o ícone é 0.

Título da caixa: é a expressão que deve ser mostrada na barra de título da caixa de diálogo. Se não for especificado, será mostrado o nome da aplicação.

Você pode criar o MsgBox de dois modos: como função ou como procedimento. A diferença entre função e procedimento é que a primeira retorna um valor e a outra, não. Desse modo, quando usamos uma função podemos tomar diferentes caminhos em nossa aplicação.

MsgBox usada como um statment:

```
Sub cmdmsg_Click ()  
    mensagem = "Esta é uma mensagem"  
    dialogo = vbOkOnly + vbExclamation  
    titulo = "Caixa de Dialogo"  
    MsgBox mensagem, dialogo, titulo  
End Sub
```

MsgBox usada como uma função:

```
Sub cmdmsg_Click()  
    Dim resposta As Integer  
    mensagem = "Voce quer sair?"  
    dialogo = vbYesNo + vbQuestion  
    titulo = "Caixa de Saida"  
    resposta = MsgBox(mensagem, dialogo,  
        titulo)  
    If resposta = vbYes Then  
        End  
    End if
```

NOTAS:

O número máximo de caracteres na mensagem é de 1024, dependendo do tamanho dos caracteres usados. Se a mensagem tiver mais de uma linha, você pode usar a função CHR(13).

Obs. Chr() - é uma função que retorna o caracter associado ao valor do código. 13 é o número correspondente a tecla ENTER.

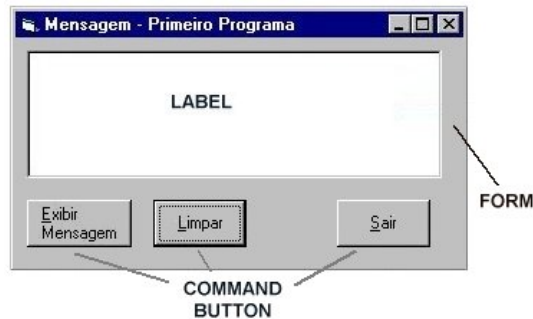
Teste!

Crie um novo Project, com botões de mensagem e para saída. Utilize os código acima.

Observe que a Caixa de Diálogo somente perderá o foco, quando você responder à questão. É possível mudar o foco para outro aplicativo. Mas o seu aplicativo ficará parado na MsgBox.

Exercício 1: Meu Primeiro Programa

Fazer um programa que exiba num Label uma mensagem qualquer e quando for ser terminado, exibir uma mensagem de saída (utilize a função MsgBox).



Tela do Exercício.

Terminando uma aplicação no VB:

Use o comando END para terminar (sair) de uma aplicação.

Propriedades Padrão:

Os objetos do Visual Basic possuem propriedades que são padrão, ou seja, quando você apenas utiliza o nome do objeto, o VB já sabe qual a propriedade que você está se referindo. No caso do objeto Label, a propriedade padrão é o Caption.

Omitindo as propriedades padrão, o seu programa fica mais rápido.

Resolução do Exercício:

Form

Caption = Mensagem – Primeiro Programa
Name = frmexerc1

CommandButton

Caption = &Limpar
Name = cmdlimpar

CommandButton

Caption = &Sair
Name = cmdsair

CommandButton

Caption = &Exibir Mensagem
Name = cmdmensagem

Label

Caption = (BRANCO)
name = lblmensagem

```
Private Sub cmdmensagem_Click()  
    lblmensagem = "Meu Primeiro Programa!"  
End Sub
```

```
Private Sub cmdlimpa_Click()  
    lblmensagem = ""  
End Sub
```

```
Private Sub cmdsair_Click()  
    MsgBox "Terminando o programa!!"  
    End 'Termina uma aplicação  
End Sub
```

Figuras no Visual Basic

Para você inserir uma figura em qualquer objeto do VB, este objeto precisa ter a propriedade Picture. Esta propriedade consegue visualizar os seguintes formatos: .BMP, .ICO, .JPG, .GIF, .WMF.

Se você precisa trabalhar em tempo de execução, é possível:

- Copiar uma figura de outro objeto;
- Copiar uma figura da área de transferência (Clipboard);
- Copiar uma figura de um arquivo.

Por exemplo:

```
MeuForm.Picture = picture1.picture  
MeuForm.Picture = Clipboard.GetData()  
MeuForm.Picture = LoadPicture("c:\temp\carro.bmp")
```

Para limpar a figura, utilize a função LoadPicture sem parâmetros:

```
MeuForm.Picture = LoadPicture( )
```

No Visual Basic existe dois controles que permitem o trabalho com figuras. São eles: Picture Box e Image Box.

O Controle Image Box gasta menos memória que a Picture Box, mas possui menos recursos de programação.

Redimensionamento de Figuras

Os controles Picture e Image Box possuem propriedades que permitem o redimensionamento das imagens inseridas.

Para o controle Picture, a propriedade é AutoSize que deve ser definida com True/False. Essa propriedade se definida como True, permite que a imagem seja exibida do seu tamanho original, redimensionando o controle Picture Box.

Para o controle Image Box, a propriedade que permite o redimensionamento da imagem é Stretch. Essa propriedade definida como True, redimensiona a imagem do tamanho que você desenhou o controle Image Box no formulário.

NOTAS:

O redimensionamento de imagens em Bitmaps resulta em perda de definição. Para evitar maiores danos, você pode utilizar imagens no formato .WMF (MetaFiles do Windows).

Teste!

Crie um novo Project com um novo Form e neste desenhe os dois controles.

Na propriedade Picture dos dois controles, selecione uma imagem. Para o Picture Box, defina a propriedade AutoSize como True e para o Image Box defina a propriedade Stretch como True. Execute o programa e confira o que acontece.



Picture Box



Image Box



Controle Image Box

Permite inserir uma imagem à sua aplicação.

Propriedades:

Picture: permite escolher o desenho que queremos mostrar. Os formatos permitidos são: .BMP, .JPG, .GIF, .WMF, .ICO.

Stretch: define se a figura deve ou não se ajustar ao tamanho do controle Image Box.



Controle Text (Caixa de Texto)

Permite a entrada e edição do texto inserido na caixa.

Propriedades:

MaxLength: determina o tamanho máximo do texto a ser digitado.

Multiline: determina que a caixa de texto terá mais de uma linha.

PasswordChar: define como o texto digitado na caixa de texto deve aparecer. Se estiver com valor True, qualquer valor digitado aparece como um * (asterisco). Apenas funciona se a propriedade **Multiline** estiver como False.

ScrollBars: determina se a caixa de texto deve ou não ter barras de rolagem quando a propriedade **Multiline** é True.

Text: é nesta propriedade que temos o texto digitado. Você também pode atribuir textos à caixa de texto em tempo de execução. Por exemplo: `text1.text = "Curso de VB"`.

Evento(s):

Change: ocorre toda vez que o conteúdo da caixa de texto for alterado.

LostFocus: ocorre quando o objeto perde o foco.

NOTAS:

Método(s):

Setfocus: este método é utilizado para dar o foco do cursor à caixa de texto.

Exemplo: `text1.Setfocus` ‘com isso, o cursor – do ponto que estiver na aplicação – passar a estar na caixa de texto Text1.

O controle Caixa de Texto é um objeto muito útil para obter os dados que o usuário insere através do teclado.



Controle Option

Permite a seleção de uma entre várias opções.

Propriedades:

Caption: legenda da opção.

Enabled: habilita ou desabilita o controle na tela para acesso do usuário.

Value: possui dois valores - True ou False. Quando está selecionado é igual a True.

Visible: determina se a opção deve ou não ser mostrada na tela.

Evento(s):

Click: ocorre toda vez houver um clique sobre o Option Button.



Controle Check Box

Apresenta opções das quais podemos escolher todas, nenhuma ou algumas.

Propriedades:

Caption: define a legenda do controle na tela.

Enabled: habilita/desabilita a opção.

Name: define o nome do controle na aplicação.

Value: esta propriedade pode ser definida com três valores:

Unchecked (não selecionado)

Checked (selecionado)

Grayed (não está selecionado nem não selecionado)

Visible: define se o controle deve ser visível ou não para o usuário.

Evento(s):

Click: ocorre toda vez houver um clique sobre o Option Button



Controle Frame

É usado para identificar um grupo de opções ou para dividir o formulário em várias funções.

Propriedades:

Caption: o título do Frame.

Visible: indica se o frame e os controles inseridos nele, devem ou não estar visíveis.

Para criar um grupo de opções ou dividir o formulário em várias funções é preciso primeiro criar o frame e depois criar os controles dentro dele.

NOTAS:



Controle ListBox

O controle ListBox mostra uma lista de itens que podem ser selecionados pelos usuários.

Propriedades:

Column: define o número de colunas que a ListBox deve mostrar. A primeira coluna é dada pelo número 0, a segundo pelo número 1 e assim por diante.

Listindex: se nenhum item for selecionado, esta propriedade terá o valor -1. O primeiro item da lista é o **ListIndex** = 0.

ListCount: retorna o número de itens que a sua lista possui. O valor desta propriedade é sempre um número a mais do valor do ListIndex.

MultiSelect: define como o usuário irá selecionar itens da ListBox. São 3 tipos de seleção:

0 – None: permite que 1 item seja selecionado por vez.

1 – Simple: permite que mais de um item seja selecionado.

2 – Extended: permite que mais de um item seja selecionado e ainda utilizar tecla como SHIFT e CTRL para selecioná-los.

Style: define o estilo de aparência da ListBox. Os estilos permitidos são:

0 – **Standard:** estilo normal de uma listbox, apenas com a lista dos itens.

1 – **CheckBox:** ao lado de cada opção da listbox é colocado uma caixa de CheckBox para seleção do item.

As propriedades **ListIndex** e **ListCount** apenas poderão ser alteradas em tempo de execução, ou seja, elas serão configuradas em programação.

NOTAS:

Controle ListBox:

Observações:

Para adicionar ou deletar itens na lista, utilize os métodos AddItem e RemoveItem.

Para adicionar itens na lista:

LstLínguas.additem "Português" → este item é o 0 (listindex)

LstLínguas.additem "Japonês" → este item é o 1 (listindex)

LstLínguas.additem "Inglês" → este item é o 2 (listindex)

LstLínguas.additem "Árabe" → este item é o 3 (listindex)

Para excluir itens da lista:

LstLínguas.removeitem lstlínguas.listindex

Usando esta sintaxe você pode remover qualquer item que você selecionar da Listox.

Ou

LstLínguas.removeitem 1

LstLínguas.removeitem 3

Utilizando esta sintaxe, apenas será removido um determinado item.

Para limpar uma Lista inteira:

LstLinguas.Clear

Utilizando este método, todos os itens da lista serão apagados.



Controle Combo Box

Um controle Combo Box é uma combinação das características do TextBox e do ListBox. Os usuários podem entrar com as informações na TextBox ou selecionar um item a partir da ListBox.

Propriedades:

Sorted: indica se a ComboBox deve ser ordenada automaticamente.

Style: retorna um valor indicando o tipo de Combo Box e o comportamento da lista. Esta propriedade é apenas para leitura em tempo de execução.

A propriedade Style pode ser definida:

0 (Default) **DropDown Combo**. É uma combinação de lista e caixa de texto. É possível selecionar o item pela lista ou digitando na caixa de texto.

1 **Simple Combo**. Combina uma caixa de texto com um caixa de Listagem. É possível selecionar o item pela lista ou digitando na caixa de texto. Por default, uma caixa Simple combo não mostra nenhum item da lista. Altere propriedade Height para mostrar mais itens da lista.

2 **DropDown List**. Este estilo apenas permite a seleção de um item pela lista.

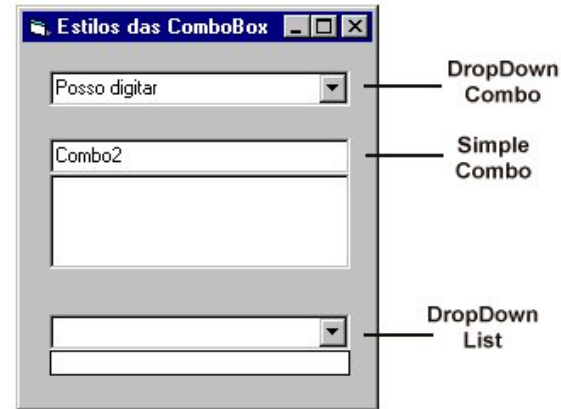
Para decidir qual dos Estilos utilizar na propriedade **Style**, é preciso levar em consideração que tipo de ação o usuário poderá ter realizar no controle:

Configure esta propriedade para 0 (DropDown Combo) ou 1 (Simple Combo) para permitir que o usuários tenha uma lista de escolha. Esses dois estilos permite que o usuário escolha o item da lista pela caixa de texto, ou seja, digite o valor do texto que ele deseja.

Configure a propriedade para 2 (DropDown List) para mostrar uma lista fixa para seleção de um item, ou seja, este estilo não permite que o usuário digite na caixa de texto o valor desejado.

NOTAS:

Para adicionar, excluir ou limpar itens do ComboBox, utilize os mesmos métodos utilizados na ListBox (AddItem e RemoveItem - página 26).



Exemplos da ComboBox

Estruturas Lógicas e Condicionais

Estruturas Lógicas e Condicionais são instruções de programa que fazem perguntas sobre alguma propriedade, objeto ou variável que retornam o valor TRUE (verdadeiro) ou FALSE (falso). Por exemplo:

```
soma = 50
```

a resposta vai ser TRUE, se a variável soma contiver 50 e FALSE se contiver um valor diferente de 50.

Estruturas de Decisão

If ... Then ...else ... end if

Com essa estrutura, é possível avaliar uma condição e de acordo com o resultado, tomar uma certa decisão. Sintaxe:

```
If <condição> then
    <instrução>
elseif <condição> then
    <instrução>
else
    <instrução>
end if
```

<condição> é uma expressão condicional e <instrução> é uma instrução na Linguagem do Visual Basic. Por exemplo:

```
if soma = 50 then
    MsgBox "A soma está entre 10 e 60".
```

Select Case

Formato:

```
Select Case <variável>
Case <expressão>
    <comandos>
Case <expressão>
    <comandos>
Case Else
    <comandos>
End Select
```

Exemplos:

‘Variável numérica

```
Select Case var1
```

```
Case 1 to 5
```

Comandos ... ‘Este bloco será executado somente se a variável var1 tiver os valores: 1,2,3,4 e 5

```
Case 32, IS <10
```

Comandos ... ‘Este bloco será executado somente se a variável var1 for 32 ou menor que 10.

```
End Select
```

‘Variável String

```
Select Case texto
```

```
Case "A" to "a"
```

Comandos ‘executará este bloco apenas se a palavra tiver letras contidas no intervalo.

```
End Select
```

Estruturas Lógicas e Condicionais Estruturas de Repetição

Do While

Sintaxe 1: Do while <condição> <comandos> Loop	Sintaxe 2: Do <comandos> Loop While <condição>
--	--

Os dois formatos do comando Do desempenham a mesma função: executar um bloco de comandos até que a condição seja verdadeira. A diferença é que na segunda sintaxe, o teste da condição é feita após a primeira execução dos comandos dentro do Do e na primeira, o teste é feito antes.

Do Until

Sintaxe 1: Do Until <condição> <comandos> Loop	Sintaxe 2: Do <comandos> Loop Until <condição>
--	--

Os dois formatos do comando Do Until desempenham a mesma função: executar um bloco de comandos até que a condição seja verdadeira. A diferença é que na segunda sintaxe, o teste da condição é feita após a primeira execução dos comandos dentro do Do e na primeira, o teste é feito antes.

FOR

Sintaxe:

```
For contador = valor_inicial to valor_final {Step incremento}
    <comandos>
Exit For
    <comandos>
Next
```

Esta estrutura de controle executa um bloco de comandos por um número fixo de vezes. O incremento pode ser negativo ou positivo.

O comando Exit For é utilizado para interromper e sair do For ... Next.

NOTAS:

Utilize o For ... Next para executar um conjunto de instruções um determinado número de vezes.

Utilize o Do While e Do Until para executar um conjunto de instruções até que uma determinada condição seja satisfeita.

Operadores Aritméticos

A tabela abaixo nos mostra os operadores que o VB aceita:

Soma: +	Ex.: var = 50 + 30.
Subtração: -	Ex.: var = valor1 - valor2
Divisão não inteira: /	Ex.: var= valor2 / 3
Divisão Inteira: \	Ex.: var = 50 \ 3
Resto da divisão: MOD	Ex.: 1 MOD 3
Exponenciação: ^	Ex.: var = valor1 ^ 2
Concatenação de String	Exe.: var = "Tes" & "te"

A Ordem de Operações

Existe uma lista de regras internas ao VB que diz qual o operador que deve ser utilizado primeiro. A tabela a seguir nos mostra do primeiro ao último operador a ser avaliado pelo VB:

() Parênteses	Os valores entre parênteses SEMPRE são avaliados primeiro.
^	A exponenciação (elevar um número a uma potência) é avaliada em segundo lugar.
-	Negação (criar um número negativo)
* /	Multiplicação e Divisão.
\	Divisão inteira
Mod	Divisão de Restos
+ -	Adição e subtração.

NOTAS:

Operadores Lógicos

Or	Se uma das condições for True, então o resultado é True
And	Se ambas as condições for True, então o resultado é True.
Not	Se a expressão condicional for True, então o resultado é false. Se a expressão condicional for false, então o resultado é true.
Xor	Se uma e apenas uma expressão condicional for True, então o resultado é True. Se ambas as condições forem True ou False, então o resultado é false.

Exercício 2: Equipando seu Escritório

Fazer um programa que a partir de uma lista de opções, você possa escolher equipamentos para informatizar seu escritório.



Tela do exercício.

```
Private Sub Form_Load()  
    lstinformatica.AddItem "Disquete"  
    lstinformatica.AddItem "Impressora"  
    cbopagamento.AddItem "Dólar"  
    cbopagamento.AddItem "Yen"  
End Sub  
  
Private Sub cmdnsair_Click()  
    End  
End Sub
```

Resolução do Exercício:

Form

Caption = Loja de Equipamentos
Name = frmexerc

ComboBox

name = cbopagamento

CommandButton

Caption = &Sair
Name = cmdnsair

CheckBox

Caption = Secretária Eletrônica
name = chkAuxiliar1

OptionButton

Caption = Macintosh
name = OptComput1

Label

AutoSize = True
Caption = Equipamentos
Auxiliares para Informática
name = lblinformatica

Image

Name = Image1
Stretch = True

Image

name = Image3
Stretch = True

Image

Name = Image5
Stretch = True

Frame

Caption = Equip. Auxiliar de
Escritório
Name = frame_aux

ListBox

name = lstinformatica

Frame

Caption = Tipo do Computador
name = frame_opt

CheckBox

Caption = Calculadora
name = chkAuxiliar

OptionButton

Caption = PC
name = OptComput

Label

AutoSize = True
Caption = Equipamentos a escolher:
name = lblsecolher

Image

Name = Image2
Stretch = True

Image

Name = Image4
Stretch = True

Label

AutoSize = True
Caption = Equipando o seu
Escritório
name = lbltitulo

Resolução do Exercício:

```
Private Sub cbopagamento_Click()
    If cbopagamento.ListIndex = 0 Then
        Image6.Picture = LoadPicture("...\dollar.wmf")
        Image6.Visible = True
    ElseIf cbopagamento.ListIndex = 1 Then
        Image6.Picture = LoadPicture("...\yen.wmf")
        Image6.Visible = True
    Else
        Image6.Visible = False
    End If
End Sub

Private Sub chkAuxiliar_Click()
    If chkAuxiliar.Value = 1 Then
        Image1.Picture = LoadPicture("...\answmach.wmf")
        Image1.Visible = True
    Else
        Image1.Visible = False
    End If
End Sub

Private Sub chkAuxiliar1_Click()
    If chkAuxiliar1.Value = 1 Then
        Image2.Picture = LoadPicture("...\calcultr.wmf")
        Image2.Visible = True
    Else
        Image2.Visible = False
    End If
End Sub
```

```
Private Sub Istinformatica_Click()
    If Istinformatica.ListIndex = 0 Then
        Image5.Picture = LoadPicture("...\disk35.wmf")
        Image5.Visible = True
    ElseIf Istinformatica.ListIndex = 1 Then
        Image5.Picture = LoadPicture("...\printer.wmf")
        Image5.Visible = True
    Else
        Image5.Visible = False
    End If
End Sub

Private Sub OptComput_Click()
    If OptComput.Value = True Then
        Image4.Picture = LoadPicture("...\computer.wmf")
        Image4.Visible = True
    Else
        Image4.Visible = False
    End If
End Sub

Private Sub OptComput1_Click()
    If OptComput1.Value = True Then
        Image4.Picture = LoadPicture("...\pcomputr.wmf")
        Image4.Visible = True
    Else
        Image4.Visible = False
    End If
End Sub
```



Controle Timer

O controle timer é utilizado para disparar procedimentos e funções automaticamente de tempo em tempo. Por exemplo, você pode utilizar um controle Timer para verificar o tempo que um usuário leva para realizar determinada função na aplicação.

Quando este controle é desenhado no Formulário, é apenas visível em tempo de Design, ou seja, em tempo de desenvolvimento. Para o usuário da aplicação, fica transparente o funcionamento do controle, ele apenas é executado internamente.

Propriedades:

Enabled: define quando o Timer deve estar ativo ou não.
Valores: False/True.

Interval: determina o período de tempo no qual o objeto deve ser ativado. O tempo deve ser inserido em segundos.

Para que uma ação seja disparada pelo Timer, a propriedade Enabled deve ter o valor True e a propriedade Interval diferente de 0.

Eventos importantes:

Timer: este evento é disparado no intervalo de tempo definido pela propriedade **Interval**.

NOTAS:

Exercício 3: Criando um Relógio

Crie um relógio que seja atualizado de segundo em segundo pelo controle Timer.



Tela do Exercício

Resolução do Exercício:

Form

Caption = Relógio
Name = frmexerc

Timer

Enabled = True
Interval= 1
Name = mrHora

CommandButton

Caption = &Sair
Name = cmd sair

Label

AutoSize = True
Caption =
name = lblHora

```
Private Sub tmrHora_Timer()  
    lblHora.Caption = Time  
End Sub
```

```
Private Sub cmdSair_Click()  
    End  
End Sub
```



Controles ScrollBars

Esse controle pode ser vertical ou horizontal. As propriedades, eventos e métodos para esses objetos são os mesmos.

Propriedades:

Value: é o valor (posição) do scroll.

Max, Min: define o valor máximo e o valor mínimo da barra, respectivamente.

LargeChange: é o salto quando clicamos no corpo do scroll.

SmallChange: é o salto quando clicamos nas setas do scroll.

Eventos mais utilizados:

Change: será executado o código escrito toda vez que o cursor for movimentado (com o mouse, setas de movimento, cliques nas setas do scroll, etc.)

Scroll: somente ocorre quando movimentamos o cursor com o mouse. Não ocorre quando usamos as setas.

NOTAS:

Tipos de Dados

Um programador cria uma variável para armazenar os resultados de um cálculo, criar nomes de arquivos, processar entrada de dados, etc. Também podem ser armazenados nomes e valores de propriedades dos objetos.

Tipos de dados de Variáveis:

Tipo	Tamanho	Caracter de Tipo
Boolean	2 Bytes	
Byte	2 bytes	
Date	8 bytes	
Integer	2 bytes	%
Long (Long Integer)	4 bytes	&
Single (Ponto Flutuante)	4 bytes	!
Double (Ponto Flutuante)	8 bytes	#
Currency	8 bytes	@
String	1 byte por caracter	\$
Variant	qualquer valor ou caracter	

Obs.: As operações com o tipo de dado Currency são mais rápidas e exatas que as com tipo Single e Double.

Declaração:

No Visual Basic não é necessário fazer a declaração de todas as variáveis que você estiver utilizando e quando isso acontece, a linguagem assume que o Tipo de Dados da variável não declarada é Variant.

Você pode declarar suas variáveis utilizando: Dim, Private, Static ou Public. Existem duas maneiras de declarar variáveis no VB:

Usando AS	Usando o Caracter de Tipo
Dim j AS Integer	Dim I%
Dim nome AS String	Dim nome\$

Para fazer declaração de mais de uma variável na mesma linha de código, utilize a sintaxe:

Dim endereco as String, nome as String

Observe que você deve especificar o tipo de dado para cada uma das variáveis declaradas. Se você utilizar o exemplo abaixo:

Dim endereco, nome as String

Apenas a variável nome será do tipo String, o VB assumirá a variável endereco como tipo Variant.

Variáveis do tipo Variant

Variant pode assumir qualquer tipo de dado, ou seja, dependendo do valor dos dados que for atribuído a essa variável, o VB fará a conversão automaticamente para o tipo correto. Pode ser numérico, data ou string.

Na inicialização do VB, o tipo de dados Variant assume um valor Empty que não é nem NULL, nem branco, nem zero.

Tipos de Dados (cont.)

Declaração e Operação de um Tipo Variant

Dim QQValor 'Com este tipo de declaração o VB assume por default o tipo Variant
QQValor = 17 'Valor Numérico
QQValor = "17" 'String de 2 caracteres
QQValor = QQValor - 15 'Valor numérico = 2 (17 - 2)
QQValor = "U" + QQValor 'String U2

Definindo variáveis Strings de tamanho variável e fixo:

Quando não se tem certeza do tamanho de uma String, é possível declará-la com tamanho variável. Caso contrário, o melhor a fazer é declarar o tamanho fixo. Exemplo:

Dim texto as String
Dim texto as String * 50 'declaração de uma variável String com 50 posições.

Inicialização de Variáveis

Quando uma aplicação em VB é inicializada, automaticamente, as variáveis numéricas recebem 0 (zero) e as variáveis String com tamanho definido, com brancos.

Convenção de Nomes de Variáveis

1. primeiro caracter deve ter uma letra (alfanumérico).
2. Podem ter letras, números e *underscore* (_).
3. Palavras reservadas do VB não podem ser declaradas como variáveis.
4. O tamanho máximo que se pode assumir é 255 caracteres.
5. Num mesmo escopo, a variável deve ser única.

Escopo das Variáveis

Local

Uma variável local é reconhecida apenas na procedure/função ou módulo em que foi criada. Para criar um vaiável local, coloque sua definição dentro da procedure/função ou módulo. As variáveis que estão dentro da seção Declarations de um Módulo.BAS são locais a este módulo.Por exemplo:

Dim nome As String

Public

Existem variáveis que são reconhecidas por todas as procedures/funções de um Form.

Para isso, selecione o Form desejado, clique no botão View Code da janela Project e declare a variável na seção General. Desse modo, você declara uma variável pública para o Formulário em questão.

Quando uma variável *Public* é declarada em um módulo esta é reconhecida por toda sua aplicação.

Para declarar uma variável pública, seja em um formulário ou em um módulo use a sintaxe:

Public <nome_variável> As <tipo_dados>

Constantes

São variáveis que contém um valor fixo durante toda a execução da aplicação.

Para declarar uma variável constante, use a palavra reservada Const, por exemplo:

Const nome = "João" 'a variável NOME conterà sempre o valor "João" durante a execução de sua aplicação.

Tipos de Dados (cont.)

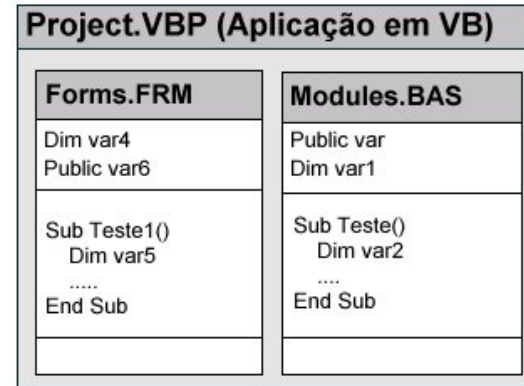
Escopo de uma Constante

Uma constante quando é definida ao nível de Formulário, será apenas local, não sendo possível mudar seu escopo. Se for preciso a declaração de uma constante pública, será necessário declará-la em um módulo. Utilize a seguinte sintaxe:

```
Public Const <nome_variável> = <valor>
```

Static

Uma variável Static não é inicializada toda vez que um procedimento ou função é chamada, será inicializada apenas uma vez quando o Formulário é inicializado (LOAD). Uma variável Static só pode ser declarada dentro de uma função/procedimento, ou seja, ela é Local e existe enquanto o formulário está ativo.



NOTAS:

Escopo das variáveis.

Na figura acima, a variável var é pública, ou seja, o projeto inteiro pode trabalhar com esta variável.

Qualquer variável declarada com a cláusula DIM é local ao objeto, então as variáveis var1 só pode ser utilizada pelo Módulo e a var2 ser utilizada apenas pelo procedimento no qual foi declarada.

O procedimento Teste – local ao Módulo - consegue trabalhar com as variáveis var, var1, var2 e var6, esta última, desde que seja declarada como pública e se faça referência ao formulário no qual ela foi declarada.

O procedimento Teste1 – local ao Formulário – consegue trabalhar com as seguintes variáveis: var, var4, var5 e var6.