EXCEL e VBA

Visual Basic for Applications

Escopo da Apresentação

- Integração do VBA com Excel
- Descrição do ambiente de programação do VBA
- Conceitos básicos de programação
- Exemplos usando VBA

Background do BASIC

BASIC = Beginner's All-Purpose Symbolic Instruction Code

Linguagem criada no início dos anos 60.

1991 ➔ Microsoft lança o VB para aplicações *standalone*.

1995 ➔ Microsoft lança Office 95 cujas aplicações (Excel, Word, PowerPoint, etc) incluem VBA.

VBA = Visual Basic for Applications ➔ linguagem *script* comum para todas as aplicações da Microsoft.

Motivação para se usar o VBA

Excel → Um programa muito eficiente (Sem concorrente!!)

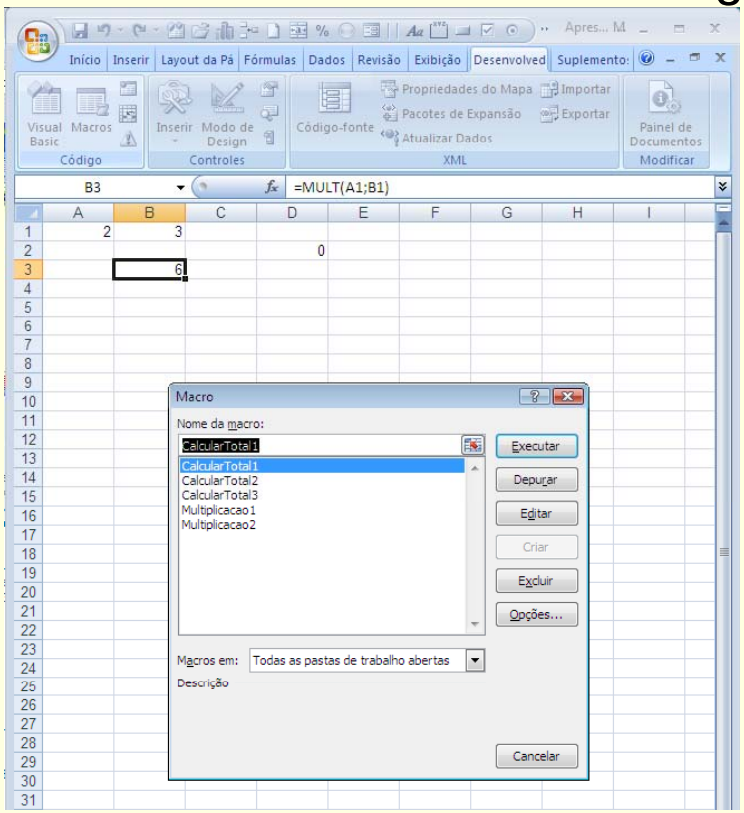
VBA → Complemento para o Excel

VBA → LOOPS: ineficiente no Excel, eficiente no VBA

Ambiente de Programação

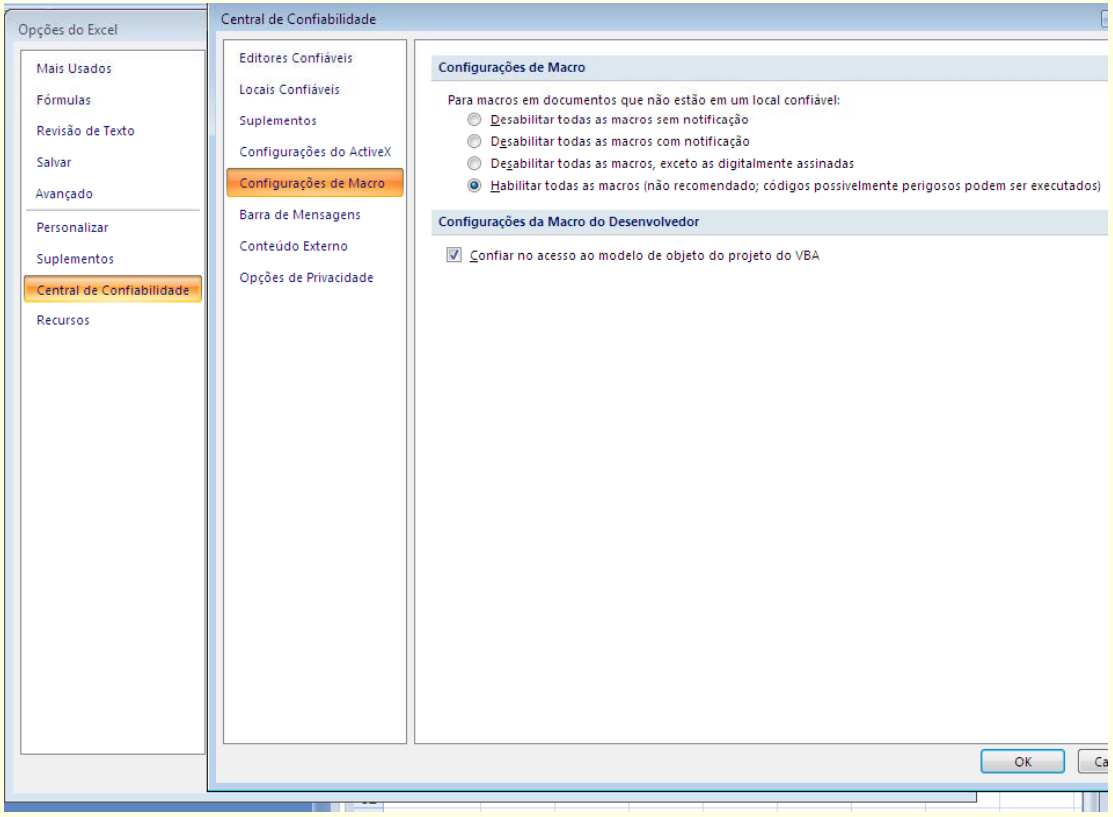
Nenhuma instalação extra é necessária para se usar o VBA

Para acessar o VBA vá: Desenvolvedor ➔ Código➔ Macro



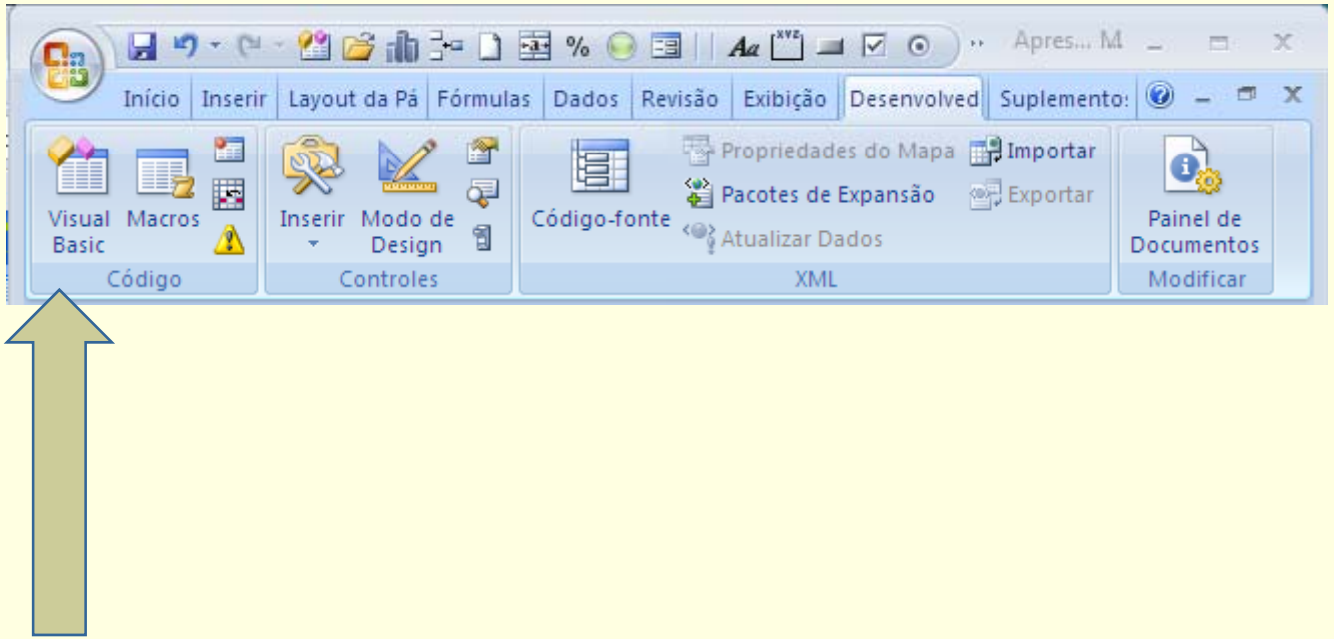
Segurança

Clique no botão do Office  e em **Opções do Excel**. Na janela *Opções do Excel* clique em **Central de Confiabilidade** e a seguir em **Configuraçãoes da Central de Confiabilidade**. Marque o botão **Habilitar todas as MACROS**.



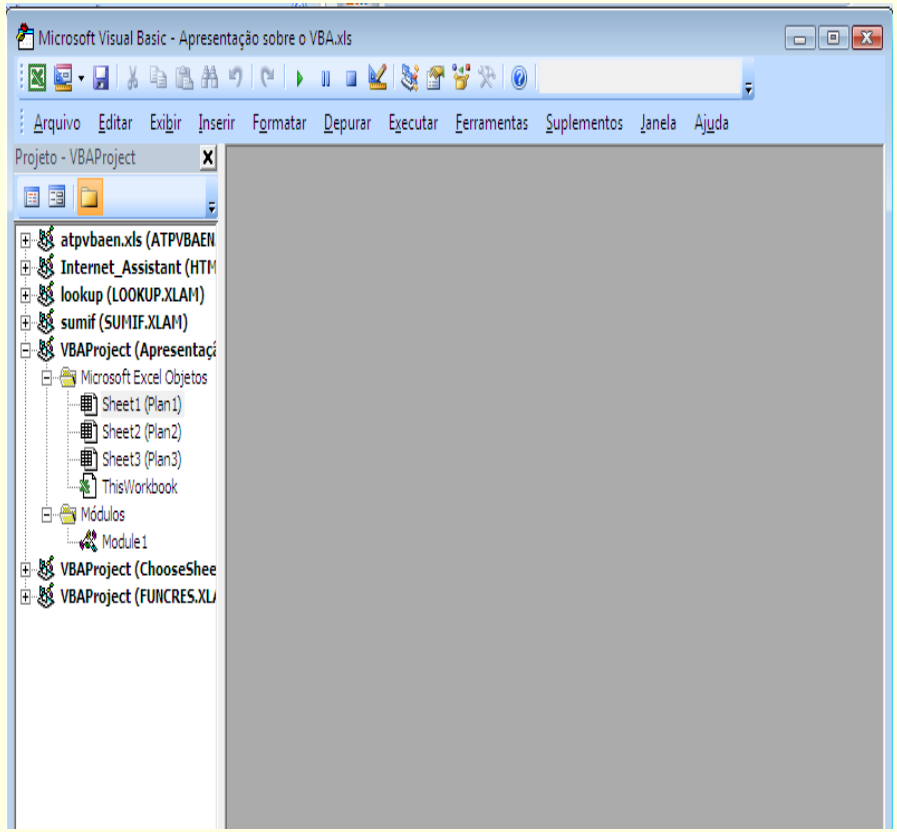
Visual Basic Editor (VBE)

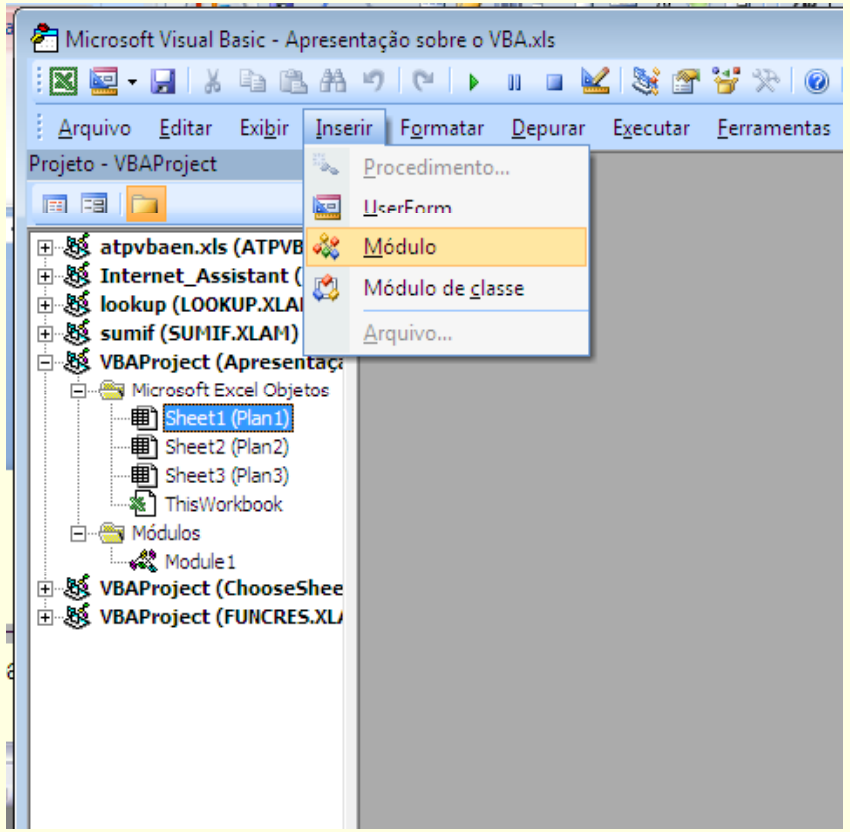
VBE pode ser acessado indo para: guia Desenvolvedor ➔ grupo Código ➔ Visual Basic ou pressionando Alt + F11



Criando um Módulo em VBA

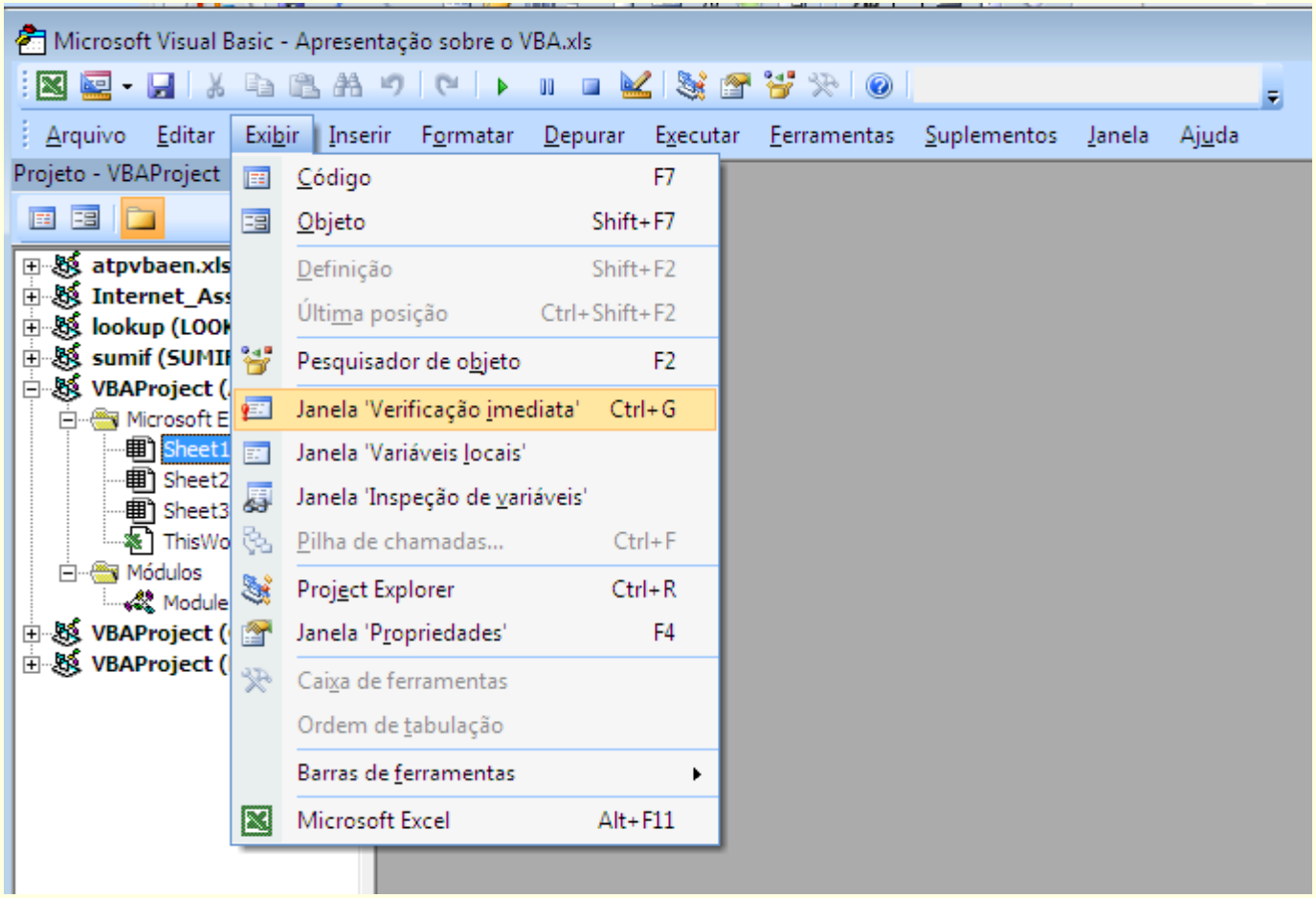
Para iniciar a programação devemos primeiro criar um Módulo na pasta (Workbook)





Janela Imediata

A Janela Imediata pode ser acesssada indo a: Exibir ➔ Janela “Verificação imediata ou pressionando Ctrl + G. Ela é como uma Janela Command Window



Objetos Básicos do VBA

Estes são os objetos mais freqüentemente usados do VBA:

- **Range**: refere-se a uma célula particular do Excel. Exemplo:
  Range("a2").Value=3
- **Cells**: uma outra maneira de se referir a uma célula particular do Excel. Exemplo:
  Cells(2,2).Value=6
- **Worksheets**: refere-se a uma planilha particular do Excel. Exemplo:
  Worksheets(“Plan3").Select
- **Worksheetfunction**: chama as funções disponíveis do Excel.
  Exemplo: Worksheetfunction.Fact(3)
- **MsgBox**: mostra uma mensagem no Excel. Exemplo:
  MsgBox Application.Name
- **?**: mostra um resultado, variável, etc Na Janela Imediata.
  Exemplo:

```
a=Worksheetfunction.Fact(3)
? a
6
```

Variáveis

- Elas contêm dados temporários
- Podemos pensar nas variáveis como “células do Excel” armazenaddas na memória RAM do PC
- Embora não seja obrigatório, é uma BOA prática de programação definir o tipo de informação que as variáveis armazenarão (em execuções longas ela economiza UM POUCO de tempo). Isto é chamado “declarar uma variável”
- O tipo de informação que podemos armazenar numa variável depende do *tipo* de dados escolhido para aquela variável

Tipos de Dados do VBA (I)

- **Byte:** armazena inteiros sem sinais entre 0 e 225
- **Boolean:** armazena Verdadeiro ou Falso
- **Integer:** armazena inteiros entre -32.768 e 32.767
- **Long:** armazena inteiros entre -2.147.483.648 e 2.147.483.647
- **Currency:** armazena inteiros na escala de 10.000 entre -922.337.203.685.477,5808 e 922.337.203.685.477,5807
- **Single:** armazena números de ponto flutuante entre -3,402823E38 e -1,401298E-45 para valores negativos e entre 1,401298E-45 e 3,402823E38 para valores positivos
- **Double:** armazena números de pontos flutuantes entre –1,7976931348623 1E308 e -4,94065645841247E-324 para valores negativos e entre 4,94065645841247E-324 e 1,7976931348623 2E308 para valores positivos

Tipos de Dados do VBA (II)

- **Date:** armazena datas (números de pontos flutuantes) entre 1º Janeiro de 100 e 31 de Dezembro de 9999 e tempo entre 0:00:00 e 23:59:59
- **String:** armazena string de caracteres. Existem duas espécies de strings:
  - Strings de tamanhos variáveis: podem conterr até 2.000 milhões (2^31) de caracteres
  - Strings de tamanho fixo: podem conter entre 1 e 64 KB (2^16) caracteres
- **Object:** armazena endereços que se referem a outros objetos
- **Variant (predeterminada):** tipo de dado default para cada variável que não é declarada como de qualquer outro tipo
- **User defined type:** tipo de dado criado pelo usuário

Declaração de Variável

- As variáveis são declaradas antes de serem usadas, geralmente no início do programa
- Os exemplos abaixo mostram os tipos de dados usados mais comuns
- Adicionando “Option Explicit” força a declaração da variável (recomendado)

Exemplos:

```
Option Explicit
Dim Result as Long
Dim Totalsum as Double
Dim Description as String
Dim Startdate as Date
```

Variáveis Especiais

Contadores e Acumuladores: variáveis especiais que armazenam certos valores quando se executa um LAÇO

- **Contadores**: armazena um, usado para contagem

  Exemplo:

```
Dim Iteration as Long
...
Iteration = Iteration + 1
...
```

- **Acumuladores**: armazena qualquer valor, usado para somatórios

  Exemplo:

```
Dim Somatotal as Double
...
Somatotal = Somatotal + Saldo
...
```

Estruturas Padrões de Programação

If – End If	Bifurcação
Select Case – End Select	
For – Next	Circulação
Do While - Loop	
Exit	Declarações Especiais
With – End With	

If – End If (I)

- Esta é a maneira mais comum para manipular bifurcação
- É análoga à função se() do Excel

Sintaxe:

```
If <Condição> then
    [Instruções  se a Condição if for Verdadeira]
Else
    [Instruções  se a Condição if for Falsa]
End If
```

If – End If (II)

Exemplo:

```
Sub Multiplicação1()
    Dim Mult as Double
    Mult = Range(“a1”).Value*Range(“b1”).Value
    If Mult > 20 Then
        MsgBox “Maior que 20"
    Else
        MsgBox “Menor que ou igual a 20"
    End If
End Sub
```

Select Case – End Select (I)

- Esta estrutura é análoga à estrutura If – End If
- Ela foi projetada para evitar os If – End If aninhados

Sintaxe:

```
Select Case <Expressão>
    Case < Condição 1>
        [Instruções se a Condição 1 if for Verdadeira]
    ...
    Case < Condition n>
        [Instruções se a Condição n if for Verdadeira]
    Case Else
        [Instruções se as n Condições Anteriores forem Falsas]
End Select
```

Select Case – End Select (II)

Exemplo

```
Sub Multiplicacao2()
    Dim Mult as Double
    Mult = Range(“a1”).Value*Range(“b1”).Value
    Select Case Mult
        Case Is < 10
            MsgBox “Menor que 10"
        Case Is < 15
            MsgBox “Maior que ou igual a 10 e menor que 15"
        Case Is < 20
            MsgBox “Maior que ou igual a 15 e menor que 20"
        Case Else
            MsgBox “Maior que ou igual a 20"
    End Select
End Sub
```

For – Next (I)

- Este é o laço mais simples e mais amplamente usado
- Especificamos quantas vezes o laço será repetido

Sintaxe:

```
For <Contador=Início> To <Fim>
    [Instruções]
Next
```

For – Next (II)

Exemplo:

```
Sub CalcularTotal1()
    Dim Contador As Integer
    Dim Total As Double
     Total=0
    For Contador = 2 To 15
        If Range(“a" & Contador).Value = "Centro" Then
            Total = Total + Range(“b" & Contador).Value
        End If
     Next
     Range(“d2").Value = Total
End Sub
```

Do While – Loop (I)

- Este é outro laço comumente usado
- Especificamos que o laço será repetido enquanto uma certa condição for verdadeira

Sintaxe:

```
Do While <Condição>
    [Instruções Enquanto a Condição for Verdadeira]
Loop
```

Do While – Loop (II)

Exemplo:

```
Sub Calcularotal2()
    Dim Contador As Integer
    Dim Total As Double
     Contador = 2
     Total = 0
     Do While Range(“a" & Contador).Value <> “Sul"
         Total = Total + Range(“b" & Contador).Value
          Contador = Contador + 1
     Loop
     Range(“d2").Value = Total
End Sub
```

Exit (I)

- Esta declaração é usada para encerrar o bloco:

For – Next	→	Exit For
Do While – Loop	→	Exit Do
Function – End Function	→	Exit Function
Sub – End Sub	→	Exit Sub

Sintaxe: (dentro de um bloco For – Next)

```
For <Contador=Inicio> To <Fim>
    [Instruções]
    If <Condição> then
        [Sair se a condição for verdadeira]
    End If
Next
```

Exit (II)

Exemplo:

```
Sub CalcularTotal3()
    Dim Contador As Integer
    Dim Total As Double
     Total=0
    For Contador = 2 To 15
        If Range(“b" & Contador).Value < 0 Then
            Msgbox “Há um erro no número registrado ” & Contador - 1
            Total=0
            Exit For
        End If
        If Range(“a" & Contador).Value = "Centro" Then
            Total = Total + Range(“b" & Contador).Value
        End If
     Next
    Range(“d2").Value = Total
End Sub
```

With – End With (I)

- Esta estrutura é usada para encurtar as declarações e acelerar a execução do código
- Embora seja similar, NÃO é uma estrutura de laço

Sintaxe:

```
With <Objeto>
    [Instruções]
End With
```

With – End With (II)

Exemplo: SEM usar a estrutura With – End With

```
Sub CalcularTotal()
    Dim Contador As Integer
    Dim Total As Double
    Contador = 2
    Total = 0
    Do While Worksheets("Plan1").Range("a" & Contador).Value <> "Sul"
        Total = Total + Worksheets("Plan1").Range("b" & Contador).Value
        Contador = Contador + 1
    Loop
    Worksheets("Plan1").Range("d2").Value = Total
End Sub
```

With – End With (III)

Exemplo: usando a estrutura With – End With

```
Sub CalcularTotal()
    Dim Contador As Integer
    Dim Total As Double
    Contador = 2
    Total = 0
    With Worksheets("Plan1")
        Do While .Range("a" & Contador).Value <> "Sul"
            Total = Total + .Range("b" & Contador).Value
            Contador = Contador + 1
        Loop
        .Range("d2").Value = Total
    End With
End Sub
```

Rodando um Programa

- Um programa é um conjunto organizado de instruções para realizar certas tarefas
- Nos programas VBA são chamados de “procedimentos”
- Existem somente dois tipos de procedimentos VBA:
  - Function procedures
  - Sub procedures
- Certas tarefas são realizadas mais eficientemente com Function procedures e algumas outras tarefas podem ser feitas mais eficientemente com Sub procedures
- Para rodar um programa ou procedure, devemos então rodar uma Function or Sub procedure

Function Procedures (I)

Function Procedures: conjuntos de instruções que realizam certas tarefas e retornam um valor (podemos pensar nas Function Procedures como funções do Excel)

- Elas NÃO podem modificar os objetos (p. ex.: modificar um valor da célula)
- Elas são usualmente executadas de uma célula automaticamente (como qualquer função Excel)

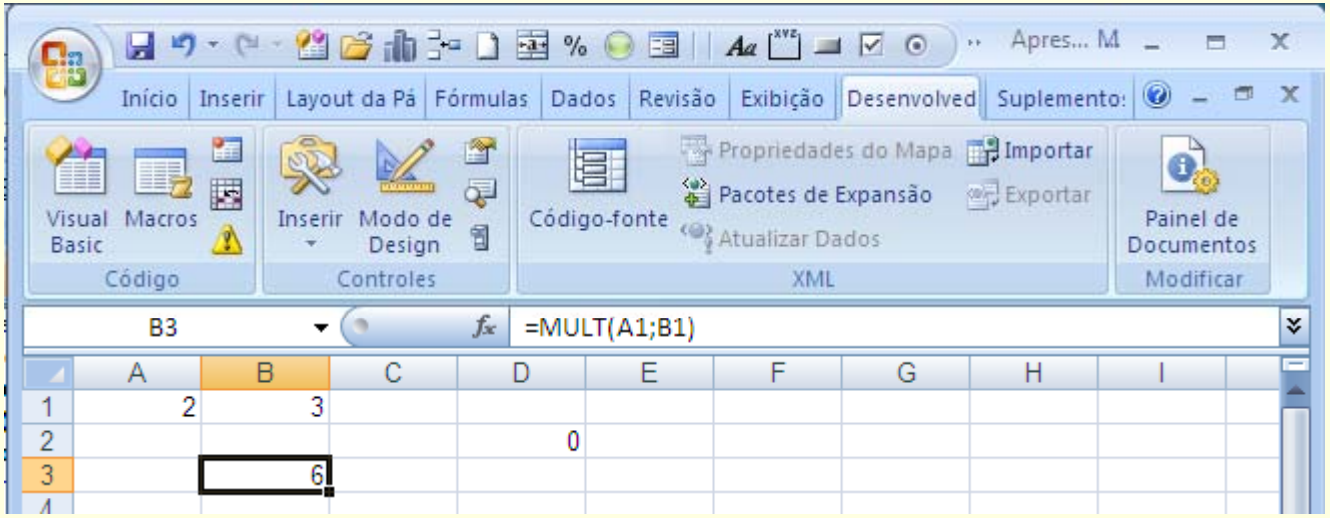
Sintaxe:

```
Function Procedure_Nome([Lista de Argumentos]) [As Tipo de Dado]
    [Instruções]
End Function
```

Function Procedures (II)

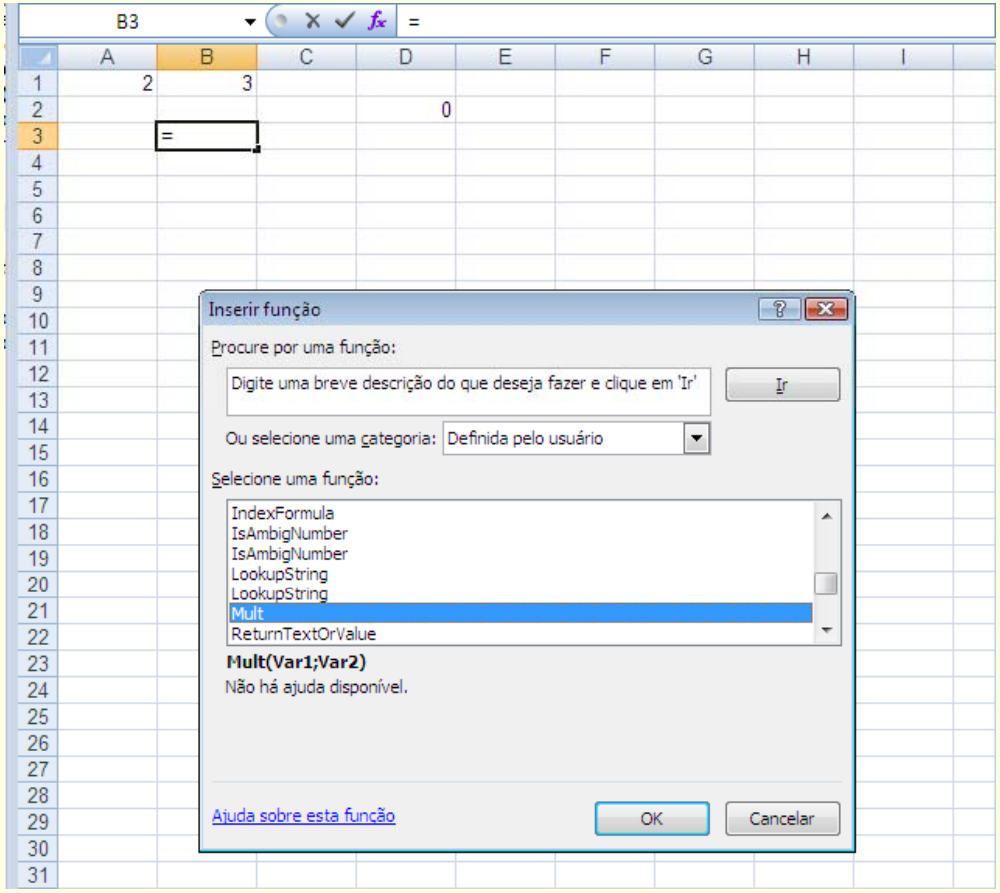
Exemplo:

```
Function Mult(Var1 As Double, Var2 As Integer) As Double
    Mult = Var1 * Var2
End Function
```



Localizando as Funções do Excel

Todas as funções que criamos podem ser encontradas e executadas em: Inserir ➔ Função… ➔ Definida pelo usuário



Sub Procedures (I)

Sub Procedures: conjuntos de instruções que realizam certas tarefas

- Elas PODEM modificar os objetos (pex.: modificar um valor de célula)
- Elas NÃO podem ser executadas de uma célula (como uma função Excel ou uma Function Procedure)
- Elas podem ser executadas manualmente do VBA Editor, da Janela Macro, usando Formulários numa planilha, etc

Sintaxe:

```
Sub Procedure_Nome([Lista de Argumentos])
    [Instruções]
End Sub
```

Sub Procedures (II)

Exemplo:

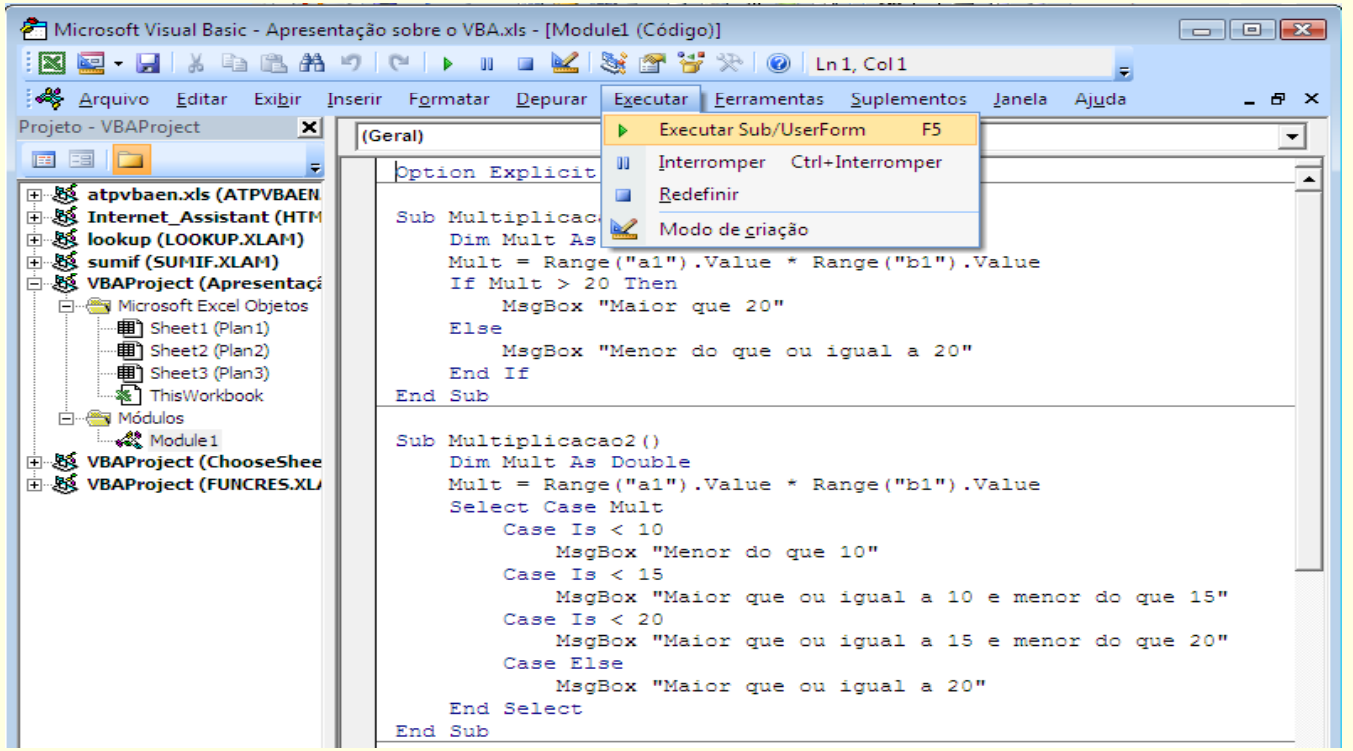
```
Sub CalcularTotal2()
    Dim Contador As Integer
    Dim Total As Double
    Contador = 2
    Total = 0
    Do While Range(“a" & Contador).Value <> “Sul"
        Total = Total + Range(“b" & Contador).Value
        Contador = Contador + 1
    Loop
    Range(“d2").Value = Total
End Sub
```

Executando um Sub Procedures no VBE

Colocando o cursor na macro podemos executar, ir para: Executar ➔ Executar Sub/UserForm, ou pressionando F5

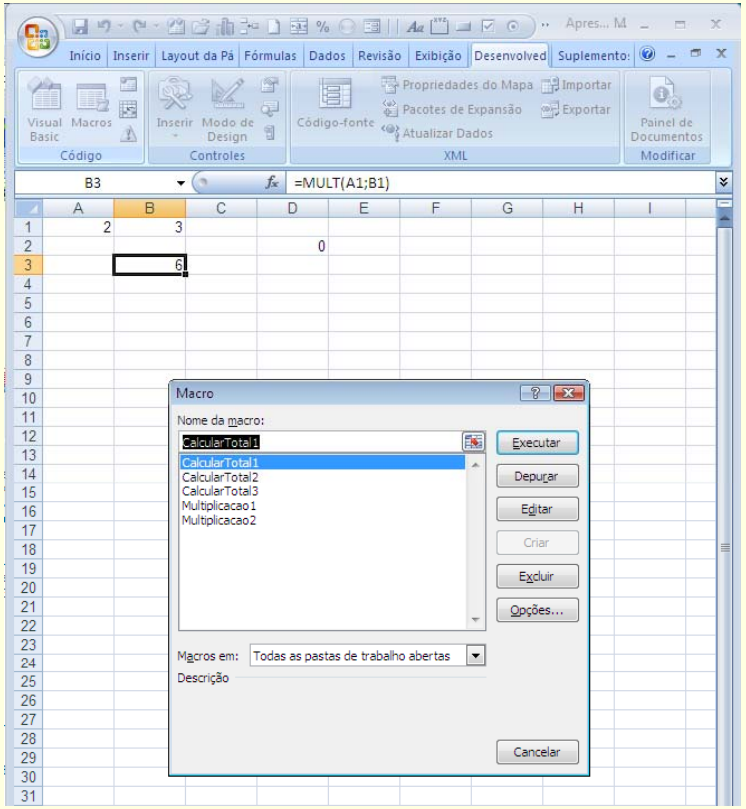
Podemos também usar a Janela Imediata para executar as nossas macros

Mas gostaríamos de executar nossas macros do Excel…



Executando Sub Procedures no Excel (I)

Todos os Sub Procedures que criamos podem ser encontrados e executados em: Desenvolvedor ➔ Código ➔ Macros. Esta maneira de executar é enrolada, mas existe uma outra muita mais fácil



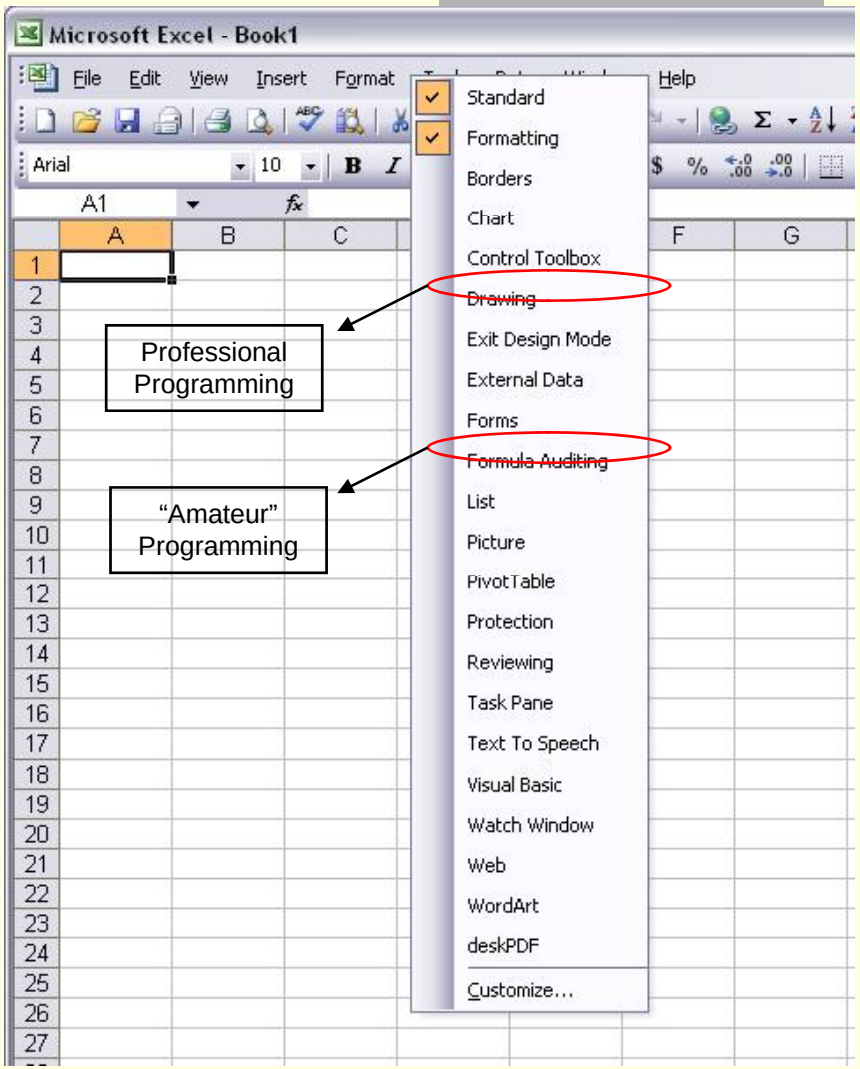
Executando Sub Procedures no Excel (II)

- Podemos executar nossas macros usando Formulários
- Formulários melhoram substancialmente nossas planilhas e dão a elas um aspecto profissional

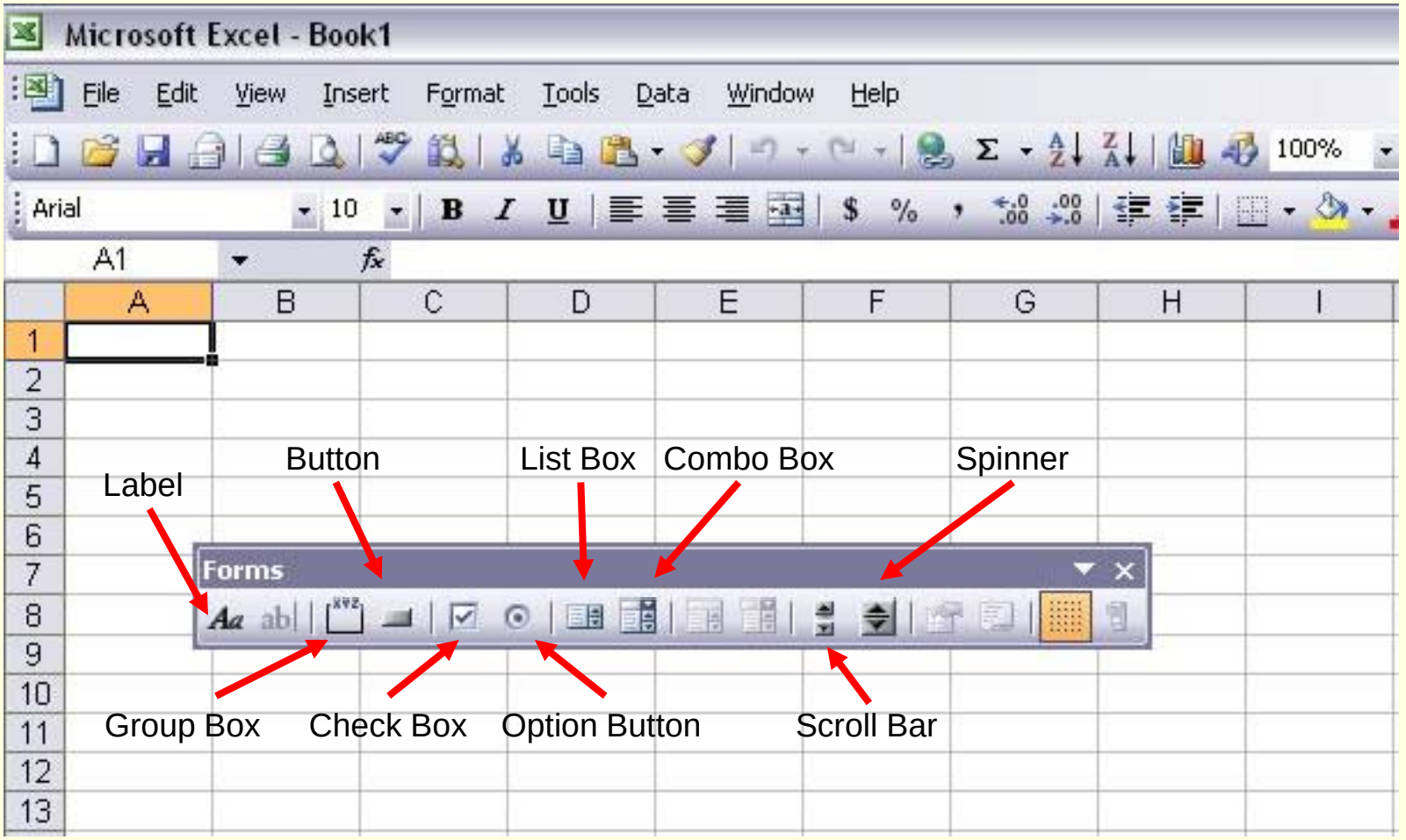
Nota: existem 2 versões de Formulários:

- Formulário "Completo": no menu de Caixa de Ferramentas de Controle
- Formulário "Simplificado": no menu Formulário

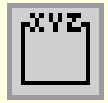
Geralmente usaremos os Formulários "Simplificados"



Forms (I)


Microsoft Excel - Book1
File Edit View Insert Format Tools Data Window Help
Arial 10
A1
Label
Button
List Box
Combo Box
Spinner
Forms
Group Box
Check Box
Option Button
Scroll Bar

Formulários (II)

- **Label:** contém apenas texto informativo 
- **Group Box:** grupos de formulários relacionados 
- **Button:** apenas executa uma macro quando pressionado 
- **Check Box:** ativa ou desativa uma alternativa 
- **Option Button:** seleciona uma entre muitas alternativas 

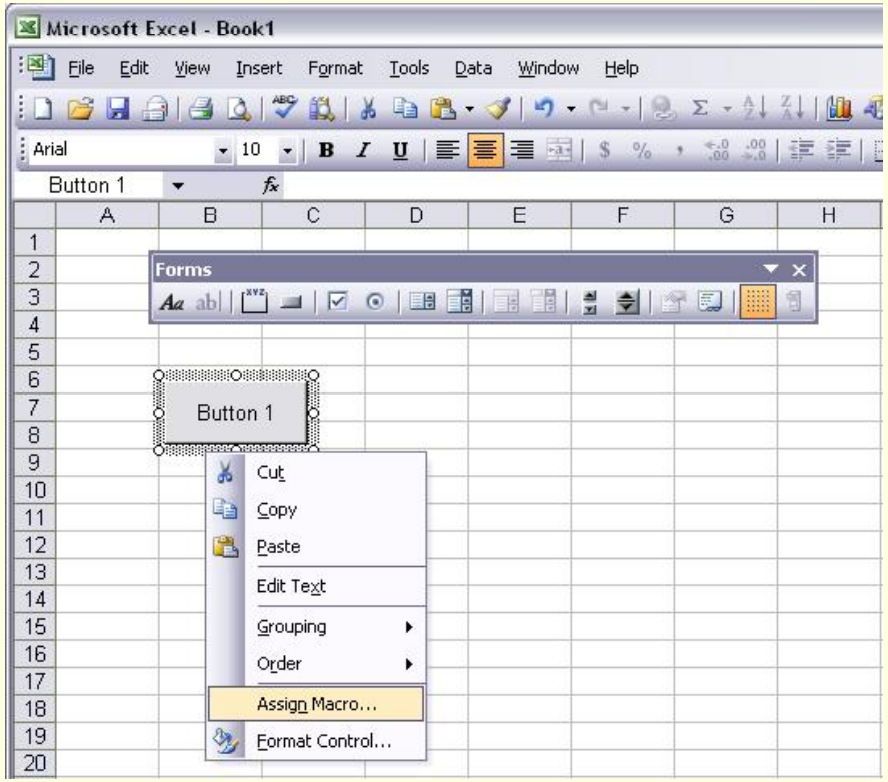
Formularios (III)

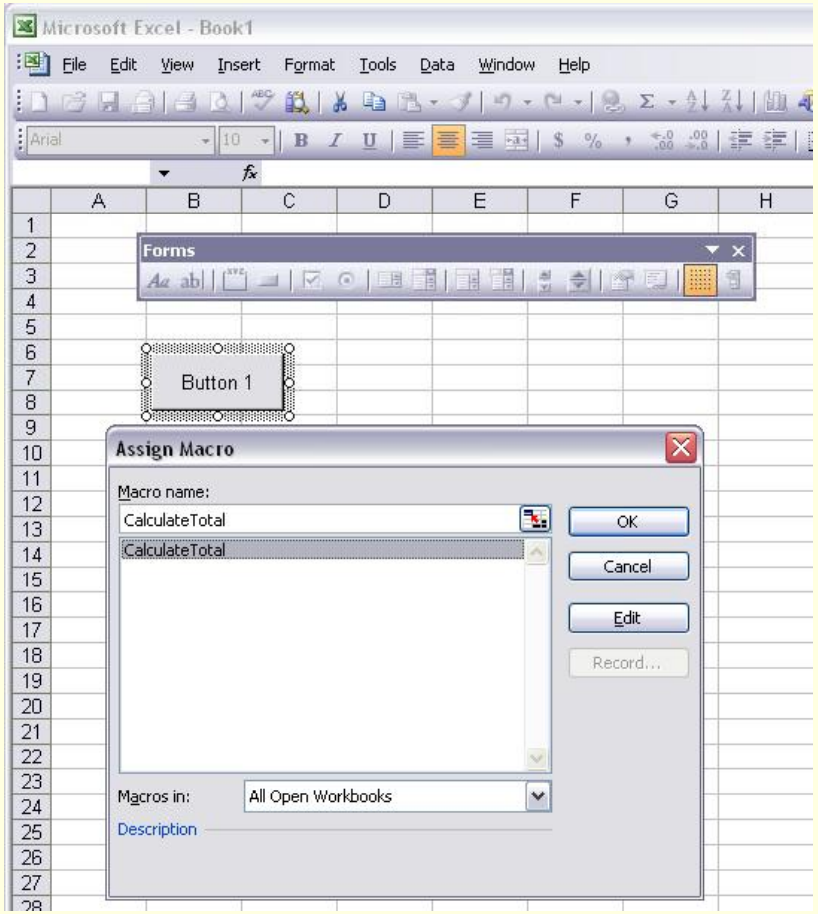
- **List Box**: contém uma lista de elementos
- **Combo Box**: contém uma lista de elementos drop-down
- **Scroll Bar**: rola através de um intervalo de valores
- **Spinner**: muda o valor de uma célula

Executando Macros com um Button

Esta é a Forma *natural* de se executarem macros de uma planilha.

Contudo, podemos atribuir uma macro a qualquer outro Formulário

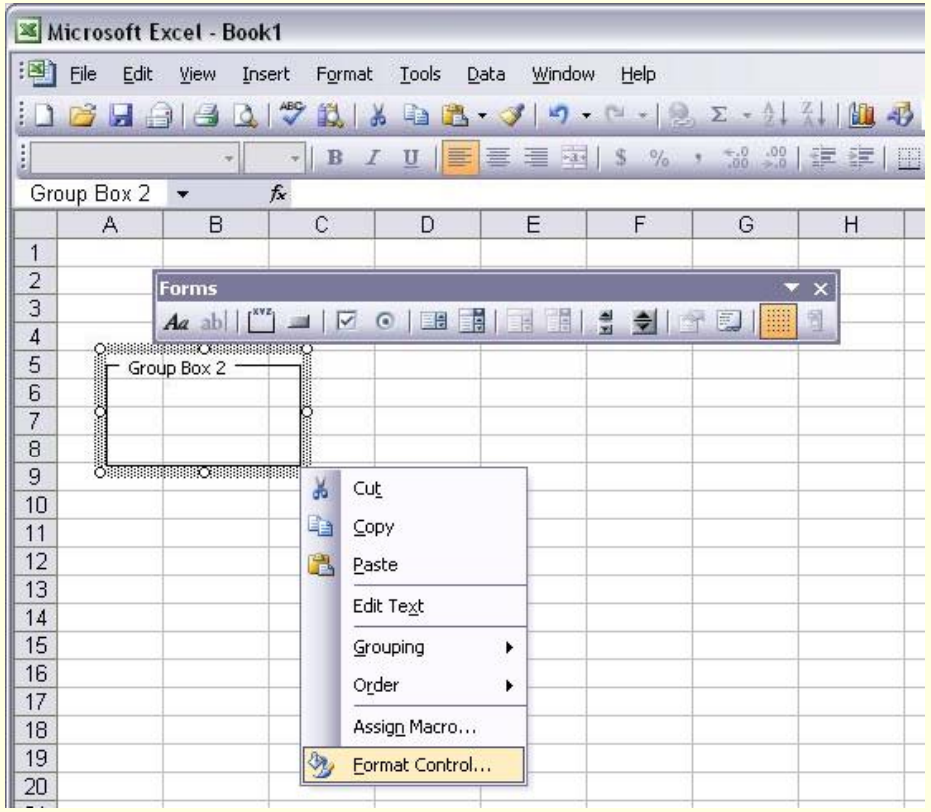




Propriedades de uma Group Box

3-D shading: dá o aspecto de um Formulário 3-D

Nota: este Formulário é usado freqüentemente para melhorar o aspecto visual da planilha. Contudo, é essencial para agrupar Option Buttons



Propriedades do Button

Esta é a Forma *natural* para executar macros da planilha.

Não existem propriedades *interessantes* para este Formulário

Propriedades da Check Box

Value: determina o estado da Check Box. Ela pode ser Não Verificada , Verificada ou Misturada (indefinida)

Cell link: refere-se à célula que armazenará o estado da Check Box. Ela armazenará False (Não Verificada), Verdadeira (Verificada) ou #N/A (Misturada)

3-D shading: dá o aspecto de Formulário 3-D

Propriedades do Option Button

Value: determina o estado do Option Button. Ele pode ser Não Verificado ou Verificado

Cell link: refere-se à célula que armazenará o estado do Option Button. Ela armazenará 0 (Não Verificado) ou 1 (Verificado). Se mais do que 1 Option Buttons forem usados (quase sempre), a célula armazenará o número de Option Button Verificados

3-D shading: dá o aspecto de Formulário 3-D

Propriedades da List Box

Input range: refere-se a uma lista de elementos numa planilha

Cell link: refere-se à célula que armazenará o elemento selecionado

Selection type: especifica a espécie de seleção. A maioria frequentemente usada é Single

3-D shading: gives the Form a 3-D aspect

Propriedades of Combo Box

Input range: refere-se a uma lista de elementos numa planilha

Cell link: refere-se a uma célula que armazenará o elemento selecionado

Drop down lines: especifica o número de elementos mostrados quando a Combo Box estiver listando os elementos

3-D shading: dá o aspecto de Formulário 3-D

Propriedades da Scroll Bar

Current Value: mostra o valor atual da Scroll Bar

Minimum value: mostra o valor mínimo da Scroll Bar

Maximum value: mostra o valor máximo da Scroll Bar

Incremental change: mostra quantas unidades a Scroll Bar muda quando uma extremidade é clicada

Page change: mostra quantas unidades a Scroll Bar muda quando a parte interna for clicada

Cell link: refere-se à célula que armazenará o valor atual da Scroll Bar

3-D shading: dá o aspecto de Formulário 3-D

Propriedades do Spinner

As mesmas propriedades da Scroll Bar, exceto para mudança de Página, que não está disponível para este Formulário

Exemplo de Excel + VBA

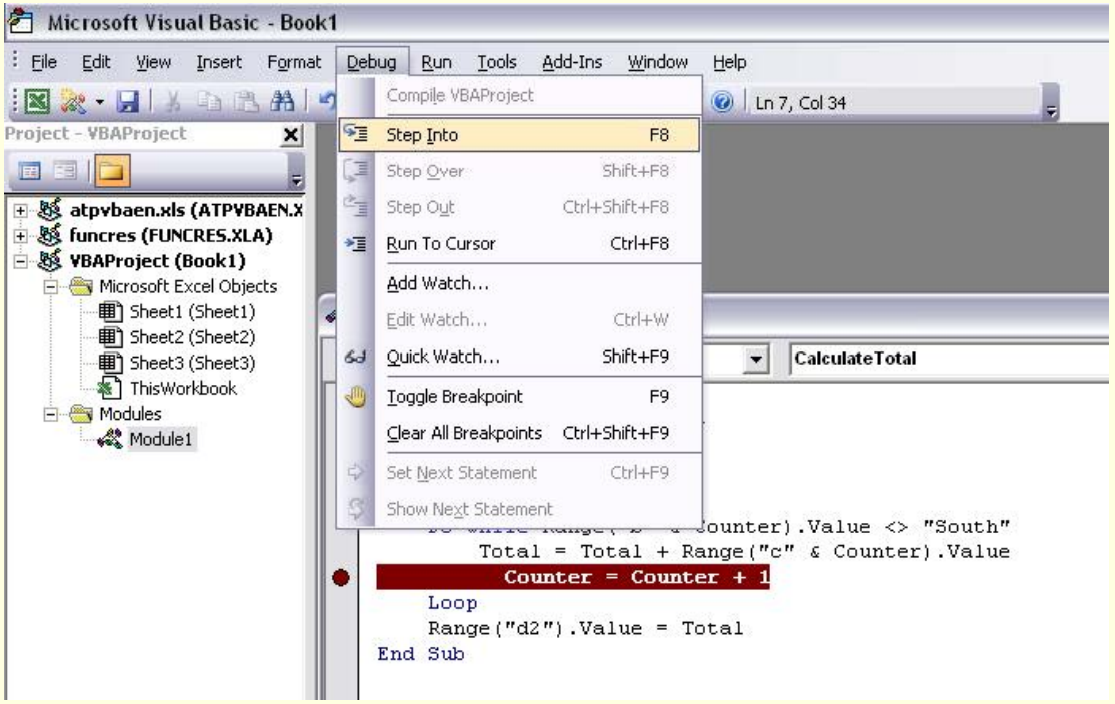
A planilha contém os preços de abertura diários de 5 anos da Bayer, Microsoft, Exxon e Bank of America

Tarefas:

1. Use uma Combo Box para escolher uma companhia
2. Usando Option Buttons crie 3 alternativas de retornos: semanal, mensal ou annual
3. Usando Option Buttons crie 2 alternativas de retornos : contínuo ou discreto
4. Usando Check Boxes crie 3 alternativas: calcule o retorno médio e/ou a variança dos retornos e/ou os parâmetros OLS (regressão linear) do CAPM
5. Use um Button para executar o programa (Sub Procedure) e armazene os resultados numa nova coluna

Depurador (Debugger) do VBA

- Depurar ou consertar programas é um aspecto da programação que NENHUM programador pode evitar. Contudo, o VBA tem um poder de depurar
- Duas ferramentas importantes de depuração são:
  - Breakpoints
  - Execução Passo por Passo, em : Debug ➔ Step Into, ou pressionando F8



Gravador de Macro

- O Gravador de Macro é uma característica muito importante do VBA (e de muitas outras linguagens)
- Sua intenção é principalmente ajudar os usuários do Excel a criarem macros (e acelerar tarefas repetitivas)
- Contudo, um uso inteligente do Gravador de Macro pode expandir o Excel além do seu escopo original e acelerar o processo de aprendizagem de programação
- Podemos acessar isto em: Ferramentas ➔ Macro ➔ Gravar Nova Macro…