

HISTÓRICO

O **BASIC** surgiu já há algum tempo, foi o pontapé inicial na carreira de Bill Gates, quando largou a faculdade com um amigo para fundar a **MICROSOFT** e fazer em duas semanas um interpretador para o **BASIC**, primeiro produto a ser comercializado pela Microsoft. Por esta empreitada conseguiu que a IBM, que estava lançando o seu PC e não esperava tanto sucesso, a permissão para fazer o sistema operacional deste computador, o MS-DOS, que marcou o início do monopólio da Microsoft.

No Brasil, muita gente entrou em contato com ele na década passada com os finados computadores MSX, TK90X, TK2000 e CP500, que o utilizavam como linguagem de programação. Nesta época era comum nos programas o uso descarado e extensivo do comando GOTO, que tornava um programa de algumas dezenas de linhas muito confuso e ilegível.

Quando o Windows se tornou padrão, pensava-se que a programação para este ambiente só seria possível com linguagens como C, sendo impossível alguma pessoa com pouca experiência fazer um programa por mais simples que fosse. Foi aí que surgiu o **VB**, com um novo **BASIC**, não mais aquele confuso, mas um estruturado e bem mais poderoso. Tornou-se a linguagem mais utilizada em todo o mundo, sendo considerada uma das causas do sucesso do Windows. Já em fevereiro de 1998 tínhamos em torno de 3 milhões de desenvolvedores de **Visual BASIC**.

Escrito originalmente em 1987 por Alan Cooper (através de um programa com nome Ruby, que unido ao **Quick BASIC** deu origem ao **Visual BASIC**). Foi lançado em 1991 com a versão 1.0. Em outubro de 1992 veio a versão 2.0. Temos a versão 5.0 e ultimamente temos a versão 6.0.

OBJETIVO DO DESENVOLVIMENTO DE SOFTWARE

“Gerar soluções em tecnologia de informação para que empresas fiquem mais organizadas, produtivas e competitivas e para que usuários trabalhem menos e com mais comodidade e prazer. Como também desenvolver soluções que venham ajudar nas mais diversas áreas que afetam indivíduos: educação, lazer, cultura, etc.”

NOVIDADES DO VB 5.0

- Novas construções de linguagem (For... Each ou With... End With);
- Public substituído por Global;
- Standard EXE (Abre EXE com Forms, **Módulos**, etc.);
- ActiveX: EXE, DLL e Control (ActiveX – construtor de servidores OLE); Controles customizados agora podem ser produzidos no **VB** e utilizados em qualquer programa para Windows que os aceite;
- Propriedade de Formulário **StartUp Position** (podemos centralizar com ela o **form**);
- Todas as janelas agora estão integradas em uma só (podendo ser separadas como antes);
- A Caixa de Ferramentas (ToolBox) pode agora receber outras Abas (Tabs), basta clicar com o botão direito sobre a ToolBox – Components – para adicionar;
- Pode abrir mais de um projeto por vez (File – Add Project);
- A janela Propertie agora pode ser listada por ordem alfabética ou por categoria (como no Access);
- Layout do Formulário – Preview do Form;

EDIÇÃO DE CÓDIGO:

- AutoList Members (opção para exibir lista de **Métodos** e properties ao escrevermos o nome de um objeto);
- Source Safe;
- Auto Quick Info (ao escrever nomes de funções aparecem seus parâmetros);
- Auto Data Tips (ao debugar vê valor da variável onde está o cursor);
- Drag-and-Drop Text Editing (copiar ou mover parte do **Código** selecionado. **Ctrl** para copiar);
- Barras de Edição (Botão direito sobre as barras de ferramentas);
- Replace All (cuidado, fazer com muita atenção);
- Compila realmente (coloca para código máquina: Menu Project – Project Propertie – Compile); existem sete tipos de compilação, além de ainda poderem ser gerado os velhos p-codes. Até o VB4, na janela Make EXE File, não restava nada mais a ser feito do que clicar o botão OK, agora você pode "regular" o executável, como:
 - * produza um compilado o mais rápido possível, embora fique de tamanho maior;
 - * produza um compilado de tamanho menor, embora fique mais lento;
 - * produza um compilado que explore melhor o processador Pentium;
 - * etc.

Num teste feito na versão beta, o executável compilado concluiu um cálculo em 2 segundos, enquanto o interpretado levou um minuto. Mas a coisa não é sempre assim.

- Executável menor e programa mais rápido;
- Gera **Código** apenas em 32 BITS (diferente do 4.0);
- Novos tipos de dados: Object, Byte, Variant Decimal, entre outros.

OUTRAS NOVIDADES:

- O editor de **Código**, aquela janela onde você escreve o programa, ficou inteligente, quando você começa a escrever um comando, ele já mostra a sintaxe e escreve para você (basta clicar), não é preciso mais ver a ajuda. Esta novidade foi relatada por um programador americano que testou a versão beta como inacreditável;

- Wizards que ajudam você a fazer o programa. Por exemplo, todo editor de texto tem uma janela principal com os menus arquivo, novo, abrir, salvar... toolbar com a figura da pasta aberta para abrir, figura do disquete para salvar, uma janela de ajuda, etc. Com o wizard, você seleciona um tipo de programa que o **VB 5.0** gera as janelas, menus, botões, toolbars que são comuns a aquele tipo de aplicação, cabendo ao programador a alterar a interface segundo as suas necessidades e escrever o **Código**. Acessados ao abrir o **VB5**.

- Criação de OCXs para a Internet; do mesmo modo que um programador de Java pode fazer uma applet, o programador de **VB** pode fazer sua OCX que rode numa página HTML, são estes os chamados controles ActiveX, que só podiam ser feitos em C ou Java.

- Você não vai mais precisar aprender Java para poder criar uma home page interativa, porque com o **VB 5**, que tem os controles TCP/IP do Internet Control Pack, seus programas vão poder ser rodados dentro do Internet Explorer.

- Controles para **INTERNET**:

Internet Transfer Control

Winsock e WebBrowser

Única propriedade: MDIChild = False

- Suporte a vários padrões gráficos, como jpg, gif, gif animado, pcx, tif e gráficos 3D.

- Interface drag n' drop.

- Suporte a múltiplos DBEngines.

- Suporte a vários resource files.

- Suporte a polimorfismo.

- Suporte a um tipo especial de herança.

OUTRAS INFORMAÇÕES:

Ainda será preciso a runtime vbrun500.dll
Lançamento em janeiro de 96.
Só vão poder ser gerados programas 32-bit.

No menu New Project agora existem várias opções, além disso, é permitido trabalhar com vários projetos ao mesmo tempo, isto quer dizer que é possível fazer um OCX enquanto ao mesmo tempo ele pode estar sendo testado em outro projeto ao mesmo tempo.

A Microsoft já disponibilizou o VB5 Control Creation Edition para download gratuito no seu site, que é uma edição "light" do VB5 e não gera executável. O tamanho é aproximadamente 6 MB, para quem tem um acesso rápido vale a pena.

Entre outras opções, a segunda versão do **Visual BASIC** para Windows 95 (a primeira foi a **4.0**), destacam-se aquelas referentes à criação de arquivos DLL e OCX, que são bibliotecas de comandos e ferramentas voltadas para o **Visual BASIC**, a ferramenta de criação de formulários referentes à banco de dados, a nova interface, que permite que se trabalhe em mais de um projeto ao mesmo tempo e muitas opções que permitem a reciclagem de **Código**, diminuindo o trabalho do programador.

Quanto a criação de DLL e OCX, este trabalho foi muito facilitado, uma vez que nas versões anteriores, era necessário se trabalhar com linguagens como o **Visual C++** e o Delphi. Agora, existe um **Módulo** que compila o **Código** fonte para DLL ou OCX.

Uma das grandes deficiências apontadas por programadores experientes era a dificuldade de se gerar um formulário no estilo Access para o **Visual BASIC**. Era necessário muito trabalho de interligação entre a tabela Access e os objetos **Visual BASIC**. Agora, existe um Wizard (assistente) que facilita e muito esta tarefa.

Na interface, houve sensíveis mudanças, com o uso de vários projetos ao mesmo tempo, o que torna a interação entre os projetos muito mais rápida. Neste sentido, a janela de propriedades, velha conhecida de qualquer programador em **Visual BASIC**, foi remodelada, apresentando as propriedades em ordem alfabética, como nas versões anteriores, ou na versão por categorias, deixando agrupadas propriedades de layout, de dados, etc., como no Microsoft Access, por exemplo.

Aproveitando o assunto de vários projetos ao mesmo tempo, uma das melhores vantagens do **Visual BASIC 5.0** foi a interligação entre os projetos: caso um form que esteja sendo usado por mais de um projeto seja alterado, o **Visual BASIC** avisará ao usuário dessa alteração e de que possivelmente poderá haver erro em algum projeto, uma vez que as propriedades não se cruzarão corretamente.

CONHECIMENTO EXIGIDO PARA APRENDIZADO

Normalmente, quando alguém pensa em fazer um programa para computador, já é um usuário com certa experiência. De programação mesmo, não é preciso saber nada (se bem que quem já sabe **BASIC** terá muito mais facilidade).

O conhecimento do inglês (mesmo que superficial) seria interessante, pois os comandos do **VB** são derivados desta língua e o help do **VB** (muito bom, por sinal) é todo em inglês. (Palavras emprestadas de uma página da INTERNET no Brasil (do programador em VB, Leandro Motta Barros, de quem tive ajuda substancial para esta apostila).

O QUE É O VISUAL BASIC?

É uma linguagem de programação **Visual** orientada a objetos (a partir da versão 4.0). Até a versão 3.0 ele era orientado a **Eventos**.

Aquela primeira frase quer dizer mais ou menos que com o **VB** é possível fazer programas para Windows utilizando programação orientada a objetos (OOP).

O que são objetos?

Objetos são coisas definidas por **CLASSES** e que possuem propriedades, **Métodos** e **Eventos**. Uma janela, um botão e um banco de dados são objetos. Mas para entender bem o que é um objetos ainda é preciso definir as tais das propriedades, dos **Métodos** e dos **Eventos**:

Comecemos pelas **Classes**. Uma janela é muito diferente de um botão, mas os dois são objetos. Isso acontece porque eles são definidos por **Classes** diferentes. **Classes** definem um objeto a grosso modo. Definem suas propriedades, seus **Métodos** e seus **Eventos**. **Classes** são os moldes dos objetos.

Um botão pode ser diferente do outro (maior ou menor, com uma legenda diferente...). Mas como isso é possível se eles pertencem à mesma classe? Porque suas propriedades são diferentes. As propriedades definem características mais específicas dos objetos. Um botão tem, por exemplo, propriedades que determinam sua altura, sua largura e sua legenda.

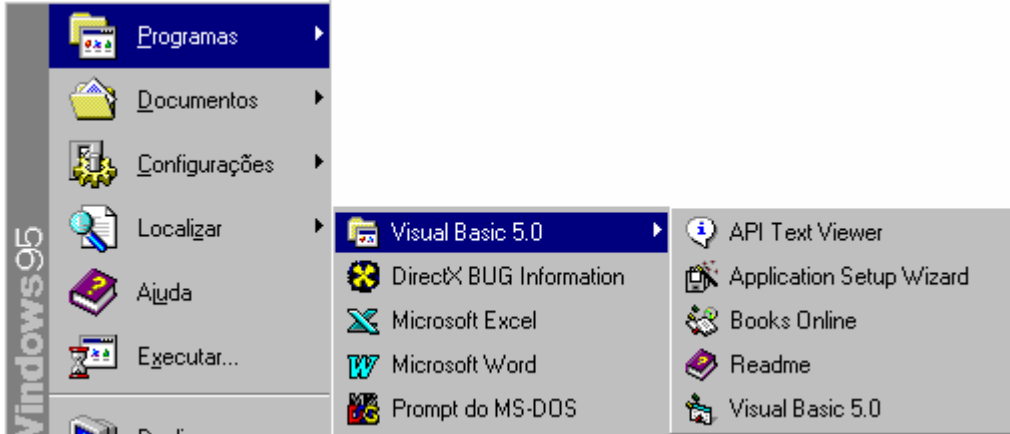
Um **Método** é uma coisa, uma função, uma ação que age sobre um objeto. Por exemplo, se tivermos uma lista (tipo as listas de tópicos no "procurar" da ajuda do Windows), adicionamos ou removemos itens através de **Métodos**. Ou ainda, se quisermos mover um certo objeto pela tela, poderíamos usar um **Método**.

Por fim, o **Evento**. Digamos que em um certo programa, pressionamos um botão e um Beep é emitido. Como o programa sabe quando apertamos o botão? Ele sabe porque quando um objeto da classe botão é clicado, é gerado um **Evento**. E é a esse **Evento** que está associado o **Código** (comandos) que produzirá o beep. Ocorrem também **Eventos**, por exemplo, ao modificarmos o texto de uma caixa de texto ou ao mover o cursor do mouse sobre uma janela.

Só para ficar mais claro, um exemplo não muito nobre: digamos que haja uma classe "Galinha". Um objeto Galinha poderia ter a propriedade "Idade", uma propriedade "CorDasPenas" e uma propriedade "Nome". Ela teria um **Método** "PoeOvo" e um **Método** "Cacareja". E ela geraria **Eventos** ao dormir ao acordar e ao ser ameaçada.

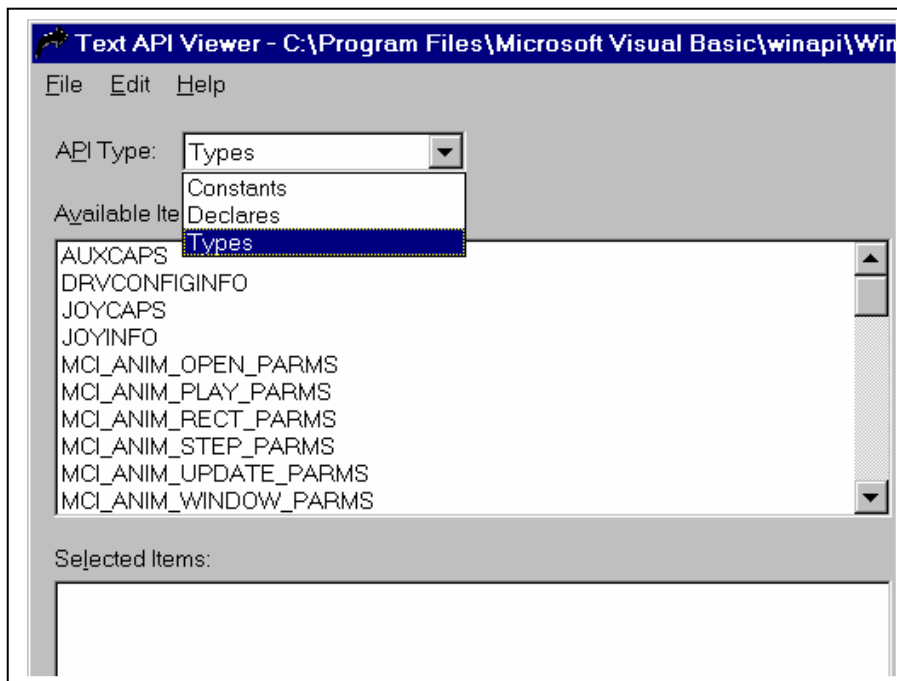
RODANDO O VB 5.0

Basta clicar em Iniciar – Programas – **Visual BASIC 5.0**.



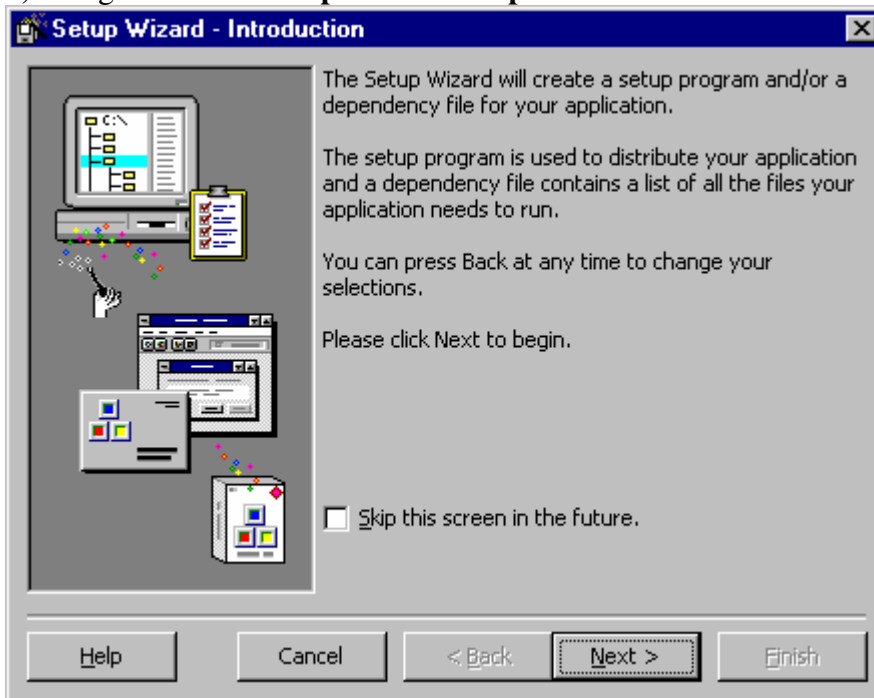
Observamos que o Grupo do **VB 5.0** tem cinco itens:

1)API Text Viewer



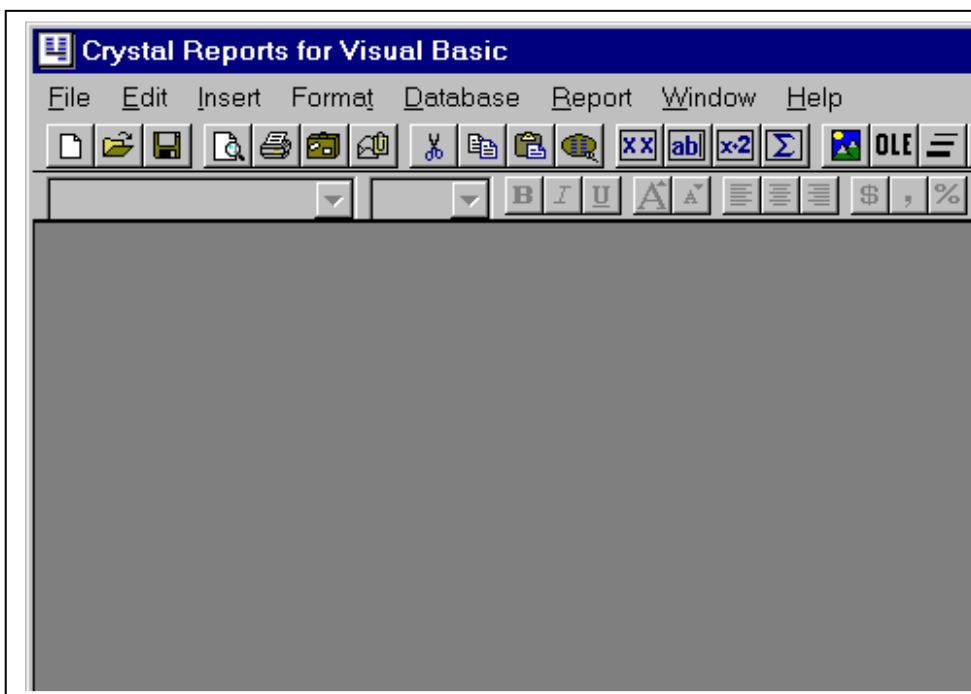
Onde podemos visualizar as APIs do **Windows**: suas **constantes**, **declarações** e **tipos de dados**. O mais importante desse utilitário é que podemos selecionar qualquer parte daqui e copiar e colar no **Código** do projeto do **VB**.

2) O segundo item é o **Application Setup Wizard**



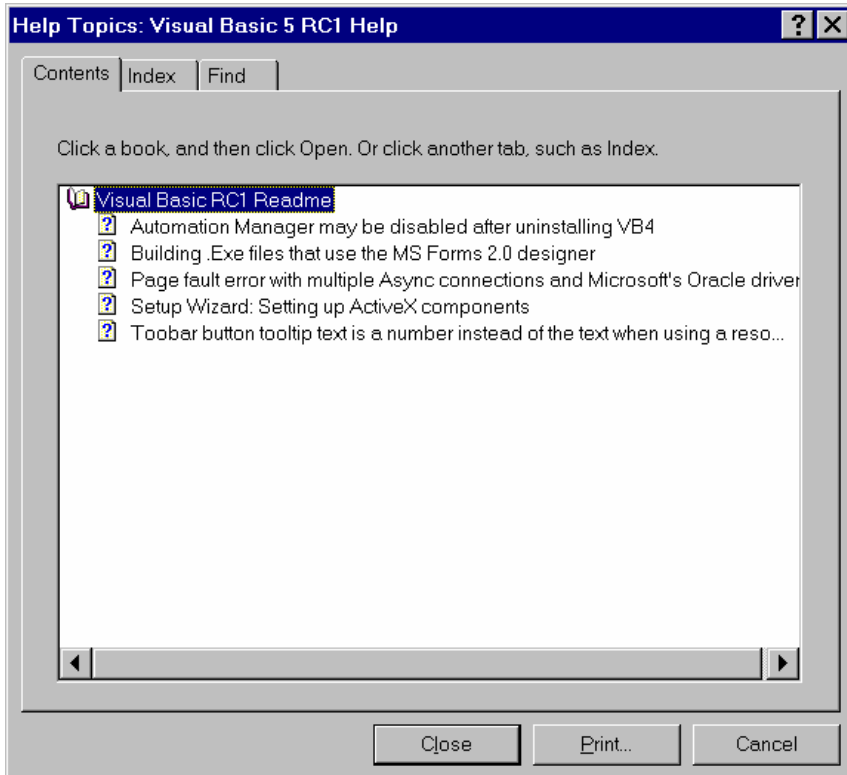
Este utilitário é o Assistente responsável pela geração dos discos (disquete ou Cd) instaladores do programa desenvolvido no **VB**. Também chega a gerar o **executável**, se assim o desejemos.

3) **Crystal Report** é o gerador de relatórios do **VB**. Desenvolvido pela Seagate (a mesma dos HDs) ele agiliza o desenvolvimento de **relatórios**, tornando essa tarefa menos trabalhosa que sua construção através de **Código**. Pode estar no Grupo do VB ou não, depende da instalação (no nosso caso não foi instalado).

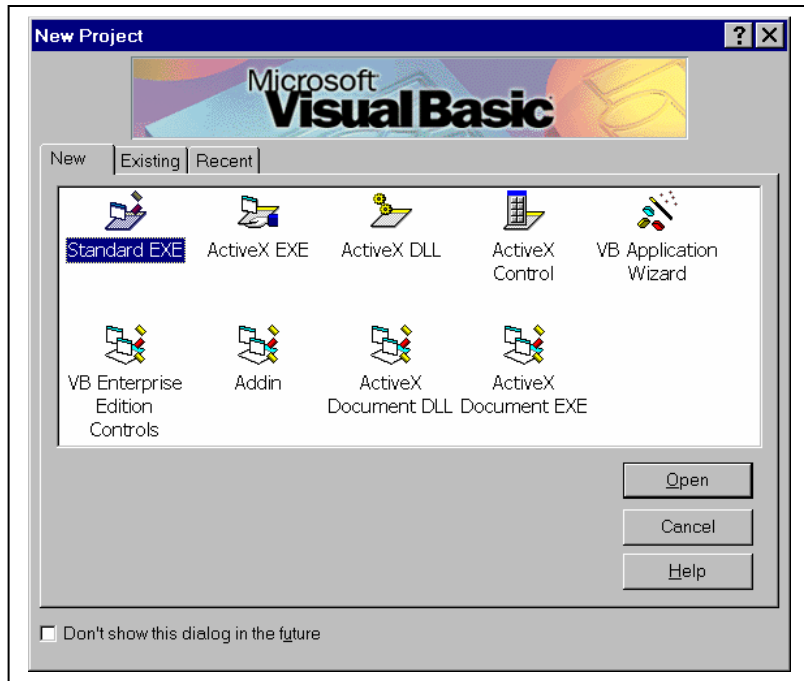


Books Online – São arquivos de Help existentes no CD de instalação e que podem ser instalados ou não no disco rígido e consultados através deste item.

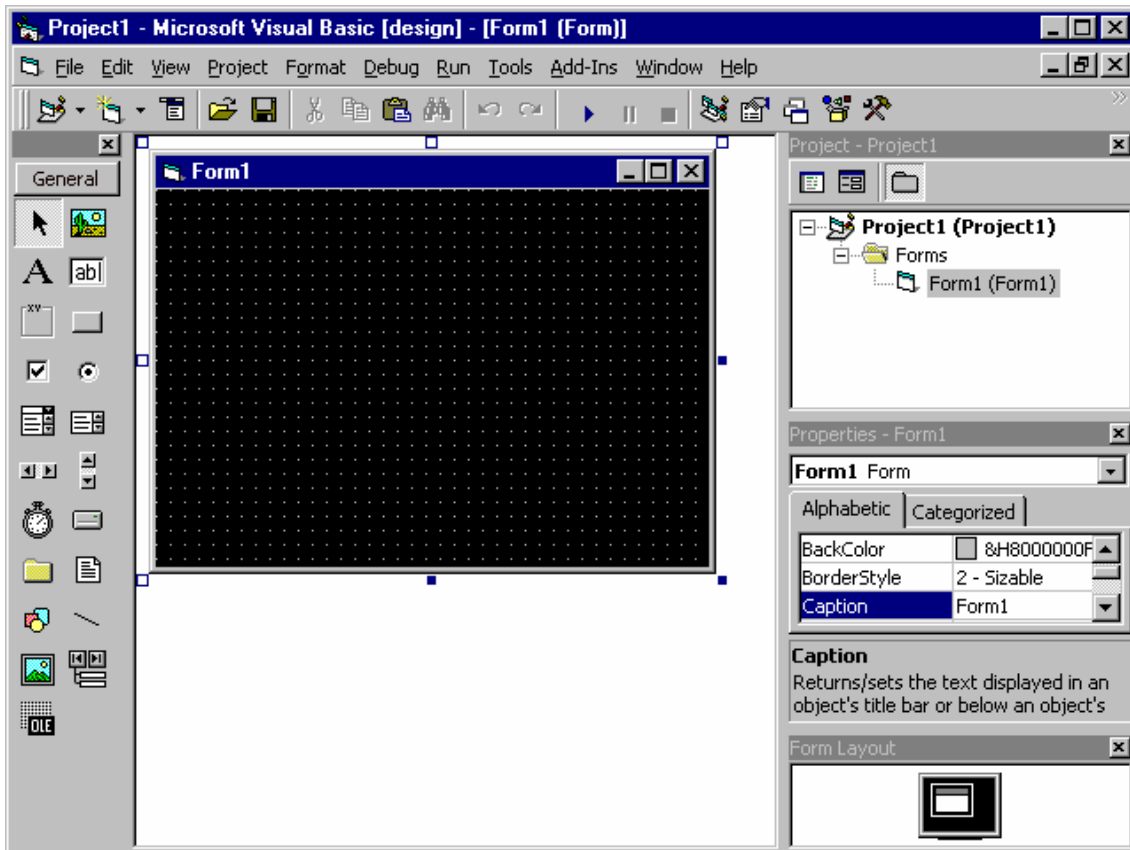
4) **Readme**, arquivo de Help que traz inúmeras informações importantes:



5) O quinto item é o próprio **VB 5.0**, que ao ser ativado abre a janela abaixo, onde devemos escolher o tipo de projeto que desejamos criar, abrir um existente ou os mais recentes:



Após escolher a opção padrão **Standard EXE** aparece a tela abaixo, que é o ambiente de programação do Visual Basic 5.0 Edição Interprise:



Todas as janelas acima (**Principal, Toolbox, Projeto, Propriedades e Formulário**) são exibidas por default.

* Toolbox - Providencia um conjunto de ferramentas que você usa durante o projeto para colocar controles dentro de seu form (janela de interface com o usuário).

* Menu Bar - Mostra os comandos usados para construir a sua aplicação.

* Form - É a janela na qual você irá construir a interface da sua aplicação. No form você irá adicionar controles , gráficos , figuras que irão criar a aparência final da sua aplicação.

- Project Window - Lista os forms , **Módulos de Código**.
-

* Properties Window - Lista as propriedades existentes para um form ou outro controle selecionado. Uma property (propriedade) é um valor ou característica associada a um objeto tais como tamanho , caption ou cor.

A TOOLBOX DO VB5:

(A ToolBox que aparece na próxima página é do VB4).



B) Agora a força do VB, a janela **TOOLBOX (Caixa de Ferramentas) - é a janela que traz os **Controles** (as ferramentas no **VB**)**



- **Pointer** – Não é um **Controle**, mas uma ferramenta que serve para mover e dimensionar **Controles**;



- **PictureBox** – Exibir figuras ou ícones nos **Formulários**. Também responde ao **Evento** click.



- **Label** – Etiqueta ou rótulo, exibe um texto que não pode ser editado pelo usuário. Uso: Títulos.



- **TextBox** – Exibe uma área onde o usuário pode digitar texto.



- **Frame** – Moldura para agrupar **Controles** para que funcionem de forma lógica.



- **CommandButton** – Botão de Comando. Executa ação ao ser clicado.



- **CheckBox** – Pode assumir valores falso ou verdadeiro. Em um mesmo **Formulário** vários **CheckBoxes** podem assumir o valor verdadeiro ao mesmo tempo.



- **OptionButton** – Pode também assumir valores falso/verdadeiro, mas em um mesmo **Formulário** somente um **OptionButton** pode assumir o valor verdadeiro.



- **ComboBox** – Combina caixa de texto com caixa de listagem. O usuário pode digitar a informação como também pode encolher da lista.



- **ListBox** – Exibe uma lista de itens que o usuário pode selecionar.



- **ScrollBar** Horizontal e Vertical – Permite selecionar com o mouse um determinado valor numa faixa de valores.



- **Timer** – Controla intervalos de tempo determinados pelo programador.



- **DriveListBox** – Exibe uma lista de drives onde o usuário pode selecionar um.



- **DirListBox** – Exibe uma lista de diretórios (pastas) e permite que o usuário selecione um.



- **FileListBox** – Idem para arquivos.



- **Shape** – Adiciona diversas formas geométricas ao **Formulário**.



- **Line** – Adiciona linhas a **Formulários**.



- **Image** – Exibe figuras ou ícones e funciona como um CommandButton quando clicado.



- **Data** – É o **DataControl**, que permite conectar-se a um **Banco de Dados** existente para extrair e manipular as informações contidas neste.



- Incorpora objetos de outros aplicativos em um **Formulário**.



- **CommonDialog** – Exibe um conjunto de caixas de diálogo comuns no **Windows**, abrir arquivo, imprimir, fontes, salvar, etc.



- **ToolBar** – Contem uma coleção de **Botões** e é utilizado para criar **Barras de Ferramentas** que estão associadas com uma aplicação.



- **StatusBar** – Cria uma **Barra de Status** para um programa com até 16 painéis contidos em uma coleção de **painéis**.



- **ProgressBar** – Barra que indica o progresso de uma operação.



- **ImageList** – Contém uma coleção de objetos **ListImage**. Para exibir imagens.



- **DBList** – Automaticamente preenche uma lista com campos de um **DataControl** e opcionalmente passa o campo selecionado para um segundo **DataControl**. O **DBCombo** é similar mas permite editar o campo selecionado.



- **DBGrid** – Exibe e manipula uma série de linhas e colunas representando **Registros** e **Campos** de um objeto **Recordset**.

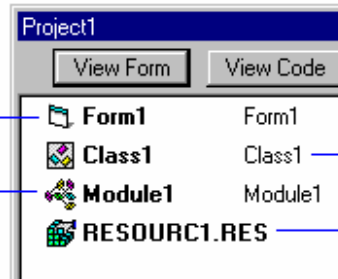
C) A JANELA DE PROJETO

As características do ambiente Windows (interface gráfica, janelas, objetos, **Eventos**, mouse, ícones, botões, etc.) são trabalhadas no **Visual BASIC**. O ponto de partida para o desenvolvimento do **Visual BASIC** é uma janela – o **Formulário**. Realmente uma janela é a interface entre o programa e o usuário. Ela interage com o usuário. Um **PROJETO** é um conjunto de arquivos que compõem a aplicação. Estes arquivos que compõem a aplicação podem ser visualizados na janela **Projeto** do VB (Project Window).

Esta é a do VB4

Formulário contém a descrição e o código associado a este.

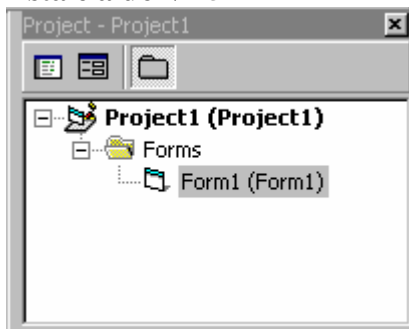
Um **Módulo** padrão contém declarações e procedimentos.



Um **Módulo Class** contém as definições características de uma **Classe** incluindo suas propriedades e **Métodos**.

Um arquivo **Resource** pode conter todos os Bitmaps, strings e outros dados de um projeto.

Esta é a do VB5

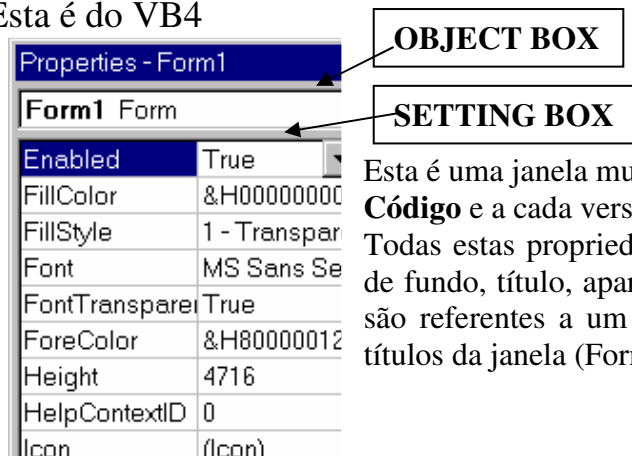


Para poder compartilhar **Código** em todo o projeto deve-se colocar este **Código** em um **Módulo** e declarar a **Procedure** como **Global**.

O ambiente de programação do **Visual BASIC** provém inúmeras ferramentas para auxiliar no desenvolvimento de aplicações gráficas: Formulários, Caixa de Ferramentas (Controles), Paleta de Cores, Propriedades, **Eventos** predefinidos, Janela de **Módulo** interativa e com help on-line, Object Browser, API Viewer, Setup Wizard, etc.

D) A JANELA DE PROPRIEDADES

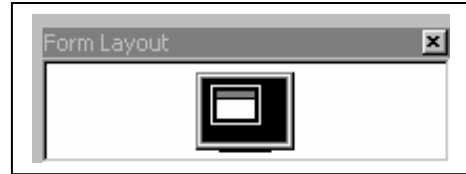
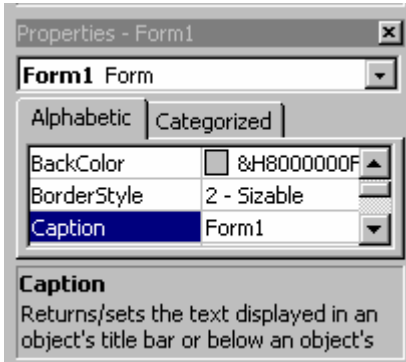
Esta é do VB4



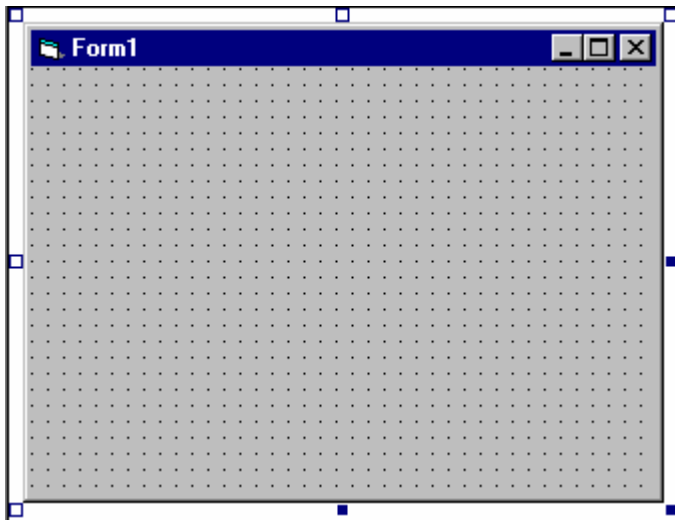
Esta é uma janela muito útil (economiza muitas linhas de **Código** e a cada versão vem mais rica.

Todas estas propriedades indicadas (nome, cor de frente, cor de fundo, título, aparência, caixa de controle, habilitada, etc.) são referentes a um objeto, cujo nome aparece na Barra de títulos da janela (Form1 no caso da janela ao lado).

Esta é a janela de Propriedades do VB5: Form Layout (Visualização do Projeto)



E) JANELA DO FORMULÁRIO



O **Formulário** é o principal objeto do **VB**, é a janela que forma a interface com o usuário. Contêm botões, menus, figuras, etc. Suas **Propriedades** principais são: *BackColor*, *BorderStyle*, *Caption*, *ControlBox*, *Max e MinButton*, *Enabled*, *Height e Width*, *Icon*, *MousePointers*, *Top, StartUp Position*, *Visible e WindowState* (0-Normal, 1-Minimizada e 2-Maximizada).

CRIANDO APLICAÇÕES

* Diferença entre uma Aplicação Tradicional e outra Event-Driven ou seja ativada por **Eventos** ou Orientada a Objetos.

Em uma aplicação tradicional ou programada de forma procedural , a aplicação por si só controla que porções do **Código** serão executadas. A execução começa na primeira linha de **Código** e segue um caminho pré-definido através de toda a aplicação , chamando subrotinas conforme for sendo necessário.

Em uma aplicação controlada por **Eventos** ou event-driven , uma ação do usuário ou do sistema , ativa um procedimento associado a este **Evento**. Assim a ordem através do qual o seu **Código** de programa é executado depende de quais **Eventos** ocorram , que por sua vez estes **Eventos** dependem das ações tomadas pelo usuário. Esta é a essência das Interfaces Gráficas e da Programação Ativada por **Eventos**.

PASSOS PARA A CRIAÇÃO DE UMA APLICAÇÃO:

- A. Criar a interface
- B. Setar as propriedades dos controles
- C. Escrever o **Código**

A. CRIAR A INTERFACE.

O primeiro passo na construção de uma aplicação **Visual BASIC** é desenhar os objetos que irão compor a interface. Para inserir o controle no seu form:

1. Clique no botão do controle dentro da toolbox.
2. Mova a seta do mouse para dentro da área do form , a seta vira uma cruz.
3. Ponha a cruz no ponto dentro do form onde irá ficar o canto superior esquerdo do controle escolhido
4. Arraste a cruz até que o controle fique do tamanho desejado (arrastar significa apertar o botão esquerdo do mouse e mante-lo apertado enquanto o objeto é movido com a mudança de posição do mouse)
5. Solte o botão do mouse e o controle aparecerá no form. Caso você dê um duplo clique num Controle da ToolBox ele será desenhado automaticamente em tamanho padrão e no centro do **Form**.

B. SETAR AS PROPRIEDADES DOS CONTROLES

Propriedades são características que um objeto pode ter. Existem as comuns (p.e.: name) e as particulares a cada objeto.

O próximo passo é setar (colocar valores de inicialização) as propriedades dos objetos que você criou. A janela Properties proporciona uma maneira fácil de inicializar as propriedades para todos os objetos do form . Para abrir a janela de propriedades , escolha o comando Properties Window no menu View ou então clique no botão Properties na Barra de Ferramentas (Toolbar) ou tecle **F4**.

OBJECT BOX - Mostra o nome do objeto selecionado. Clique a seta a direita do box para selecionar o form ou nome de controle a partir da lista de objetos presentes no form atual.

SETTINGS BOX - Permite que você edite a inicialização da propriedade selecionada na lista de propriedades. Algumas inicializações podem ser trocadas clicando-se na seta sublinhada existente a direita do box ; será mostrada então uma lista de opções . Você poderá clicar em um item da lista para selecioná-lo.

LISTA DE PROPRIEDADES - A coluna esquerda mostra todas as propriedades para um objeto selecionado, a coluna da direita mostra a inicialização atual para cada uma das propriedades.

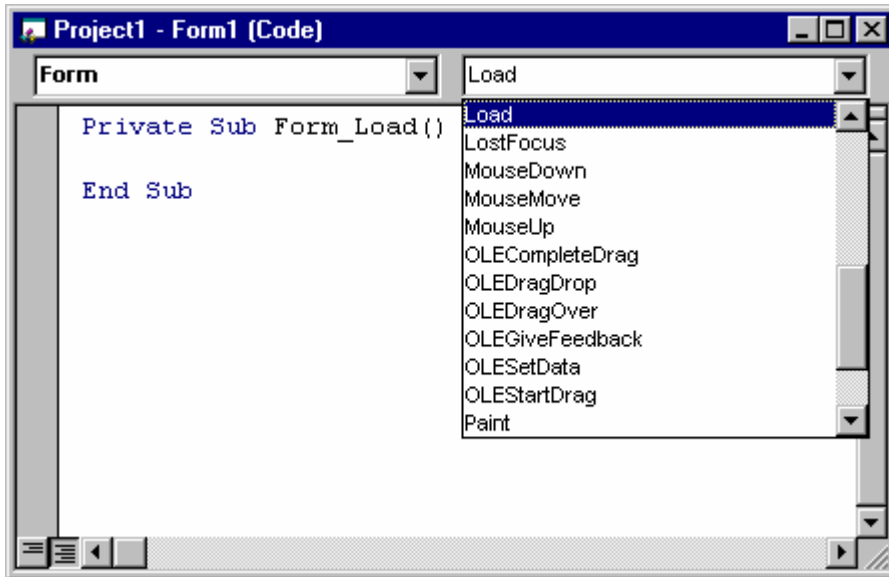
C. ESCREVER O CÓDIGO.

MÓDULOS – são objetos que contêm código de programação (procedures, variáveis, constantes, tipos de dados, etc).

A janela de **Código** é o local onde você escreve o **Código Visual BASIC** para a sua aplicação. **Código** consiste em sentenças da linguagem , constantes e declarações. Usando a janela de **Código** (Code Window) , você pode rapidamente ver e editar qualquer parte do seu **Código** dentro de sua aplicação.

Para abrir a janela de **Código** você deve clicar duas vezes no form ou no controle para o qual você deseja escrever **Código** ou então a partir da janela Project , selecione o nome do form e clique no botão View Code.

JANELA DE CÓDIGO (Code Window)



A janela de **Código** (Code Window) contém os seguintes elementos :

Object Box (Object) - Mostra o nome do objeto selecionado. Clique na seta a direita para mostrar a lista de todos os objetos associados a este form.

Procedures List Box (Proc) - Lista os procedimentos existentes para um objeto. O box mostra o nome do procedimento selecionado - no exemplo acima Click . Clique na seta a direita do box para mostrar todos os procedimentos associados a este objeto.

O **CÓDIGO** (ou programa) em uma aplicação **Visual BASIC** é dividido em pequenos blocos chamados procedures (procedimentos). Um event procedure (procedimento associado a um **Evento**) contém **Código** que é executado quando um **Evento** ocorre (como por exemplo quando o usuário clica em um botão) .

Cada controle tem uma parte do código associada a si. Existem controles que tem apenas propriedades sem a necessidade de código (acontece muito com os Labels), sem qualquer código

CRIANDO UM EVENT PROCEDURE:

1. No Object box , selecione o nome do objeto no form ativo (form que atualmente tem o foco)
2. Na Procedure List box , selecione o nome do **Evento** desejado. No exemplo acima a procedure Click já estava selecionada, já que ela é a procedure default para o botão de comando. Note que uma máscara (template) para a escrita do **Código** para este **Evento** está sendo mostrada na janela de **Código**
3. Digite o **Código** desejado entre as sentenças *Private Sub* e *End Sub* e a procedure se parecerá com o texto abaixo.

```
Private Sub Command1_Click ()
```

```
Text1.text = "Hello World!"
```

```
End Sub
```

4 - CONECTANDO FORMULÁRIOS

A adição de novos formulários na sua aplicação é feito através da opção File na barra de menu , comando New Form ou clicando no Botão **Form**.

Comandos para controle do form :

Comando	Ação
Load form	Carrega o form mas o deixa invisível
Form.Show [modo]	Mostra o form se ele estiver invisível, se ele não estiver carrega então ele primeiro carrega o form e depois o mostra , estilo refere-se a se o form que fez a carga do próximo form , fica parado até a desativação do novo form (modo =0 ou modeless) ou se continua a sua execução sem se importar com o próximo form (modo=1 ou modal)
Unload form	O form é descarregado da memória e a sua execução é encerrada

CONTROLES, MÉTODOS, EVENTOS E PROPRIEDADES

PRINCIPAIS CONTROLES DO VB5



CommandButton (Botão de Comando) – É um dos **Controles** mais utilizados em programas do **Windows**, não apenas em **VB**. Sua função principal é executar comandos quando clicado. O **Evento Click** é o mais importante para ele. As **Propriedades *Caption*, *Name*, *Enabled*** são importantes entre outras. A propriedade ***Enabled*** indica se o botão responderá ao **Evento Click** ou não.



PictureBox (Caixa de Imagem) - Este **Controle** é utilizado para exibir figuras nos **Formulários** e também responde ao **Evento Click**. A propriedade ***Picture*** é a que abre a caixa de diálogo para indicarmos o ***path*** do ícone ou da figura que queremos exibir.



Label (Etiqueta ou Rótulo) – Exibir um texto onde o usuário não possa alterar. Sua propriedade principal é a ***Caption*** (que é o texto que o **Controle** exibirá). Outras **Propriedades** de formatação do texto: ***FontName*, *FontBold*, *Alignment***, etc.



TextBox (Caixa de Texto) – Exibir um texto onde o usuário possa editar. Sua propriedade principal é o ***Text*** (que é o texto que o **Controle** contém). As **Propriedades** de formatação do texto são idênticas ao **Label**.



CheckBox (Caixa de Verificação) – Útil quando necessitamos informar ao sistema que determinadas informações são verdadeiras ou falsas. Em um mesmo **Formulário** vários **CheckBoxes** podem assumir o valor verdadeiro ao mesmo tempo. Algumas **Propriedades**: ***Visible*, *Enabled* e *Caption***.



OptionButton (Botão de Opção ou botão de rádio) – Com este **Controle** somente uma das opções pode assumir o valor verdadeiro em um mesmo conjunto. Para Ter mais de um conjunto verdadeiro temos que utilizar o **Controle Frame**.



ListBox (Caixa de Listagem) – Exibir e selecionar uma lista de itens. As **Propriedades *ListCount* e *ListIndex*** são utilizadas para identificar os itens da lista. Os **Métodos *AddItem* e *RemoveItem*** são utilizados para adicionar e remover itens.



ComboBox (Caixa de Combinação) – Combinação de uma **TextBox** com uma **ListBox**. Pode-se editar um texto na parte superior ou selecionar um item da lista. **Propriedades** idênticas as da **ListBox**.



ScrollBar (Barra de Rolagem) Vertical e Horizontal - Estes **controles** assumem valores máximos ou mínimos de acordo com as **propriedades *Max e Min***, respectivamente. Estes controles podem receber um deslocamento pequeno (**propriedade *SmallChange***), que ocorre quando se clica nas setas e um deslocamento grande (**propriedade *LangeChange***), que ocorre quando se clica na barra. O **controle** recebe o valor da posição atual.



Timer (Temporizador) – Útil para se controlar intervalos de tempo. **Evento** mais importante é o ***Timer*** e a propriedade mais importante é o ***Interval*** (que determina em milisegundos o intervalo de tempo d **controle**).



Shape (Formas Geométricas) – Adiciona figuras geométricas aos **Formulários**. Com a **Propriedade *Shape*** escolhemos o tipo de figura (forma). Outras **Propriedades *BorderColor, BorderStyle e BackColor***.



Line (Linha) – Adicionar linhas geométricas a formulários. **Propriedades *BorderColor, BorderWidth, BorderStyle, X1, X2, Y1 e Y2***.



Image (Imagem) – Exibe ícones ou figuras e também responde ao **Evento *Click***. Diferentemente do **Controle *Picture*** este controle tem a **Propriedade *Stretch*** que ajusta o tamanho da imagem para preencher a moldura.



Data Control – Conectar-se a um **Banco de Dados** para extrair e manipular informações deste BD. **Propriedades** mais importantes ***DatabaseName e RecordSource***. Controles como **TextBox** podem ser vinculados a um **DataControl** através das **Propriedades *DataField e DataSource***.



DBList – Automaticamente preenche uma lista com campos de um **DataControl** e opcionalmente passa o campo selecionado para um segundo **DataControl**.



DBCombo é similar mas permite editar o campo selecionado. **Propriedades** importantes: ***DataField, DataSource, BoundColumn e RowSource***.



OLE – Incorpora e aninha objetos de outros aplicativos em um formulário. As **Propriedades *Class, OLETypeAllowed e SourceDoc*** definem o objeto.

EVENTOS COMUNS

O **Visual BASIC** apresenta alguns **Eventos** de objetos que são usados freqüentemente, e se apresentam na maior parte dos objetos de um form. Abaixo segue uma relação dos principais e suas respectivas funções:

- **Click**: É um **Evento** que executa determinadas ações especificadas pelo programador quando alguém clica sobre alguma coisa.

- **Dblclick**: Executa determinadas ações especificadas pelo programador quando alguém clica duas vezes sobre algum objeto.

- **Resize**: executa algum comando pré estabelecido quando o usuário redimensiona o form.

- **Load**: executa ordens sempre que o programa for carregado na memória.

- **Gotfocus**: quando um objeto, um botão por exemplo, apenas ganha o foco, sem executar as suas funções em outros **Eventos**.

- **Lostfocus**: quando o foco passa para um outro objeto. Quando um objeto perde o foco.

- **KeyPress**: quando é pressionada alguma tecla, geralmente em caixas de texto, são acionados determinados comandos.

- **Mousemove**: sempre que se move o ponteiro do mouse, algo será ativado. Um exemplo prático, são os protetores de tela, que são desativados quando mexemos o mouse.

PROPRIEDADES GERAIS

As principais propriedades dos objetos do **Visual BASIC**, que estão presentes em praticamente todos os objetos do **Visual BASIC**, são as seguintes:

- **Caption** - Nos botões, form, e frames, altera o seu título a ser exibido no form.
- **Enabled** - Presente em todos os objetos, permite que os objetos sejam acessados, caso ela esteja definida como False.
- **Visible** - Torna o objeto invisível caso esta propriedade esteja configurada como False.
- **Font** - presente em objetos como textbox, label, botões command e botões check e option. Permite que seja alterada a fonte de um objeto. Que agora no VB5 configura tudo sobre fonte.
- **Name** - altera o nome do objeto, para fins de referência durante a programação.
- **Forecolor, Backcolor** - define a cor da fonte e do fundo de uma textbox e uma label, por exemplo.
- Propriedades do form: **Height** (altura do form), **Weight** (largura do form), **Maxbutton** (ativa ou desativa a presença do botão Maximizar), **Minbutton** (ativa ou desativa presença do botão Minimizar), **Left** (altera a distância do form a partir da extrema esquerda do vídeo) e **Top** (altera a distância do form a partir do topo do vídeo).

Em geral, todas as propriedades dos objetos presentes no **Visual BASIC** são bastante intuitivas, no que se refere aos seus nomes. Isto pode ser notado claramente nas propriedades Visible, Enabled, Name, entre outras.

Bem, já que sabendo o que são **Classes, Propriedades, Métodos, Eventos e Objetos**, vamos usá-los! Inicie o **VB**. Você certamente já notou que o **VB** cria uma janela automaticamente. Bem, esta janela é um objeto. Você pode ver suas propriedades na janela "Properties" (se esta janela não estiver visível, use o menu View / Properties ou a tecla F4). Na verdade nem todas as propriedades estão listadas.

Note que a janela Properties tem, logo abaixo da barra de título, uma "caixa de seleção" dizendo "Form1 Form". Esta caixa lista todos os objetos colocados na janela ativa. O texto em negrito "Form1" indica o nome do objeto. O texto ao lado, "Form", indica a classe à qual pertence o objeto. Logo abaixo há a lista das propriedades. À esquerda ficam os nomes das propriedades e a direita os seus respectivos valores.

Note que no **VB** a janela é chamada de Form (a partir de agora uma janela que você criar será chamada de Form). Então, quer ver os **Eventos** que um Form possui? Dê-lhe um duplo clique. Abre-se uma janela de edição. Aliás, em janelas como estas que o seu programa será escrito. No topo desta janela há duas "caixas de seleção"; a primeira ("Object:") lista os objetos que estão na janela. A Segunda ("Proc:"), lista todos os **Eventos** do objeto. Dando um duplo clique no form, a janela de edição se abrirá. O **Evento** "Load" do Form estará selecionado. Já haverá 2 linhas incluídas: "Private Sub Form_Load()" e "End Sub". "Private" por default é coisa do VB4 em diante.

Aquela 1^a linha inicia um subprocedimento; a Segunda termina o mesmo subprocedimento. Qualquer coisa que você colocar entre aquelas duas linhas será executado quando aquele form for carregado na memória (porque o **Evento** é o Load-Carrega).

EXEMPLO DE UM PROGRAMA

Vamos fazer um programinha: você clica em um botão e o título da janela passa a ser "O Botão Foi Clicado"; dê um duplo-clique no form e o botão se moverá na tela.

Como exemplo serve! A primeira coisa a fazer é criar um botão. Para um botão de tamanho "normal", dê um duplo-clique na ferramenta "CommandButton" na caixa de ferramentas (toolbox). Deixe o botão deste tamanho, no meio da tela. No **VB** cada objeto tem um nome; quando você cria um controle (um objeto como um botão, uma barra de rolagem, etc.), o **VB** lhe dá um nome "genérico" automaticamente. O nome do objeto é uma propriedade. Vamos trocar o nome do botão para "MeuBotão" e o do form para "MinhaJanela". Clique em uma área vazia do form e vá para a janela das propriedades. Ache a propriedade "Name" e troque-a para "MinhaJanela". Da mesma forma, troque o Name do botão para "MeuBotão".

Vamos agora trocar a legenda do botão para "Clique-me". Ache a propriedade "Caption" do botão e troque para "Clique-me". Essas propriedades que você alterou foram alteradas durante o Design Time, ou seja, enquanto o programa estava sendo criado (e não executado). Você já deve imaginar que para trocar o título da janela deveríamos trocar a sua propriedade "caption". Certo, mas isto só deve acontecer ao clicar o botão. Dê um duplo-clique no botão que você criou (para abrir a janela de edição). O **Código** (ou seja, os comandos) para trocar o caption do form deverá ser colocado no **Evento** Click do botão; assim, estes comandos só serão executados quando o botão for clicado.

Ao criar um controle, a propriedade que tem o foco é a **Caption**, ou seja, logo que criamos um **Label** (p.e.) podemos imediatamente digitar seu **Caption** (mesmo que a janela de propriedades não esteja visível).

Para alterar uma propriedade durante o Run-Time (enquanto o programa é executado), o **VB** usa a seguinte sintaxe:

```
Objeto.Propriedade = NovoValor
```

Então para trocar o caption do form usaríamos o seguinte **Código** (note que o **Evento** é o click):

```
Private Sub MeuBotão_Click ()  
    MinhaJanela.Caption = "O Botão Foi Clicado"  
End Sub
```

Veja bem: como o valor do caption é um texto, deve estar entre aspas. Com isto você já é capaz de se virar com as propriedades e ter uma boa noção sobre os **Eventos**. Agora vamos aos **Métodos**:

Para mover o botão vamos usar o **Método** Move do botão. Como a movimentação acontecerá ao duplo-clicar o form, o **Código** vai no **Evento** DblClick do Form. A seguinte sintaxe é usada no **VB** para **Métodos**:

```
Objeto.Método Atributos    (Nem sempre há atributos)
```

Para mover o botão, usaremos o seguinte **Código** (lembre-se **sempre** de colocar o **Código** em seu devido **Evento**):

```
Private Sub MinhaJanela_DblClick ()  
    MeuBotão.Move 100, 100  
End Sub
```

Os atributos 100, 100 vão colocar o botão perto do canto superior esquerdo do form. Execute o programa (tecla F5 ou o botãozinho do tipo play na barra de ferramentas). Clique o botão e depois dê um duplo clique no form. Se tudo funcionou direito, parabéns.

O **VB** tem dezenas de **Classes** com centenas, milhares de propriedades, **Métodos** e **Eventos**. É muito difícil falar sobre todos em pouco espaço. Mas todos eles estão muito bem explicados no help on line do **VB**. Procure Methods, Properties e Events e brinque bastante com eles. Brinque também com outros controles, não só botões. Comece a usar **Labels** ("etiquetas" que mostram um texto), PictureBoxes (para mostrar desenhos) e outros.

VARIÁVEIS

O QUE SÃO VARIÁVEIS?

Bem, variáveis são coisas que, em programação, servem para armazenar dados temporariamente na memória. Digamos que você queira, fazer uma "calculadora"; os números serão armazenados em variáveis.

Como eu crio uma variável?

Normalmente não é preciso criar uma variável; basta usá-la que ela é criada automaticamente. Uma variável é identificada por um nome. Este nome deve obedecer às seguintes regras:

- Não pode haver repetição de nomes;
- Deve começar com uma letra;
- Não pode conter espaços, pontos, vírgulas e outros caracteres do tipo: !, \$, %.
- Deve ter, no máximo, 255 caracteres.

Muitíssimas vezes você usará variáveis para guardar valores de propriedades. E depois poderá atribuir o valor da variável a uma outra propriedade. A sintaxe é esta:

Variável = Objeto.Propriedade => Coloca na variável o valor da propriedade

Objeto.Propriedade = Variável => Coloca na propriedade o valor da variável

EXEMPLO DE MÁQUINA DE SOMAR

Para dar um exemplo, vamos fazer uma "máquina de somar": em um form coloque duas "text boxes", uma "label" e um botão. Nas caixas de texto serão colocados os números. Um clique no botão e a soma aparecerá na "label". Vamos usar os nomes padrão dos controles (já que o projeto é pequeno). Troque a caption do botão para "Soma" e apague o texto das text boxes.

Todo o **Código** estará no **Evento** Click do botão:

```
Private Sub Command1_Click ()  
    Numero1 = Val(Text1.Text)  
    Numero2 = Val(Text2.Text)  
    Soma = Numero1 + Numero2  
    Label1.Caption = Str(Soma)  
End Sub
```

Acho que até deu para entender. Mas o que são aqueles "Val (...)" e "Str (...)"?

Val e Str são funções. Funções são comandos que retornam algum dado ou valor. Seguidamente uma função requer atributos (que são colocados entre parênteses). Eis o porquê de usar essas funções: o **VB** diferencia números de textos. Mas "12" pode ser um texto. Normalmente, quando algo está entre aspas, é considerado texto. A propriedade "Text" (assim como a "Caption") sempre contém um texto. Uma soma entre os textos "1" e "1" resultaria em "11"! Por isso, usamos a função Val, que pega um texto e transforma-o em um valor numérico. A função Str, que aparece depois faz o contrário, pega um número e transforma em uma string (texto).

Note que você também pode colocar um valor numa variável diretamente:

```
Valor = 10   ou   Msg = "Isto aqui é uma string."
```

DECLARANDO VARIÁVEIS

É possível (e importante) "declarar" variáveis, ou seja, dizer ao programa que você irá usá-las antes que elas sejam "automaticamente criadas".

Mas para que "declarar" se o processo é automático?
Em algumas linguagens é obrigatório (C e Java).

Principalmente por dois motivos: velocidade e economia de memória. Existem vários tipos de dados no **VB** (veja tabela na próxima página). Cada um destes tipos tem uma limitação em termos de valores que pode adquirir, mas também se diferenciam pela quantidade de memória que ocupam.

CONSTANTES:

CORES:

vbBlack, vbRed, vbGreen, vbBlue, etc.

TIPOS DE DADOS DO VB5

<u>Data type</u>	<u>Storage size</u>	<u>Range</u>
Byte	1 byte	0 to 255
Boolean	2 bytes	True or False
Integer	2 bytes	-32,768 to 32,767
Long		
(long integer)	4 bytes	-2,147,483,648 to 2,147,483,647
Single		
(single-precision floating-point)	4 bytes	-3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values
Double		
(double-precision floating-point)	8 bytes	-1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values
Currency		
(scaled integer)	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	14 bytes	+/-79,228,162,514,264,337,593,543,950,335 c/ no decimal point; +/-7.9228162514264337593543950335 with 28 places to the right of the decimal; smallest non-zero number is +/-0.00000000000000000000000000000001
Date	8 bytes	January 1, 100 to December 31, 9999
Object	4 bytes	Any Object reference
String		
(variable-length)	10 bytes + string length	0 to approximately 2 billion
String		
(fixed-length)	Length of string	1 to approximately 65,400
Variant		
(with numbers)	16 bytes	Any numeric value up to the range of a Double
Variant		
(with characters)	22 bytes + string length	Same range as for variable-length
String		
User-defined		

(using Type) Number required by elements The range of each element is the same as the range of its data type.

Note Arrays of any data type require 20 bytes of memory plus 4 bytes for each array dimension plus the number of bytes occupied by the data itself. The memory occupied by the data can be calculated by multiplying the number of data elements by the size of each element. For example, the data in a single-dimension array consisting of 4 Integer data elements of 2 bytes each occupies 8 bytes. The 8 bytes required for the data plus the 24 bytes of overhead brings the total memory requirement for the array to 32 bytes.

A Variant containing an array requires 12 bytes more than the array alone.

Quando você não declara uma variável, ela é considerada do tipo Variant. Note que este tipo ocupa pelo menos 16 bytes de memória, bem mais que os outros. A sintaxe para declarar uma variável pode mudar um pouco, dependendo do "escopo" que você deseja. ("escopo" é a disponibilidade da variável em cada um dos "níveis" do programa - veja o próximo capítulo). Generalizando, a sintaxe é esta:

ComandoDeDeclaração NomeDaVariável as TipoDeDado

De um modo geral, o "ComandoDeDeclaração" será o comando "Dim". Por exemplo, se quiser criar uma variável do tipo Integer chamada MinhaVariavel, usaríamos:

Dim MinhaVariavel as Integer

Mas, será que faz muita diferença gastar 2, 4, 8 ou 16 bytes da memória? Hoje os computadores têm **muita** memória.

Tudo bem, em programas pequenos isto não será uma diferença crucial, mas em um programa maior, com centenas de variáveis, isto pode fazer a diferença. Além disto, devemos lembrar que o Windows pode trabalhar com vários programas abertos ao mesmo tempo; e se seu programa for "gordo" poderá atrapalhar a performance dos outros.

Além disto, quanto menos memória ocupa um certo tipo de dado, mais rapidamente ele é processado. Por isso, procure sempre usar o "menor" tipo de dado possível. Mas lembre-se: se você ultrapassar o limite da variável acontecerá um erro (tente criar uma variável do tipo Byte e atribua a ela um valor maior que 255).

ESTRUTURAS E LOOPS

Loops e estruturas permitem que um **Código VB** (ou de qualquer outra linguagem) seja repetido várias vezes. Permitem também analisar dados e tomar decisões diferentes dependendo do resultado. Na verdade loops são criados por estruturas (as estruturas de loop).

Afinal, como é que eu faço para utilizar uma estrutura?

Não é difícil; as estruturas são criadas através de alguns grupos de comandos. A seguir listo e comento as estruturas disponíveis no **VB**.

If... Then... Else...

Isto, em bom português, quer dizer Se... Então... Senão... Eis um exemplo de como utilizá-la:

```
If MinhaVariavel <= 4 Then
    Beep
ElseIf MinhaVariavel = 5 Then
    Beep
    Beep
Else
    Beep
    Beep
    Beep
End If
```

Neste exemplo, se a MinhaVariavel for menor ou igual (<=) a 4, então um beep será emitido. O "bloco" ElseIf (em verde) é opcional. Ele serve para dizer que se MinhaVariavel for igual a 5, algo será feito (no caso, dois beeps serão emitidos). Podem ser incluídos quantos ElseIf forem desejados, mas talvez seja melhor usar a estrutura Select Case (veja adiante) do que vários ElseIf. O bloco em azul (Else), também opcional, será executado somente se nenhum dos blocos anteriores (If e ElseIf) for executados. End If finaliza a estrutura.

Select Case

Vamos direto ao exemplo:

```
Select Case MinhaVariavel
  Case 0
    Beep
  Case Is <= 5
    Beep
    Beep
  Case Else
    Beep
    Beep
    Beep
End Select
```

Esta estrutura, inicialmente, "seleciona" o valor de MinhaVariavel. Depois verifica vários casos (quantos forem desejados). Caso MinhaVariavel seja igual a 0, um beep será emitido. Caso seja menor ou igual a 5, teremos dois beeps. O bloco Case Else, que é opcional e está em azul, será executado somente caso nenhum dos blocos case anteriores tenha sido executado. End Select finaliza a estrutura.

Do... Loop

Esta estrutura executa um certo grupo de comandos até (Until) ou enquanto (While) que uma certa condição é verdadeira. Por exemplo:

```
Randomize Timer
Contador = 0
Do While MinhaVariavel <> 5           'Enquanto MinhaVariavel <>5
  Contador = Contador + 1
  Aleatório = Int (Rnd * 10)         'Sorteia número inteiro entre 0 e 9
  If Aleatório = 5 Then
    MinhaVariavel = 2
  End If
Loop
MsgBox Str(Contador)
```

Aqui há algumas observações interessantes: a primeira é que é possível colocar estruturas dentro de outras. Aqui, tivemos uma If... Then... dentro de uma Do... Loop. Isto é chamado de estruturas aninhadas.

Quando o computador chega na linha `Do While...` ele verifica se a condição (`MinhaVariavel <> 5`) é verdadeira. Se for ele seguirá até encontrar o comando `Loop`. Aí ele volta à linha `Do While...` e o ciclo se repete enquanto a condição for verdadeira (quando for falsa, a execução segue na linha logo abaixo de `Loop`). Se tivéssemos usado `Do Until` ao invés de `Do While`, o loop seria executado até que a condição se tornasse verdadeira.

Continuando a análise: as linhas em azul criam um número aleatório. Note que se este número for 5, `MinhaVariavel` receberá o valor de 2 (que é diferente de 5 - satisfaz a condição do loop; na próxima vez que a execução chegar no `Do While`, o loop será encerrado). A variável `Contador`, inicializada em 0, é acrescida de 1 a cada execução do loop. No final, uma caixa de mensagem (`MsgBox`) é exibida mostrando quantas vezes o loop precisou ser executado.

Lembre-se que para testar este exemplo é preciso colocar o **Código** no **Evento** de algum objeto (como no `Click` de um botão).

Mais uma coisinha: dentro de uma estrutura `Do... Loop` pode ser colocado o comando `Exit Do`. Quando um `Exit Do` for encontrado, a execução automaticamente vai para a linha logo a seguir do `Loop`, independentemente da condição estar falsa ou verdadeira.

`For... Next`

A estrutura `For... Next` é ideal para realizar tarefas repetitivas. Que tal somar todos os números pares de 50 a 100 (`50+52+54+...+100`)?

```
For Cont = 50 To 100 Step 2
    Soma = Soma + Cont
Next Cont
MsgBox Str(Soma)
```

Não é difícil entender este programinha: O laço (laço é o mesmo que loop) iniciado pelo `For` utiliza uma variável (que eu chamei de `Cont`) que inicia em 50. Quando a execução chega no `Next`, volta à linha `For...` Ali, `Cont` será incrementado de 2 ("Step x" identifica de quanto será o incremento. A ausência do `Step x` é equivalente a `Step 1`). Quando `Cont` passar de 100 e a execução chegar à linha `For Cont...`, a execução continua a partir da linha logo após o `Next Cont`. É possível sair do loop antes que a variável contador chegue ao valor planejado. Basta que o computador veja o comando `Exit For`.

Bem, basicamente é só isso.

Na verdade existem outras estrutura.

OBSERVAÇÃO

Conhecendo o uso básico de objetos, o uso básico de variáveis as estruturas do **VB** e os níveis do programa, você já estará pronto para fazer programas bem bonitinhos. Como já disse antes, não pretendo analisar cada detalhe, cada controle do **VB**.

Tentar, errar, corrigir, errar de novo, corrigir de novo. Fazer programinhas simples, depois alguns mais complexos. Abrir exemplos e alterar. É assim que se chega lá.

SUBPROCEDIMENTOS E FUNÇÕES

Mas voltando aos subprocedimentos: você já sabe que quando o usuário interage com o seu programa (clikando em um botão, por exemplo) é gerado um **Evento**. Para responder a este **Evento** é executado um subprocedimento.

EXEMPLO

Então um subprocedimento é tudo aquilo que colocamos entre as linhas "Private Sub..." e "End Sub" de um "respondedor de **Eventos**"?

Bom, aquilo é um subprocedimento sim. Mas subprocedimentos não é só isto. Na verdade existem dois tipos de subprocedimentos: Subs e Functions.

Vamos a um exemplo (outro daqueles inúteis). Em um Form (Form1), coloque uma ScrollBar (horizontal ou vertical, tanto faz). Ajuste seu Name para "BarraRol", seu Max para 15 e seu LargeChange para 5.

Abrindo um pequeno parênteses: as propriedades Max e Min de uma ScrollBar determinam o valor máximo e mínimo, respectivamente, que a ScrollBar pode possuir. A propriedade SmallChange determina de quanto será o incremento (ou decremento) quando uma daquelas setinhas nas pontas da ScrollBar é clicada. LargeChange determina de quanto será o incremento (ou decremento) quando a ScrollBar for clicada fora das setinhas ou daquela "caixinha" que pode ser arrastada. Coloque no Form1 também uma Label (LabelValor) e ajuste seu Caption para 0. Agora, ao **Evento** Change da BarraRol, acrescente o seguinte **Código**:

```
Private Sub BarraRol_Change()  
    LabelValor.Caption = Str(BarraRol.Value)  
End Sub
```

Neste exemplo criamos uma Sub (note que todo o **Código** está delimitado pelas linhas Private Sub... e End Sub).

E aquele "Private" tem algo a ver com escopo?

Tem sim. Agora inclua um botão (Zerador será seu name). Quando ele for clicado a ScrollBar irá "zerar":

```
Private Sub Zerador_Click()  
    BarraRol.Value = 0  
End Sub
```

Certo. Agora, vamos incrementar. Coloque um Shape (é aquele botão da caixa de ferramentas com o desenho de um círculo e dois quadradinhos coloridos) no Form1. Nomeie este Shape como Figura e seu FillColor para azul (&H00FF0000&) e seu FillStyle para 0 (Solid). O Shape fica azul.

Digamos que a cor deste Shape mude a cada vez que o botão for clicado e cada vez que a ScrollBar tenha seu valor modificado. Como faríamos isto?

Daria para colocar o **Código** necessário nos subprocedimentos BarraRol_Change e Zerador_Click.

Daria, mas estaríamos escrevendo o mesmo **Código** duas vezes. Isto não é muito interessante. A melhor solução é criar um subprocedimento (uma Sub) que não esta diretamente associado a um **Evento**. Para fazer isto vá até a janela de edição do Form1 (dando um duplo clique em qualquer controle ou no próprio Form1). Na caixa de seleção "Object" escolha General. Aí, entre com o seguinte **Código**:

```
Private Sub MudaCor()  
    Figura.FillColor = Figura.FillColor Xor &HFFFFFFF  
End Sub
```

Note que ao terminar de escrever a primeira linha o **VB** automaticamente inclui o "End Sub". O que este subprocedimento faz é uma operação lógica XOR entre a cor atual do Shape e o número &HFFFFFFF (255 em hexadecimal). Este novo valor é atribuído à cor do Shape. Devido às características do operador XOR, a cor ficará variando entre o azul original e o amarelo.

Agora só falta fazer com que esta Sub seja "chamada" nos momentos adequados. Isto é feito alterando as Subs BarraRol_Change e Zerador_Click. Elas ficarão assim:

```
Private Sub BarraRol_Change()  
    LabelValor.Caption = Str(BarraRol.Value)  
    MudaCor  
End Sub
```

```
Private Sub Zerador_Click()  
    BarraRol.Value = 0  
    MudaCor  
End Sub
```

Na verdade só incluímos a chamada à Sub MudaCor que acabamos de criar.

Outra coisinha: uma Sub pode ser escrita tanto na seção General de um Form como em um Module.

E aquela história de escopo?

Ah, sim. Subprocedimentos também têm escopo. Um subprocedimento (seja Sub ou Function), quando declarado como Private pode ser "chamado" apenas por subprocedimentos que estejam no mesmo Form ou Module que ele. Para permitir que seu subprocedimento seja chamável de qualquer ponto do programa declare-o como Public (é idêntico ao que fazemos com variáveis).

APROFUNDANDO

Talvez você esteja pensando o que faz aquele par de parênteses vazios ao lado de cada nome de subprocedimento.

Dentro daqueles parênteses colocamos algum parâmetro que será utilizado pelo subprocedimento. Vamos tornar nosso programa-exemplo ainda mais fantástico: Quando a Sub MudaCor for chamada pela ScrollBar, a espessura da borda do Shape ficará com uma espessura igual ao seu Value. Quando for chamada pelo Botão, a borda ficará maior ainda (Uau!).

A primeira coisa a fazer é rescrever a Sub MudaCor:

```
Private Sub MudaCor(MeuValor As Byte)
    Dim c As Byte
    Figura.FillColor = Figura.FillColor Xor &HFFFFFF
    Figura.BorderWidth = MeuValor
End Sub
```

Agora sim! Temos alguma coisa dentro dos parênteses!

Aquilo que está entre os parênteses significa o seguinte: quando esta Sub for chamada, deverá ser passado um valor (no caso um número, um Byte). Este valor estará disponível para a Sub na forma de uma variável (no exemplo, "MeuValor"). Não é difícil compreender isto analisando o exemplo. A estes valores "passados" a um subprocedimento damos o nome de argumento.

O próximo passo é alterar os outros subprocedimentos (os associados a **Eventos**) para que eles forneçam o argumento necessário:

```
Private Sub BarraRol_Change()
    LabelValor.Caption = Str(BarraRol.Value)
    MudaCor BarraRol.Value
End Sub
```

```
Private Sub Zerador_Click()
    BarraRol.Value = 1
    MudaCor 30
End Sub
```

Na primeira Sub é passado como argumento a propriedade Value da BarraRol. Na segunda é passado um número (30) diretamente.

Ainda é preciso ajustar o Min da BarraRol para 1. Isto evitará que o programa tente dar à Border do Shape um valor igual a 0 (o que geraria um erro).

Ainda é interessante dizer que podemos criar subprocedimentos que admitem mais de um argumento.

...FUNCTIONS

Functions são o segundo tipo de subprocedimento que podemos criar com o **VB**. Elas se diferem das Subs porque elas retornam um valor.

Como assim?

Vamos logo a um exemplo: uma função-dado, ou seja uma função que retorna um número inteiro entre 1 e 6:

```
Private Function Dado() As Byte
    Dado = Int(6 * Rnd + 1)
End Function
```

Esta é uma função simples, com apenas uma linha de **Código**. O mais importante é observar como fazemos para que a função retorne um valor: a função retorna um valor que está contido em uma variável cujo nome é igual ao nome da Function.

Usar esta função é simples: Por exemplo:

MinhaVariavel = Dado + Dado, coloca em MinhaVariavel o que seria o valor do lançamento de dois dados.

Com Functions também é possível utilizar argumentos. Por exemplo, a seguinte função retorna a soma de três números:

```
Private Function SomaTres(Num1 as Integer, Num2 as Integer, Num3 as
integer) As Integer
    SomaTres = Num1 + Num2 + Num3
End Function
```

E para usar esta função:

MinhaVariavel = SomaTres (300, 20, 12) , colocaria 332 em MinhaVariavel.

CONCLUINDO

Saber quando e como utilizar Subs e Functions adequadamente é uma coisa que só vem com o tempo, com a prática. Aliás, isto não acontece só com Subs e Functions...

Uma ultima observação é que subprocedimentos são comumente chamados de Procedures. Se algum dia você vir algum texto falando em uma Procedure Sub ou Procedure Function, já sabe do que se trata.

CRIANDO BASE DE DADOS VIA DATA MANAGER

- Abrir o Visual Basic 5.0
- Add-Ins
- Visual Data Manager...
- Se for a primeira vez que Acessa o Data Manager responda Não para a Pergunta do VB (Add SYSTEM.MD? to INI file?)
- É então aberta a janela do VisData
- File, New, Microsoft Access, Versão 7.0
- Digitar o nome do Banco de Dados a ser criado e Save
- Na janela DataBase Window clicar com o botão direito e New Table
- No campo Table Name digitar o nome da Tabela;
- Clicar em Add Field para abrir a janela de adição de Campos;
- Onde se deve entrar com o nome, tipo, tamanho e outras característica do campo e clicar em OK
- Inserir características de outros campos e clicar em OK, até completar os campos
- Clicar em Close para fechar a janela de criação dos campos
- Adicionar índice se necessário
- Clicar em Build the Table para construir a Tabela
- Aparece então o nome da tabela na janela DataBase Window
- Caso desejemos inserir dados na tabela devemos dar um duplo clique no nome da mesma na janela DataBase Window
- Será aberta a janela Dynaset para que sejam inseridos os dados, clicando em Add e digitando
- Fechar então o Visual Data manager e o Banco de Dados está criado, podendo ser utilizado em um projeto do VB através do Data Control, Caixas de Texto e outros controles.