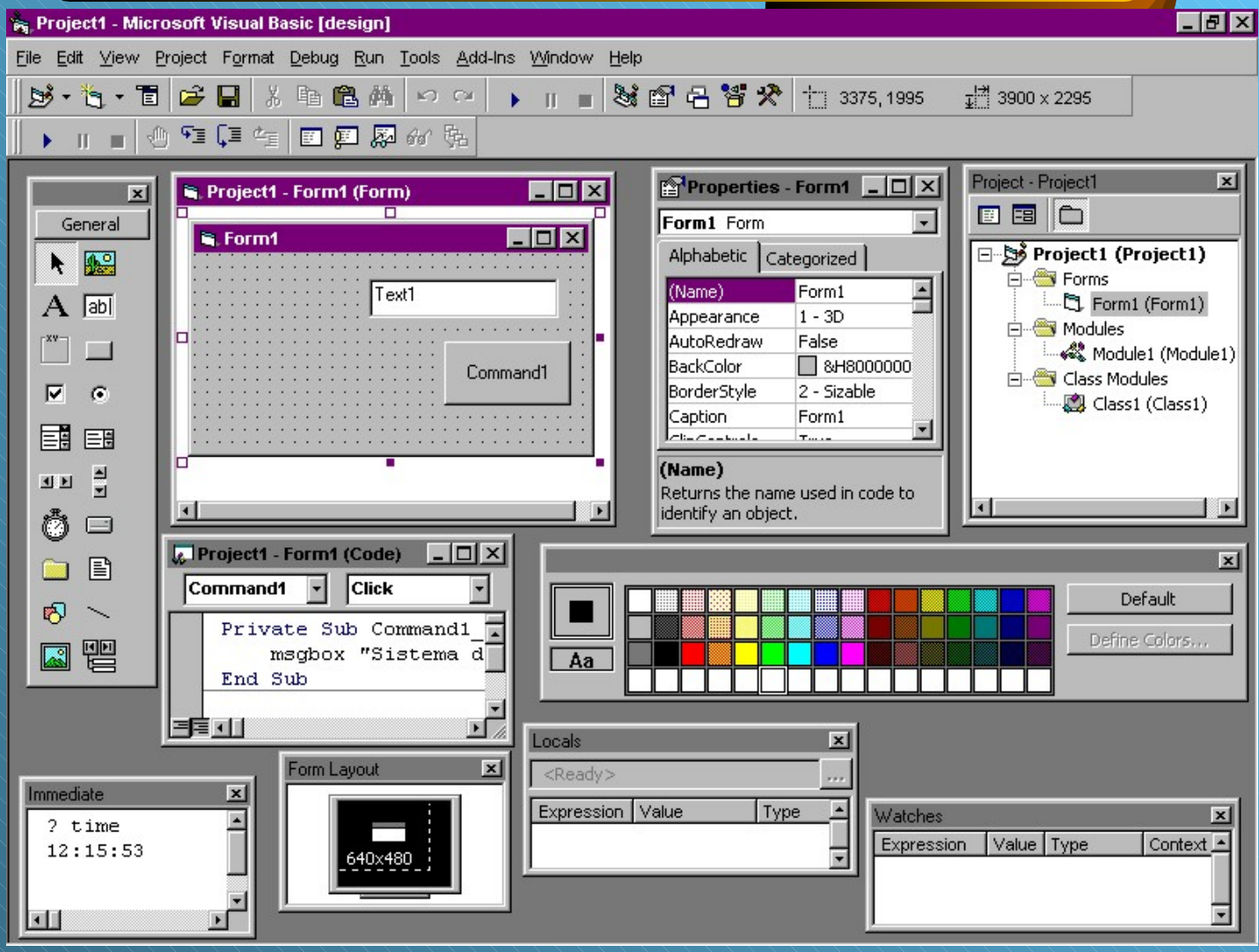
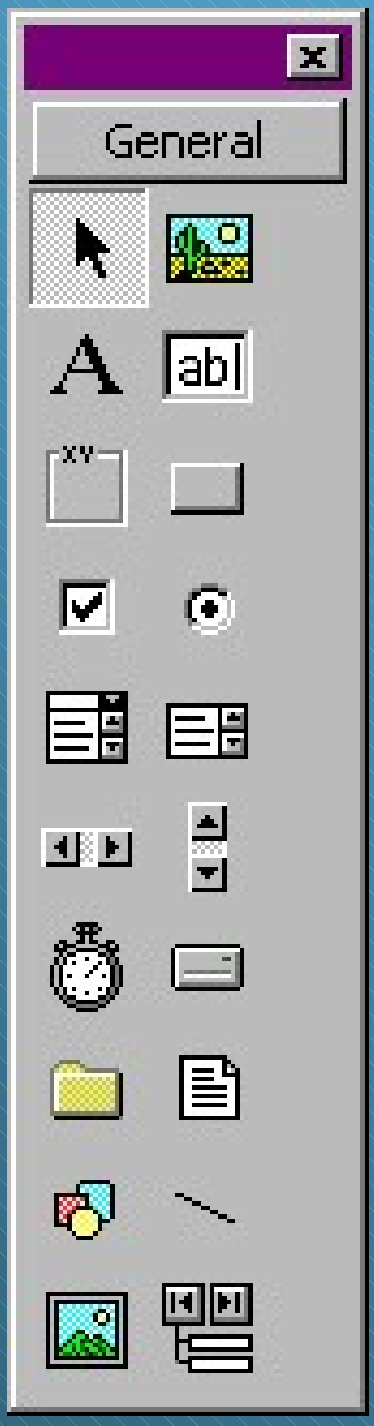
Ambiente Visual Basic



- ToolBox (Ferramentas)
- Properties Window
- Code Window
- Immediate Window
- Local Window
- Forms
- Project Windows
- Color Palette
- Form Layout
- Watches Window

CAIXA DE FERRAMENTAS



Tab(Geral)

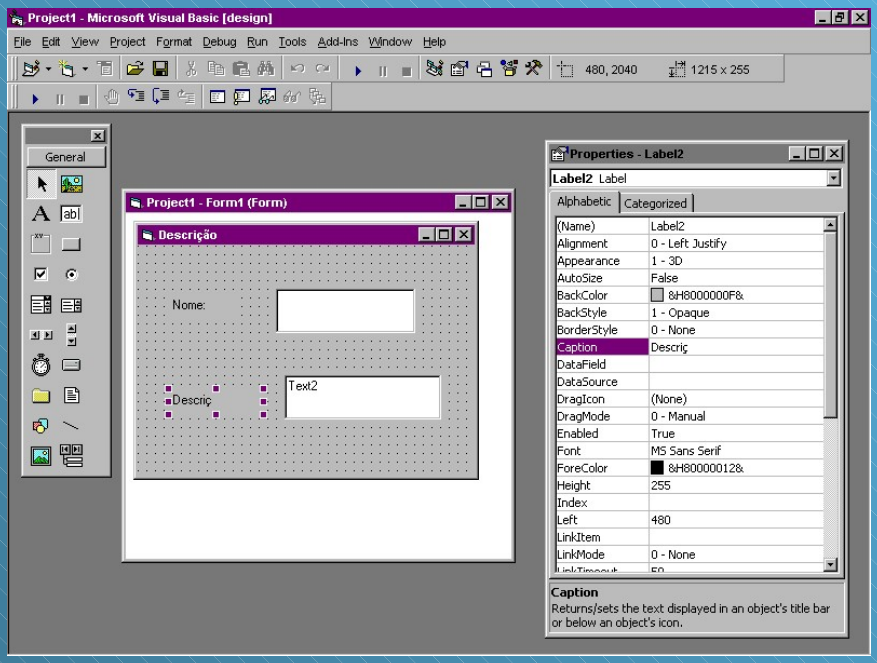
Ponteiro	PictureBox
Label	TextBox
Frame	CommandButton
CheckBox	OptionButton
ComboBox	ListBox
HScrollBar	VScrollBar
Timer	DriveListBox
DirListBox	FileListBox
Shape	Line
Image	Data

PROPRIEDADES

É um atributo ou valor de um objeto

Para atribuir:

- Em tempo de design na janela de propriedades:



- Em tempo de execução por meio de código:

  Label2.caption = “Descrição”

  txtTotal.text = txtSubTotal.text * 1.2

Para verificar valor em tempo de execução:

Variável = Objeto.Propriedade

Ex: strNomeUsuario = txtNome.Text

MÉTODOS

Um Método causa um objeto a realizar uma ação ou um procedimento

Em geral os métodos utilizam a seguinte sintaxe:

Objeto.Metodo(arg1, arg2,...)

Exemplos:

Método Setfocus -> Não possui argumentos

txtNome.Setfocus

Método Move -> Possui os argumentos: esquerda, topo, largura, altura

FrmAbertura.Move 0,0

Método Show -> Possui um argumento: tipo

frmAbre.Show 0

frmAbre.Show

frmAbre.Show VbModeless

EVENTOS

Um Evento é uma ação reconhecida por um form ou controle.

Eventos podem ser invocados:

- De um resultado de uma ação de um usuário
- De um código de programa
- Pelo sistema

Em Visual Basic podemos dizer que qualquer ação possível de se realizar está associado a um evento.

Alguns exemplos de eventos em Visual Basic:

Activate	DragOver	Load
Change	GotFocus	LostFocus
Click	KeyDown	MouseDown
DblClick	KeyPress	MouseMove
DragDrop	KeyUp	MouseUp

Nota: Cada objeto em Visual Basic possui sua própria lista de eventos que ele reconhece, ou seja, nem todos os objetos reconhecem todos os eventos.

FORMS

Form é a principal forma de interação com o usuário. Usuários interagem com os controles inseridos num form para obter resultados.

Os eventos, propriedades e métodos mais utilizados:

Propriedades	**Métodos**	**Eventos**
ActiveControl	Hide	Activate
ActiveForm	Move	Click
BackColor	Print	DblClick
BorderStyle	PrintForm	Deactivate
Caption	Refresh	DragDrop
ControlBox	SetFocus	DragOver
Enabled	Show	GotFocus
Icon		Load
Left		LostFocus
MaxButton		MoseDown
MinButton		MouseMove
Name		MouseUp
Top		Unload
WindowState		

ACESSO AOS CONTROLES

Sempre providencie acesso aos controles nos forms por meio do teclado

O controle que está com o foco é o controle que pode aceitar dados por meio do teclado vindo do usuário.

Tab Order (Ordem de Tabulação) é a ordem de foco dos controles quando o usuário pressiona a tecla TAB ou SHIFT + TAB.

Para setar o Tab Order (a ordem de tabulação)

Por meio da propriedade TabIndex do controle.

O Usuário pode **mover o foco** pelos controles pressionando ALT + a chave de acesso:

Digite & - e comercial (ampersand) antes da letra da propriedade Caption do Controle.

Para setar a tecla de acesso de um controle

É só colocar & no caption

Ex: cmdSair.caption = Sai&r

Irá aparecer Sair no botão

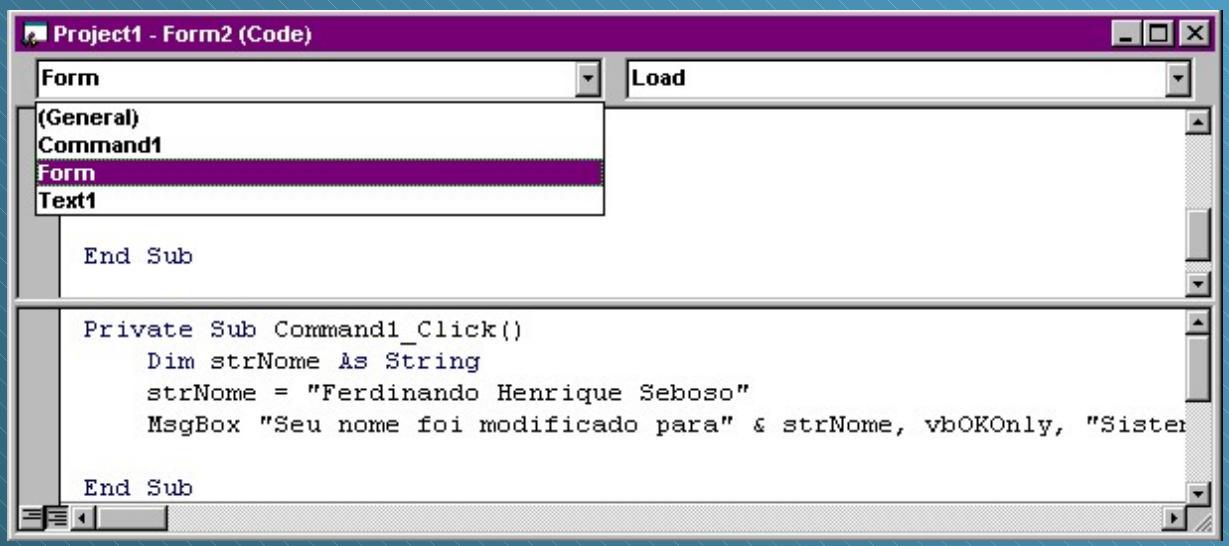
JANELA DE CÓDIGO

A Janela de Código é utilizada para escrever, mostrar e editar códigos Visual Basic. A Janela de código inclue:

Object Box -> Mostra o nome do objeto selecionado. Clicando a seta deste ComboBox será mostrado todos os objetos associados no form corrente.

Procedure Box -> Mostra todos os Eventos reconhecidos pelo Visual Basic para o form ou controle seleconado. Quando é selecionado um evento, a procedure (procedimento) associada para este evento é mostrado na janela de código

Spit Bar -> É localizado na parte abaixo da barra de título no topo da barra vertical de scroll.



Convenção
Edição de Código

Objetivo: Deixar o código de fácil visualização com a utilização de indentação, caracter de continuação de linha, comentários.

Código escrito correto, pois funciona desta forma:

```
Private Sub MudaSinal()
If imgVerde.Visible = True Then
imgVerde.Visible = False
imgAmarelo.Visible = True
ElseIf imgAmarelo.Visible = True Then
imgAmarelo.Visible = False
imgVermelho.Visilbe = True
Else
imgVermelho.Visible = False
imgVerde.Visible = True
End If
MsgBox “Foi modificado o sinal do Sistema”, 16, “Sinal”
End Sub
```

Convenção
Edição de Código

O mesmo código, mais fácil de visualizar e entender:

```
Private Sub MudaSinal()
    ‘se estiver verde fica amarelo
    ‘se estiver amarelo fica vermelho
    ‘se estiver vermelho fica verde

    If imgVerde.Visible = True Then
        imgVerde.Visible = False
        imgAmarelo.Visible = True
    ElseIf imgAmarelo.Visible = True Then
        imgAmarelo.Visible = False
        imgVermelho.Visilbe = True
    Else
        imgVermelho.Visible = False
        imgVerde.Visible = True
    End If
    MsgBox “Foi modificado o sinal do Sistema”, _
            16, _
            “Sinal”
End Sub
```

Escondendo um Form

Enquanto seu programa é executado, talvez seja necessário remover um form da tela para mostrar outro. Há duas maneiras para fazer isto: o método Hide e o procedimento Unload.

Método Hide (esconder)

O método Hide esconde o form da tela mas não o "descarrega" da memória, ou seja, quando um form está escondido ele não é visível para o usuário, mas todos os valores de suas váriaveis, objetos, etc mantêm seus valores.

Sintaxe: frmUsuario.Hide

O procedimento Unload

Ao contrário do Hide, onde o form está na memória, o procedimento Unload retira o form da memória.Se é o único ou o último form a aplicação é encerrada.

Sintaxe: Unload frmUsuario

Dica: Se o Sistema utiliza o mesmo form várias vezes, é mais rápido esconder (frm.Hide) e mostrar (frm.Show) o form do que ler (Load frm) e descarrega-lo (Unload

Startup Form

Setando o Startup Form ou Procedure (o form ou procedure de abertura do Sistema)

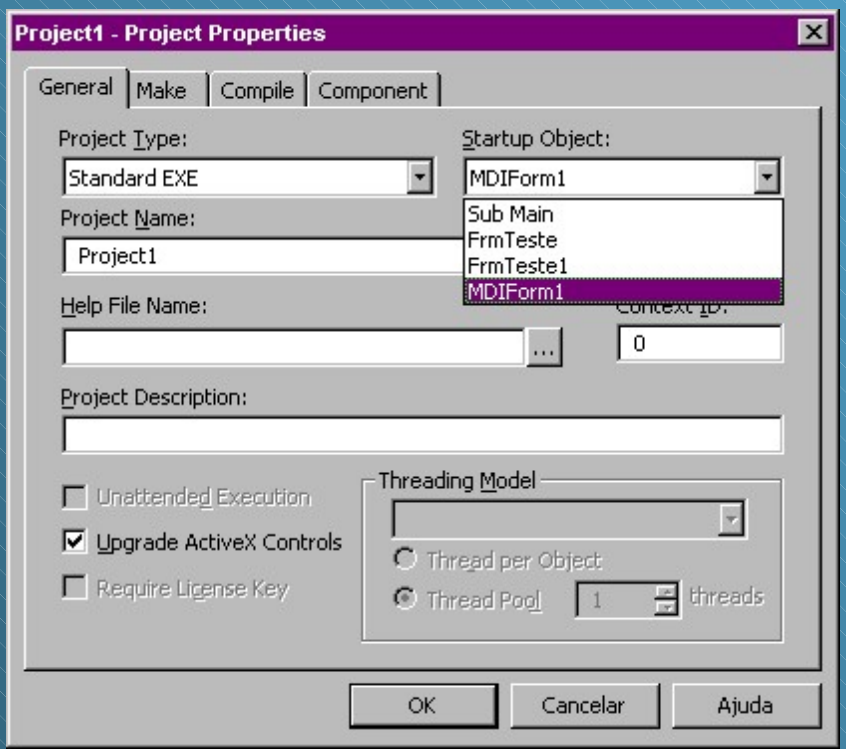
Por default, o VB irá ler o primeiro form criado no seu projeto quando a aplicação é executada. No entanto, você pode especificar qualquer form que você desejar que seja o primeiro.

Para especificar:

1 - Do menu Project, escolha “Nome_Projeto” Properties

2 - Selecione o Tab General

3 - Selecione o form/procedure na Lista de Startup Object



Finalizando a Aplicação

Utilizando o Procedimento End ou Fechando o último form

Ex:

```
Sub Fechar_Click()
    End
End Sub
```

Dica: O usuário pode tentar fechar o último form utilizando o botão close na barra de título (padrão windows), isto irá disparar o evento Unload no form.

O Evento Unload possui um argumento que permite que descartemos que o form seja fechado. Este argumento é o **Cancel.**

Ex:

```
Sub Form_Unload (Cancel as Integer)
If MsgBox("finalizar o programa ?", _
                VbOKCancel)= vbCancel  Then
          Cancel = True  'O Form não será removido
End If
```

Variáveis e Constantes (Variables and Constants)

Variável é um local de armazenamento com um nome, que contem dado que pode ser modificado durante a execução de um programa.

Constante é um local de armazenamento com um nome, que contem dado que permanece igual durante a execução de um programa.

Podem conter diferentes tipos de dados, como texto, números, objetos ou valores boleanos (verdadeiro ou falso).

O local e forma de criação define o escopo, ou seja, aonde a variável pode ser visualizada dentro do programa.

TIPOS DE DADOS

Tipo	Tamanho	Faixa
Byte	1 byte	0 a 255
Boolean	2 bytes	True ou False
Integer	2 bytes	-32,768 a 32,767
Long (Long integer)	4 bytes	-2,147,483,648 a 2,147,483,647
Single (Single precision floating point)	4 bytes	-3.402823E38 a -1.401298E-45 (para negativos) 1.401298E-45 a 3.402823E38 (para positivos)
Double (Double precision floating point)	8 bytes	-1.79769313486232E308 a -4.94065645841247E-324 (neg) 4.94065645841247E-324 a 1.79769313486232E308 (pos)
Currency (Scaled integer)	8 bytes	-922,337,203,685,477.5808 a 922,337,203,685,477.5807

TIPOS DE DADOS

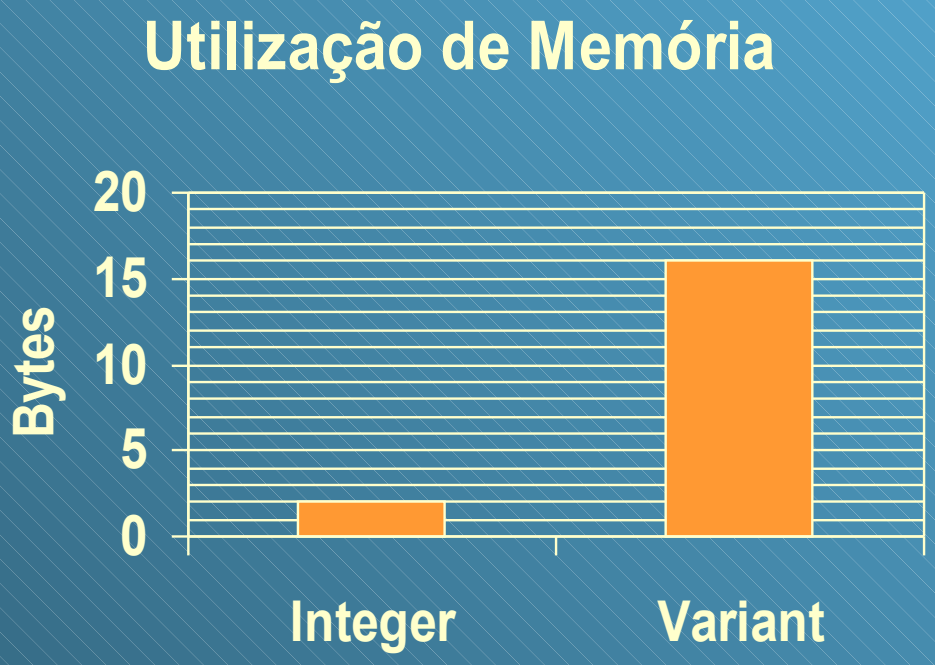
Tipo	Tamanho	Faixa
Decimal	14 bytes	79,228,162,514,264,337,593,543,950,335 (sem ponto decimal); 7.9228162514264337593543950335 (com 28 posições para a decimal); 0.0000000000000000000000000001 (menor número não zero).
Date	8 bytes	January 1, 100 a December 31, 9999
Object	4 bytes	Qualquer referência
String	10 bytes + tamanho	0 a 2 bilhões
String (fixa-*)	Tamanho da string	1 a 65,400
Variant (Numeric)	16 bytes	Qualquer valor numérico até o range do Double
Variant (Text)	22 bytes + tamanho	Mesmo range do tipo String
Type (User-defined)	Depende dos elementos	Mesmo range dos elementos internos do Type

VARIÁVEIS

Por que devemos utilizar o tipo apropriado de variável?

- Variant, o tipo default no VB, armazena qualquer tipo de dado mas requer mais memória.
- Outro detalhe que devemos levar em consideração é a performance, pois utilizando variant existe um processo de conversão quando o dado é manipulado !

Ex: A diferença da utilização de inteiro e variant.



Convenção Microsoft para nomes de Variáveis

- Tem que ser único no mesmo escopo
- Não pode ser maior que 255 caracteres
- Não pode conter caracteres específicos(@, &, &, !, #, $)
- Tem que começar com um caracter do alfabeto

Tipo de Dado	Prefixo
Byte	b
Boolean	f (de flag)
Integer	i
Long	l
Single	s
Double	dbl
Currency	c
Date	dt
Object	Usar o prefixo do obj. (frm, txt,etc)
String	str
Variant	v

Ex: Dim fligado as Byte
Private mstrNome as String

Convenção Microsoft para nomes de Variáveis

Escopo	Prefixo
Global	g
Modulo/Form	m
Local	l

Ex:

Global gfligado as Byte (Public gfligado as Byte)

Private mlstrNome as String

Dim dblvalor as Double

Dica: Para nomes de constantes utilize sempre todas as letras maiúsculas para fácil identificação.

Option Explicit

Com esta opção, todas as variáveis necessitam ser "explicitamente" declaradas!

Forma Automática :

Menu Tools/ Ítem de menu Options/ Tab Editor/

Opção Require Variable Declaration

Forma Manual:

Seção General Declaration/ Digitar Option Explicit

Implicitamente significa: Utilizar a variável sem declará-la

```
Sub Form_Load()
    Vtemp = 100
End Sub
```

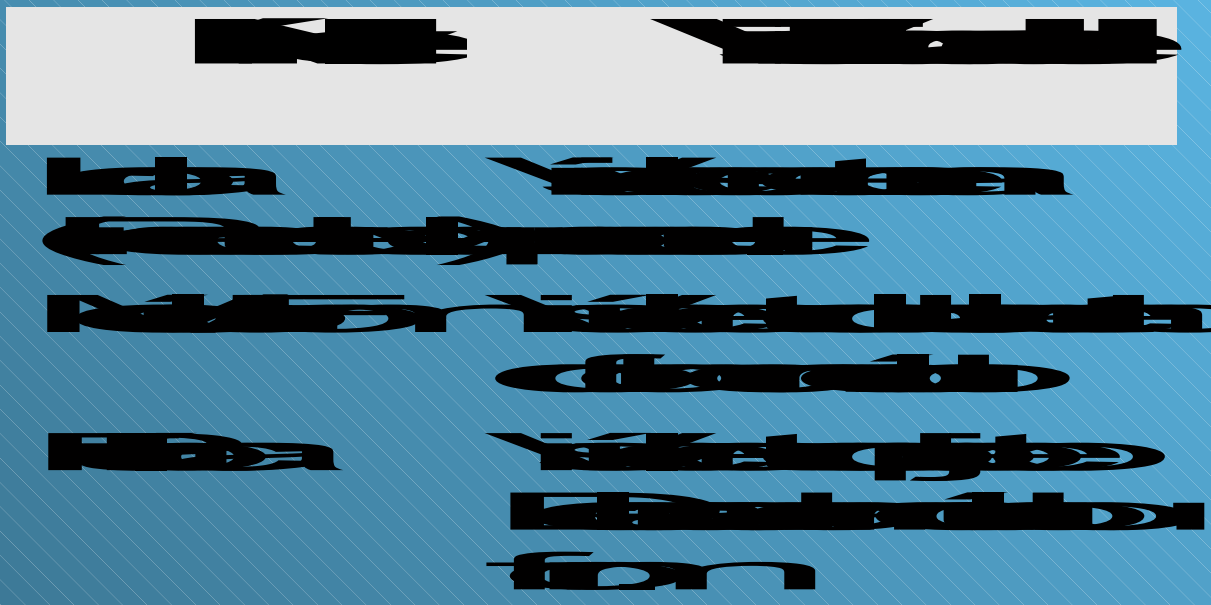
Explicitamente ficaria:

```
Sub Form_Load()
    Dim vtemp as Integer
    Vtemp = 100
End Sub
```

Dica: Colocar todas as variáveis explicitamente e esquecer o que é declaração explicita ou implicita!

Escopo (Scope)

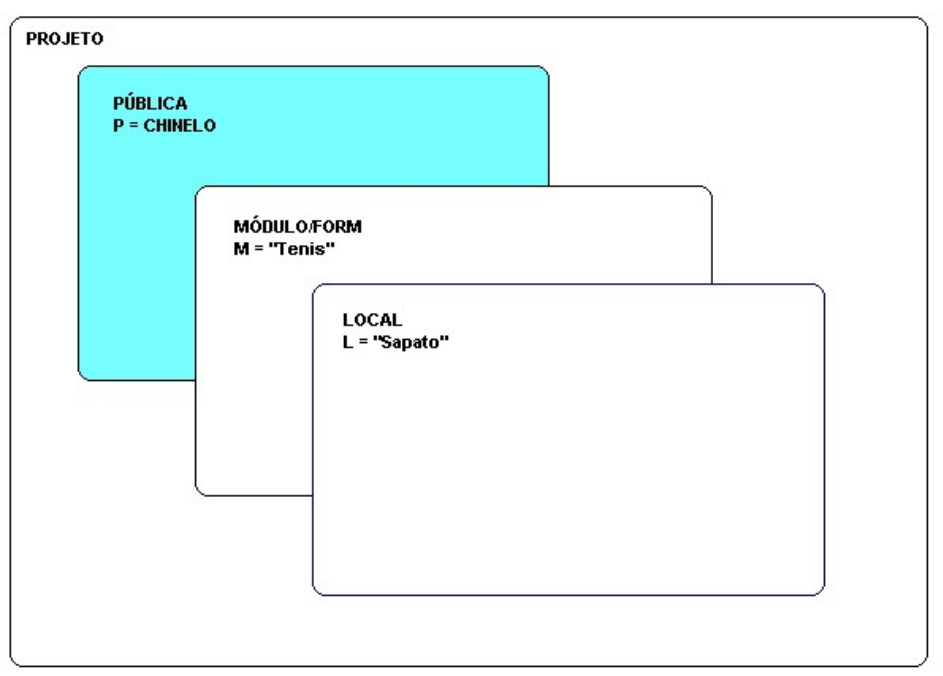
Escopo define a visibilidade da variável, procedure ou objeto



- Uma variável local é declarada dentro de uma procedure utilizando Dim.
- Uma variável de Módulo/Form é declarada em general declarations do Módulo/Form utilizando Dim/Private.
- Uma variável Pública é declarada em general declarations do Módulo/Form utilizando Public.

Dica: Não esquecer que o escopo também se aplica a constantes

Escopo (Scope)



Project1 - FrmTeste (Code)

(General) | (Declarations)

```
'Publica
Public gfSexo    As Boolean

'Privadas
Private mfsinal As Boolean
Dim mftipo      As Boolean    'Forma antiga
Private Sub Command1_Click()
    Dim strNome As String
    strNome = "Ferdinando Henrique Seboso"
    MsgBox "Seu nome foi modificado para" & strNome, vbOKOnly, "Sistema"
    mftipo = True
End Sub
```

Variáveis Estáticas (Static)

Tempo de Vida (**Lifetime**) é a chave do porque existem variáveis estáticas.

As variáveis locais retêm seu valor somente enquanto a procedure durar, ou seja, toda vez que a procedure acaba, o valor das variáveis locais desta procedure retorna para 0 ou nulo.

Utilizando variáveis estáticas o valor se mantêm enquanto o aplicativo existir.

Ex:

```
Sub cmdContador_Click()
    Static iCount as Integer
    ICount = iCount + 1
    MsgBox “Você já pressionou  “ & icount & “ este botão.”
End Sub
```

Dica: Uma variável de escopo maior (nível de módulo/form ou pública), também não perde o valor só que esta variável pode ser acessada por todos as outras procedures, diferente da variável do tipo estática que só pode ser acessada pela procedure que a criou.

Declarando Variáveis

Declarando **Variáveis Locais**

- Declaração na procedure utilizando Dim.

```
Sub cmdDezPorcento_Click()
        Dim ivalor as Double
        Ivalor = Cint(txtvalor.Text) * 1.10
End Sub
```

Variáveis de Módulo/Form e Variáveis Públicas são

- Declaradas no mesmo local: Na Seção Geral de Declaração (General Declarations) que se encontra no Form, Módulo ou Class Module.

Diferente das variáveis locais, que são sempre privadas, variáveis declaradas na Seção General Declarations podem ser tanto private ao módulo/form que foram declaradas ou podem ser públicas.

Private - Privada ao módulo ou form:

```
Dim variavel [As tipo]
Private variavel [As tipo]
```

Public - Disponível para todos os módulos do projeto:

```
Public varname [as tipo]
```

Nota: Uma variável pública declarada num form tem que ser prefixada pelo nome do form quando chamada de outro form ou módulo (Ex: Form1.icount)

FUNÇÃO MSGBOX

Caixa de Mensagem (MessageBox) oferece uma simples e fácil forma de mostrar ao usuário uma informação ou permitir ao usuário tomar decisões sobre qual curso que o programa deve percorrer.

Sintaxe:

MsgBox(prompt [, buttons] [,title] [,helpfile, context])

prompt - único campo requerido - a mensagem.

Buttons - Expressão númerica, determina quantos e quais botões serão mostrados + o ícone que irá aparecer.

Title - Título da mensagem.

HelpFile - Arquivo help associado.

Context - Contexto do arquivo help associado.

Msg = “Você quer continuar ?“

Estilo = vbYesNo + vbExclamation + vbDefaultButton2

Titulo = “Demonstração de Utilização do MsgBox”

Resposta = **Msgbox**(Msg, Estilo, Titulo)

Para este exemplo o valor da varíavel Resposta pode conter vbYes ou vbNo (6 ou 7)

FUNÇÃO MSGBOX

Constante	Valor	Descrição
vbOKOnly	0	Botão OK
vbOKCancel	1	Botões OK e Cancelar
vbAbortRetryIgnore	2	Botões Abortar, Repetir e Ignorar
vbYesNoCancel	3	Botões Sim, Não e Cancelar
vbYesNo	4	Botões Sim e Não
vbRetryCancel	5	Botões Repetir e Cancelar
vbCritical	16	Ícone Mensagem Crítica
vbQuestion	32	Ícone Consulta de Aviso
vbExclamation	48	Ícone Mensagem de Aviso
vbInformation	64	Ícone Mensagem Informativa
vbDefaultButton1	0	Primeiro botão padrão
vbDefaultButton2	256	Segundo botão padrão
vbDefaultButton3	512	Terceiro botão padrão
vbApplicationModal	0	Aplicativo modal; o usuário deve responder à caixa de mensagem antes de continuar o trabalho no aplicativo atual
vbSystemModal	4096	Sistema modal; todos os aplicativos são suspensos até o usuário responder à caixa de mensagem

Respostas	
vbOK	1 OK
vbCancel	2 Cancelar
vbAbort	3 Abortar
vbRetry	4 Repetir
vbIgnore	5 Ignorar
vbYes	6 Sim
vbNo	7 Não