INSTITUTO FEDERAL
TRIÂNGULO MINEIRO
Campus Uberlândia Centro

# CST em Logistica

Decisões Apoiadas em Planilha Eletrônica

# Usando Macros no

# Excel


**Prof. Walteno Martins Parreira Jr**

**www.waltenomartins.com.br**
**waltenomartins@iftm.edu.br**


**2015**

# 1 - Introdução

Uma macro é uma coleção de comandos que você pode aplicar com um único clique. As macros podem automatizar quase tudo que seja possível executar no programa que você está usando e até mesmo permitem fazer coisas que talvez você não soubesse que fossem possíveis.

As macros são programação, mas para usá-las, você não precisa ser um desenvolvedor e nem mesmo ter conhecimento de programação. A maioria das macros que você pode criar nos programas do Office é escrita em uma linguagem chamada Microsoft Visual Basic for Applications, ou VBA.

As macros economizam tempo e ampliam os recursos dos programas que você usa diariamente. Elas podem ser usadas para automatizar tarefas repetitivas de produção de documentos, simplificar tarefas cansativas ou criar soluções, como automatizar a criação de documentos que você e seus colegas usam regularmente. Os usuários com experiência em VBA podem usar macros para criar suplementos personalizados que incluem modelos, caixas de diálogo e até mesmo armazenam informações para uso repetido.

Considere o exemplo usado neste artigo de formatação de várias tabelas em um documento do Word. Imagine o documento tenha 50 tabelas e elas precisem ser reformatadas. Mesmo se você for um usuário avançado e leve apenas cinco minutos para formatar cada tabela, isso significa mais de quatro horas somente para essa tarefa. Se você gravar uma macro para formatar as tabelas e editar essa macro para repetir as alterações por todo o documento, poderá concluir a tarefas em questão de minutos e não horas.

Cada macro criada dá origem a um procedimento ou rotina. Existem dois tipos de rotinas:

1. As sub-rotinas ou rotinas Sub.

2. As funções ou rotinas Function.

As Sub-rotinas são aquelas cuja definição é delimitada pelas palavras-chave *Sub* e *EndSub*. Assim se reparar todas as macros que grava no Excel é deste tipo. Repare ainda como é que são definidas:

*Sub <nome_da_macro> ( )*
*<corpo_da_macro>*
*End Sub*

Estas Sub-rotinas são designadas pelo nome que lhe atribuímos e não recebem parâmetros do exterior, têm como função desempenhar um conjunto de tarefas que compõem o seu corpo. O corpo da macro, é assim composto por um conjunto de instruções, sendo que cada instrução diferente necessita de estar numa linha diferente. Contudo, quando se trata de instruções demasiado grandes o editor faz a sua partição por diversas linhas, recorrendo ao operador "_", de forma a facilitar a leitura.

As Funções são rotinas cuja definição começa com a palavra-chave Function e termina com as palavras End Function. Todas as funções que são utilizadas no Excel são deste tipo de rotina. A sua definição tem a seguinte estrutura:

*Function <Nome da Função> ( <parametro1>, <parametro2>,…)*
*…*

*<Nome da Função> = <Valor / Expressão>*
*…*
*End Function*

A função é identificada pelo nome, pelo número e tipo de parâmetros recebidos, e tem como objetivo executar um conjunto de instruções e produzir um valor final. Isto é, sempre que se pretender executar uma função é sabido à priori que ela produzirá um valor. Recorde-se como exemplo a função SOMA esta recebe por parâmetro um conjunto de valores que se pretendem somar, sabe-se que o resultado da aplicação dessa função ao conjunto de valores será o respectivo somatório.

Para definir o valor produzido por uma função basta no seu interior, atribuir ao nome da função um determinado valor ou expressão.

As funções são similares às sub-rotinas, existem simplesmente três diferenças:

1. As funções começam com a palavra-chave *Function* e terminam com as palavras *End Function*
2. As funções podem ser chamadas a partir de fórmulas introduzidas numa planilha.
3. As funções retornam valores para as fórmulas ou sub-rotinas que as chamarem.

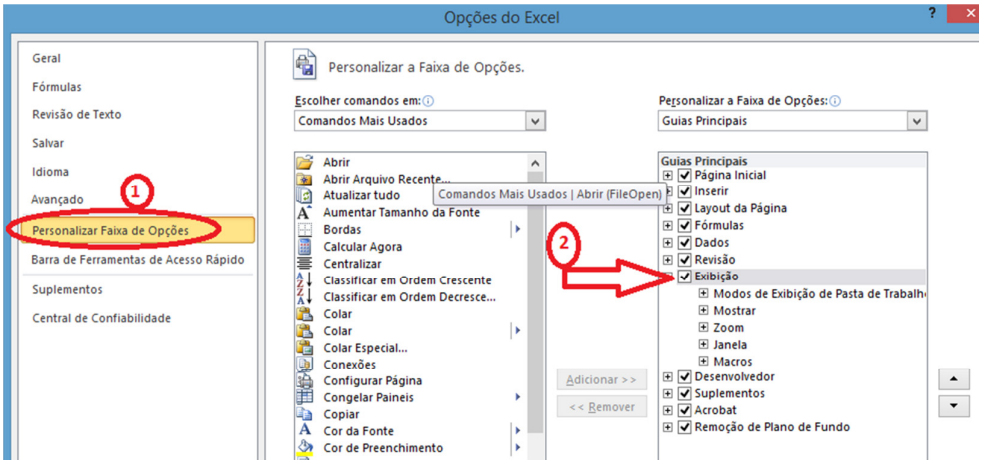
As variáveis constituem repositórios temporários de dados, podendo ser utilizadas para diversos fins. Quando se pretende atribuir valores a variáveis dever-se-á indicar o nome da variável, o operador "=" e o valor que se pretende que a variável armazene.

*<Nome_Variável> = <Valor>*

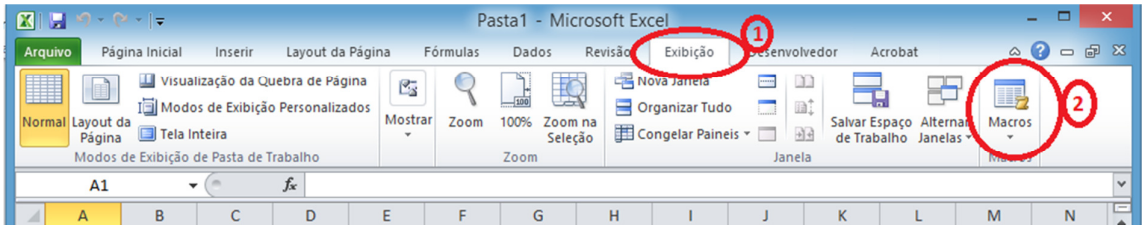
O nome da variável representa o conteúdo da mesma, isto é, sempre que mencionar o nome da variável é o seu conteúdo que será considerado.

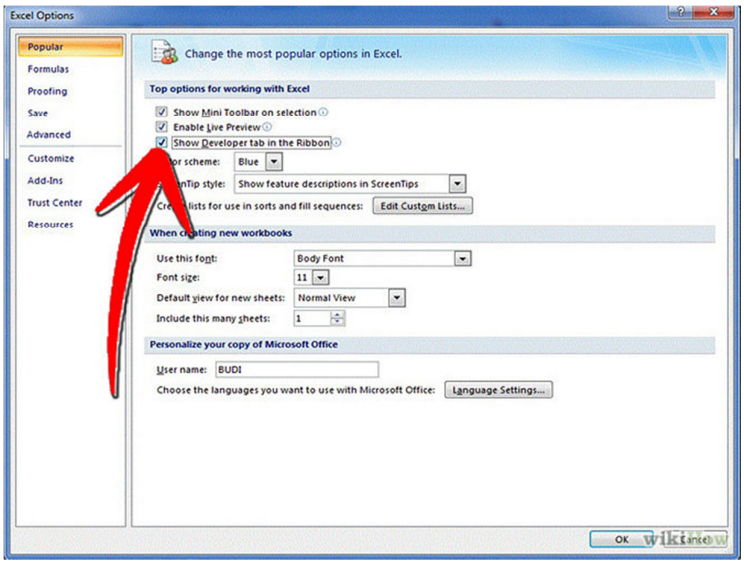
## 1.1 – Usando a Aba Desenvolvedor

Se a aba **Desenvolvedor** não estiver aparecendo, você pode adicioná-la da seguinte forma: inicialmente na aba **Arquivo** e então escolher no menu o item Opções e na janela que é exibida, clicar na opção **Personalizar faixa de opções** (ver na Figura o item 1). E na janela que vai aparecer, marcar a opção indicada pelo item 2.

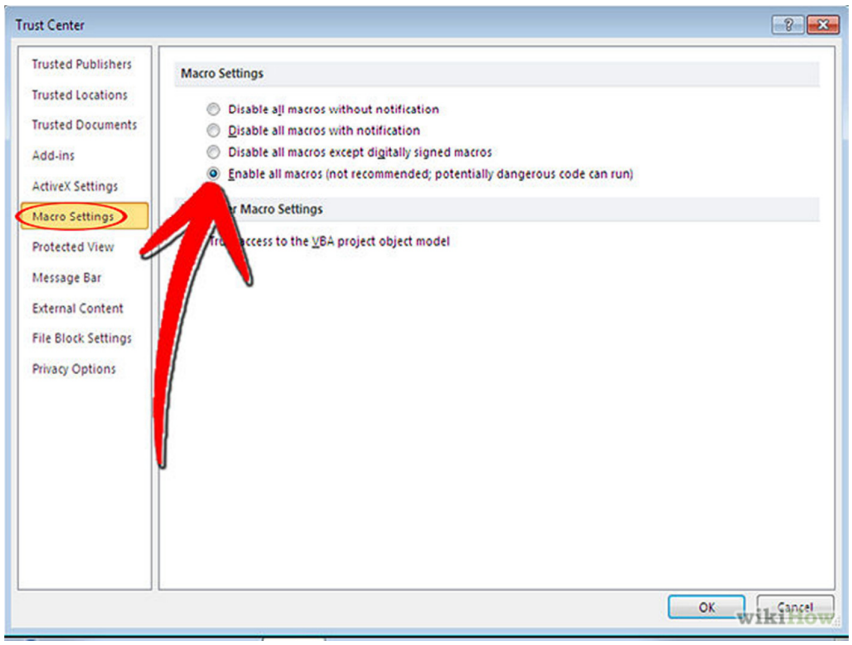



Outra opção é utilizar a aba Exibição (item 1 na Figura) e em seguida clicar no ícone Macro (item 2 na Figura).




Para o Excel 2007: clique no botão do Microsoft Office, em seguida Opções do Excel. Na categoria **Popular,** na seção **Mais Usados**, escolha **Mostrar guia desenvolvedor** na faixa de opções.

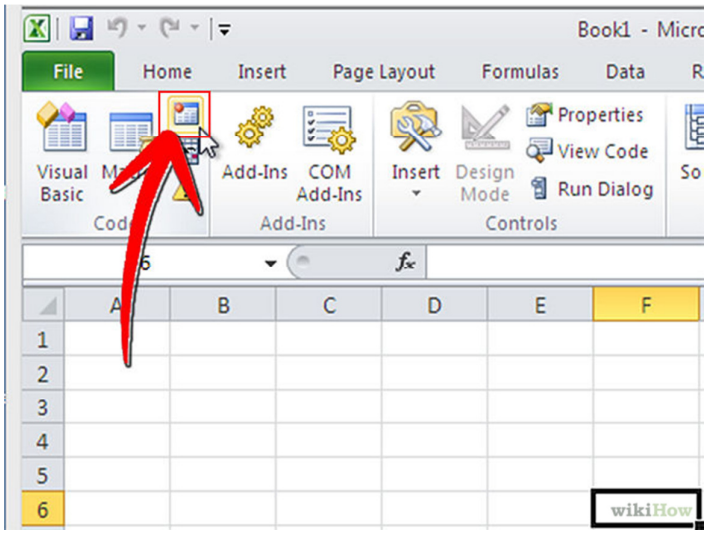

Trust Center
Trusted Publishers
Trusted Locations
Trusted Documents
Add-ins
ActiveX Settings
Macro Settings
Protected View
Message Bar
External Content
File Block Settings
Privacy Options
Macro Settings
Disable all macros without notification
Disable all macros with notification
Disable all macros except digitally signed macros
Enable all macros (not recommended; potentially dangerous code can run)
Macro Settings
access to the VBA project object model
OK
Cancel
wikiHow

## 2 – Configurando e Usando o Gravador de Macro

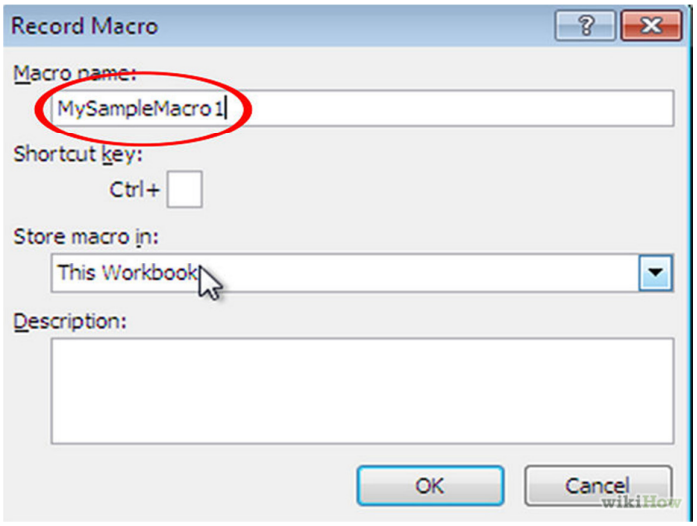
2.1 - **Clique na aba Desenvolvedor na parte de cima da tela.**

2.2 - **Mude as configurações de segurança.** No grupo Código da aba Desenvolvedor, clique em Segurança de Macro. Nas Configurações de macro, clique em Habilitar toas as macros e depois clique em OK.

Perceba que esta configuração de segurança não é ideal para o uso básico. Quando tiver terminado de criar sua macro, você pode voltar e desmarcar Habilitar todas as macros.

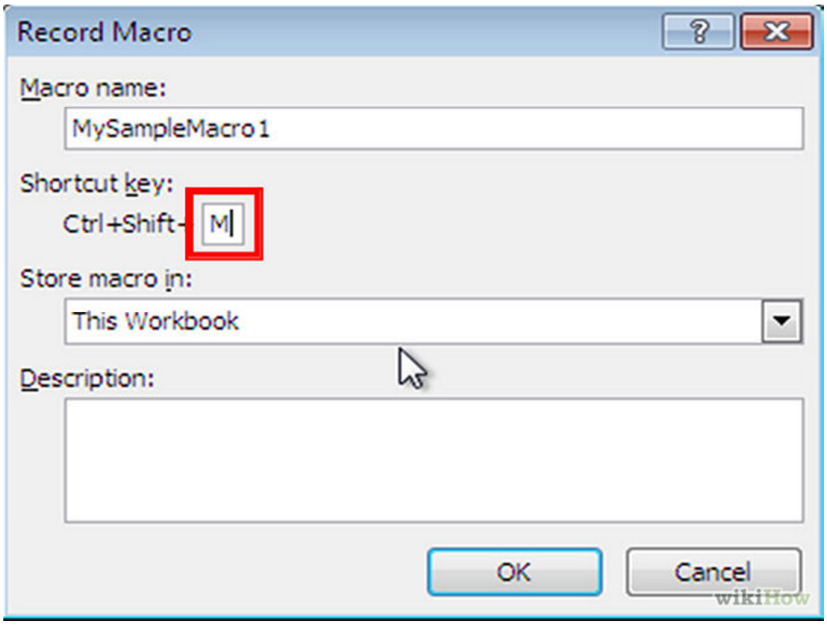



2.3 - **Clique em Gravar Macro.** Você irá encontrar esta opção na aba Desenvolvedor no grupo Código.

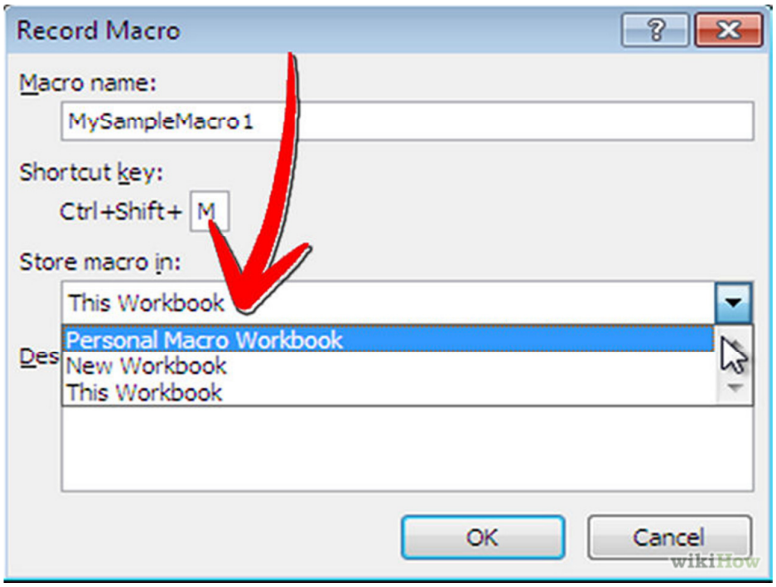



2.4 - **Dê um nome a sua macro.** O primeiro caractere da macro deve ser uma letra; depois disso você pode usar letras, números e sublinhas. Os nomes das macros não podem ter espaços.

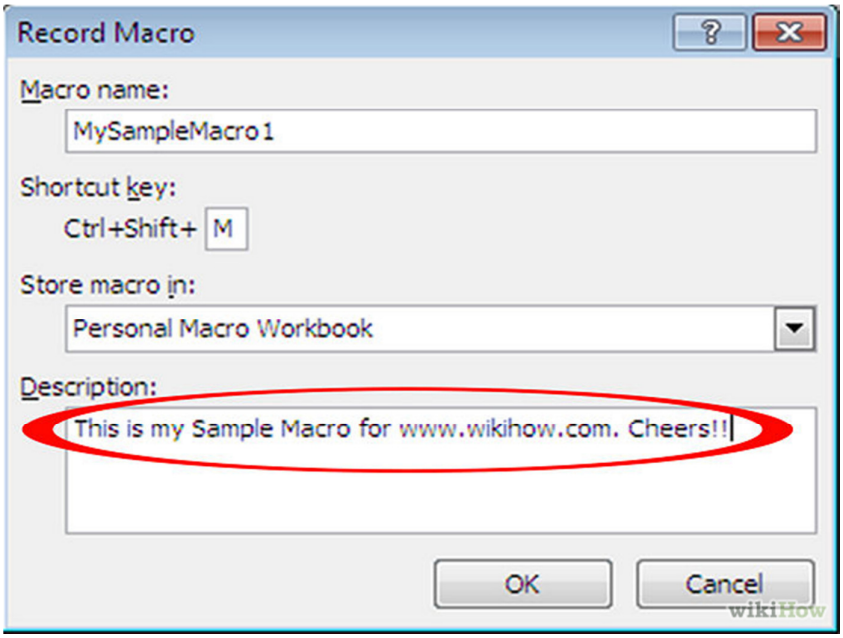
Não use o mesmo nome da sua macro que o nome de uma referência de célula existente.



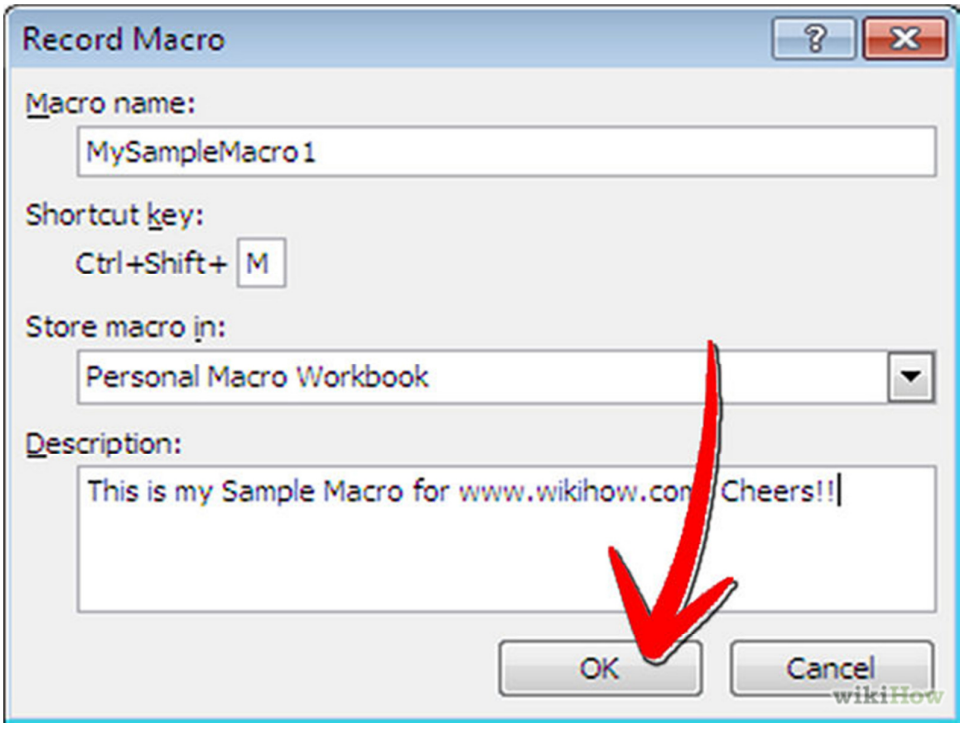

2.5 - **Crie um atalho para a sua macro (opcional).** Aperte uma letra na caixa de Tecla de Atalho. Uma tecla em minúscula irá ser traduzida como CTRL + letra; uma letra em maiúscula será interpretada como CTRL + ShIFT + letra.



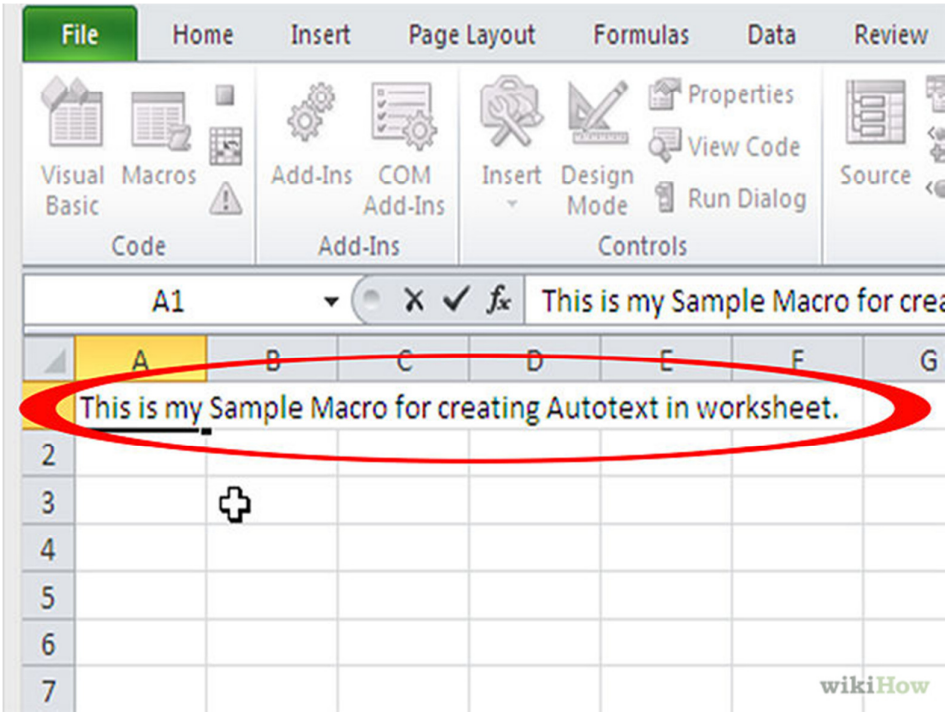

2.6 - **Escolha onde você irá guardar a sua macro.** Na lista Armazenar Macro em, escolha a pasta de trabalho onde você quer armazená-la. Se você quer que ela esteja disponível sempre que você executar o Excel, escolha Pasta de trabalho pessoal de macros.



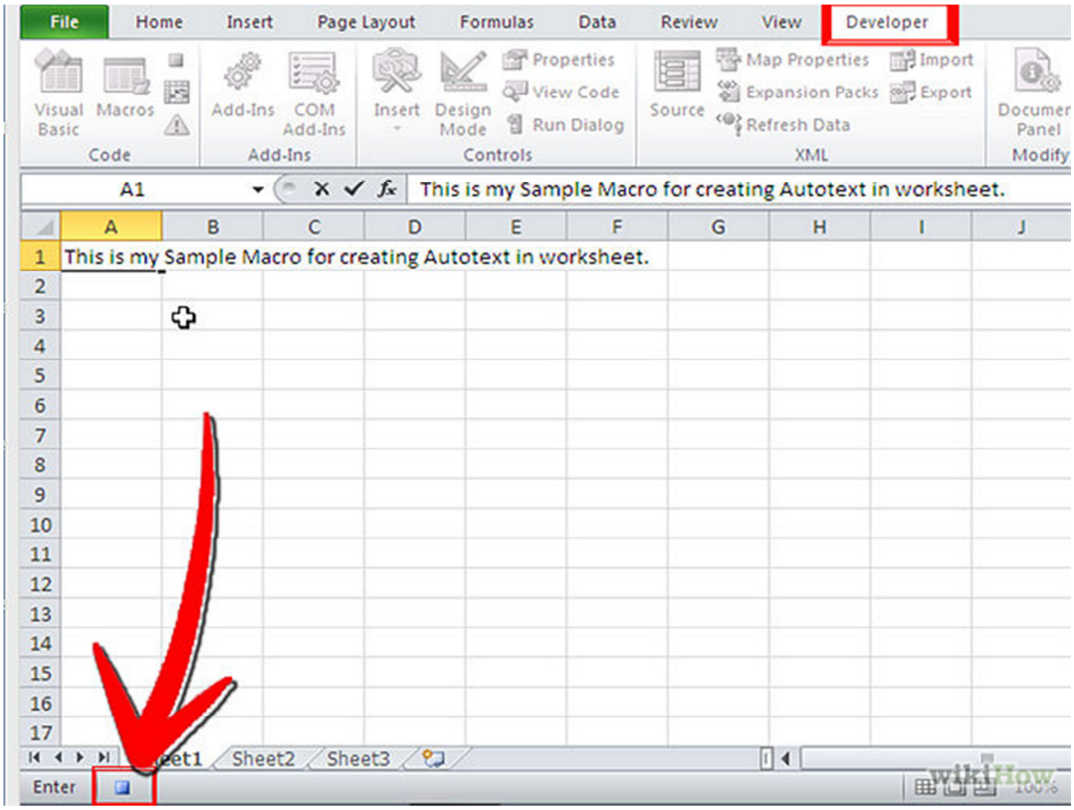

2.7 - **Descreva sua macro.** Escreva esta informação na caixa de Descrição.




2.8 - **Clique em OK para começar a gravar a sua macro.**

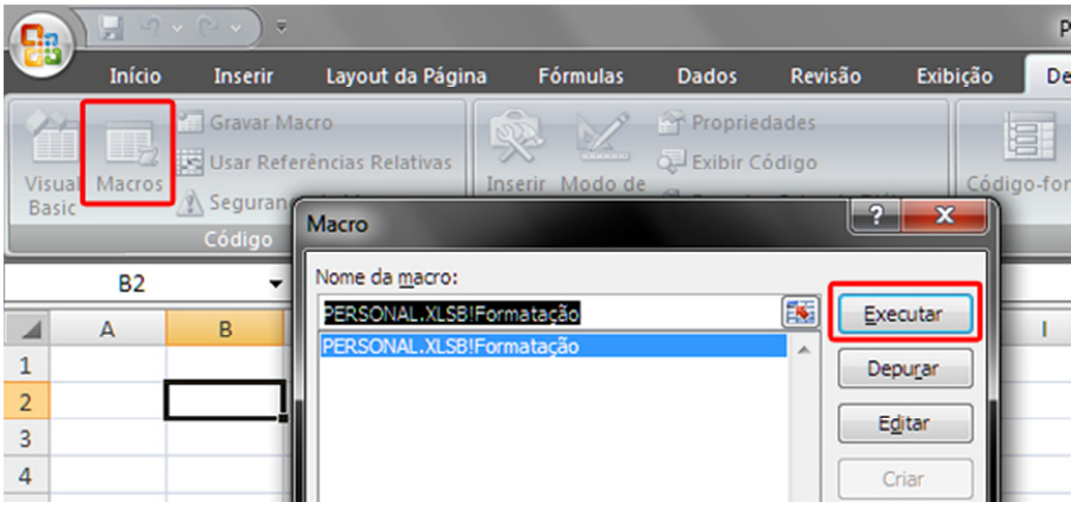



2.9 - **Faça as ações que você quer que a macro grave.**




2.10 - **Pare de gravar.** Vá até a aba Desenvolvedor, clique em Código e em seguida Parar Gravação.

2.11 - Para executar uma macro, clique em “Macros”, selecione a macro desejada (por isso é importante nomeá-las) e clique em “Executar”. Todas as ações gravadas serão repetidas na planilha que você desejar.




## 3 – Escrevendo uma Macro

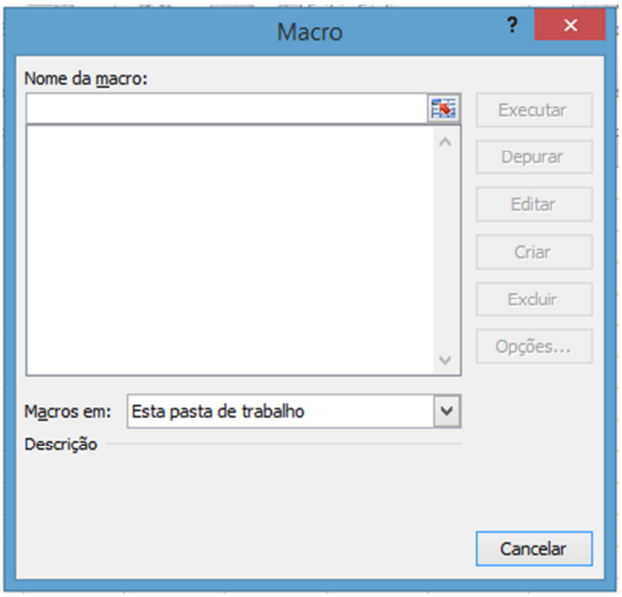
Se já existe alguma Macro criada, pode-se abrir o editor de macros para construir novas Macros, para isto, na aba **Desenvolvedor** clicar no ícone **Macros** e clicar no botão **Editar**, observe que abriu uma janela com o código fonte da primeira macro desenvolvida.

Vamos criar uma nova macro denominada Macro02 e deve ter como comando a sequencia de teclas ctrl+x. Clique no final do código existente e insira uma linha em branco e em seguida digite o código. Por exemplo, digite o código a seguir e depois selecione o menu **Arquivo** e em seguida a opção **Salvar**.

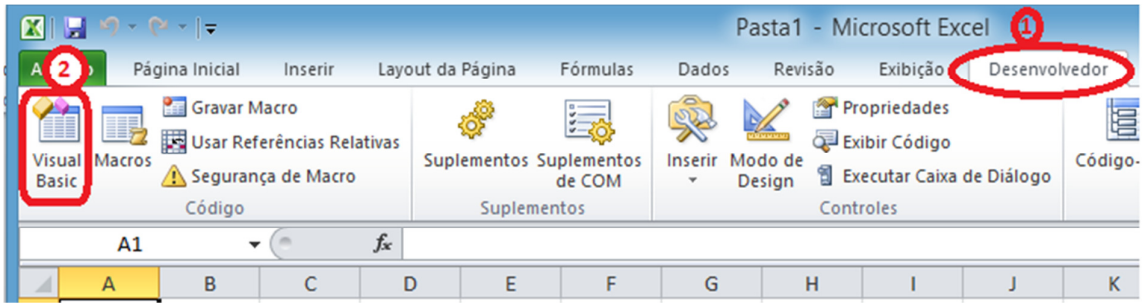
```
Sub Macro02()
    Rows(2).Insert
    Range("A2").Select
    ActiveCell.Value = "999"
End Sub
```

Para definir a sequencia de teclas que executa a macro, vá na aba **Desenvolvedor**, clique no ícone **Macro** e agora clique no nome da macro desejada (Macro02) e clique no botão **Opções**. Coloque a letra indicada e clique no botão **Ok** para salvar a alteração. Agora execute a macro.

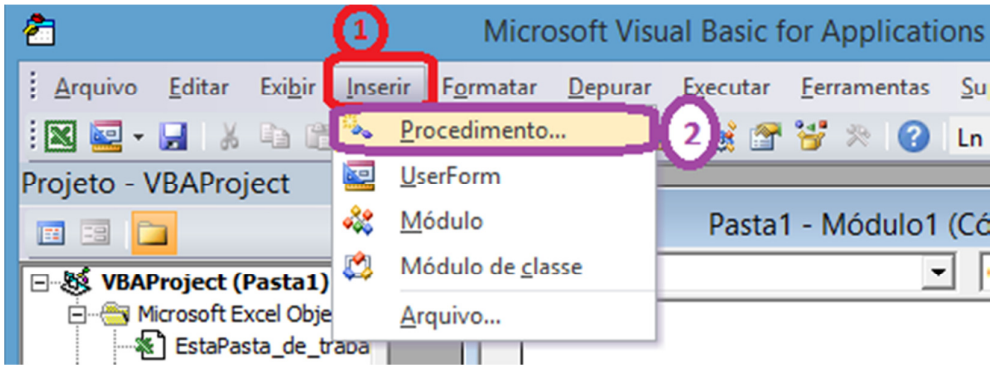
Se não existe nenhuma Macro codificada, ao clicar no ícone Macro, abrirá a janela com todas as opções desabilitadas. Neste caso, não é possível começar a codificação de um Macro, então primeiramente escreva o nome na caixa de texto para a nova macro e depois clique no ícone **Criar** que agora estará disponível.



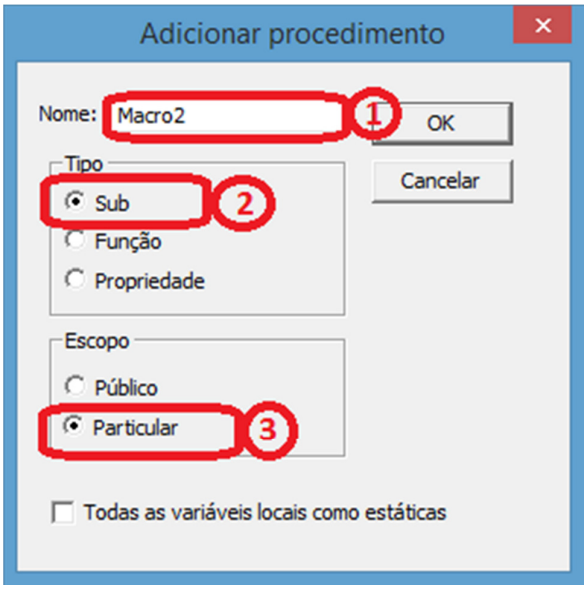

A opção é clicar no ícone **Visual Basic** (item 2 da figura) da aba **Desenvolvedor** (item 1) e escrever os comandos no editor do VBA.



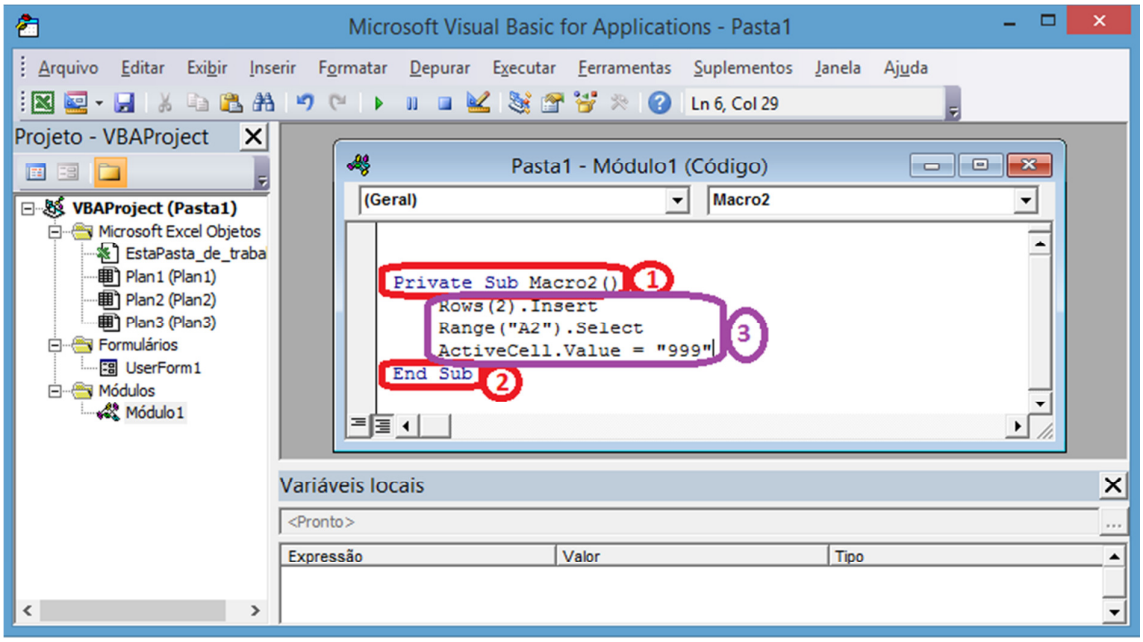

Vai aparecer uma janela para a inserção do código. Um caminho é usar a aba **Inserir** (item 1 da figura) e no menu escolher a opção **Procedimento** (item 2 da figura).




Vai abrir a janela do formulário **Adicionar procedimento** e então, preencha o campo **Nome** (item 1 da figura) com a informação do nome de sua Macro, escolha o tipo **Sub** (item 2) e escolha o Escopo como **Particular** (item 3) que define que ela será somente de uso restrito e então clique no botão **Ok**.




Vai abrir a janela para a programação, nesta janela já estará o início (item 1 da figura) e o final (item 2) da Macro e então é só digitar o corpo da Macro (ou seja os comandos que estão identificados com o numeral 3.



## 4 – Comandos

Um **método** é um termo do Visual Basic para uma palavra-chave que representa uma **ação** que você quer impor a um **objeto**. Por exemplo, Copiar, Recortar, e Colar são todos exemplos de métodos do Visual Basic. Como uma ação está associada com um objeto, o Visual Basic requer que você especifique o objeto, um ponto e daí então o método. Um sinal de igual não é usado. Na declaração *Range("B9").Select*

o **objeto** é a célula B9 e o **método** é a palavra-chave *Select*, que faz o ponteiro de célula mover-se para esta célula.

Exemplo:

```
Sub Teste1 ()
    Range("A2").Value= 2
    ThisWorkbook.Close Application.
    Quit
End Sub
```

Estas cinco linhas do exemplo constituem um **procedimento** (macro) chamado "Teste1". "Range("A2")", "ThisWorkbook" e "Application" são **objetos**, "Value" é uma **propriedade** do objeto e "Close" e "Quit" são **métodos**.

Este procedimento VBA será atribuído a um **objeto** botão (controle) e quando o usuário clicar nele (o evento) o **procedimento** VBA rodará (o **método** será aplicado).

## Alguns Comandos:

4.1 - ActiveCell.Value → atribui a célula ativa o valor indicado

*ActiveCell.Value = "99"* → atribui a célula ativa o valor 99

*varx = ActiveCell.Value* → atribui a variável varx o valor da célula ativa.

4.2 - ActiveCell.Offset(x, y).Activate → desloca a célula ativa em x linhas e y colunas.

*ActiveCell.Offset(1, 0).Activate* → desloca o foco para a célula abaixo da atual célula ativa.

4.3 – ActiveCell.Copy → copia o conteúdo da célula ativa para a memória;

4.4 – Selection.Font.Bold = True → coloca a fonte da célula em negrito;

4.5 - Selection.Font.Italic = True → coloca a fonte da célula em itálico;

4.6 - Selection.Font.Underline = xlUnderlineStyleSingle → coloca a fonte da célula sublinhada simples; (xlUnderlineStyleDouble) → sublinhada dupla

4.7 - Rows(x).Insert → insere uma nova linha na planilha na linha x indicada.

*Rows(2).Insert* → insere uma nova linha na posição 2.

4.8 - Selection.EntireColumn.Insert → insere uma coluna a esquerda da célula (coluna) ativa.

*Selection.EntireColumn.Delete* → Deleta a coluna ativa.

4.9 - Range("x:y").Select → seleciona um conjunto de células, iniciando na célula x e até a célula y

*Range("A2").Select* → selecionou a célula A2

*Range("A2:B4").Select* → selecionou o intervalo de células A2 até B4

4.10 - Now → atribui o valor da data e hora do sistema operacional.

*Range("A1") = Now* → atribui a célula A1 o valor atual da data e hora do sistema operacional.

4.11 – HorizontalAlignment = (xlCenter ou xlLeft ou xlRight) → define o alinhamento do conteúdo da célula), pode ser centralizado, a esquerda ou a direita. No exemplo abaixo, está centralizando as células selecionadas:

```
With Selection
  .HorizontalAlignment = xlCenter
End With
```

4.12 – Selection.Font → seleciona a fonte da célula ou seleção de células e depois pode trocar a fonte (.Name) ou trocar o tamanho da fonte (.Size). No exemplo abaixo, trocando a fonte e o tamanho:

```
With Selection.Font
  .Name = "Arial"
  .Size = 15
End With
```

4.13 - MsgBox "texto" → é um mecanismo de saída e permite ao utilizador visualizar as mensagens geradas pelo VBA.

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

*MsgBox "Macro desenvolvida"* → apresenta a mensagem “Macro desenvolvida” para o usuário.

4.14 - InputBox é uma função que permite ao usuário introduzir dados no programa – é portanto um mecanismo de entrada.

*InputBox(prompt[,title][,default][,xpos][,ypos][,helpfile,context])*

*Varx = InputBox(“Entre com o valor”)*

4.15 – For – Next → Executa uma determinada tarefa um determinado número de vezes.

*For counter = start To end [Step step] ... Next [counter]* → cria um laço iniciando em x e incrementando até y de 1 em 1. O exemplo abaixo, um laço iniciando em 1 e variando de 1 em 1 até 10 vai executar os comandos indicados.

```
For nx = 1 To 10 Step 1
      ActiveCell.Value = nx
      ActiveCell.Offset(1, 0).Activate
   Next nx
```

4.16 - If – Then – Else → Testa uma condição e executa um determinado conjunto de instruções dependendo do resultado dessa avaliação.

*If condition Then [statements] [Else elsestatements]* → teste lógico. Se a condição for verdadeira executa o(s) comando(s). Exemplos:

```
If val = 0 Then
  AlertLabel.ForeColor = "Red"
  AlertLabel.Font.Bold = True
  AlertLabel.Font.Italic = True
End If
```

```
If val = 0 Then
  AlertLabel.ForeColor = vbRed
  AlertLabel.Font.Bold = True
  AlertLabel.Font.Italic = True
Else
  AlertLabel.Forecolor = vbBlack
  AlertLabel.Font.Bold = False
 AlertLabel.Font.Italic = False
End If
```

4.17 - With – End With → Permite que você execute uma série de instruções sem precisar requalificar um objeto. Sintaxe:

```
With object
   [statements]
End With
```

Exemplo:

```
Sub teste6()
    With Worksheets("Plan1").Range("A1:C10")
      .Value = 30
      .Font.Bold = True
      .Interior.Color = RGB(255, 255, 0)
    End With
  End Sub
```

4.18 - Select – Case → Seleciona um dos segmentos de código a ser executado mediante a avaliação consecutiva de condições. Sintaxe:

```
Select Case testexpression
   [Case expressionlist-n
   [statements-n]] ...
   [Case Else
   [elsestatements]]
End Select
```

Exemplo de uma função que calcula o bônus dos funcionários em função de um valor fornecido pela variável performance.

```
Function Bonus(performance, salary)
  Select Case performance
    Case 1
      Bonus = salary * 0.1
    Case 2, 3
      Bonus = salary * 0.09
    Case 4 To 6
      Bonus = salary * 0.07
    Case Is > 8
      Bonus = 100
    Case Else
      Bonus = 0
  End Select
End Function
```

4.19 - While-Wend → Executa uma determinada tarefa enquanto que uma determinada condição permaneça verdadeira, isto é, com o valor *True*. Sintaxe:

```
While condition
   [statements]
Wend
```

Exemplo de um procedimento que imprime uma mensagem após contar até 20.

```
Dim Counter
Counter = 0 ' Inicialize variável.
While Counter < 20 ' Teste valor de Contador.
     Counter = Counter + 1 ' Incremente contador.
Wend ' Encerre o loop While quando Counter > 19.
MsgBox ("Imprima 20 na janela" & Counter)
```

4.20 – GoTo → é uma declaração que transfere o controle para a localização especificada denominada Rótulo. Um rótulo consiste de um nome seguido por dois pontos. Sintaxe:

```
GoTo segue
   [statements1]
Segue:
   [statements2]
```

Exemplo:

```
Resp = MsgBox("Continuar?", vbYesNo)
If Resp = vbYes Then
   GoTo vai
End If
....
vai:
MsgBox("novos comandos")
```

4.21 – Format → Retorna uma expressão formatada de acordo com instruções contidas em uma expressão de formato. Sintaxe:

***Format(***expression*[,*format *[,*firstdayofweek *[,*firstweekofyear*]]])***

Onde: *expression.* → Obrigatória. Qualquer expressão válida.

*Format* → Opcional. Uma expressão de formato nomeada ou definida pelo usuário válida.

Firstdayofweek → Opcional. Uma constante que especifica o primeiro dia da semana. Valores: vbUseSystem (valor = 0), vbSunday (1), vbMonday (2), vbTuesday (3), vbWednesday (4), vbThursday (5), vbFriday (6), vbSaturday (7).

Firstweekofyear → Opcional. Uma constante que especifica a primeira semana do ano. Valores: vbUseSystem (valor = 0), vbFirstJan1 (1) ← Inicia na semana em que 1º de janeiro cair (padrão), vbFirstFourDays (2) ← Inicia na primeira semana do ano a ter pelo menos quatro dias, vbFirstFullWeek (3) ← Inicia na primeira semana completa do ano.

Exemplos:

*varx = Format(MyTime, "h:m:s")* → Retorna: "17:4:23".

*varx = Format(MyDate, "dddd, mmm d yyyy")* → Retorna: "Wednesday, Jan 27 1993".

*varx = Format(5459.4, "##,#0.00*") → Retorna: "5,459.40".

*varx = Format(5, "0.00%*") → Retorna: "500.00%".

## Exemplos:

Macro que mostra o uso do teste (if) e Box de mensagem

```
Sub teste3()
    Dim iResposta As Integer
    'mostrar uma mensagem
    iResposta = MsgBox("Selecione um botão", vbYesNoCancel)
    If iResposta = vbCancel Then
      MsgBox "Você selecionou Cancelar"
    End If
    ' Teste do Sim
    If iResposta = vbYes Then
      MsgBox "Você selecionou Sim"
    End If
    ' Teste do Não
    If iResposta = vbNo Then
      MsgBox "Você selecionou Não"
    End If
End Sub
```

Macro que mostra o uso do with que é usado para substituir parte do objeto comum e o uso do selection

```
Sub teste4()
     Selection.Font.Bold = True
     Selection.Font.Italic = True
     Selection.Font.Underline = xlUnderlineStyleDouble
     With Selection
     .HorizontalAlignment = xlCenter
     End With
     With Selection.Font
      .Name = "Arial"
      .Size = 15
     End With
End Sub
```

Macro que mostra o uso do InputBox e MsgBox

Uso do InputBox e MsgBox

```
Sub teste5()
    Varx = InputBox("Entre com o valor")
    MsgBox ("teste" & Varx)
End Sub
```

Macro que mostra o uso de vários comandos (InputBox, MsgBox, For/Next, GoTo)

Uso do InputBox e MsgBox

```
Sub teste7()
    novoInicio:
    Varx = InputBox("Entre com quantidade de termos")
    vary = ActiveCell.Value
    MsgBox ("Valor da célula: " & vary)
    For nx = 1 To Varx Step 1
       ActiveCell.Offset(1, 0).Activate
       ActiveCell.Value = vary * nx
    Next nx
    iResposta = MsgBox("Continuar", vbYesNoCancel)
    If iResposta = vbCancel Then
      MsgBox "Você selecionou Cancelar!"
    End If
    ' Teste do Sim
    If iResposta = vbYes Then
      GoTo novoInicio
    End If
    ' Teste do Não
    If iResposta = vbNo Then
      MsgBox "Você selecionou Não!"
    End If
End Sub
```

## Referencias

AIOSA, Rodrigo. **O que é uma macro?** Escolaexcel. 04 dez. 2010. Disponível em <http://www.escolaexcel.com.br/2010/12/macros.html>, acesso em 09 nov. 2015.

BERTOLO, L. A. **Lições de VBA do Excel**. Disponível em < http://www.bertolo.pro.br/FinEst/SemanaContabeis2007/MacroExcel.pdf>, acesso em 09 nov. 2015.

CARDOSO, Ramon. **Como criar macros no Excel?** Techtudo. 05 mai. 2012. Disponível em <http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2012/05/como-criar-macros-no-excel.html>, acesso em 09 nov. 2015.

SANTOS, C. F. **Planilha Eletrônica** (Excel). UFLA. 2011. Disponível em <http://www.apostilando/download/3358_excel_2010.pdf>, acesso em 09 nov. 2015.

WIKIHOW. **Como escrever uma macro simples no Microsoft Excel**. Disponível em <http://pt.wikihow.com/Escrever-uma-Macro-Simples-no-Microsoft-Excel>, acesso em 09 nov. 2015.

**Observação:**

Esta apostila foi desenvolvida a partir de textos consultados na internet e que estão indicados nas referencias e em exemplos e experiências desenvolvidas pelo autor.

Ela é somente um guia inicial e o aluno deve consultar os livros indicados no Plano de Aula da disciplina.

Versão 23/11/2015