

Disciplina: Programação em microinformática

VBA com Word / Excel / Access

Office 2007

(Versão 3.1)

Prof. Hamilton Martins Viana

Esta apostila pode ser encontrada no site:
<http://sites.google.com/site/hamiltonmv>

Página Assunto

03	Primeira parte - Word – Aula 1. Criação de uma macro
07	O ambiente VBA
08	Exercício sobre macros
13	Aula 2 de programação com Word - Uso de formulários
17	Segunda parte - Excel – Aula 1
19	Criação de macros no Excel sem uso do gravador de macros
20	Uso de formulários em Excel
24	Uso de tratadores de evento em VBA com Excel
26	Aula 2 de programação com Excel
26	Criação e uso de funções - a função SaldoEstoque
31	Aula 3 de programação com Excel
32	Exercícios VBA com Excel
33	Terceira parte - Access – Aula 1
34	Uso de formulários com Access
38	Exercícios para aula 1 com Access
41	Aula 2 de Access
41	Inclusão de botões no formulário do Access:
44	Uso de DoCmd
45	Exercícios Access Aula 2
49	Aula 3 de Access
54	Aula 4 de Access
54	Uso de relatórios com Access
61	Aula 5 de Access
65	Automatização de tarefas
71	Projeto final da disciplina.
75	Complemento - Alguns programas do projeto Access e dos exercícios de script.
80	Bibliografia

Disciplina Programação em Microinformática (Prg Micro)

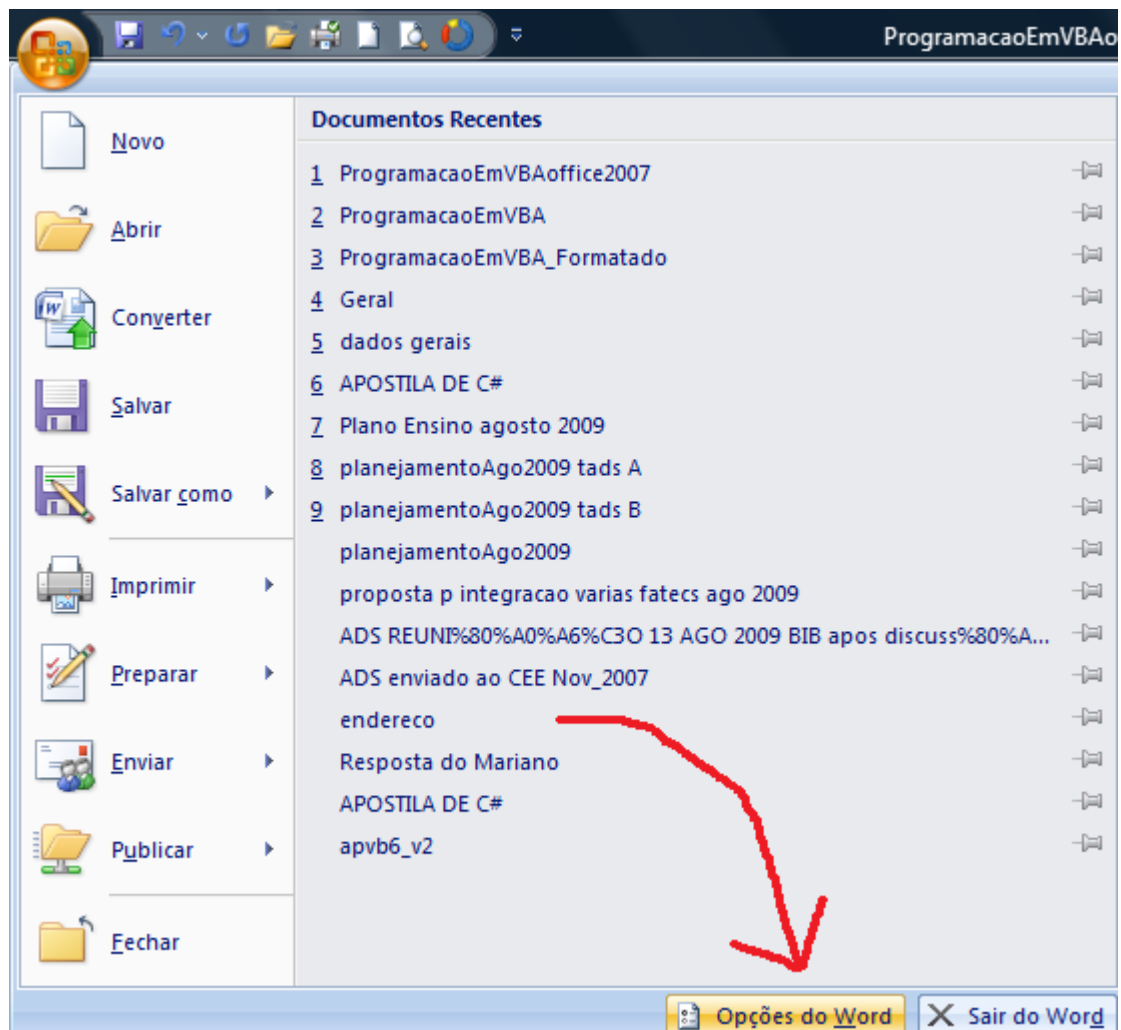
Segundo a Microsoft, foram vendidas mais de 120 milhões de licenças do “Office-2007”. O pacote “Office” é composto de quatro aplicativos (Power Point, Word, Excel e Access). Apesar de serem os programas mais utilizados em todo o mundo, poucos conhecem e usam seus recursos mais avançados que podem ser utilizados através da programação com VBA (Visual Basic for Applications).

Primeira Parte – Word – Aula 1

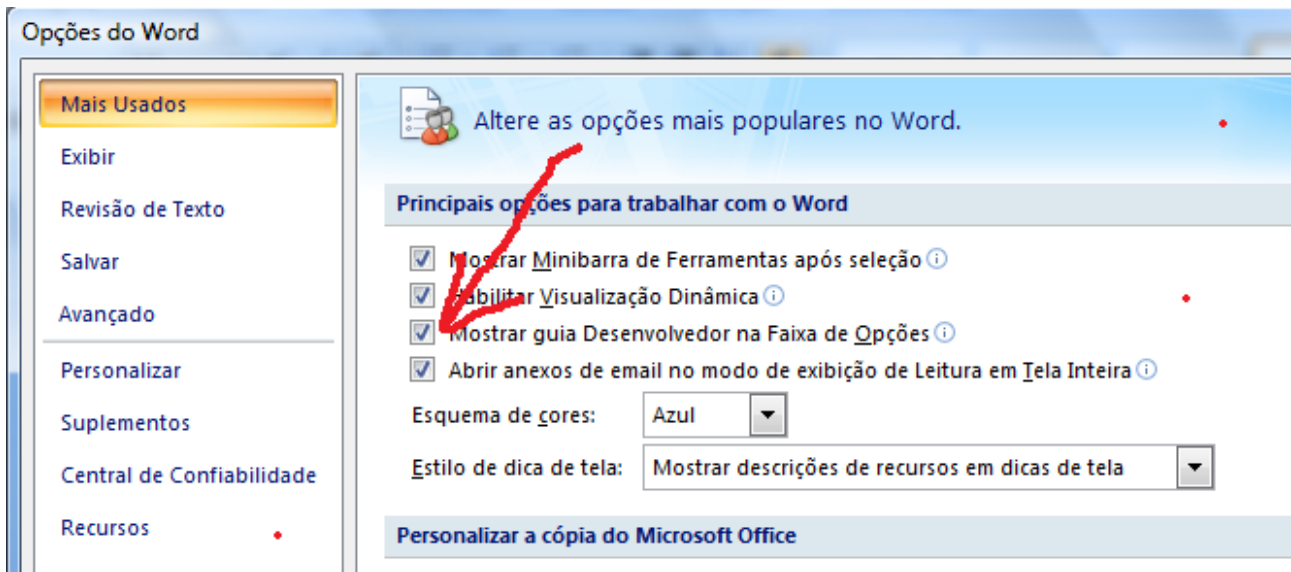
No Word, a programação é feita através de Macros, da seguinte maneira:

Criação de uma macro

Para criar uma nova macro deverá estar disponível a guia “Desenvolvedor”, que fica visível clicando-se no botão “Office” e em seguida no botão “Opções do Word”. Veja as janelas abaixo:



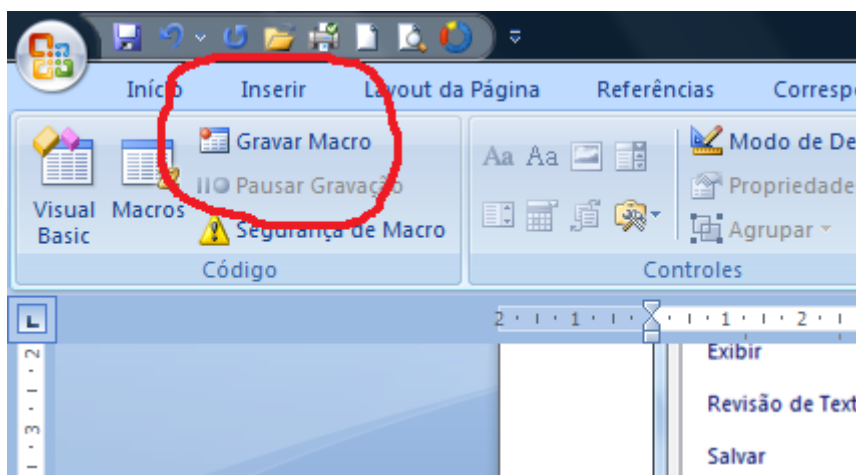
Em seguida dê um click em “Mais Usados” e ative a caixa “Mostrar guia Desenvolvedor na Faixa de Opções”.
Veja a janela a seguir:



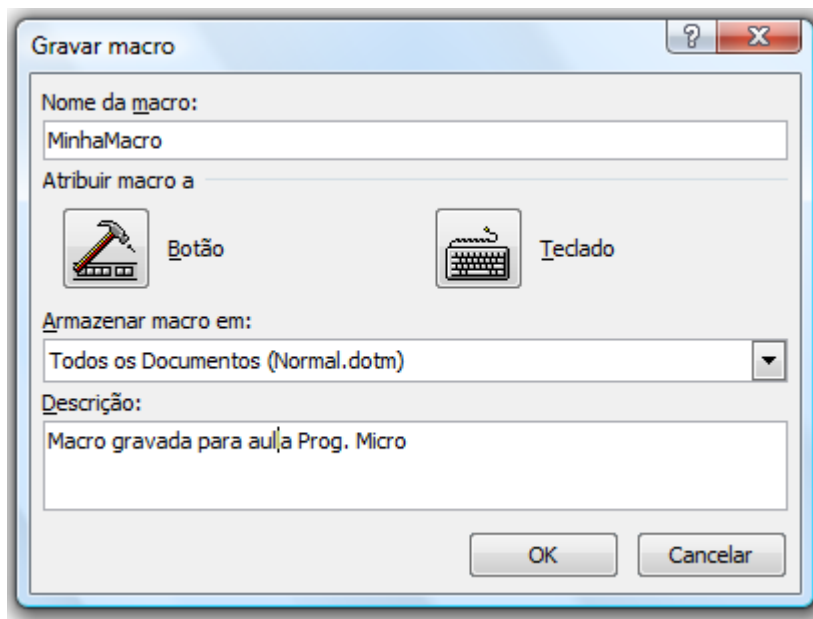
Click em Ok p/ fechar as janelas e note que agora está disponível a guia “Desenvolvedor”, que utilizaremos em nossos trabalhos.

Uma macro contém um conjunto de instruções que serão processadas passo a passo, quando a macro for executada.

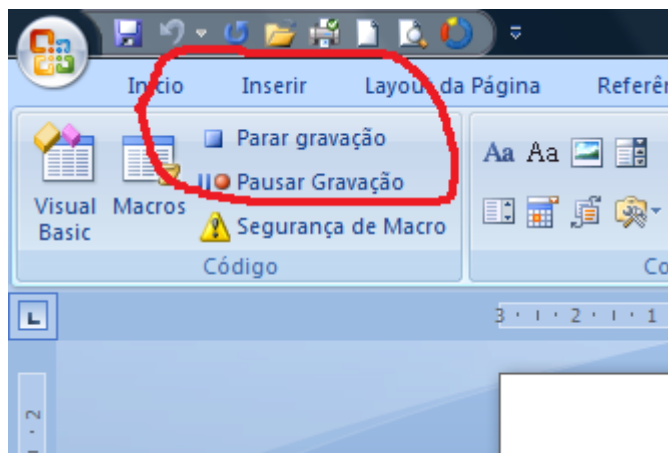
Para criar uma nova macro, dê um click na guia Desenvolvedor e um click no item “Gravar Macro”:



Na janela que abrir, escolha o nome da nova macro (pode ser por exemplo, “MinhaMacro”) e escolha o seu documento, na caixa “Armazenar macro em:”. Se você escolher “Todos os documentos”, a macro ficará gravada no Word que está instalado no micro. Faremos isso e ao final desta lição, nós a apagaremos. A janela para gravação da macro, ficará como o exemplo abaixo:



Dê um click no botão OK. Note que o apontador do mouse ficou um pouco diferente. A partir desse momento todas as ações que você executar estarão sendo gravadas, até que você pare a gravação, dando um click no item retangular da janela “Parar gravação”

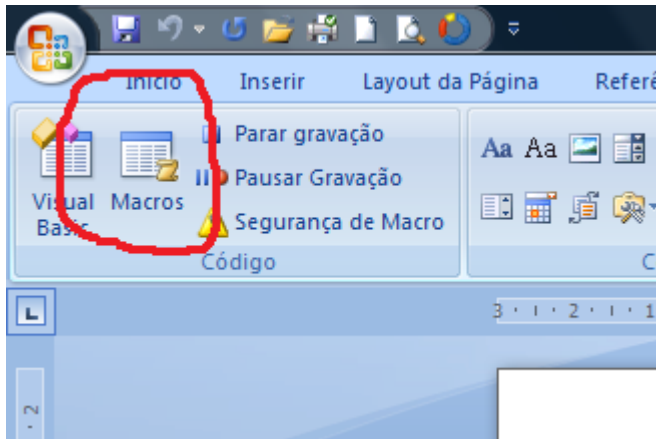


Antes de parar a gravação vamos realizar algumas ações que ficarão gravadas em nossa macro. Digite por exemplo seu nome, como por exemplo: “Ricardão, o Bãããã!!!”, formate-o com tamanho 16, tachado e cor vermelha. Pare a gravação da macro.

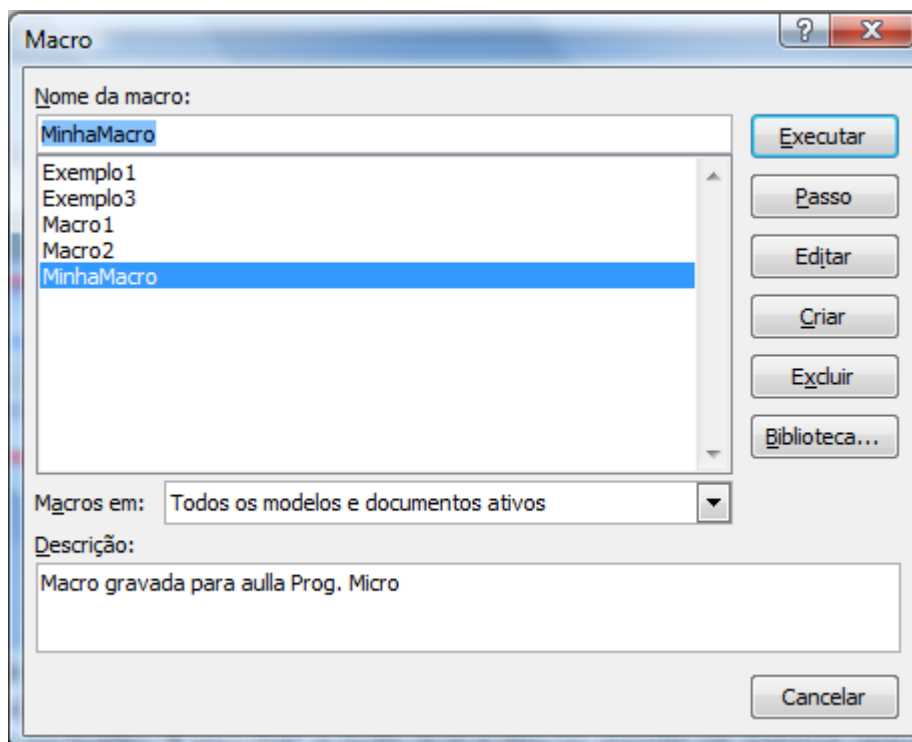
Obs: Como está sendo utilizado o gravador de macros, a seleção com o mouse não é possível. Para selecionar o texto, use a tecla Shift + as setas de direção.

Pronto podemos parar agora a gravação da macro. A partir de agora, toda vez que a macro for executada, repetirá as operações que você realizou. Vamos testar a macro MinhaMacro:

- Apague o texto impresso pela macro em seu documento.
- Dê um click para acionar o item Macros, na guia Desenvolvedor e na janela que surgir, selecione a macro que quer executar.



- Selecione a macro desejada (neste caso é a macro MinhaMacro, o nome que você escolheu).
- Dê um click no botão “Executar” e note que a macro repete os passos gravados.



Inclusão de uma macro na Barra de Ferramentas de Acesso Rápido.

Vamos automatizar um pouco mais esse processo.

Podemos atribuir nossa nova macro à “Barra de Ferramentas de Acesso Rápido” para podermos executá-la com mais comodidade.

Para isso, dê um click no botão Office e em Opções do Word.

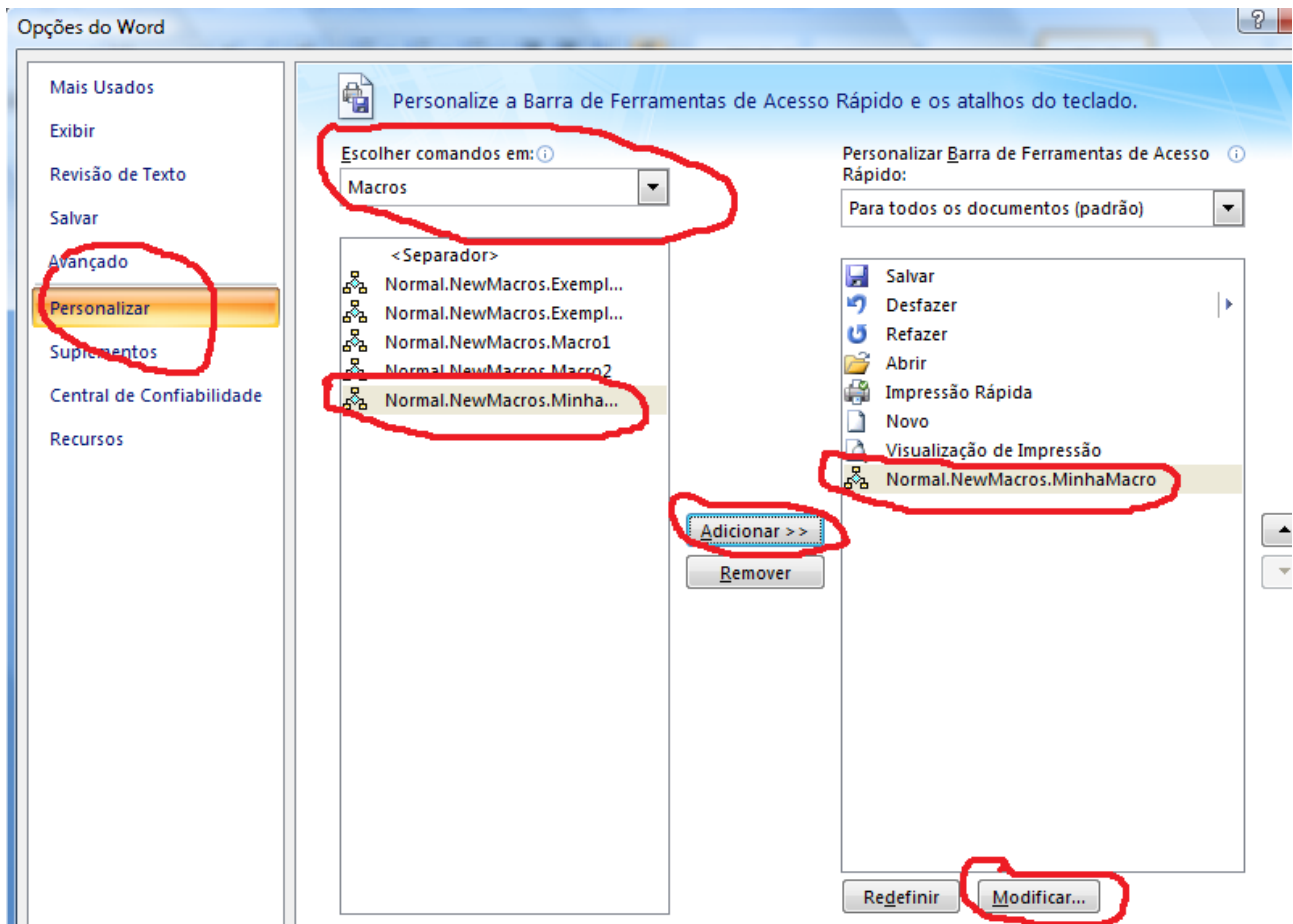
Na janela que abrir, na caixa da esquerda, click em “Personalizar”

No item “Escolher comandos em:” selecione “Macros” e em seguida escolha a macro desejada (neste caso é “MinhaMacro”). Click em “Adicionar”.

Na caixa da direita, selecione a macro desejada e click em “Modificar”

Escolha o ícone mais adequado e mude o “Nome para exibição”. Pronto! Dê um click em Ok e veja o botão p/ acionar a macro na Barra de Ferramentas de Acesso Rápido. Quando terminar, você pode utilizar a macro, com apenas um click do mouse.

Abaixo há um exemplo da janela “Opções do Word” com os itens a serem configurados:



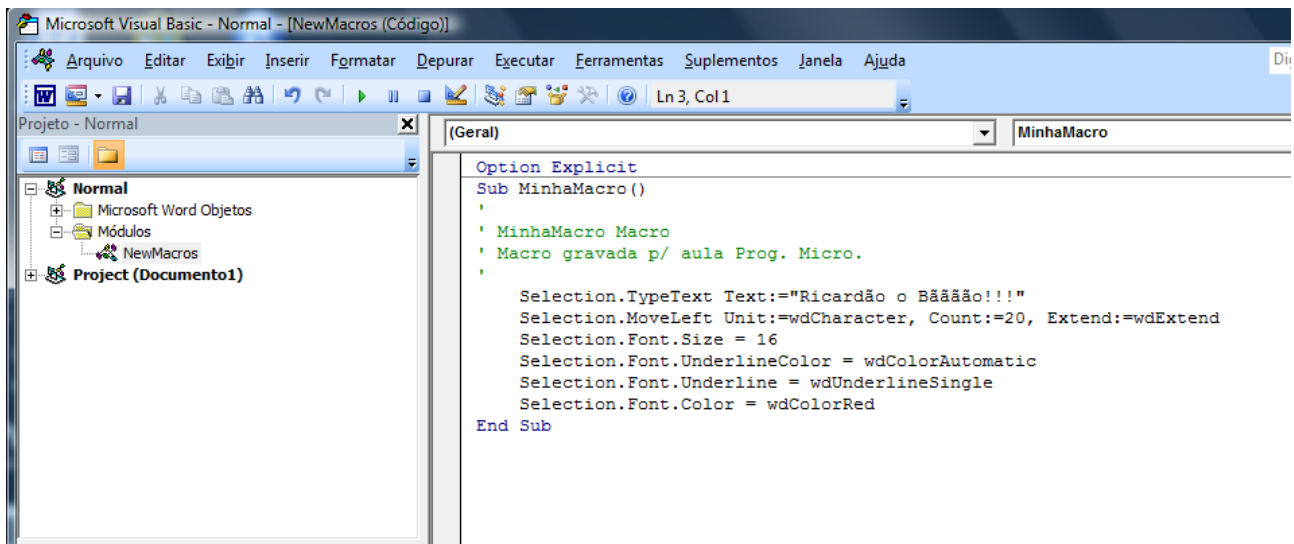
Obs: Dê um click com o botão direito do mouse na Barra de Acesso Rápido e veja as opções.

Programação usando o VBA (Visual Basic for Applications):

Vamos agora verificar a parte mais poderosa do uso de macros através do VBA.

Para isso, selecione a guia Desenvolvedor e dê um click no item “Macros”. Selecione a macro que interessa e dê um click sobre o botão “Editar”. Nesse momento o Word abre o ambiente do VBA, que é muito parecido com o ambiente do Visual Basic 6.0.

O ambiente VBA.



Através desse ambiente, podemos fazer a programação que quisermos.

Note que as frases em cor verde são comentários (iniciam por apóstrofe) sendo utilizados apenas para documentação.

Obs.: O gravador de macros inclui, na forma de instruções, todas as ações que executamos quando criamos a macro.

Podemos modificar diretamente as instruções ou incluir novas instruções mudando o comportamento da macro.

Exercício sobre macros

EXERCÍCIOS

Inclua manualmente na macro, as instruções abaixo:

```
Selection.MoveRight
Selection.Font.Color = wdColorBlack
Selection.TypeParagraph
Selection.TypeParagraph
Selection.Font.Size = 12
```

Salve as modificações efetuadas: Arquivo / Salvar normal

Nossa macro agora ficará com a seguinte aparência:

```
Sub MinhaMacro()
'
' MinhaMacro Macro
' Macro gravada p/ aula Prog. Micro.
'

Selection.TypeText Text:="Ricardão o Bãããã!!!"
Selection.MoveLeft Unit:=wdCharacter, Count:=20, Extend:=wdExtend
Selection.Font.Size = 16
Selection.Font.UnderlineColor = wdColorAutomatic
Selection.Font.Underline = wdUnderlineSingle
```



```
Selection.Font.Color = wdColorRed
```

'Inclusões manuais, sem uso do gravador de macros.

```
Selection.MoveRight  
Selection.Font.Color = wdColorBlack  
Selection.TypeParagraph  
Selection.TypeParagraph  
Selection.Font.Size = 12
```

End Sub

Execute-a, verifique como funciona, analise as instruções da macro e explique o funcionamento.

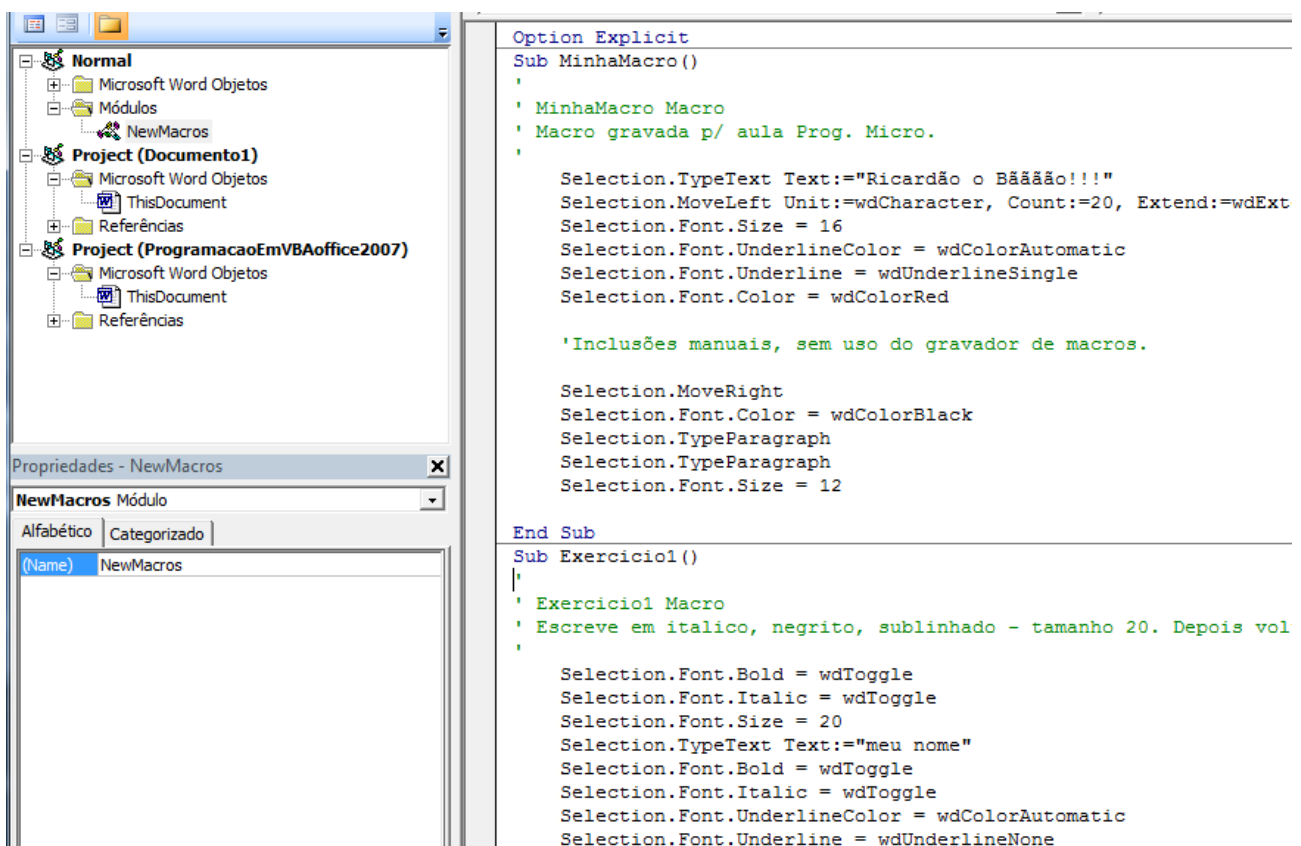
Obs.: Estando no ambiente do VBA você pode obter ajuda sobre quaisquer instruções ou objetos que desejar. Para isso, selecione a instrução ou objeto e pressione a tecla F1.

Por exemplo, selecione a palavra *MoveRight* e pressione F1.

Obs.: Para alternar entre a janela do VB e a janela do Word, pressione Alt + F11.

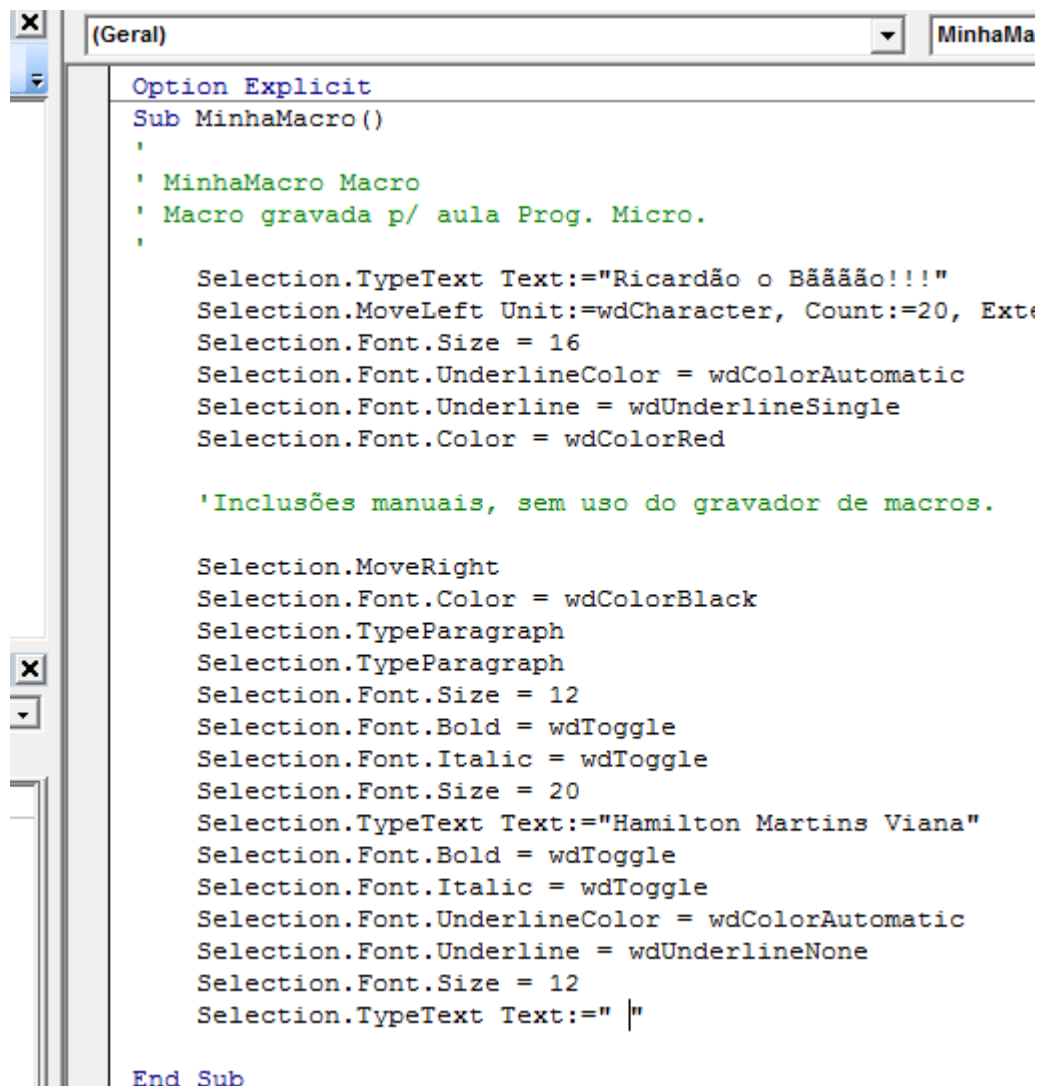
Exercício: Altere manualmente a macro para que, após exibir o texto anterior, escreva seu nome em tamanho 20, em negrito, itálico e sublinhado. Após isso, o texto deve voltar ao normal e com tamanho do fonte 12.

Sugestão: Crie uma nova macro p/ ver como se faz negrito, itálico e sublinhado, copie o código na macro MinhaMacro e em seguida, apague a macro de exemplo. Veja abaixo:



Obs.: Note que as macros são para o VB, apenas PROCEDIMENTOS, dentro da pasta “Módulos”

Após a cópia e adequação das instruções de MinhaMacro, esta deverá estar mais ou menos assim:

The image shows a screenshot of the Microsoft Visual Basic for Applications (VBA) editor. The window title is '(Geral) - MinhaMa'. The code editor contains the following VBA code:

```
Option Explicit
Sub MinhaMacro()
'
' MinhaMacro Macro
' Macro gravada p/ aula Prog. Micro.
'
    Selection.TypeText Text:="Ricardão o Bãããã!!!"
    Selection.MoveLeft Unit:=wdCharacter, Count:=20, Ext:=wdMoveFromEnd
    Selection.Font.Size = 16
    Selection.Font.UnderlineColor = wdColorAutomatic
    Selection.Font.Underline = wdUnderlineSingle
    Selection.Font.Color = wdColorRed

    'Inclusões manuais, sem uso do gravador de macros.

    Selection.MoveRight
    Selection.Font.Color = wdColorBlack
    Selection.TypeParagraph
    Selection.TypeParagraph
    Selection.Font.Size = 12
    Selection.Font.Bold = wdToggle
    Selection.Font.Italic = wdToggle
    Selection.Font.Size = 20
    Selection.TypeText Text:="Hamilton Martins Viana"
    Selection.Font.Bold = wdToggle
    Selection.Font.Italic = wdToggle
    Selection.Font.UnderlineColor = wdColorAutomatic
    Selection.Font.Underline = wdUnderlineNone
    Selection.Font.Size = 12
    Selection.TypeText Text:=" |"

End Sub
```

Exercício : Suponhamos que desejamos acionar a calculadora do Windows.

Crie uma nova macro chamada Calculadora, coloque um botão na “Barra de Ferramentas de Acesso Rápido”, com o ícone de uma calculadora, sendo que quando esse botão for acionado, é executado o programa “calc.exe”.

Passos:

- Guia Desenvolvedor / Gravar Macro
- Nome da macro = Calculadora
- Armazenar nova macro em = Todos os Documentos
- Descrição : Chama a calculadora do Windows
- Botão / OK – configurar o botão.
- Parar gravação

Em seguida, vamos editar manualmente a macro Calculadora:

Itens:

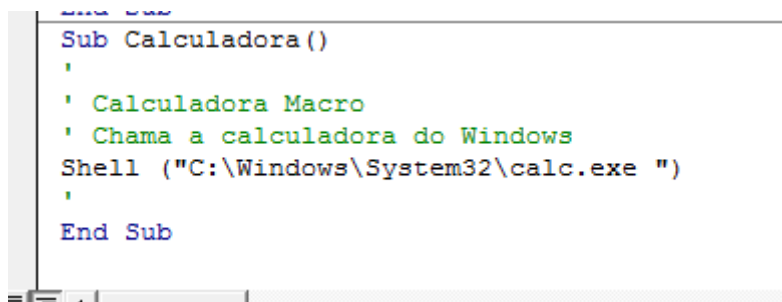
- Guia Desenvolvedor / Macros / Editar

Digitar na Sub Calculadora(): Shell ("C:\Windows\System32\calc.exe ")

Obs.:

- a) A instrução Shell(" caminho / programa ") executa programas externos
- b) A instrução acima executa o programa calc.exe que está na pasta c:\windows.

Nossa Procedure ficará assim:



```
Sub Calculadora()  
'  
' Calculadora Macro  
' Chama a calculadora do Windows  
Shell ("C:\Windows\System32\calc.exe ")  
'  
End Sub
```

Obs.: Caso a Calculadora não esteja na pasta indicada acima, localize-a e corrija o programa.

- Voltar ao ambiente do Word (alt + F11).
- Testar a macro.

Observações importantes:

- a) Todos os exercícios serão apresentados ao professor, para avaliação.
- b) Ao salvar o documento, escolha salvar com o tipo "Documento Habilitado para Macro do Word (*.docm)". Se isso não for feito, todas as macros que você fez serão perdidas.

Exercícios:

1-Fazer uma macro de nome "TesteInputOutput" que utilize:

- variáveis - é a maneira que manipulamos valores em programação.
- inputbox – é uma função do VBA que nos permite a entrada de dados em um programa.
- msgbox – é uma função do VBA que nos permite exibir mensagens para o usuário.

Essa macro deverá declarar uma variável do tipo string que receberá uma frase digitada pelo usuário.

Em seguida, será exibida uma caixa de mensagem com a frase digitada. Quando for dado o click no botão Ok da caixa de mensagem, a frase deverá ser enviada como texto para o documento do Word onde se executou a macro.

A codificação da macro será mais ou menos como a seguir:

```
Sub TesteInputOutput()  
,  
' TesteInputOutput Macro  
' Abre InputBox, exibe MsgBox e põe string no documento  
,  
  
    Dim frase As String  
  
    frase = InputBox("Digite uma frase para o documento do Word:")  
  
    MsgBox "A frase digitada foi: " + frase, vbExclamation, "A T E N Ç Ã O"  
  
    Selection.TypeText Text:="A frase digitada na inputbox foi : " + frase  
End Sub
```

Complementação:

- a) Na msgbox a frase digitada deverá aparecer na segunda linha da caixa de mensagem - use chr\$(13) para concatenação de strings e mudar a linha.
- b) O string a exibir no documento do Word deve ser no tipo de fonte "Times New Roman" (Selection.Font.Name="Times New Roman"), O tamanho deve ser 16, negrito, itálico, azul.
- c) Após a exibição, o tamanho deve ser 12, preto, não itálico não negrito.

2-Fazer uma macro que quando é executada, exibe uma "InputBox" solicitando o texto para o cabeçalho do documento do Word. Se nada for digitado, ao ser pressionado o botão Ok, o cabeçalho padrão "Cabeçalho de Documento - Para Aula Prg Micro" é exibido. Se um texto for digitado, esse texto será o cabeçalho.

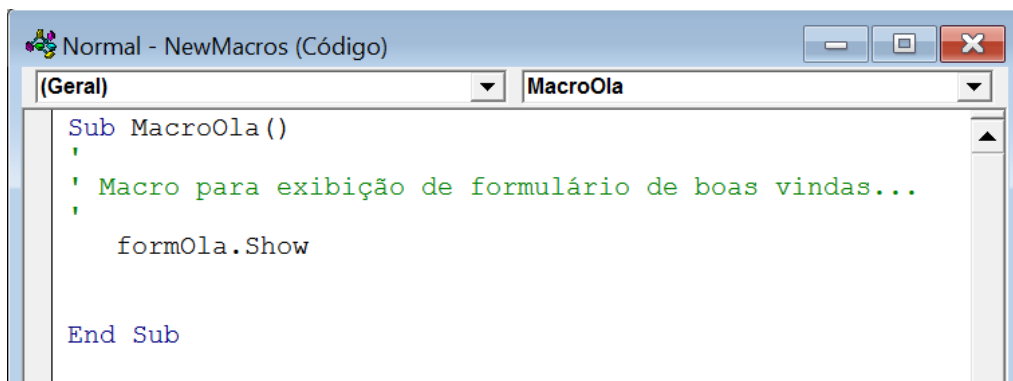
3-Uma boa maneira de transitar dados entre aplicações Windows, é pela Área de Transferência. Modifique a macro que aciona a calculadora e faça modificações para que o cálculo efetuado na calculadora e mandado para a Área de Transferência (Editar / Copiar) apareça no texto, como no exemplo: "Valor calculado pela calculadora = 9999".

Obs. O fato de a macro ter exibido a calculadora não significa que a execução da macro parou nesse ponto e que esteja aguardando alguma ação do usuário. As instruções após a exibição da calculadora foram todas executadas e o processamento da macro foi encerrado. Se após (ou antes) de exibir a calculadora você escreveu instruções para pegar dados da Área de Transferência e "Colar" no documento, essas instruções são executadas imediatamente sem interrupções até o fim da macro.

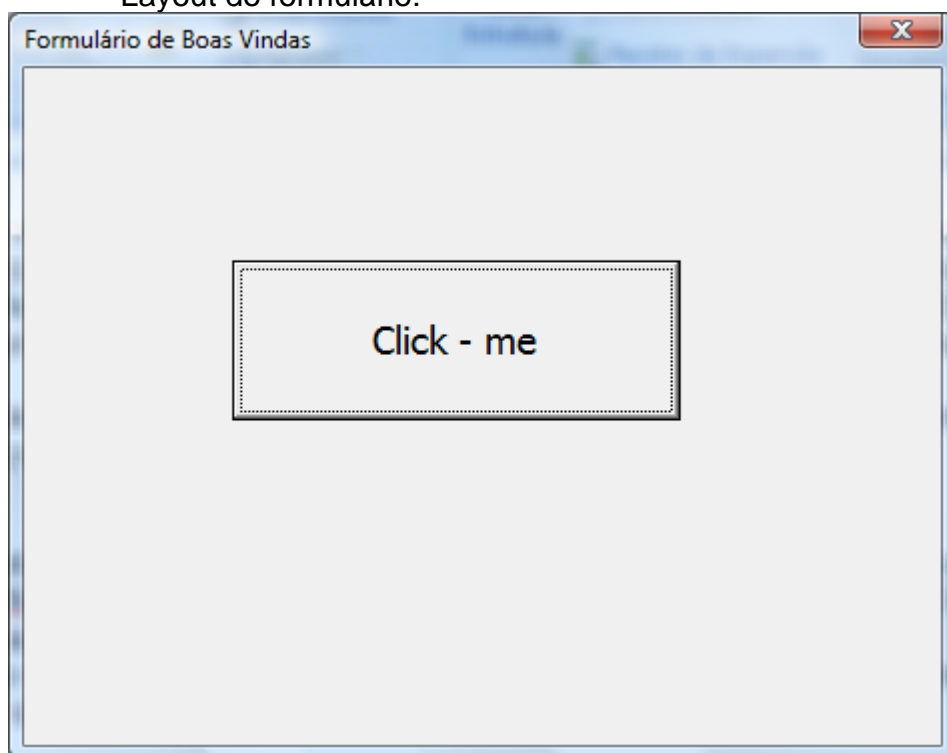
Se você quiser que o processamento dê uma pausa enquanto você faz os cálculos com a calculadora e "Copie" o resultado para depois "Colar" no documento, utilize uma "inputbox" ou uma "msgbox" que interrompem o processamento até que ocorra o click no botão Ok.

Aula 2 de programação com Word - Uso de formulários

4-Fazer uma macro diretamente no Módulo (sem usar o gravador) que exiba um formulário com layout e programação mais ou menos como a seguir.



Layout do formulário:



Programação associada ao formulário:

Option Explicit

```
Private Sub btnClickme_Click()  
    MsgBox "B E M    V I N D O S    A O    V B A", vbExclamation, "B O A S    V I N D A S"  
    End  
End Sub
```

O procedimento acima exibe uma caixa de mensagem dando boas vindas. Ao ser dado o click no botão OK da caixa de mensagem, esta é encerrada e o processamento passa à instrução seguinte, o "End", que encerra o processamento do VBA.

5-Fazer diretamente no Módulo (sem usar o gravador de macros), uma macro de nome "CalculaNota" que exibe um **formulário de acordo com o layout a seguir**. O usuário digita as duas notas de um aluno e o programa calcula a média aritmética. O programa exibe no documento do Word a nota e a informação se o aluno está "Aprovado" ou "Reprovado".

A macro será mais ou menos assim:

```
Sub CalculaNota()  
    formNotas.Show  
    MsgBox "A situação do aluno será exibida no documento do Word..."  
End Sub
```

Layout do formulário:

The screenshot shows a VBA form window titled "Cálculo de Média de Duas Notas e Situação Final". Inside the form, there are three input fields arranged horizontally. The first field is labeled "Nota1" and contains the number "6". The second field is labeled "Nota2" and contains the number "9". The third field is labeled "Resultado" and contains the value "7,5". Below these fields is a button labeled "Calcula". At the bottom of the form, there is a label that displays the word "Aprovado".

Exemplo de programação associada ao formulário acima:

```
Private Sub btnCalcula_Click()  
  
    Dim result As Single  
    result = (Val(txtNota1.Text) + Val(txtNota2.Text)) / 2  
    lblResultado.Caption = result  
    If result >= 5.0 Then  
        lblSituacao.Caption = "Aprovado"  
    Else  
        lblSituacao.Caption = "Reprovado"  
    End If  
  
    Selection.TypeText Text:=lblSituacao.Caption ' Exibe resultado no documento do Word  
  
End Sub
```

Complementação:

Exercícios:

- a) As caixas de texto txtNota1 e txtNota2 devem aceitar apenas valores numéricos e o ponto decimal. A programação é a seguinte:

```
Private Sub TextBox1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
Dim tecla As String
    tecla = Chr$(KeyAscii)
    If (tecla < "0" Or tecla > "9") And tecla <> "." Then
        Beep
        KeyAscii = 0
    End If
End Sub
```

- b) As caixas de texto txtNota1 e txtNota2 devem aceitar apenas números e apenas **um único ponto decimal**.

Obs.: a função InStr(txtNota1.Text, ".") retorna 0 se não achou o "." ou retorna a posição onde encontrou o ".". Verifique o Help on line.

- c) Se uma tecla inválida for digitada, o programa deverá exibir mensagem de alerta.
- d) As caixas de texto txtNota1 e txtNota2 devem aceitar no máximo 4 caracteres (propriedade maxlength).
- e) Inclua no formulário do VBA uma caixa de texto para ser digitado o Nome do Aluno. Todos os caracteres devem ficar em maiúsculo. Use as funções do VBA: Asc, Ucase, Chr. Exemplo:

```
KeyAscii = Asc ( UCase ( Chr ( KeyAscii ) ) )
```

Ao ser pressionado o botão "Calcula", o nome do aluno deverá **aparecer no documento do Word, em cor preta, negrito, tamanho 14**.

- f) Ao ser pressionado o botão "Calcula", exibir **no documento do Word**, o nome do aluno que está na caixa de texto. Em seguida, exibir a informação se o aluno está ou não aprovado. O nome do aluno deverá estar formatado em cor preta, negrito e tamanho 14.

- Se o aluno estiver aprovado, o texto "**APROVADO**", será na cor azul.
- Se reprovado, o texto "**REPROVADO**" deverá ser na cor vermelha. Após a impressão, o texto voltará a ser preto.

Após efetuar seus testes, vamos eliminar as nossas macros do Word. Para isso, dê um click com o botão direito do mouse no ícone da Barra de Ferramentas e os remova.

Em seguida, dê um click no item Macros, selecione cada macro e dê um click no botão Excluir.

Alguns programas deste projeto:

```
Option Explicit
Private Sub CommandButton1_Click()
UserForm1.Hide
End Sub

Private Sub btnCalcula_Click()
Dim n1 As Single
Dim n2 As Single
Dim result As Single
result = (Val(txtNota1.Text) + Val(txtNota2.Text)) / 2
lblResultado.Caption = result
If result >= 5# Then
    lblSituacao.Caption = "Aprovado"
Else
    lblSituacao.Caption = "Reprovado"
End If
End Sub

Private Sub btnFim_Click()
formNotas.Hide
End Sub

Private Sub txtNota1_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
    Dim tecla As String
    tecla = Chr$(KeyAscii)
    If (tecla < "0" Or tecla > "9") And tecla <> "," Then
        Beep
        KeyAscii = 0
    End If
    If tecla = "," And InStr(txtNota1.Text, ",") <> 0 Then
        MsgBox "Vírgula já digitada..."
        KeyAscii = 0
    End If
End Sub
```


Segunda Parte – Excel – Aula 1

Para iniciarmos, abra o Excel, para começarmos a criação de nossa macro, que irá acrescentar um cabeçalho em qualquer planilha na qual for chamada.

No Excel o processo de criação de macro é muito parecido com o Word. Para testar, vamos seguir os mesmos passos usados no Word para criar uma nova macro:

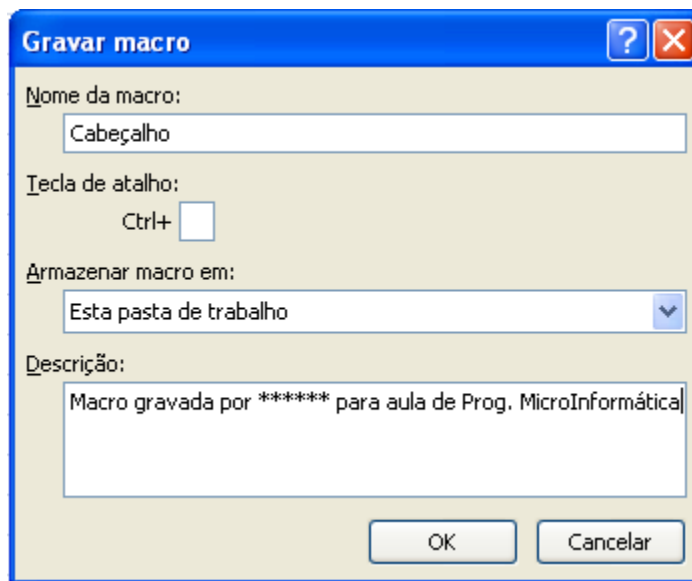
Para isso, faça ficar disponível a guia “Desenvolvedor”, selecione-a e dê um click no item “Gravar macro”.

Na janela que abrir, configure o seguinte:

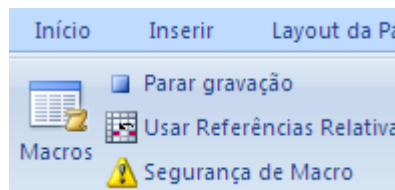
- Mude o nome da macro para “Cabeçalho”;
- No item “Armazenar macro em”, escolha “Esta pasta de trabalho”.

Obs.: Se você escolher “Pasta de trabalho pessoal de macros”, a macro ficará gravada no Excel que está instalado no micro. Não faremos isso. A macro que criarmos será disponibilizada apenas na planilha que estamos desenvolvendo. Para isso, na caixa “Armazenar macro em:”, escolha “Esta pasta de trabalho”.

- Em “Descrição”, substitua o ***** por seu nome.
- Click em Ok.



A partir deste momento, tudo que você fizer estará sendo gravado na macro, até que você dê um click no botão .Parar gravação.



Quando parar a gravação, a macro será gravada na forma de comandos VBA, com o nome que você escolheu.

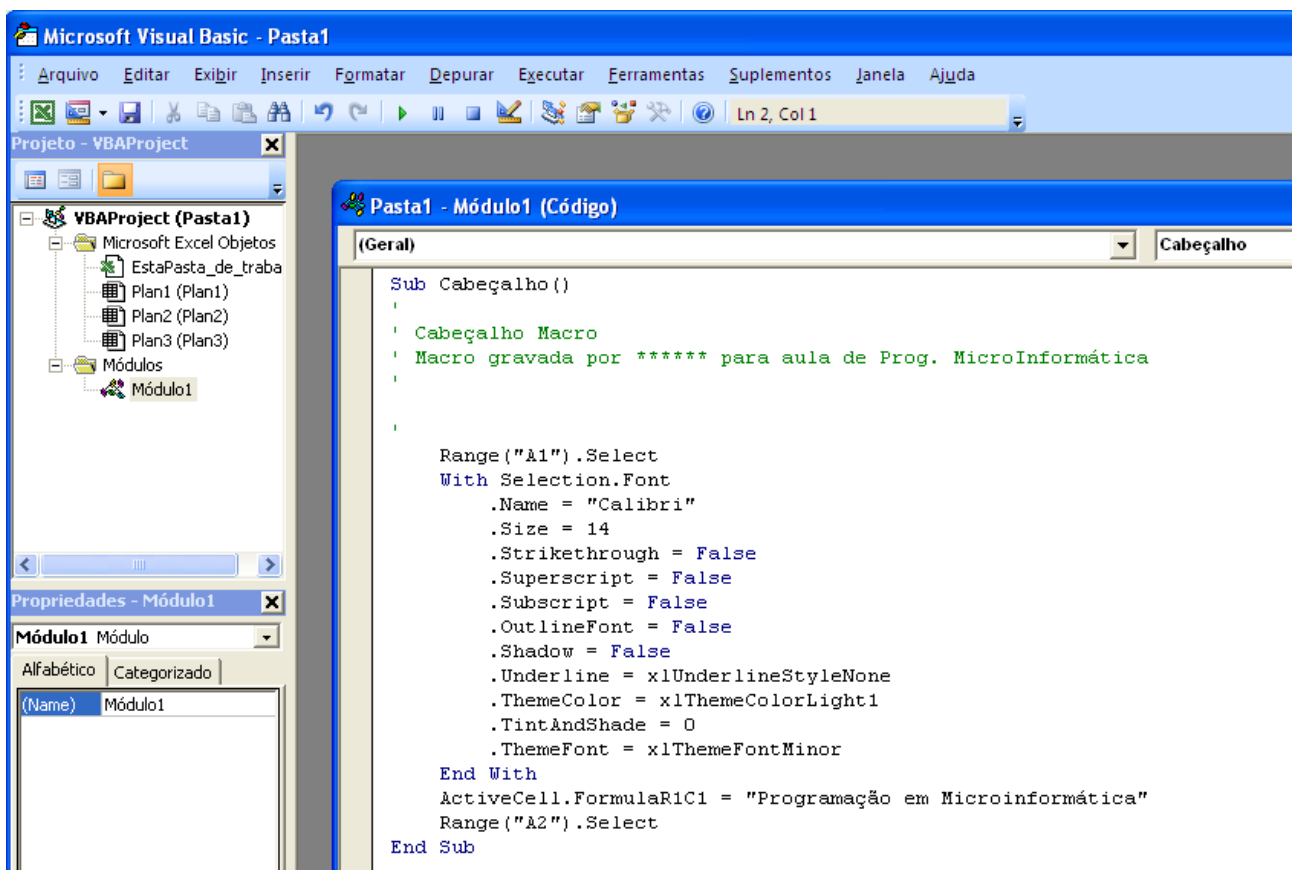
A macro que iremos criar fará o seguinte:

- Insere uma linha a partir da primeira linha;
- Posiciona o foco na célula A1 e a formata com tamanho 14;
- Digita o texto “Programação em microinformática”, na célula A1;

Então no Excel, siga os seguintes passos:

- . Selecione a primeira linha e insira uma nova linha;
- . Posicione o cursor na célula A1 e formate-a com tamanho 14;
- . Na célula A1 digite o texto “Programação em microinformática” e pressione a tecla “Enter”.
- . Encerre a gravação da macro, dando um click no botão “Parar Gravação”.

Pronto: Nossa macro “Cabeçalho” está gravada. Vamos vê-la. Dê um click no botão “Macros”, selecione a macro “Cabeçalho” e dê um click no botão “Editar”
Aparecerá uma janela parecida com a abaixo:



A figura acima mostra uma parte do ambiente de desenvolvimento do VBA. Explore a macro Cabeçalho...

Obs. Comentários: Em VB iniciam com apóstrofo, ficam na cor verde e servem apenas para a documentação do programa.

A macro “Cabeçalho” está gravada em nossa planilha do Excel. Vamos executá-la. Para voltar ao Excel ou vice-versa, use a combinação de teclas Alt + F11.

Apague o conteúdo da célula A1 (Editar / Limpar / Tudo) e execute a macro Cabeçalho que você criou, para ver seu efeito.

Para executá-la, abra selecione a guia “Desenvolvedor”, dê um click no botão “Macros”, escolha a macro “Cabeçalho” e dê um click em “Executar”.

Após executá-la, volte ao ambiente do VBA, com a combinação de teclas Alt + F11 para estudá-la um pouco melhor.

Obs. “O gravador de macros do Excel gera a parte mais “pesada” da codificação”, gerando inclusive algumas instruções desnecessárias, que deixam a execução da macro mais lenta.

Podemos analisar e modificar as instruções da macro para melhor adequá-la a nossas necessidades.

Por exemplo, podemos eliminar as instruções:

```
.Strikethrough = False  
.Superscript = False  
.Subscript = False  
.OutlineFont = False  
.Shadow = False
```

Podemos também substituir a instrução “ActiveCell.FormulaR1C1 = “Prog. Micro”,
Por ActiveCell.FormulaR1C1 = inputbox (“Digite o Cabeçalho desejado...”)

Faça essas modificações, execute a macro e veja o efeito.

Obs.: Você pode executar a macro, pressionando a tecla F5. Se você estiver no ambiente do VBA e pressionar F5, a macro será executada o foco volta ao ambiente do VBA.

-Para ver o efeito da execução da macro, use Alt + F11 (Mostra a planilha).

-Limpe novamente os textos da planilha e execute a macro novamente.

Criação de macros no Excel sem uso do gravador de macros

Criando macros sem o uso de gravador:

Abra o editor do VBA (Alt + F11), e digite após o “End Sub” do procedimento “Sub Cabeçalho”:

```
Sub Teste_Cabecalho_Com_Data()
```

Note que após você pressionar Enter, o programa inclui automaticamente o “End sub”.
Digite em seguida, os comandos abaixo, deixando a procedure da forma:

```
Sub Teste_Cabecalho_Com_Data()
```

```
    Range("A1").Select  
    ActiveCell.FormulaR1C1 = InputBox("Digite o Cabecalho : ", "A T E N Ç Ã O") + Str(Date)  
End Sub
```

Obs. : A expressão Str (Date) é concatenada à string que será digitada na InputBox. Ela é composta por duas funções do VBA:

Str – converte número em texto.

Date – retorna a data do computador.

Pressione F5 para executá-la e veja o efeito. Obs. Quando se pressiona F5, é executado o procedimento onde está o foco (cursor do editor).

Volte para o ambiente do Excel (Alt + F11) e verifique que o procedimento **Teste_Cabecalho_Com_Data** aparece junto com a macro anterior (Cabeçalho).

A qualquer momento, o ambiente do VBA dispõe de um Help, de onde se podem obter informações valiosas. Para acionar o Help, basta selecionar o objeto ou instrução que se necessita ajuda e pressionar a tecla F1. Tente com “**Inputbox**” da macro acima.

Uso de formulários em Excel:

Utilização de formulários na programação com VBA: Utilizam-se formulários de maneira similar à programação em VB. No formulário, são incluídos componentes tais como botões, caixas de texto, labels etc. que respondem a eventos provocados pelo usuário.

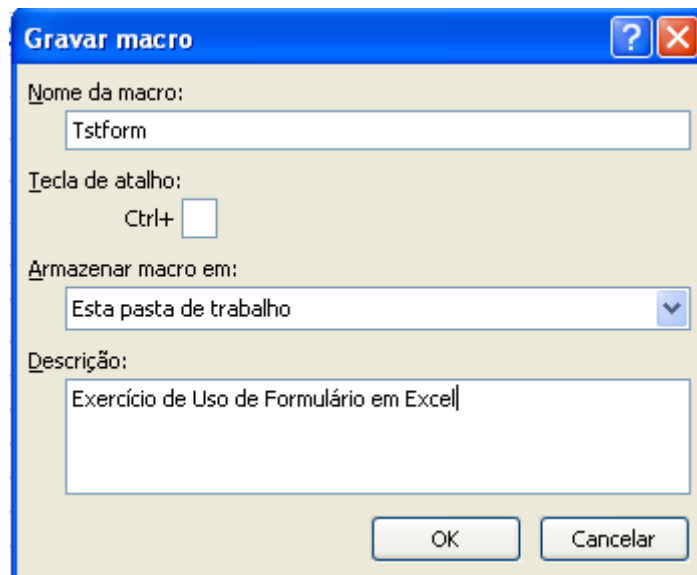
Para testar isso, feche o Excel, e através do “Windows Explorer” **copie para sua pasta ou seu pendrive**, o arquivo “Livros”.

Abra o arquivo “Livros” de sua pasta ou pendrive, contendo a planilha “Estoque”, igual à abaixo:

	A	B	C	D	E	F
1	Código	Nome	Autor	Editora	Preço	Quantidade
2	605	C++	Valmir	Mack	R\$ 67,00	10
3	805	Office	Fabiani	Erica	R\$ 92,00	21
4	1005	VB net	Viana	Makron	R\$ 77,00	6
5	104	Delphi	Cantú	Sams	R\$ 130,00	3
6	204	C Builder	Kent	Makron	R\$ 78,00	5
7	304	Access	Pádua	Érica	R\$ 45,00	2
8	505	V. Basic	Nunes	QUE	R\$ 88,00	7
9						

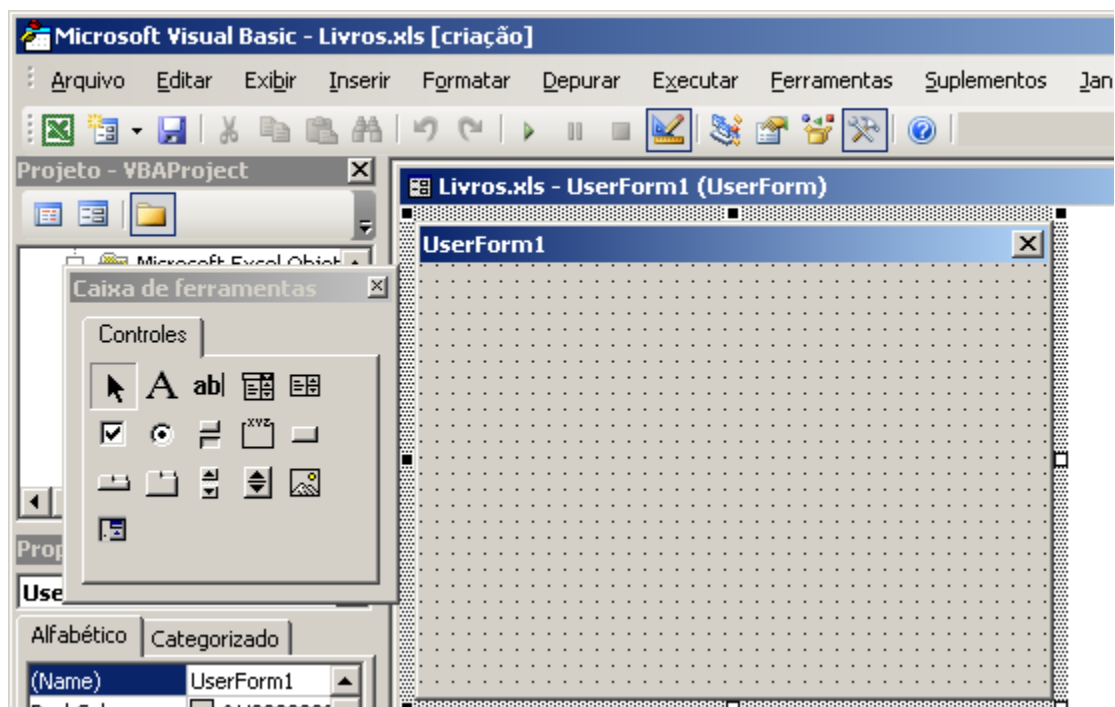
Obs.: No Office 2007, o nível de segurança aumentou. Ao salvar uma planilha que contenha macros, no item de menu “Salvar Como” escolha uma pasta “Pasta com Macros Excel”. Se isso não for feito, a planilha será salva sem as macros e você perderá todo o seu trabalho.

Crie uma macro de nome TstForm, sem nenhuma programação (click no botão “Gravar Macro”).



Pressione o botão OK e em seguida pare a gravação. Pronto! Gravamos uma macro de nome TstForm, sem nenhuma programação. Agora, abra o ambiente do VBA, com a combinação Alt + F11. Na janela “Project Explorer”, abra a pasta Módulos/Módulo1, que é onde estão as macros.

Inclua um formulário associado à sua planilha. vá ao ambiente do VBA e dê click nos itens Inserir / UserForm. Você verá a seguinte janela:



Através dessa janela você irá elaborar e programar a interface com a qual o usuário irá manipular dados na planilha.

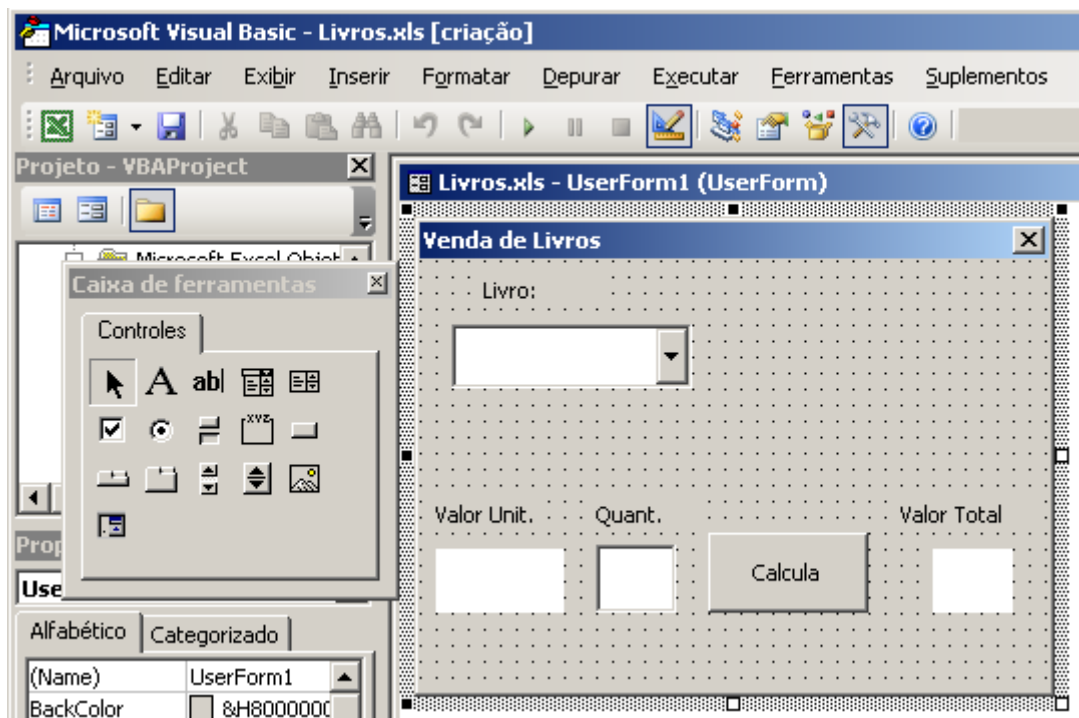
Obs.: Estrutura de comandos e programação em VBA – aos objetos são associadas propriedades, que são as características do objeto. Por exemplo, o tamanho, cor etc. são propriedades de um objeto. Aos objetos também são associados métodos (funções) – exemplo Range(“A2:A6”).Select – deixa selecionadas as células A2 até A6.

Variáveis – É a maneira pela qual manipulamos valores em programação. Exemplo de criação de variáveis em VBA :

```
Private Sub UserForm_Click()  
    Dim x As String  
    x = "Bem vindos ao VBA..."  
    FrmVendaLivros.Caption = x  
End Sub
```

Selecione o formulário e na janela “Propriedades” digite, na **propriedade Caption**, o texto “Venda de Livros”.

Inclua no formulário os componentes abaixo e configure o formulário de acordo com o seguinte layout:



Obs.: vamos mudar todos os nomes dos componentes, de acordo com seu tipo e objetivo:

Exemplos:

Formulários – inicia com Frm – exemplo FrmVendaLivros

Caixa de Combinação - Inicia com Cxc – exemplo CxcLivro

Label – Inicia com Lbl – exemplos LblValorUnit, LblValortotal

Caixa de Texto – Inicia com Txt – exemplo TxtQuantidade

Botão de comando – inicia com Cmd – exemplo CmdCalcula

Para isso, selecione cada componente e altere a propriedade Name na janela de propriedades.

Para executar o formulário, vamos criar nossa procedure, digitando diretamente na Macro TstForm:

```
Sub TstForm()
```

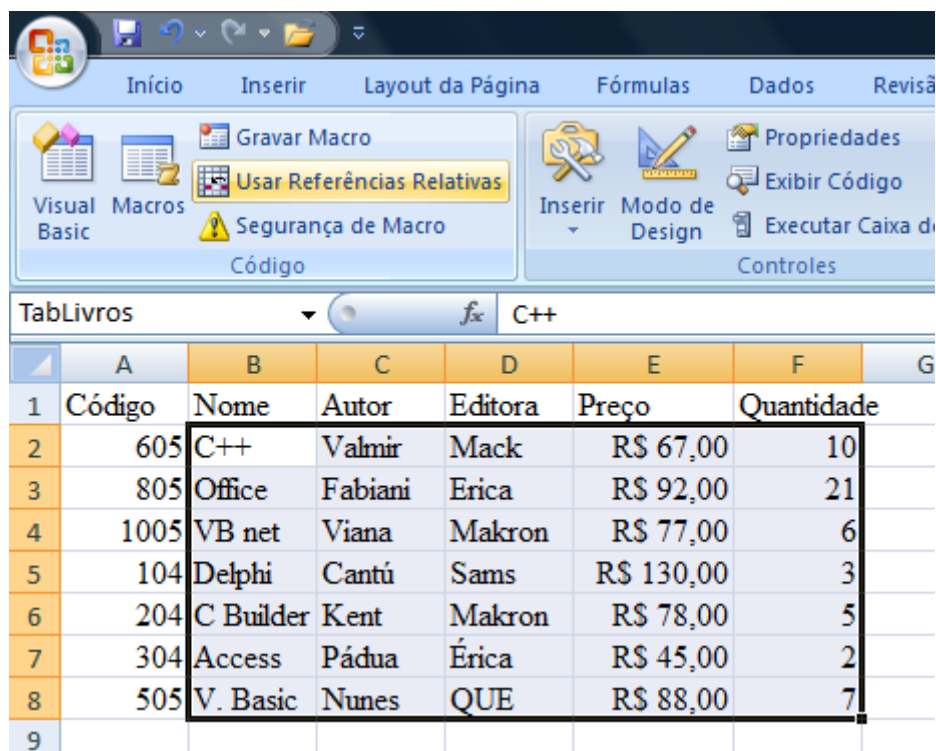
```
    FrmVendaLivros.Show ' Esta instrução exibe e executa o formulário FrmLivros.
```

```
End Sub
```

Pressione a tecla F5 para ver o programa funcionando.

Note que os componentes estão prontos e funcionando, porém não respondem aos eventos provocados pois estes (os eventos) não estão sendo tratados. Faremos o tratamento a seguir.

Feche o formulário e volte à janela da planilha (Alt + F11), selecione a área igual à abaixo, dê o nome a essa seleção de “TabLivros” e pressione a tecla Enter.



	A	B	C	D	E	F	G
1	Código	Nome	Autor	Editora	Preço	Quantidade	
2	605	C++	Valmir	Mack	R\$ 67,00	10	
3	805	Office	Fabiani	Erica	R\$ 92,00	21	
4	1005	VB net	Viana	Makron	R\$ 77,00	6	
5	104	Delphi	Cantú	Sams	R\$ 130,00	3	
6	204	C Builder	Kent	Makron	R\$ 78,00	5	
7	304	Access	Pádua	Érica	R\$ 45,00	2	
8	505	V. Basic	Nunes	QUE	R\$ 88,00	7	
9							

Volte ao ambiente do VBA, selecione o componente CxcLivros e coloque em sua propriedade RowSource, o texto “TabLivros”, que é o nome que você colocou na região selecionada da planilha.

Execute novamente o programa (tecla F5) e dê um click no componente CxcLivro.

Note que o componente responde ao evento, trazendo a tabela TabLivros.

Vamos continuar nossa programação. Para voltar à fase de projeto, interrompa a execução do programa.

Duas funcionalidades de nosso programa serão:

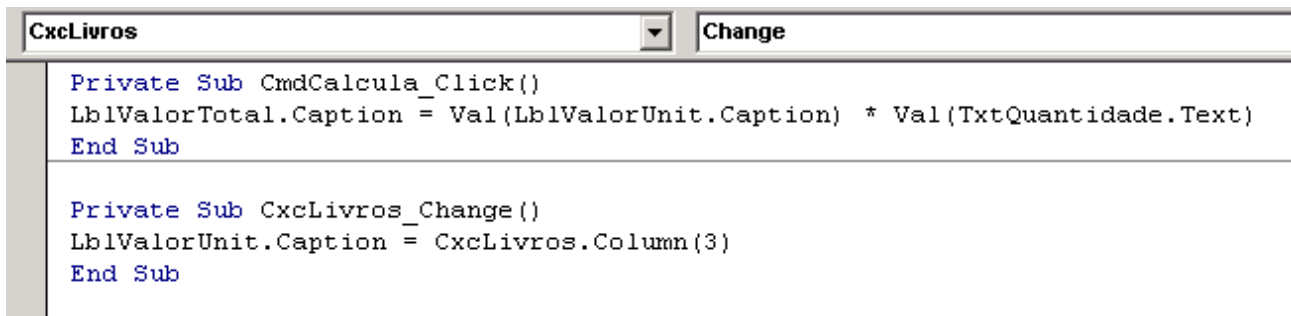
- Quando ocorrer o click no nome do livro, deverá surgir o valor unitário no label de nome LblValorUnit.

- b) Se digitarmos a quantidade de livros e pressionarmos o botão “Calcula”, o programa deverá exibir o valor Total da compra, no label LblValorTotal.

Uso de tratadores de evento em VBA com Excel

Programação dos “Tratadores de Eventos” do VBA:

A janela de programação abre-se ao ocorrer o click duplo no componente que desejamos programar. Essa janela tem a seguinte aparência e a programação será a seguinte:



Faça o teste e verifique o funcionamento do programa.

Selecione um livro na caixa de combinação, digite a quantidade desejada e pressione o botão “Calcula”. Ao fazer isso, o tratador do evento Click do botão é executado e o valor total dos livros é calculado.

Exercícios:

- 1) Fazer o valor unitário aparecer formatado, com centavos e separador de milhares. Isso pode ser feito com ajuda da instrução “format”. Veja o exemplo abaixo:

```
Private Sub cxcLivro_Change()  
    lblValorUnit.Caption = Format(cxcLivro.Column(3), "R$##,###.00")  
End Sub
```
- 2) Note que o valor unitário fica muito bonito formatado, só que o programa não funciona mais. Verifique o motivo e conserte-o.
- 3) O campo “Quantidade” deve aceitar apenas números. Lembra do programa de cálculo da média que fizemos quando estudamos o VBA p/ Word? É bem parecido.
- 4) Formatar em formato de moeda (R\$) com centavos, o campo “Valor Total”.
- 5) Inclua uma caixa de texto para que o operador possa oferecer um desconto no preço da compra. Caso o desconto exceda 20% o programa deve alertá-lo sobre esse fato, mas deve realizar o cálculo assim mesmo. Use a procedure do VBA msgbox. Use o Help para ver seu funcionamento.


```

UserForm
Click

Private Sub CmdCalcula_Click()
If Val(TxtDesconto.Text) > 20 Then MsgBox "Atenção.... desconto acima de 20%"
LblValorTotal.Caption = Val(LblValorUnit.Caption) * Val(TxtQuantidade.Text)

End Sub

Private Sub CxcLivros_Change()
LblValorUnit.Caption = CxcLivros.Column(3)
End Sub
    
```

- 6) Inclua um label p/ exibir separadamente o valor do desconto.em moeda.
- 7) O campo em que o operador vai informar o desconto, deve aceitar somente números e o ponto decimal. Note que o desconto deve ser um valor real. Se você for usar variáveis reais, a declaração é, por exemplo: ***dim desconto as single***.
- 8) Formate adequadamente todos os valores em formato de moeda para exibição nos campos do formulário.

Use a instrução Format. Exemplo, supondo-se que valorTotal é variável real, a instrução de formatação do campo no formulário, será mais ou menos como abaixo:

LblValorTotal.Caption=Format (valorTotal, "R\$#,###.00")

Formato sugerido do formulário “Venda de Livros” até agora:

Para finalizar, selecione o botão e atribua para “True” sua propriedade “Default”. Se a propriedade default de um botão estiver True, você pode executar o programa associado ao evento Click desse botão, pressionando a tecla “Enter”.

Ainda no botão, coloque a letra “C” na propriedade “Accelerator” e note que a letra C do “Calcula”, ficou sublinhado. Agora você pode executar o programa do botão com a combinação de teclas Alt + C.

Salve a planilha, Teste o programa e prepare-se para apresentá-lo.

Aula 2 de programação com Excel

Vamos continuar desenvolvendo nossa aplicação com VBA.

Abra o arquivo Livros.xls que utilizamos na aula anterior. Ative o ambiente do VBA (Alt + F11)

Criação e uso de funções:

Podemos criar funções em VBA e utilizá-las na planilha Excel.

Por exemplo, suponha que precisamos da informação da situação de nosso estoque, da seguinte maneira:

- se o estoque estiver igual ou abaixo de 3 unidades, isso significa que **o estoque estará baixo**.
- se estiver entre 4 e 14 unidades, **estará bom (normal)**.
- com 15 unidades ou mais, **estará muito alto**.

Podemos usar as funções nativas do Excel e criar uma expressão para resolver isso. Uma solução melhor, seria criar a seguinte função em VBA, que deve ser digitada no mesmo Módulo que contém a procedure TstForm:

A função SaldoEstoque:

Function SaldoEstoque(estoque As Integer) As String

```
If estoque <= 3 Then SaldoEstoque = "Estoque Baixo"  
If estoque > 3 And estoque <= 15 Then SaldoEstoque = "Estoque Normal"  
If estoque > 15 Then SaldoEstoque = "Estoque Em Excesso"
```

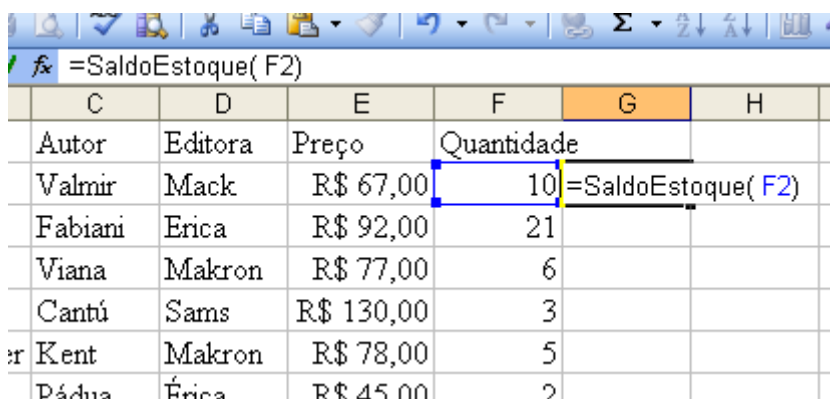
End Function

Sub TstForm()

```
FrmVendaLivros.Show ' Esta instrução exibe e executa o formulário FrmLivros.
```

End Sub

Agora basta chamar a função SaldoEstoque, na planilha e utilizá-la:



The screenshot shows an Excel spreadsheet with a table of book data. The formula bar at the top displays the formula `=SaldoEstoque(F2)`. The table has columns for Autor, Editora, Preço, and Quantidade. The data rows are as follows:

Autor	Editora	Preço	Quantidade
Valmir	Mack	R\$ 67,00	10
Fabiani	Erica	R\$ 92,00	21
Viana	Makron	R\$ 77,00	6
Cantú	Sams	R\$ 130,00	3
Kent	Makron	R\$ 78,00	5
Dádua	Erica	R\$ 45,00	2

Verifique que o formulário FrmVendaLivros deve estar mais ou menos com a aparência:

Lembre-se que se o desconto oferecido for maior que 20%, o programa exibe uma mensagem de aviso, porém faz a operação assim mesmo. Vamos alterá-lo para que o usuário possa cancelar a operação. Para isso, usaremos a msgbox na forma de função, onde será retornado o valor do botão Yes ou No, que o usuário clicou.

O código ficará mais ou menos igual ao abaixo:

```
Private Sub CmdCalcula_Click()  
    Dim ValTotal As Single  
    Dim ValDesconto As Single  
    Dim ValComDesconto As Single  
    If Val(txtDesconto.Text) > 20 Then  
        If MsgBox(" Atenção: Desconto maior que 20%...", vbYesNo) = vbNo Then  
            Exit Sub  
        End If  
    End If  
  
    ValTotal = Val(TxtQuantidade) * Val(LblValorUnit.Caption)  
    ValDesconto = Val(txtDesconto.Text) / 100 * ValTotal  
    ValComDesconto = ValTotal - ValDesconto  
    LblDesconto.Caption = Format(ValDesconto, "#,###.00")  
    LblValorTotal.Caption = Format(ValComDesconto, "#,###.00")  
  
End Sub
```

Exercício: Se o percentual do desconto for maior que 20% e o usuário pressionar o botão "No", o programa deverá limpar os valores dos labels LblDesconto, LblValorTotal e deverá mandar o foco para a caixa de texto txtDesconto, deixando os valores dessa caixa selecionados.

Para isso, utilizar o método setfocus e as funções:

```
txtDesconto.SetFocus  
Txtdesconto.SelStart=0  
Txtdesconto.SelLength=len(Txtdesconto.text)
```

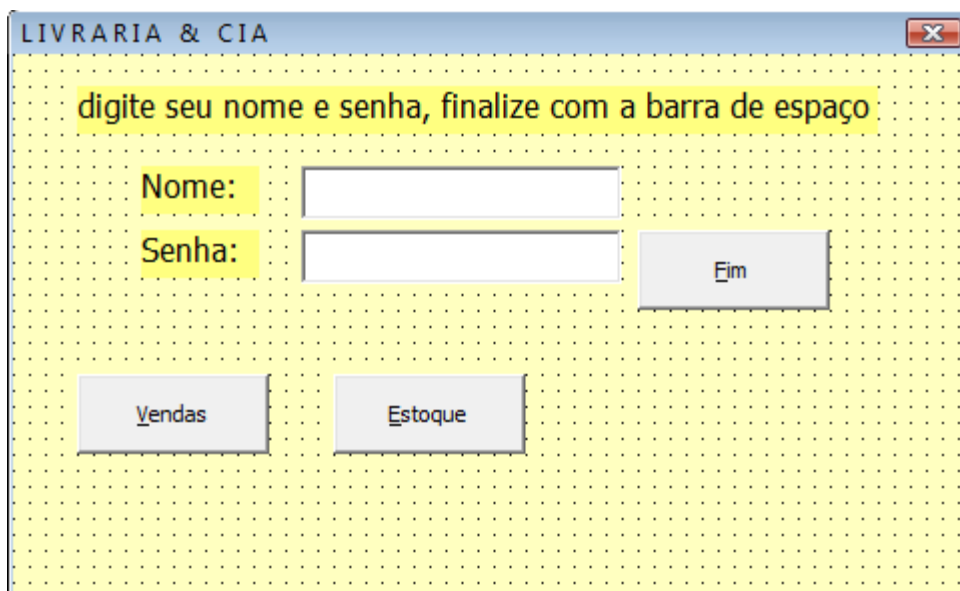
Essa parte do programa ficará mais ou menos assim:

```
If Val(TxtDesconto.Text) > 20 Then
    If MsgBox(" Atenção: Desconto maior que 20%...", vbYesNo) = vbNo Then
        LblDesconto.Caption = ""
        LblValorTotal.Caption = ""

        TxtDesconto.SetFocus

        TxtDesconto.SelStart = 0
        TxtDesconto.SelLength = Len(TxtDesconto.Text)
    Exit Sub
End If
End If
```

Continuando nosso projeto, inclua um novo formulário (Inserir UserForm) e configure-o como abaixo:



Os nome dos componentes serão:

Name do formulário: FrmLivraria,
Name dos botões: cmdVendas, cmdEstoque, CmdFim.

Os botões cmdVendas e cmdEstoque deverão iniciar com a propriedade “Visible” false. Ou seja, não estarão visíveis. Ficarão disponíveis, ou seja, visíveis, somente quando o usuário acertar a senha.

Name das caixas de texto: txtNome, txtSenha.

Insira na propriedade Picture do formulário FrmLivraria, uma das fotos da pasta ...Server2000 / Prg_Micro.

Obs.: Podemos adequar o tamanho da figura ao formulário, através da propriedade PictureSizeMode.

Especificações e exercícios associados a este formulário:

- 1) Ao iniciar a execução da macro, o formulário “frmLivraria” é exibido, solicitando ao usuário que digite seu nome e senha.
- 2) As letras digitadas no campo nome, devem ficar todas em maiúsculo. Vide exemplo abaixo:

```
Private Sub txtNome_KeyPress(ByVal KeyAscii As MSForms.ReturnInteger)
Dim tecla As String
tecla = Chr$(KeyAscii)
KeyAscii = Asc(UCase(tecla))
End Sub
```

- 3) Se o foco estiver em txtNome e for pressionada a barra de espaço, o foco deve mudar para txtSenha. Exemplo:

If KeyAscii = 32 Then txtSenha.SetFocus

Obs.: Para digitação e verificação da senha, o projeto deve obedecer às especificações a seguir:

- a) Se a senha for correta, o programa exibe a mensagem: “Bem Vindo Senhor **nome**!”. Caso contrário, o programa exibe: “Caia Fora!...”.
- b) Os caracteres digitados na senha aparecem como “*”.
- c) A sequência de caracteres do nome e da senha são finalizados pela barra de espaço.
- d) A senha correta é “1234”.
- e) O usuário pode fazer até três tentativas p/ acertar a senha. A cada tentativa, o programa informa que a senha informada foi incorreta e quantas tentativas faltam. Se o usuário acertar a senha, veja o item "g" desta lista. Se ocorrerem mais de 3 tentativas, o programa deve encerrar a execução.
- f) Inclua o botão “cmdFim” com a propriedade Caption Fim e com a propriedade “Accelerator” = “F”, que ao receber o click do mouse encerra o programa. A instrução é “End”.
- g) Inclua os botões “cmdVendas” com caption “Vendas” e o botão “cmdEstoque” com caption “Estoque”. Esses botões estarão inicialmente invisíveis e só ficarão visíveis quando o usuário acertar a senha.

Ao ser dado o clique no botão cmdVendas, é exibido **na forma modal**, o formulário FrmVendaLivros. Deverá ter a forma:

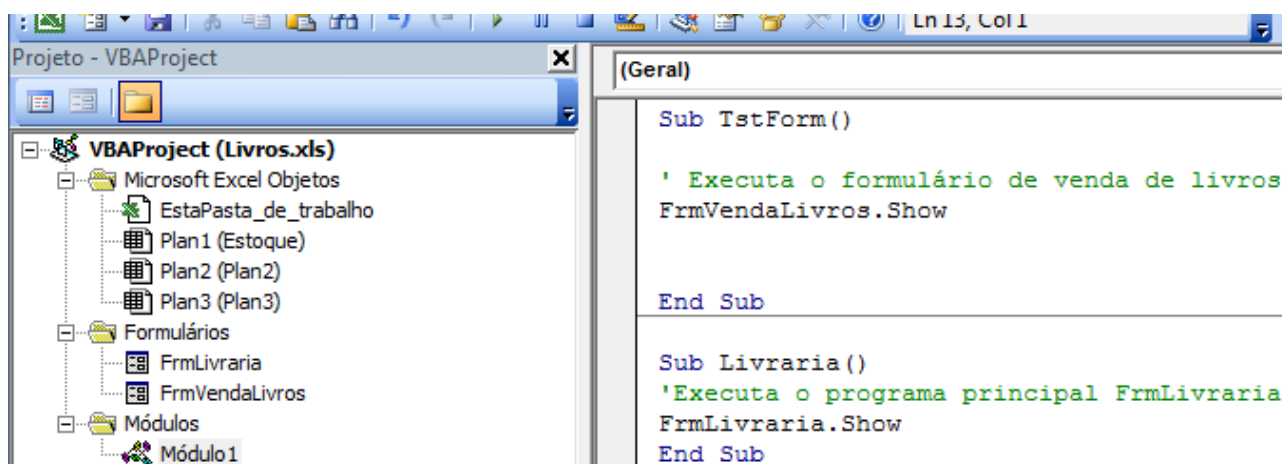
```
Private Sub CmdVendas_Click()
    FrmVendaLivros.Show 1
End Sub
```

Ao ser dado o clique no botão CmdEstoque, é exibido o formulário FrmEstoque que desenvolveremos posteriormente.

Ao ser dado o clique no botão CmdFim, o programa é encerrado. Veja o exemplo abaixo:

```
Private Sub CmdFim_Click()  
    Unload FrmLivraria  
End Sub
```

Crie no Módulo1, a procedure “Livraria”, que ao ser executada, exibe o formulário FrmLivraria. Veja a figura:



Salve a planilha, Teste o programa e prepare-se para apresentá-lo.

Aula 3 de programação com Excel

Inclua na Barra de Ferramentas de Acesso Rápido, o ícone mais adequado para executar a macro “Livraria” que você fez na aula anterior (Veja como você fez no Word – página 7 da apostila).

Teste a execução do programa para certificar-se que está tudo bem.

A interface de seu programa deverá estar mais ou menos assim:

The image shows two Excel VBA forms overlaid on a spreadsheet. The first form, titled 'LIVRARIA & CIA', has a yellow background and contains a login section with labels 'Nome:' and 'Senha:', input fields containing 'HAMILTON' and '****', and a 'Fim' button. Below this are two buttons labeled 'Vendas' and 'Estoque'. The second form, titled 'Venda de Livros', has a light gray background and features a 'Livro:' dropdown menu. Below it are input fields for 'Valor Unit:', 'Quant:', and 'Desconto:', followed by a 'Calcula' button. To the right of the 'Calcula' button are two more input fields labeled 'Valor Desconto:' and 'Valor Total:'. The background spreadsheet shows columns for 'Quantidade' and 'Situação do Estoque'.

Exercícios:

- 1) Quando o usuário estiver fazendo o logon e acertar a senha, desabilitar as caixas de texto de nome e senha.
- 2) Inclua no formulário FrmVendaLivros, o botão “Volta”, que ao receber o click do mouse fecha esse formulário e o foco volta ao formulário FrmLivraria.

Crie o formulário FrmEstoque, com a seguinte aparência:

The image shows an Excel VBA form titled 'ESTOQUE'. It features six input fields arranged horizontally, each with a label above it: 'Codigo', 'Nome', 'Autor', 'Editora', 'Preço', and 'Quant.'. Below these fields are two buttons labeled 'Ok' and 'Cancela'.

Esse formulário será acionado quando o botão CmdEstoque, do formulário FrmLivraria, for clicado.

Exercícios VBA com Excel:

- 1) Os campos Código e Quant. só devem aceitar números.
- 2) O campo Preço deve aceitar apenas números e uma única vírgula decimal.
- 3) Nos campos Nome Autor e Editora todos os caracteres devem ficar em maiúsculo.
- 4) Incluir instruções para que quando for pressionado o botão Ok, o programa pergunta ao usuário se quer incluir mais algum livro. Se o usuário quiser continuar, todas as caixas de texto serão limpas e o foco vai para TxtCodigo. Se o usuário quiser parar, o formulário é fechado e o controle volta para frmLivraria.
- 5) Não deve aceitar inclusão de registros com campos em branco.
- 6) Para evitar a entrada de caracteres branco ou nulo, usar a função "trim". Essa função elimina brancos da esquerda e da direita do string.

Exemplo: if trim(txtNome.text) = "" then msgbox "Nome em branco..."

Programação inicial associada a esse formulário:

```
Private Sub CmdCancela_Click()  
    Unload FrmEstoque  
End Sub  
  
Private Sub CmdOk_Click()  
    Dim endereco As String  
  
    Sheets("Estoque").Select  
    Range("A65536").End(xlUp).Offset(1, 0).Select  
    ActiveCell.Offset(0, 0).Value = TxtCodigo.Text  
    ActiveCell.Offset(0, 1).Value = TxtNome.Text  
    ActiveCell.Offset(0, 2).Value = TxtAutor.Text  
    ActiveCell.Offset(0, 3).Value = TxtEditora.Text  
    ActiveCell.Offset(0, 4).Value = TxtPreco.Text  
  
    ActiveCell.Offset(0, 5).Value = TxtQuant.Text  
    ActiveCell.Offset(0, 5).Select  
  
    endereco = ActiveCell.Address  
    ActiveCell.Offset(0, 1).Value = "=SaldoEstoque(" + endereco + ")"  
  
    FrmEstoque.Hide  
  
End Sub
```

Este é apenas um protótipo de programa em VBA para o Excel, visando vislumbrar uma pequena parcela do potencial dos recursos que o software pode oferecer.

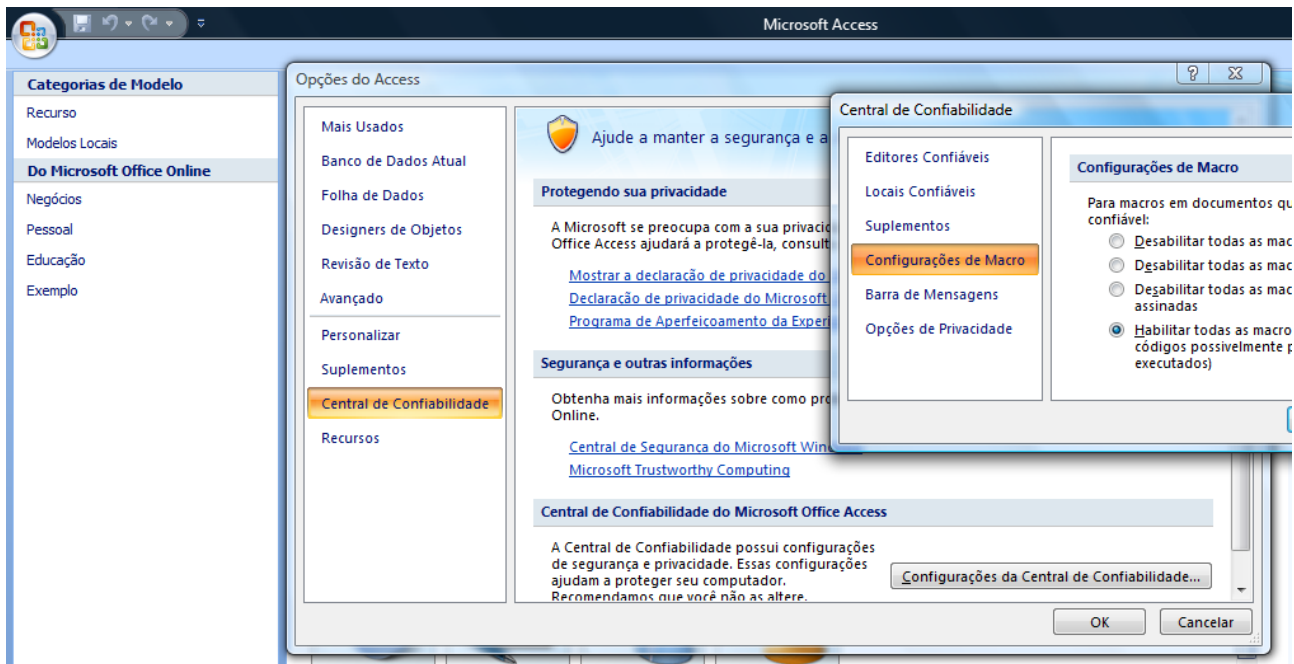
Salve a planilha Livros, teste todo o projeto e prepare-se para apresentá-lo ao professor.

Terceira Parte – Access –Aula 1

Aula 1 de Access.

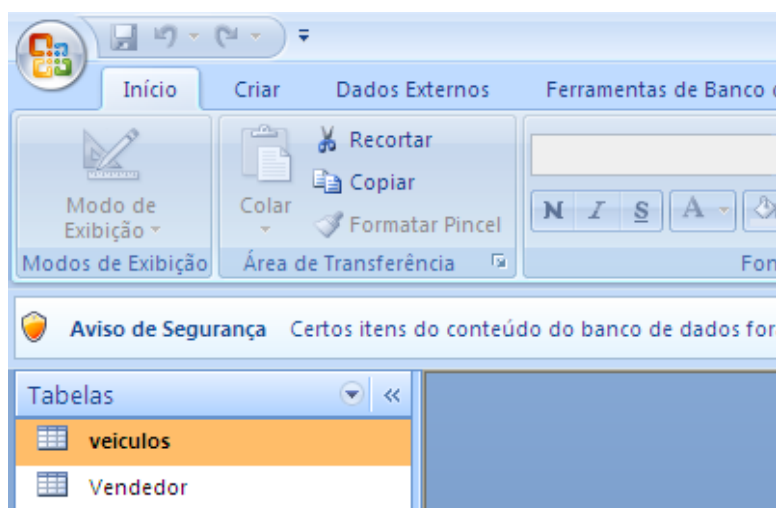
Para trabalharmos com a programação do Access, iremos inicialmente reduzir o nível de segurança para evitarmos problemas com a execução de macros. Para isso, localize e abra o Access. No botão Office, pressione o botão “Opções do Access” e selecione as janelas:

Central de Confiabilidade/Configurações da Central de Confiabilidade/Configurações de Macro. Em seguida, habilite todas as macros e **feche o Access**. Confira a figura abaixo:



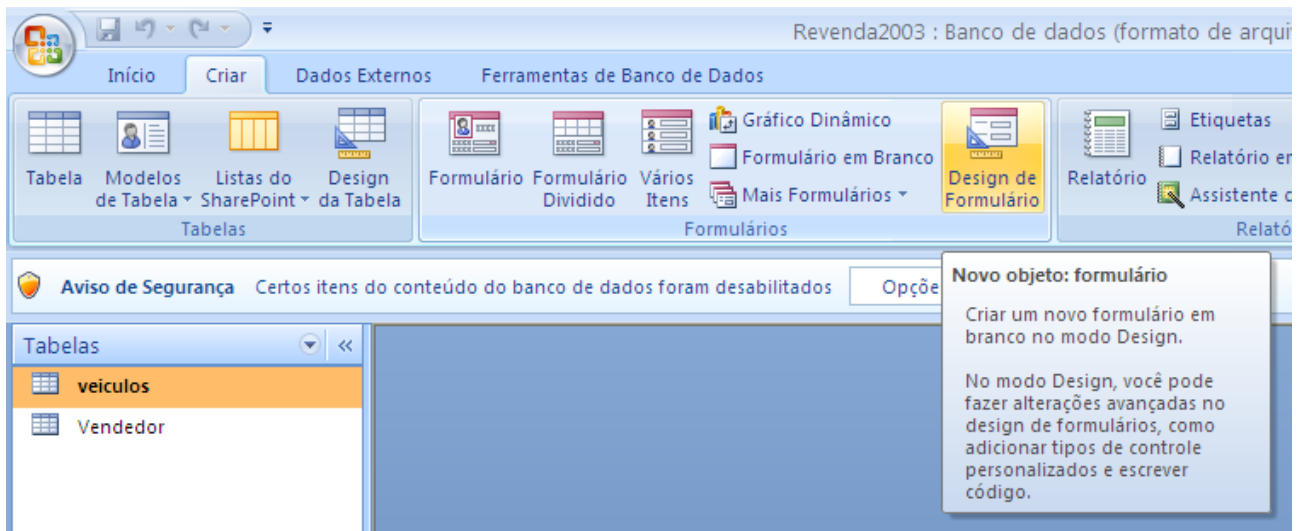
Copie o arquivo do Access **Revenda.accdb** do diretório “server2000\Prg Micro” para seu pendrive e abra o arquivo dando-lhe um duplo click

Esse arquivo possui tabelas contendo registros de veículos e de vendedores em uma revenda de veículos usados. Veja na figura abaixo:



Uso de formulários com Access:

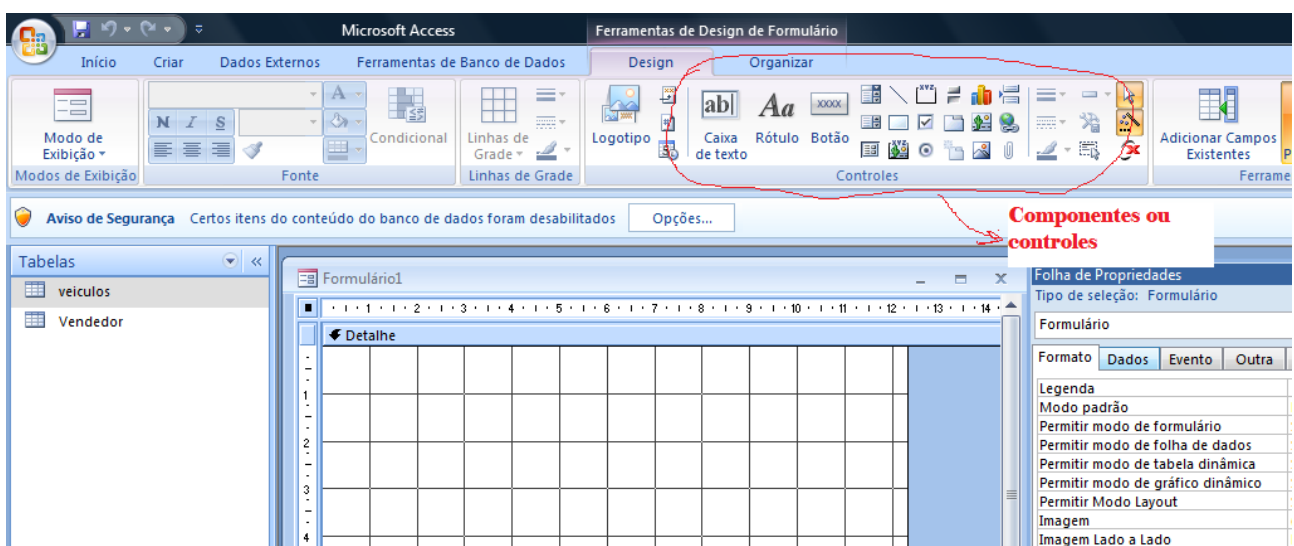
Selecione a tabela “veículos”, selecione a guia “criar” e dê um click no ícone “Design de Formulário”



Crie o formulário a seguir:

Na figura abaixo é exibida a janela “Propriedades”, o Formulário e os controles que você pode colocar em seu formulário.

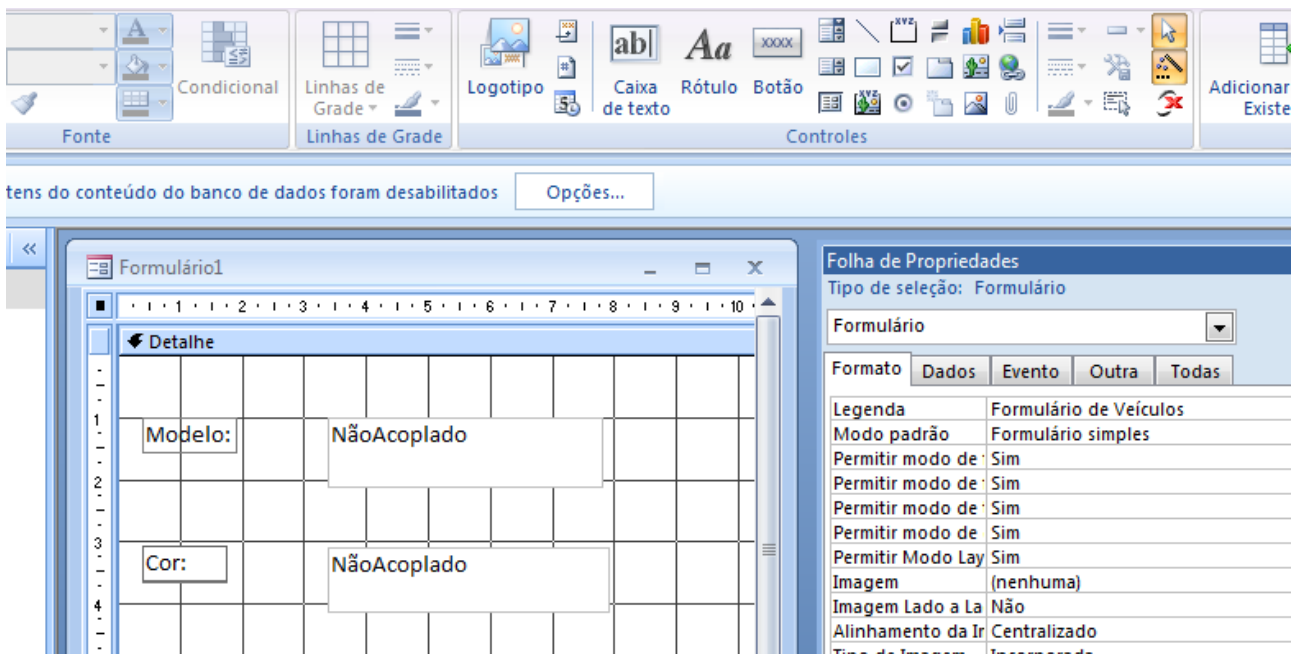
Se a janela Propriedades não estiver visível, dê um click com o botão direito do mouse no formulário e, no menu popup que abrir, selecione o item “Propriedades”.



Na caixa de listagem da janela de propriedades, selecione “Formulário” e na guia Formato selecione o item Legenda.

Digite o texto para a barra de título do formulário: “Formulário de Veículos”.

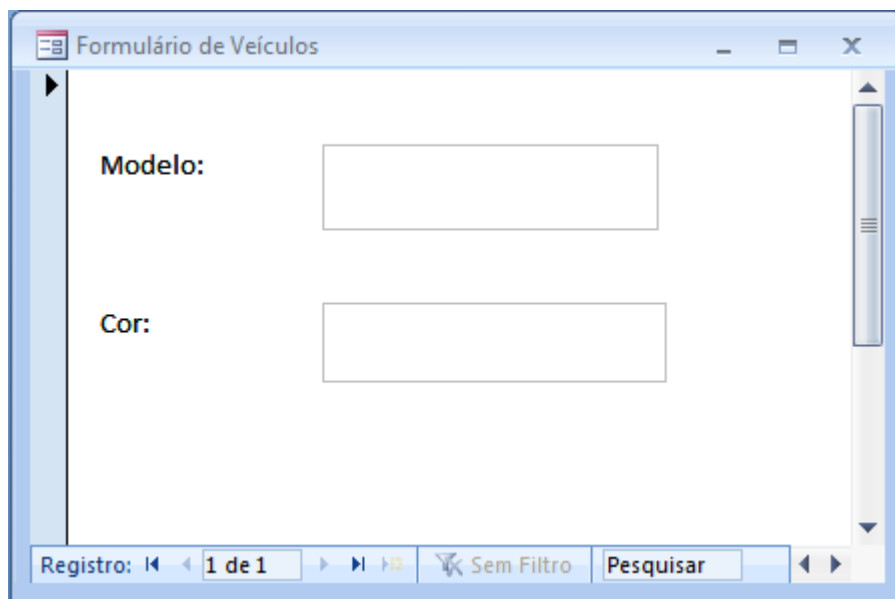
Em seguida, coloque duas caixas de texto no formulário e formate-as como abaixo:



Execute o formulário, pressionando a tecla F5. Seu formulário, em execução, deverá ficar parecido com a figura a seguir:

Obs.:

- 1) Se o formulário estiver maximizado, não será exibida a barra de título;
- 2) Se na guia "Todas", a propriedade do formulário "estilo da borda" estiver "Nenhum", não será exibida a barra de título.
- 3) Para que o formulário não fique maximizado, na guia "Todas", configure para "Sim", a propriedade "PopUp" do formulário.



Para voltar ao modo de design, click no formulário com o botão direito do mouse e selecione "Modo Design".

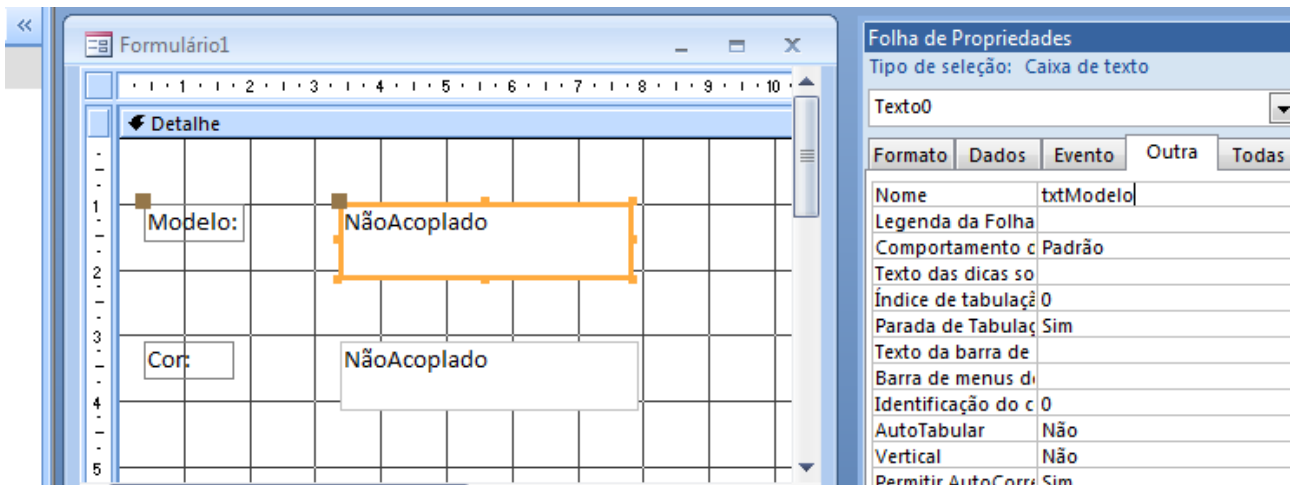
Vamos colocar nomes coerentes nos componentes do formulário. Os nomes deverão representar o tipo do componente e o que irão conter. Para representar o tipo de componente, usaremos a seguinte padronização:

- Nomes de caixa de texto devem iniciarão pelas letras txt – exemplo: txtModelo, txtCor etc.
- Nome de Label iniciará pelas letras lbl – exemplo: lblModelo, lblCor etc.
- Botão de comando iniciará pelas letras cmd – exemplo: cmdOk, cmdCancela etc.
- Botão de opção deverá iniciar pelas letras opt – exemplo: optopcao1, optopcao2 etc.

Outros tipos de nomes de componentes serão vistos oportunamente.

A padronização de nomes para componentes é muito importante para documentação e tem por objetivo facilitar o desenvolvimento e manutenção de programas.

Vamos atribuir nomes às caixas de texto de nosso formulário. Clique na caixa de texto que conterá o Modelo do veículo para selecioná-la, selecione a guia “Outra” e altere o Nome para txtModelo. Faça o mesmo para a caixa que conterá a cor do veículo.

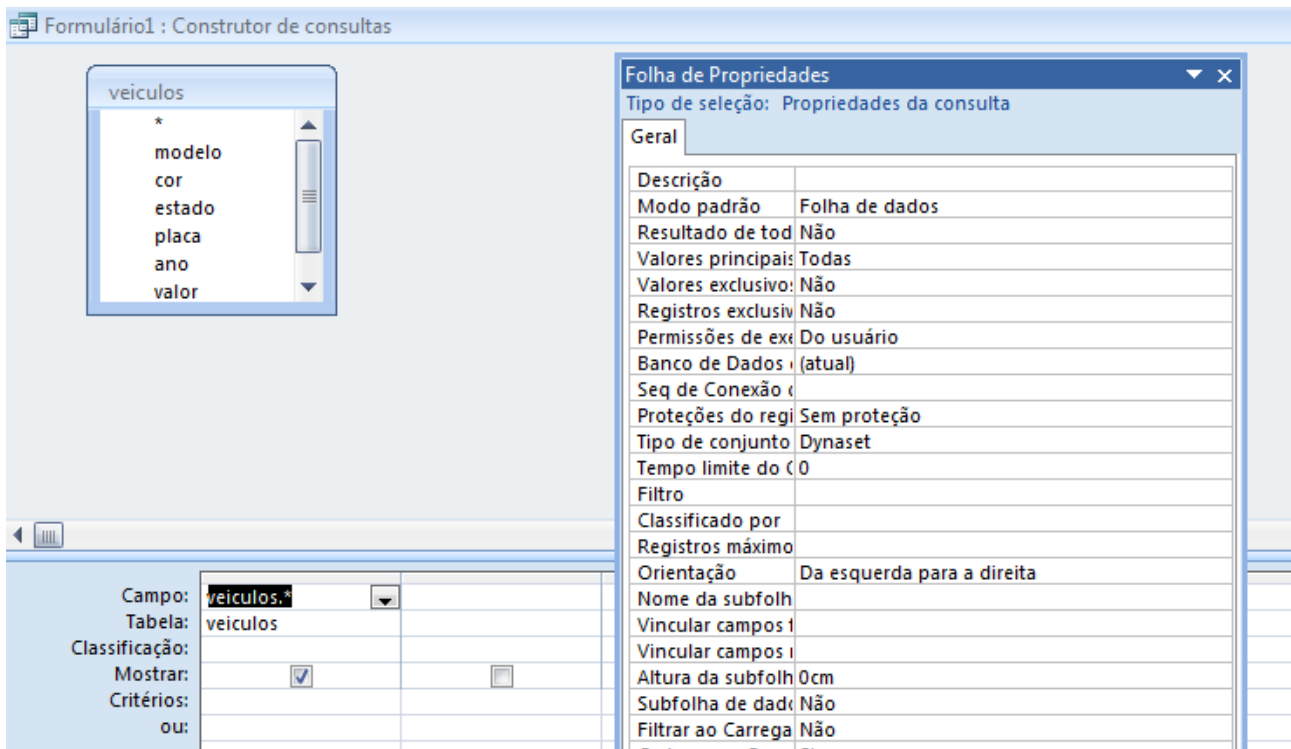


Vamos incluir em nosso formulário, a tabela de onde virão os dados que irão aparecer nas caixas de texto. Para isso, selecione o item “Formulário” na caixa de listagem da janela de propriedades, e selecione a guia “Dados”.


Selecione o item Fonte de registros e dê um click no ícone “...”. Selecione a tabela “veículos”, dê um click em Adicionar, em seguida dê um click no item Fechar.

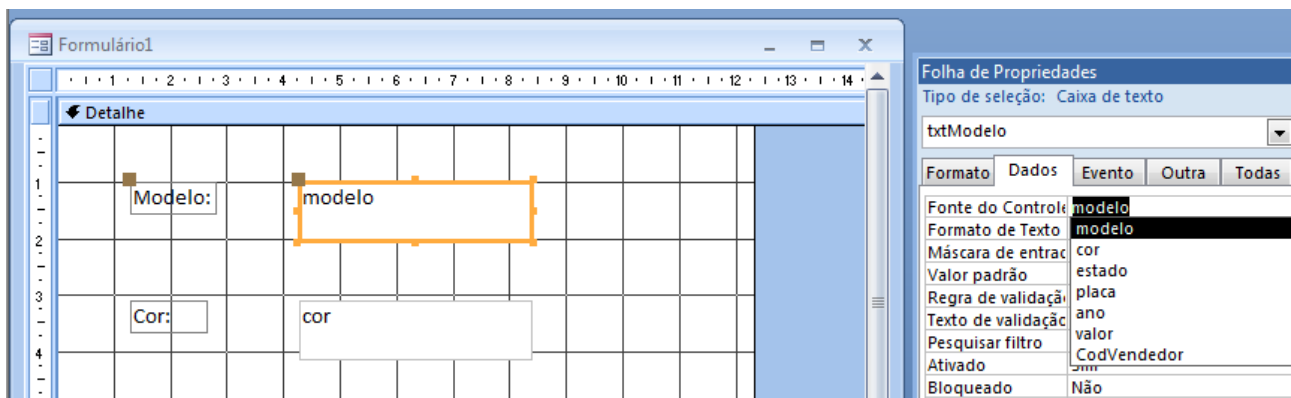
Na escolha do campo, selecione o item “veiculos.*”. Isso irá incluir em seu formulário, a tabela inteira.

Feche a janela “Formulário1 – construtor de consulta” e salve-a. Veja a figura a seguir:

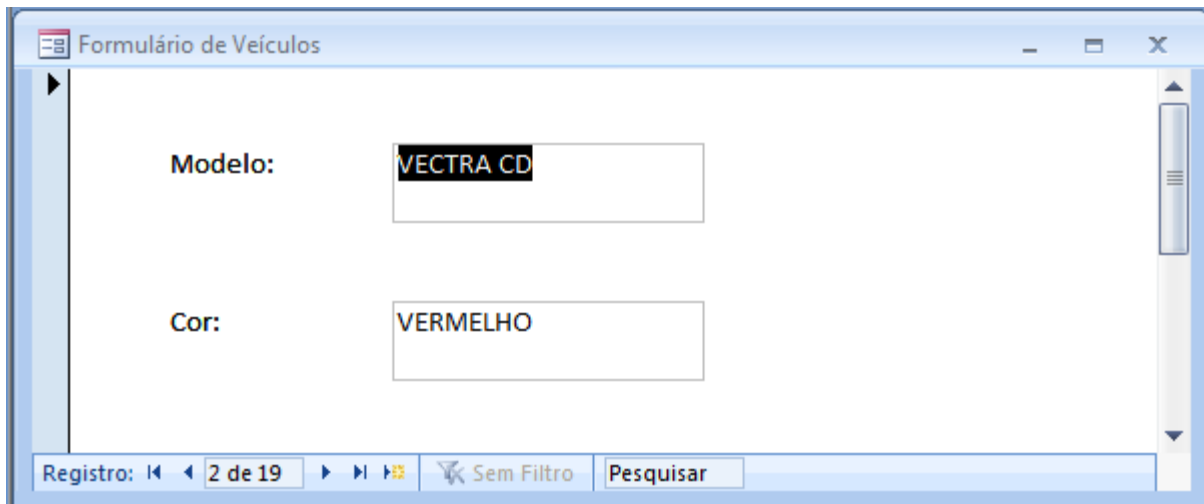


Vamos agora incluir os campos do banco de dados nos componentes do formulário:

- Selecione a caixa de texto txtModelo, clicando nela com o botão direito do mouse.
- Escolha o item propriedades.
- Na guia Dados, abra o ícone  da “Fonte de controle” e selecione o campo modelo,
- Faça o mesmo para o campo cor.



Ponha o foco no formulário e pressione a tecla F5 para ver o formulário em execução:



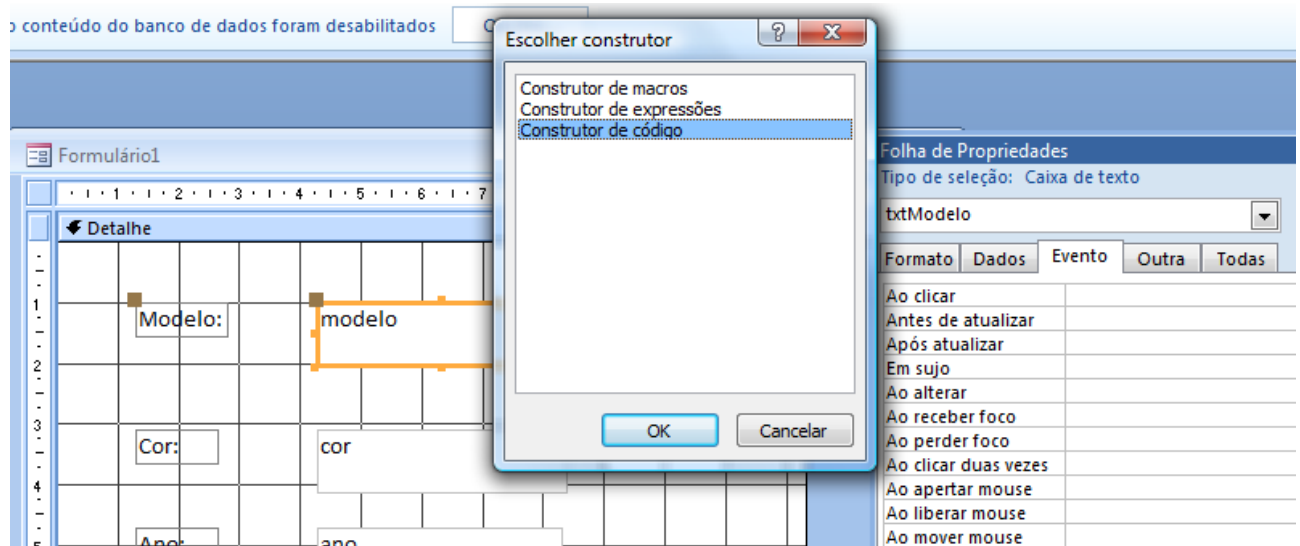
Obs. Se usarmos o assistente para criação de formulário, esses passos serão feitos automaticamente. Optamos por fazer manualmente visando a aquisição de habilidades que usaremos posteriormente.

Exercícios para aula 1 com Access:

- 1 - Inclua os campos ano e valor no formulário, **com nomes txtAno e txtValor**. O campo ano deve ser no formato número geral com zero decimais e o campo valor deve ser no formato unidade monetária com 2 decimais. As especificações do tipo do campo são definidas na guia “Formato”.
- 2 - Fazer programação para que ao ser inserido um novo registro, o que for digitado no campo modelo, fique em maiúsculo.

Procedimento:

- a) Selecione a caixa de texto txtModelo e na janela “Folha de Propriedades” escolha a guia Evento.
- b) Escolha o item “Ao pressionar tecla” e dê um click nos “...” da direita da linha.
- c) Na janela “Escolher construtor”, selecione “Construtor de código” e dê um click no botão Ok.



Será oferecido para programação, o tratador do evento KeyPress, que deverá ser programado da seguinte forma:

```
Private Sub txtModelo_KeyPress( KeyAscii As Integer )  
    KeyAscii = Asc ( UCase ( Chr$ ( KeyAscii ) ) ) ' Deixa os caracteres em maiúsculo.  
End Sub
```

Obs: Para alternar entre a janela do VBA e do banco de dados, pressione Alt + F11.

3 - Fazer programação para que o ano de um novo veículo a ser digitado seja validado. Ou seja, o ano só é válido se estiver entre 1900 e o ano atual + 1.

Para isso, selecione a caixa de texto txtAno e programe o evento “Antes de atualizar”, como abaixo:

```
Private Sub txtAno_BeforeUpdate(Cancel As Integer)  
    If Val(txtAno.text) < 1900 Or Val(txtAno.text) > Year(Date) + 1 Then  
        MsgBox " Ano inválido... Digite o ano correto!", vbCritical, "A T E N Ç Ã O"  
  
        Cancel = True 'Cancela a atualização  
    End If  
End Sub
```

4 – O campo “valor” deve aceitar apenas valores numéricos e a vírgula decimal. A programação é a seguinte:

```
Private Sub txtValor_KeyPress(KeyAscii As Integer)  
    Dim TECLA As String  
    TECLA = Chr$(KeyAscii)  
    If (TECLA < "0" Or TECLA > "9") And TECLA <> "," Then  
        Beep  
        KeyAscii = 0  
    End If  
End Sub
```

5 – O campo valor deve aceitar apenas números e apenas **uma única vírgula decimal**.

Obs.: a função InStr(txtValor.Text, ",") retorna 0 se não achou a “,” ou retorna a posição onde encontrou a “,”. Verifique o Help on line.

6 – O campo ano deve aceitar apenas números e no máximo 4 caracteres. Se for digitado mais de 4 caracteres, exibir mensagem de alerta. Utilize a função LEN que retorna o comprimento de um string – use o help para verificar.

7 – No campo “cor” todos os caracteres digitados devem estar em maiúsculo.

8 – Os campos “ano” e “valor” devem aceitar a tecla BackSpace. O código ASCII dessa tecla é 8. Portanto, se keyAscii for igual a 8, a tecla será válida.

9 – Se uma tecla inválida for digitada, o programa deverá exibir mensagem de alerta.

- 10 – Inclua no formulário, os campos Estado e Placa. Todos os caracteres devem ficar em maiúsculo.
- 11 – Insira no banco de dados, através dos botões de navegação do formulário, os registros abaixo:

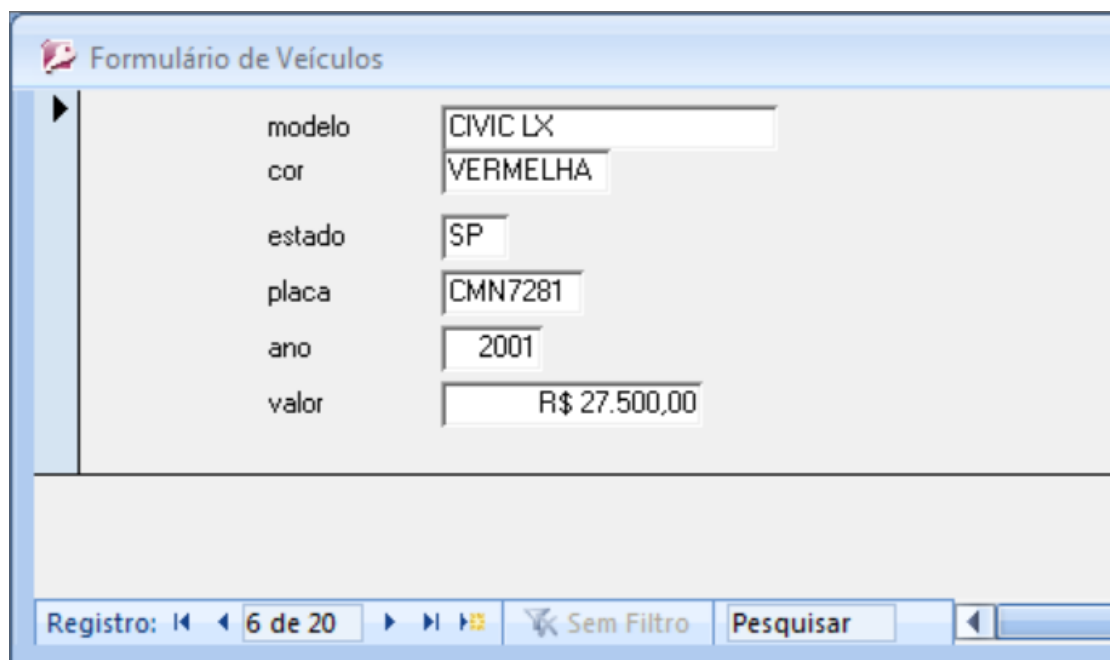
	MODELO	COR	ESTADO	PLACA	ANO	VALOR
1	CORSA	BEGE	RJ	CAU3254	1998	8000,00
2	CORSA	VERDE	SP	BBJ2429	2000	12000,00
3	CORSA	BRANCO	BA	EBF9553	2002	15000,00
4	CORSA	PRETO	SP	GTH8865	1998	7500,00

- 12 – Todos os campos devem aceitar a tecla “Tab”. O valor ascii em decimal da tecla Tab é 9. Pressionando-se a tecla Tab, a ordem de tabulação deve ser adequada ao usuário (de cima p/ baixo) – verifique.

Salvar o arquivo “Revenda” e preparar para apresentar o formulário ao professor.

Aula 2 de Access.

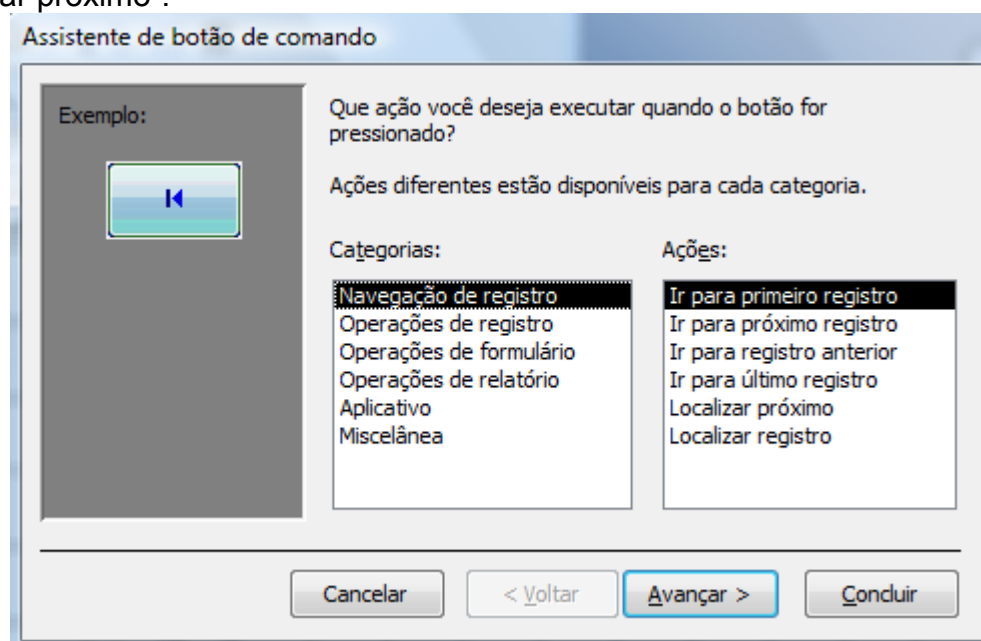
Abrir o arquivo Revenda utilizado na aula anterior. O formulário deve estar com a seguinte aparência:



Vamos personalizar um pouco nosso formulário e depois iremos remover o componente padrão de navegação de registros. Para isso, vamos selecionar o ícone do componente “botão de comando” na guia Design, e vamos incluí-lo no formulário. Note que o VBA abre uma janela para escolhermos qual a categoria de botão nós queremos. Vamos selecionar a categoria Navegação de registro.

Inclusão de botões no formulário do Access:

Vamos inserir em nosso formulário, **todos os botões dessa categoria**, exceto o botão “localizar próximo”.



Para cada botão que você inserir, click no botão Avançar e configure as propriedades do botão a seu gosto. É muito importante atribuir um nome de acordo com nossa padronização, como por exemplo, para o botão “Ir para primeiro registro”, o nome cmdPrimeiroRegistro, cmdProximo etc..

Faça isso para todos os botões.

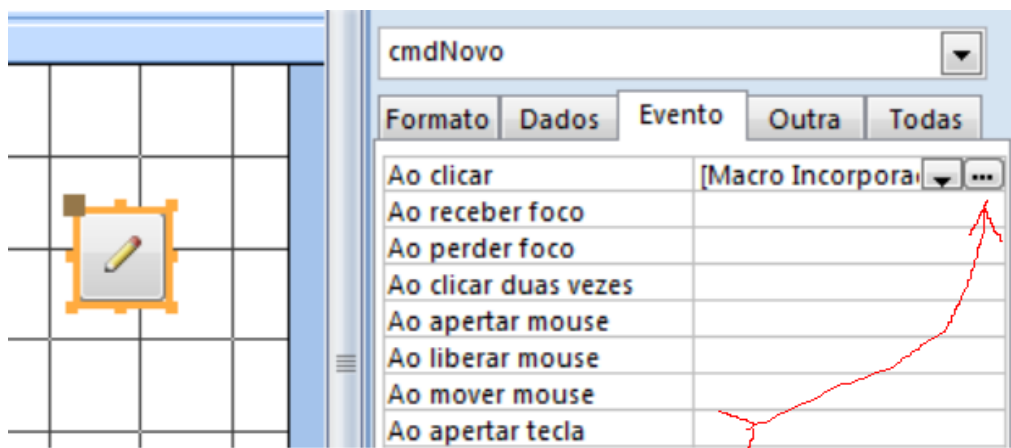
Exercício: Coloque mais dois botões da categoria Operações de registro: Adicionar novo registro e Excluir registro. Mude seus nomes para cmdAdicionaNovo e cmdExcluir, por exemplo.

Nosso formulário ficará mais ou menos assim:



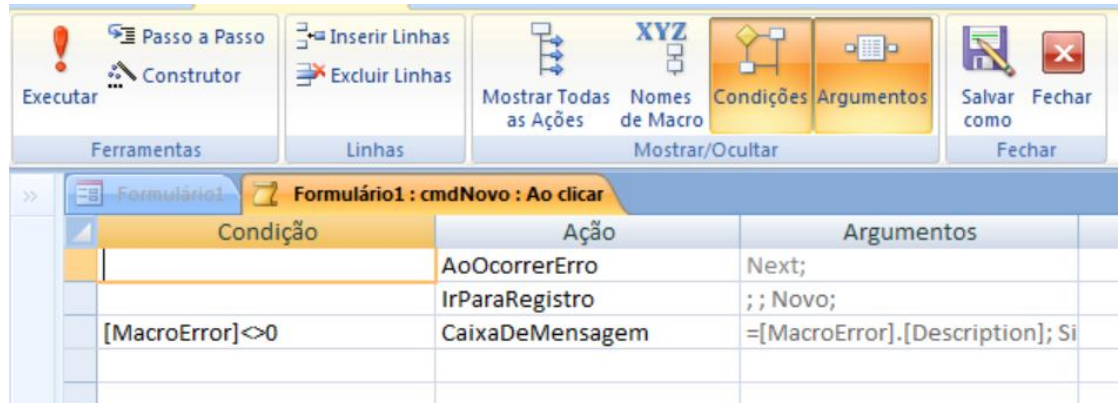
Agora que já temos nossos próprios botões de controle dos registros, não precisamos mais do controle fornecido automaticamente com o relatório. Para eliminá-lo, no modo de design do formulário, coloque “Não” no item Formulário / Formato / Botões de navegação.

Execute o formulário para verificar e teste todos os botões que você colocou. Note que para cada botão que você incluiu, o editor de VBA do Access providenciou uma "Macro Incorporada" que faz o trabalho de acordo com as funções do botão.



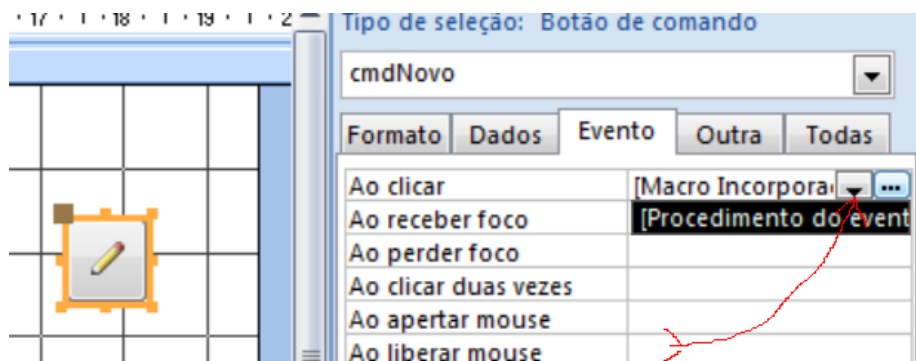
Se for essa a única ação esperada quando ocorrer o click no botão, então está tudo pronto para uso desse botão pois essa macro faz apenas isso. Veja as instruções da macro, dando um click nos (...), apontado pela seta da figura acima.

Aí está a macro incorporada! Ela é feita com instruções especiais diferentes de VBA.



Suponha que após pressionarmos o botão cmdNovo, queremos que o foco vá para a caixa de texto txtModelo ou queremos realizar algumas outras ações. Nesse caso não devemos usar o recurso de macro incorporada e sim o procedimento do evento.

Para isso, dê um click no botão apontado na figura abaixo, selecione o "Procedimento do evento" e click no botão com os (...).



E lá está nosso velho conhecido, tratador do evento Click do botão cmdNovo.

Digite nele o código abaixo e teste-o:

```
Private Sub cmdNovo_Click()  
On Error GoTo Err_cmdNovo_Click  
  
docmd.GoToRecord , acNewRec  
  
Exit_cmdNovo_Click:  
Exit Sub  
  
Err_cmdNovo_Click:  
MsgBox Err.Description  
Resume Exit_cmdNovo_Click  
End Sub
```

Verifique que o código para inserir um novo registro no banco de dados funciona do mesmo modo que quando havia a macro incorporada. Porém, agora podemos incluir a codificação que quisermos, como por exemplo, após o novo registro ser criado, mandamos o foco para a caixa de texto txtNome. A instrução que faz isso é:

```
txtNome.setfocus.
```

Onde você irá incluí-la? Faça isso e teste novamente o uso do botão.

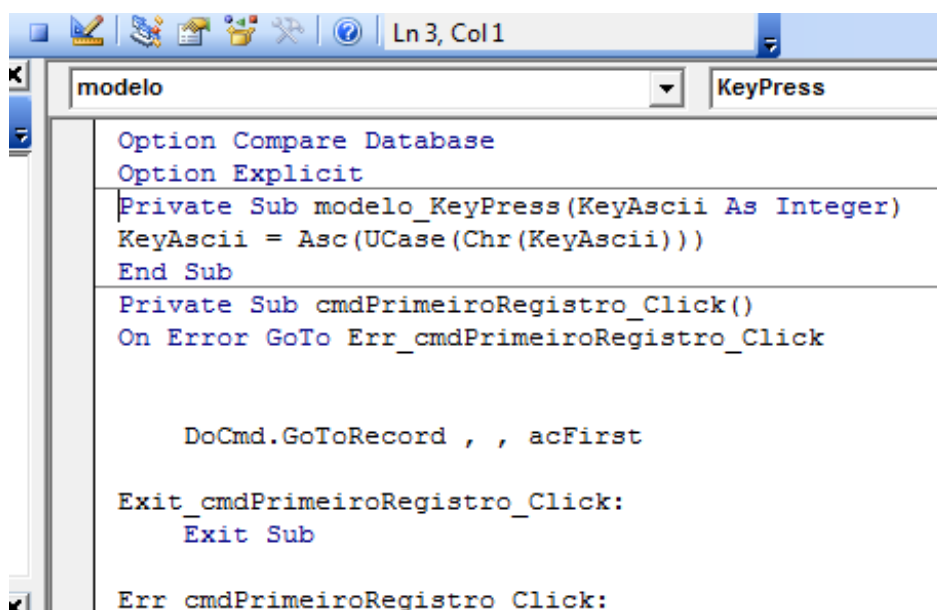
Uso de DoCmd:

Note a palavra “DoCmd” no código acima. Dizemos que DoCmd é um objeto do modelo de objetos do Access. O objeto DoCmd suporta vários “métodos” que provocam várias ações na programação em VBA para o Access.

Veja a seguir, a programação em VBA associada ao evento click no botão cmdProximo, onde o objeto DoCmd executa o método GoToRecord, com o argumento acNext:

```
Private Sub cmdProximo_Click()  
On Error GoTo Err_cmdProximo_Click  
  
DoCmd.GoToRecord , , acNext  
  
Exit_cmdProximo_Click:  
Exit Sub  
  
Err_cmdProximo_Click:  
MsgBox Err.Description  
Resume Exit_cmdProximo_Click  
  
End Sub
```

Na figura abaixo vemos um trecho de código associado ao botão que ao receber o click, exibe no formulário o primeiro registro.



Outros exemplos de uso de DoCmd:

DoCmd.Maximize ' maximiza o formulário onde está inserido.

DoCmd.Close ' fecha o formulário.

DoCmd.Quit ' fecha o Access.

DoCmd.OpenForm "frmMenu" 'exibe o formulário de nome frmMenu.

Exercícios Access Aula 2:

Se um veículo do banco de dados for vendido, o vendedor deverá informar a venda, a fim de receber sua comissão.

Para que o vendedor possa fazer isso, inclua no formulário, o botão “Vender”, que ao ser pressionado solicita o código do vendedor. Esse código deverá ser inserido no campo “codvendedor” do banco de dados Veículos. A digitação do código poderá ser feita por uma InputBox.

Inclusão de um botão comum no formulário (sem programação automática):

Para incluir um botão comum no formulário proceda da seguinte maneira: Inclua um botão no formulário e na janela apresentada, onde aparecem as categorias de botão oferecidas, pressione a tecla Cancelar.

Na janela de propriedades, selecione a guia Outra e mude o nome para cmdVender.

Na guia Formato, mude a legenda para &Vender. Com a legenda dessa maneira, você poderá executar o programa associado ao botão, pressionando Alt + V.

O código do vendedor a ser digitado deverá possuir dígito de controle que será verificado através de cálculo pelo módulo 11.

O dígito verificador (dv), ou dígito de controle (dc) calculado pelo módulo 11 é bastante seguro, e é usado para validação de CPF, RG, CGC e vários outros códigos oficiais.

Um dígito de controle é calculado pelo módulo 11 da seguinte maneira : Por exemplo, supondo-se o código de vendedor 200506003, fazem-se os cálculos.

```
3 x 2 = 6
0 x 3 = 0
0 x 4 = 0
6 x 5 = 30
0 x 6 = 0
5 x 7 = 35
0 x 8 = 0
0 x 9 = 0
2 x 10 = 20
+ -----
      91
```

Pega-se o resultado da soma (91), divide-se por 11 e pega-se o resto, que neste caso é 3.

$$91 / 11 = 8 \text{ Resto} = 3$$

Daí, faz-se $11 - 3 = 8$. Pronto. O dígito calculado é 8, e o código do vendedor, com dígito, ficará: 200506003-8

Se o resto da divisão por 11 for 0 ou 1, o dígito é o próprio resto.

Na programação usamos a função Mod do VBA, que retorna o resto da divisão de um número inteiro por outro. Exemplo `DigitoLido=CodigoLido mod 10`

Nas divisões, para truncar o número em sua última casa, usamos a função Int para que o VBA não cause arredondamento no resultado da divisão. Por exemplo, `codigo = Int(codigo / 10)`

Acompanhe o programa a seguir:

```
Private Sub cmdVender_Click()  
    Dim codigolido As Long  
    Dim digitolido As Integer  
    Dim codigo As Long  
    Dim digito As Integer  
    Dim soma As Integer  
    Dim mult As Integer  
    mult = 2  
    soma = 0  
  
    codigolido = Val(InputBox(" Digite o código do vendedor: ", " V e n d a   d e   V e í c u l o"))  
    digitolido = codigolido Mod 10  
    codigo = Int(codigolido / 10)  
  
    While codigo > 0  
        soma = soma + (codigo Mod 10) * mult  
        mult = mult + 1  
        codigo = Int(codigo / 10)  
    Wend  
  
    digito = soma Mod 11  
    If digito > 1 Then digito = 11 - digito  
  
End Sub
```

O programa acima recebe um código numérico de vendedor com dígito, separa esse dígito e calcula, com o restante do número lido, um outro dígito com a finalidade de verificar se o dígito calculado confere com o dígito lido.

Exercícios:

- 1 – Verificar se o usuário digitou apenas números. Caso tenha sido digitado algum caractere não numérico, exibir mensagem de erro e abandonar o procedimento. Utilize a função IsNumeric. A programação ficará assim:

```
Dim codigostr As String  
codigostr = InputBox(" Digite o código do vendedor: ", "   V e n d a   d e   V e í c u l o")  
' verifica se foram digitados só numeros.  
If Not IsNumeric(codigostr) Then
```

```
MsgBox "Número inválido! Digite apenas números...", vbCritical, "ATENÇÃO"
Exit Sub
End If
```

2 - Verificar se o dígito calculado é igual ao dígito lido. Se for igual, aceitar o código do vendedor e incluí-lo na tabela "Veículos", no campo codvendedor, e exibir mensagem "Venda efetuada com sucesso!"

3 - Se o dígito não "bater", exibir mensagem de erro ao usuário e abandonar o procedimento.

O código será mais ou menos como abaixo:

```
If digitolido = digito Then
    codvendedor = codigolido ' codvendedor é campo do Banco de Dados.

    ' A instrução abaixo salva o registro corrente - grava o código do vendedor.
    DoCmd.RunCommand acCmdRefresh

    MsgBox "Venda efetuada com sucesso!"
Else
    MsgBox "Dígito não bate. Tente novamente....", vbExclamation, "A T E N Ç Ã O"
Exit Sub
End If
```

4 – Se a venda for concretizada e após ser inserido com sucesso o código do vendedor, o botão cmdVender deverá ficar **desabilitado**, com caption "Vendido". Usar a instrução cmdVender.enabled = false.

5 – Ao navegar pelo banco de dados, se o veículo estiver à venda, ou seja, se o campo de código de vendedor ainda estiver em branco, o botão cmdVender deverá ficar **habilitado**, e com caption = &Vender. Caso contrário, ou seja, se o veículo já foi vendido, consequentemente o campo "codvendedor" estará preenchido. Nesse caso, o botão cmdVender deverá estar **desabilitado**, com caption "Vendido".
A programação será mais ou menos como a seguir:

Obs.: No Access, para verificar se um campo do banco de dados contém o caráter nulo, usamos a função IsNull. Veja o exemplo abaixo:

```
If IsNull(codvendedor) Then
    cmdVender.Caption = "&Vender"
    cmdVender.Enabled = True
Else
    cmdVender.Caption = "Vendido"
    cmdVender.Enabled = False
End If
```

A programação acima deverá ser codificada em todos os botões de navegação.

SALVE O PROJETO E PREPARE-SE P/ APRESENTÁ-LO AO PROFESSOR.

AULA 3 DE ACCESS:

Abra o arquivo de Access Revenda da aula anterior e teste-o p/ verificar se tudo funciona de acordo com as especificações anteriormente fornecidas.

Exercícios:

1 – Com objetivo de deixar o código do evento Click do botão cmdVender mais enxuto e mais elegante, criar a função “CalculaDC” que recebe o código do vendedor, sem dígito de controle, calcula e retorna o dígito calculado com base no código recebido. Utilize essa função no evento click do botão cmdVender, ao invés daquele código enorme.

2 – Ao pressionar o botão “Inserir Registro” o foco deve ir para a caixa de texto txtModelo. Verifique se a ordem de tabulação está correta. Se não estiver, corrija-a. A propriedade que especifica a ordem de tabulação é “Índice de tabulação”. Esta inicia por zero e vai seqüencialmente de acordo com a tabulação desejada.

Caso você tenha dúvidas quanto a partes do código, veja exemplos nas próximas páginas.

Lembre-se que a programação final do evento Click do botão cmdVender estava mais ou menos assim (sem uso da função CalculaDC):

```
Private Sub cmdVender_Click()  
    Dim codigostr As String  
    Dim codigolido As Long  
    Dim digitolido As Integer  
    Dim codigo As Long  
    Dim digito As Integer  
    Dim soma As Integer  
    Dim mult As Integer  
    mult = 2  
    soma = 0  
  
    codigostr = InputBox(" Digite o código do vendedor: ", " V e n d a   d e   V e í c u l o")  
  
    ' verifica se foram digitados só numeros.  
    If Not IsNumeric(codigostr) Then  
        MsgBox "Número inválido! Digite apenas números...", vbCritical, "ATENÇÃO"  
        Exit Sub  
    End If  
  
    codigolido = Val(codigostr)  
    digitolido = codigolido Mod 10  
    codigo = Int(codigolido / 10)  
  
    While codigo > 0  
        soma = soma + (codigo Mod 10) * mult  
        mult = mult + 1  
        codigo = Int(codigo / 10)  
    Wend  
  
    digito = soma Mod 11
```



```
If digito > 1 Then digito = 11 - digito
```

```
If digitolido = digito Then
```

```
    codvendedor = codigolido ' codvendedor é campo do BD
```

```
    ' A instrução abaixo salva o registro corrente - grava o código do vendedor.
```

```
    DoCmd.RunCommand acCmdRefresh
```

```
    MsgBox "Venda efetuada com sucesso!"
```

```
Else
```

```
    MsgBox "Digito não bate. Tente novamente....", vbExclamation, "A T E N Ç Ã O"
```

```
Exit Sub
```

```
End If
```

```
End Sub
```

Quando você fizer e utilizar a função CalculaDC, o programa deve ficar mais ou menos assim:

Function CalculaDC(novocodigo As Long) As Integer

```
Dim codigo As Long
```

```
Dim digito As Integer
```

```
Dim soma As Integer
```

```
Dim mult As Integer
```

```
codigo = novocodigo
```

```
mult = 2
```

```
soma = 0
```

```
While codigo > 0
```

```
    soma = soma + (codigo Mod 10) * mult
```

```
    mult = mult + 1
```

```
    codigo = Int(codigo / 10)
```

```
Wend
```

```
digito = soma Mod 11
```

```
If digito > 1 Then digito = 11 - digito
```

```
CalculaDC = digito 'Retorna o valor do digito calculado
```

End Function

Private Sub cmdVender_Click()

```
Dim codigostr As String
```

```
Dim codigolido As Long
```

```
Dim codigo As Long
```

```
Dim digitolido As Integer
```

```
On Error GoTo ExibeErro:
```

```
codigostr = InputBox(" Digite o código do vendedor: ", " V e n d a d e V e í c u l o")
```

```
    ' verifica se foram digitados só numeros.
```

```
If Len(codigostr) > 10 Then
```

```
    MsgBox "Código com mais de 10 dígitos - redigite!"
```

```
Exit Sub
```

```
End If
```

```
If Not IsNumeric(codigostr) Then
    MsgBox "Número inválido! Digite apenas números...", vbCritical, "ATENÇÃO"
    Exit Sub
End If

codigolido = Val(codigostr) 'converte o código p/ número
digitolido = codigolido Mod 10 'separa o dígito lido
codigolido = Int(codigolido / 10) 'tira o dígito do código - trunca

If digitolido = CalculaDC(codigolido) Then 'Chamada da função VerificaDC
    'o dígito bateu...
    codvendedor = Val(codigostr) 'codvendedor é campo do BD

    'A instrução abaixo salva o registro corrente - grava o código do vendedor.

    DoCmd.RunCommand acCmdRefresh 'Grava registro corrente

    MsgBox "Venda efetuada com sucesso!"
    cmdVender.Caption = "Vendido"
    cmdProximo.SetFocus
    cmdVender.Enabled = False
Else 'No else, o dígito não bateu...
    MsgBox "Dígito não bate - redigite!", vbCritical, "A T E N Ç Ã O"
    Exit Sub
End If
Exit Sub
ExibeErro:

MsgBox Error(Err)

End Sub
```

Alguns problemas poderão ocorrer, que deverão ser solucionados:

Verifique os aspectos da estética do formulário: alinhamento e tamanho dos componentes, bom balanceamento de cores (não deve ser um carnaval e nem totalmente cinza), etc.

A seguir algumas dicas sobre problemas observados em projetos anteriores:

- 1) Ao inserir um novo registro o foco deve ir para txtModelo (txtModelo.SetFocus).
- 2) Verifique que ao inserir um novo registro, se o campo de código do vendedor estiver com o valor zero, mesmo assim o botão cmdVender deverá estar habilitado.
- 3) Se o campo do banco de dados de código de vendedor estiver preenchido, significa que aquele veículo já foi vendido. Portanto o botão cmdVender deverá estar desabilitado com o caption "Vendido".
- 4) Efetue a venda do primeiro veículo da tabela, notando que o botão cmdVender torna-se desabilitado, com caption "Vendido". **Feche o Access e abra-o novamente**, acessando o formulário de vendas. O botão cmdVender está desabilitado e com caption "Vendido"? Para resolver esse problema, programe adequadamente o tratador de evento FormLoad. Ele é executado quando o

formulário é carregado na memória e tem por objetivo atribuir condições iniciais a objetos, inicializar variáveis, etc.

Faça uma revisão geral até este ponto do projeto. O formulário **deve estar minimamente** com a seguinte aparência:

Continuação: Dando continuidade a nosso projeto, faça os exercícios a seguir:

- 1 - Incluir no projeto um formulário de nome frmVendedor, contendo todos os campos da tabela "vendedor". Mude os nomes das caixas de texto desse formulário, para nomes compatíveis com caixas de texto, de acordo com o padrão adotado por nós: txtCodVendedor, txtNome, txtEndereco, txtEstadoCivil, txtSalarioFixo.
- 2 - Incluir botões para navegação no banco de dados e um botão do grupo "Operações de registro" com ação "Adicionar registro", para podermos incluir novos vendedores em nosso banco de dados.
- 3 – Eliminar o componente padrão de navegação do relatório.
- 4 – A caixa de texto "Salário Fixo" deve aceitar somente números e uma única vírgula decimal.

Esse formulário deverá **ficar minimamente com a seguinte aparência:**

- 4 - Para que o usuário não possa alterar os registros já existentes no banco de dados, programar o evento Form_load() para desabilitar as caixas de texto do formulário:

```
Private Sub Form_Load()  
    txtcodvendedor.Enabled = False  
    txtNome.Enabled = False  
    txtEndereco.Enabled = False  
    txtEstadoCivil.Enabled = False  
    txtSalarioFixo.Enabled = False  
End Sub
```

- 5 – Programar o evento click do botão cmdAdicionaVendedor para que:

- Desabilite todos os botões, inclusive ele próprio (exceto o botão Ok).
- Solicite o código do novo vendedor (via InputBox)
- Verifique se o código digitado é composto só de números (função IsNumeric)
- Solicite que o código seja digitado novamente, para verificar se é igual ao anterior.
- Utilize a função CalculaDC para calcular o dígito de controle do código do novo vendedor.
- Inclua esse dígito no código do novo vendedor, coloque-o no campo codvendedor do banco de dados de um registro novo.
- Libere as demais caixas de texto para digitação (propriedade enabled = true) e exiba mensagem para o usuário digitar os dados do novo vendedor e em seguida pressionar o botão “Ok”.
- Ponha o foco na caixa de texto txtNome, para o usuário começar a digitação.
- Após o botão Ok ser pressionado (evento click), os conteúdos das caixas de texto: código, nome, endereço etc., deverão ser gravados no banco de dados e este deve ser atualizado (DoCmd.RunCommand acCmdRefresh).
- Os botões deverão ser novamente habilitados e as caixas de texto deverão ser novamente desabilitadas.
- A caixa de texto SalarioFixo deve aceitar apenas números e um único ponto decimal.

Obs.: Para que uma função ou procedimento possam ser utilizados por todos os formulários do banco de dados, necessitam estar dentro de um “Módulo”. Portanto, inclua um módulo em seu projeto e copie a função CalculaDC para dentro dele.

- 6- Inclua na tabela Vendedor os registros abaixo, através do formulário frmVendedor que você acabou de criar. Note que o código de vendedor da tabela abaixo já está com dígito de controle. **Você deve fornecê-lo sem o dígito.** O sistema irá calcular o dígito de controle pelo módulo 11 e o incluirá no código do vendedor:

	codvendedor	nome	endereço	estado civil	salário fixo
▶	2004100095	Washington Luiz	Pça Patriarca, 237	Casado	R\$ 800,00
	2005060038	Aluno DTI-ADS	Pça Cel F. Prestes 30	Solteiro	R\$ 2.000,00
	2005080128	Marlene Casagrande	R. Itumbiara, 87	Solteira	R\$ 600,00
	2007050031	Arthur Pujol Viana	R. Alcides Ribeiro	Solteiro	R\$ 1.500,00

Após verificar se tudo funciona, faça os acertos finais nas tabelas Veiculos e Vendedor, do banco de dados Revenda:

Verifique que na tabela Veiculos não deve haver registros de veículos usados para teste, ou seja, com nomes do tipo “ABC”. Se um veículo foi vendido o conteúdo do campo “codvendedor” deve ser um código válido, constante na tabela de vendedores.

Efetue a venda de alguns veículos, verifique se as tabelas Veiculos e Vendedor estão com os dados corretos.

SALVE O PROJETO E PREPARE-SE PARA APRESENTÁ-LO..

Aula 4 de Access.

Abra o arquivo Revenda, teste-o e verifique se tudo funciona a contento de acordo com o que foi especificado.

Até agora, temos dois formulários: Formulário de Veículos e Cadastro de Vendedor.

Nesta aula criaremos um relatório de vendas efetuadas e um relatório de veículos com totais e quebras. Para exibição desses dois relatórios, criaremos um formulário.

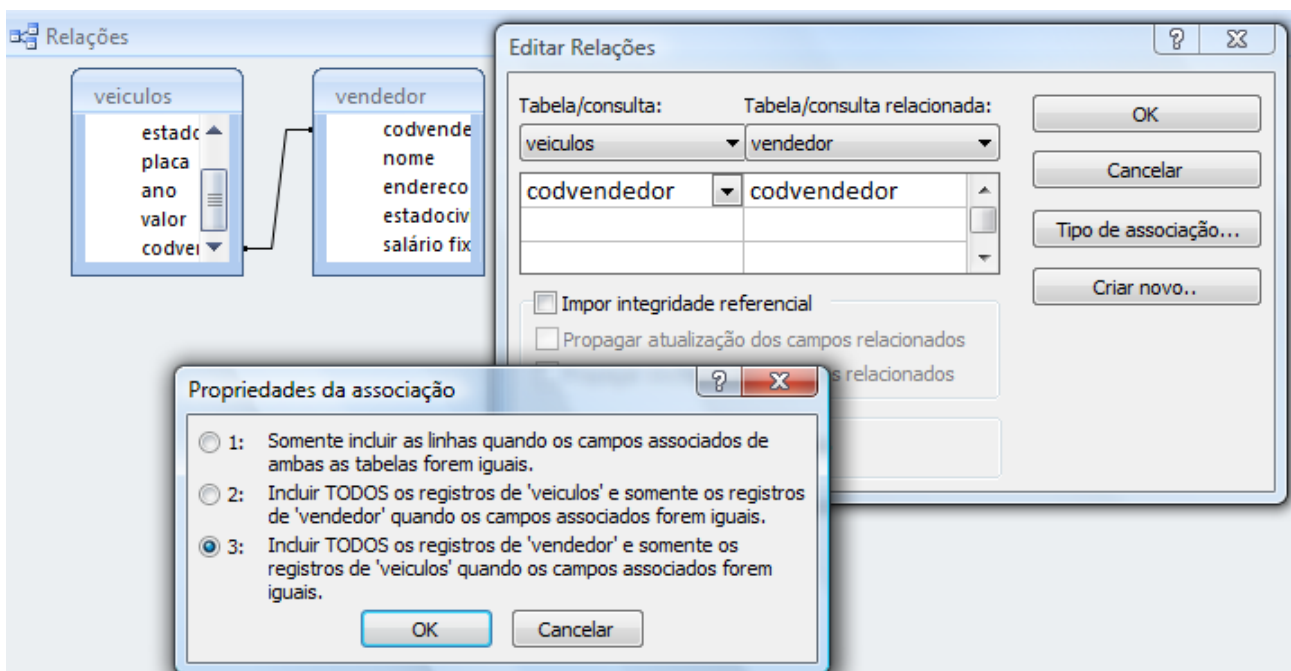
Uso de relatórios com Access: Execute os procedimentos a seguir:

Criação do primeiro relatório.

Nosso próximo passo será criar um **RELATÓRIO** que listará os vendedores e seus salários, que serão compostos pelo salário fixo (da tabela Vendedor), somados com a comissão de 10% do valor de cada veículo (da tabela Veículos) vendido pelo vendedor.

Para iniciar, acione a guia Ferramentas de Banco de Dados e dê um click no ícone “Relações”.

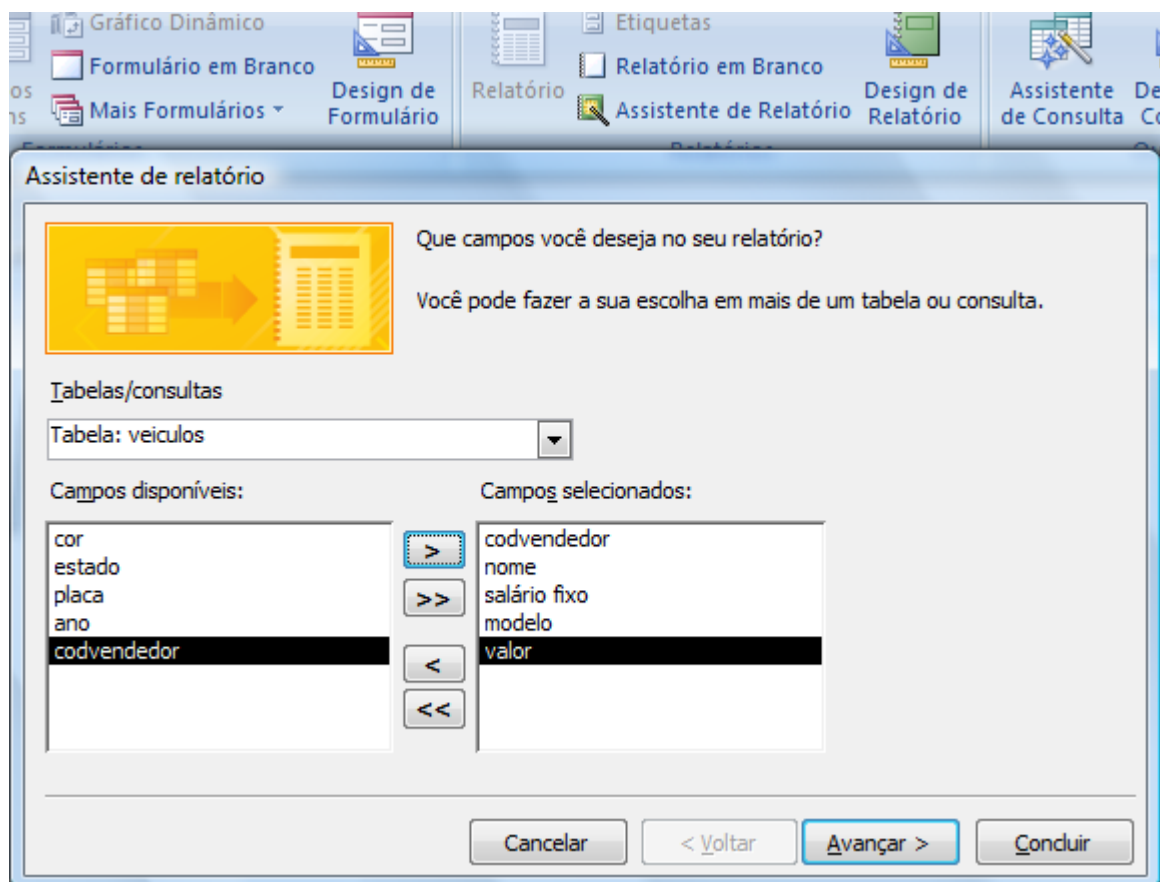
Na janela de Relações do Access crie um relacionamento entre os campos codvendedor entre a tabelas Veiculos e Vendedor. Não imponha integridade referencial.



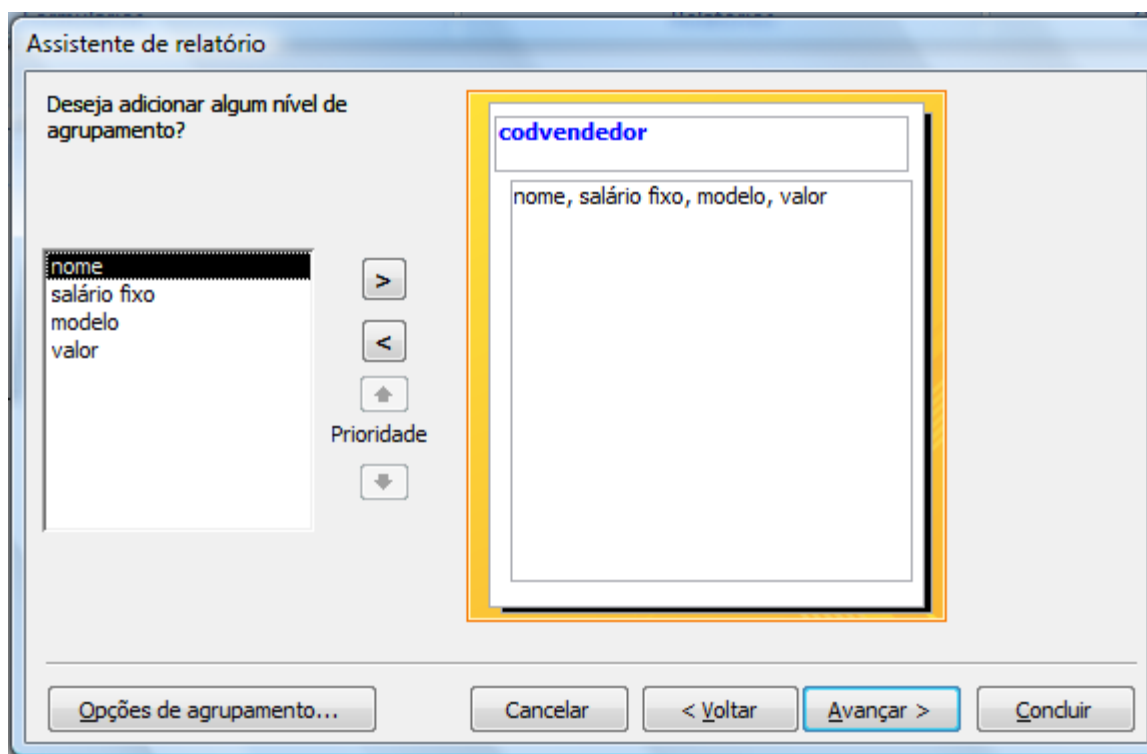
Dê um click no botão “Tipo de associação...” e em “Propriedades de associação”, escolha o item “**Incluir TODOS os registros de `vendedor` e somente...**”. Click nos botões Ok para fechar essas janelas.

Agora estamos prontos para criar nosso relatório. Volte à janela de objetos, selecione a guia “Criar” e dê um click no ícone “Assistente de Relatório”. Vamos criar nosso relatório usando o assistente e posteriormente iremos personalizá-lo.

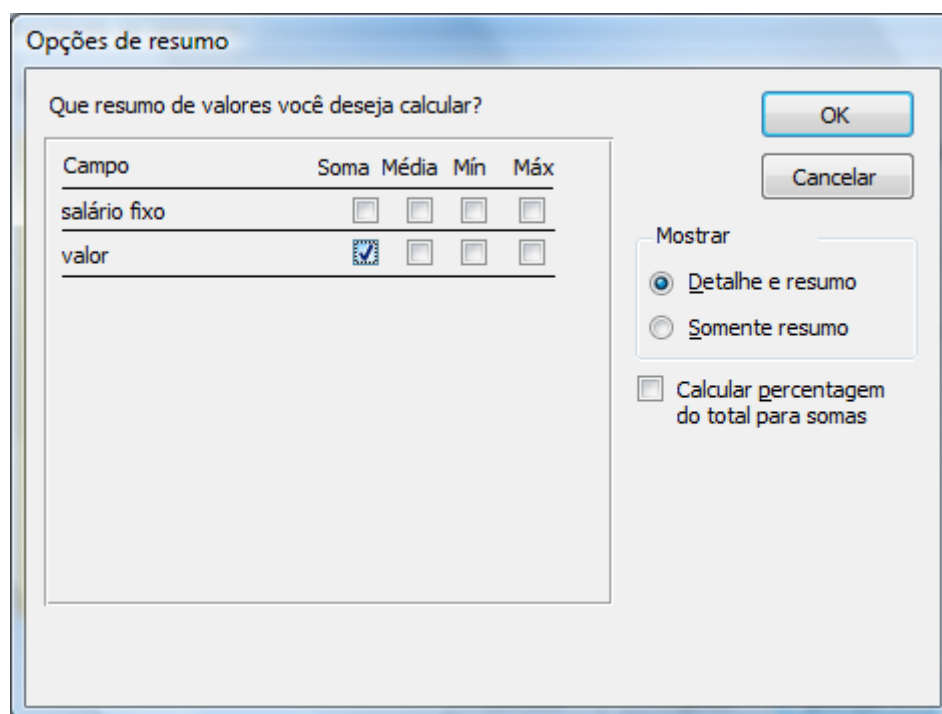
Crie o relatório de nome Relat_Salário, usando o assistente. Selecione a tabela “vendedor” e inclua os campos codvendedor, nome e salário fixo. Selecione agora a tabela “veiculos” e inclua os campos modelo e valor.



Dê um Click no botão Avançar e na próxima tela escolha o agrupamento por codvendedor. Click novamente em Avançar.

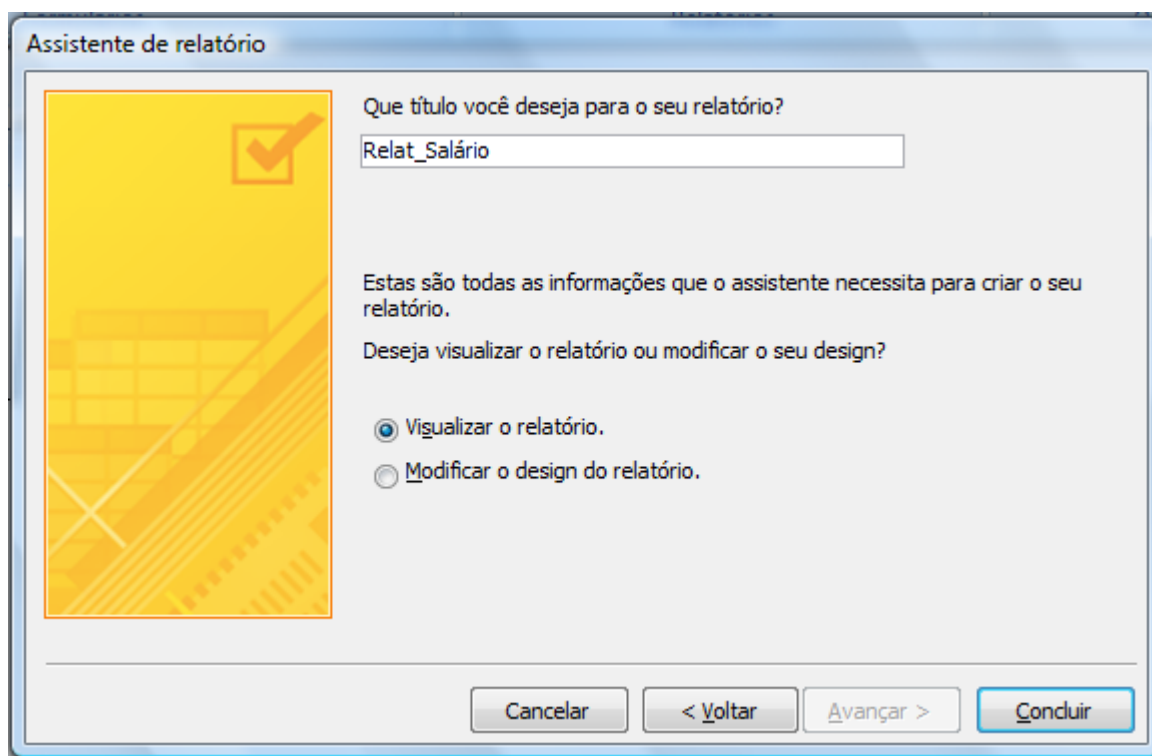


Na próxima tela escolha o campo “valor” como ordem de classificação e em “Opções de resumo” escolha Soma dos campos “valor”.



Click novamente no botão Avançar, e na próxima janela escolha layout Bloco.

Na próxima janela escolha o estilo que gostar mais. Na última janela escolha um título sugestivo para o relatório, como por exemplo “Relat_Salario”.



Dê um click no botão “Concluir” para ver o relatório criado:

Relat_Salário

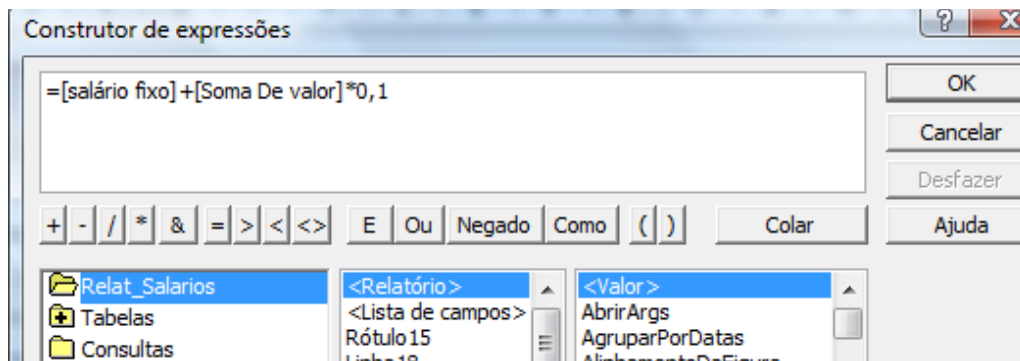
codvendedor	valor	nome	salário fixo	modelo
2004100095	45500	Washington Luiz	R\$ 800,00	PASSAT 2.8
	32500	Washington Luiz	R\$ 800,00	ASTRA GLS
Resumo para 'codvendedor' = 2004100095 (2 registros de detalhe)				
Soma		78000		
2005060038	80000	Arthur Pujol Viana	R\$ 1.500,00	MONDEO GHIA
	24500	Arthur Pujol Viana	R\$ 1.500,00	COROLLA
Resumo para 'codvendedor' = 2005060038 (2 registros de detalhe)				
Soma		104500		
2005080128	82000	Marlene Casagrande	R\$ 600,00	AUDI A4
	76800	Marlene Casagrande	R\$ 600,00	PAJERO
	37500	Marlene Casagrande	R\$ 600,00	VECTRA CD
	19500	Marlene Casagrande	R\$ 600,00	VECTRA CD
Resumo para 'codvendedor' = 2005080128 (4 registros de detalhe)				

Em seguida entraremos no modo de edição do relatório para adequá-lo a nossas necessidades. Para isso, dê um click no relatório com o botão direito do mouse e escolha “Modo Design”

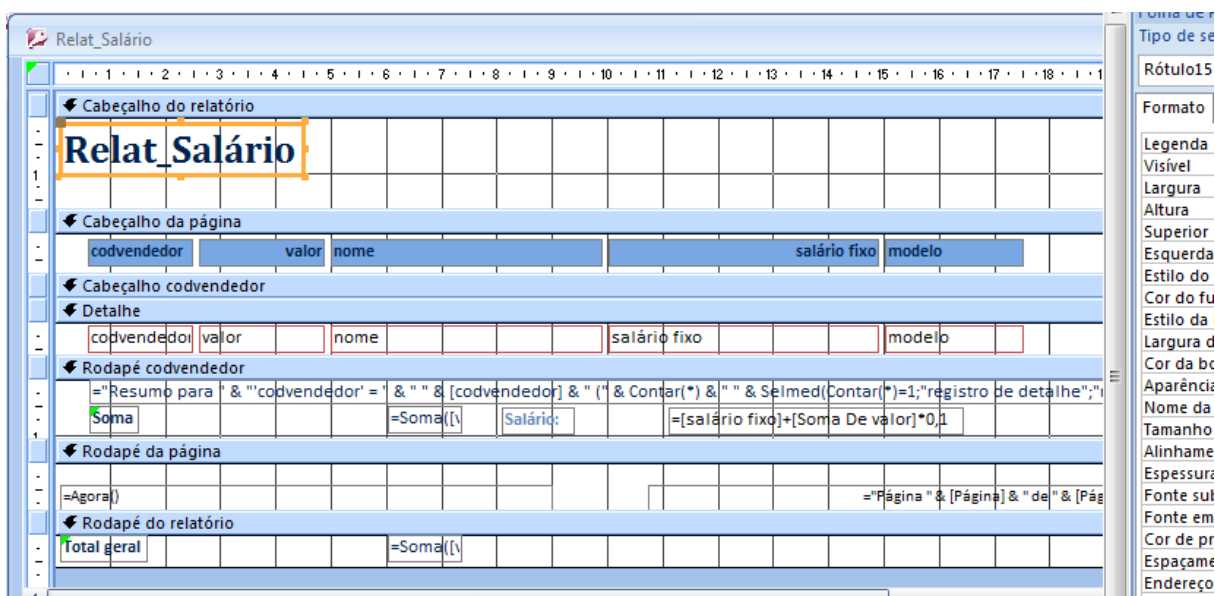
Estando no modo de edição do relatório, inclua uma caixa de texto, onde exibiremos o valor do salário do vendedor. Para isso, incluiremos a caixa de texto txtSalario na seção “Rodapé codvendedor”.

Para isso, selecione a caixa de texto, exiba a janela de propriedades, mude o nome da caixa de texto para txtSalario, e o formato para Unidade monetária.

Pegue a guia Dados e configure o conteúdo dessa caixa de texto (Fonte de Controle [...]) de acordo com o seguinte:



Visualize o relatório, que deverá exibir o salário fixo de cada vendedor, acrescido com a comissão pela venda dos veículos, por ele vendidos. Se quiser formatá-lo de maneira melhor, faça-o a seu gosto.



Obs.: Note que o relatório está numa forma um tanto bagunçada e não muito lógica para consulta por um usuário. Deixe-o de uma forma mais fácil para consulta. Use seu bom senso.

Obs.: No caso de um vendedor não vender nenhum veículo, note que não é impresso o campo salário. Está errado, pois o vendedor deveria receber o salário fixo. Isso ocorre porque o Access não faz a conversão automática de nulo para zero em "soma de valor". Para corrigir isso, use a função NZ, que faz a conversão de nulo p/ zero. Ficará assim:

$$=[\text{salário fixo}]+nz([Soma De valor];0)*0,1$$

Criação do segundo relatório.

Ainda através do assistente, crie agora o relatório de nome Relat_Veiculos para exibição dos dados da tabela “Veiculos”. Os registros deverão ser exibidos em ordem alfabética de modelo de veículo, com os agrupamentos (quebras com totais): A cada quebra de modelo deverá haver um subtotal de valor por modelo. Dentro de cada modelo deverá haver um sob total por ano de veículo, daquele modelo. Vide exemplo de relatório abaixo:

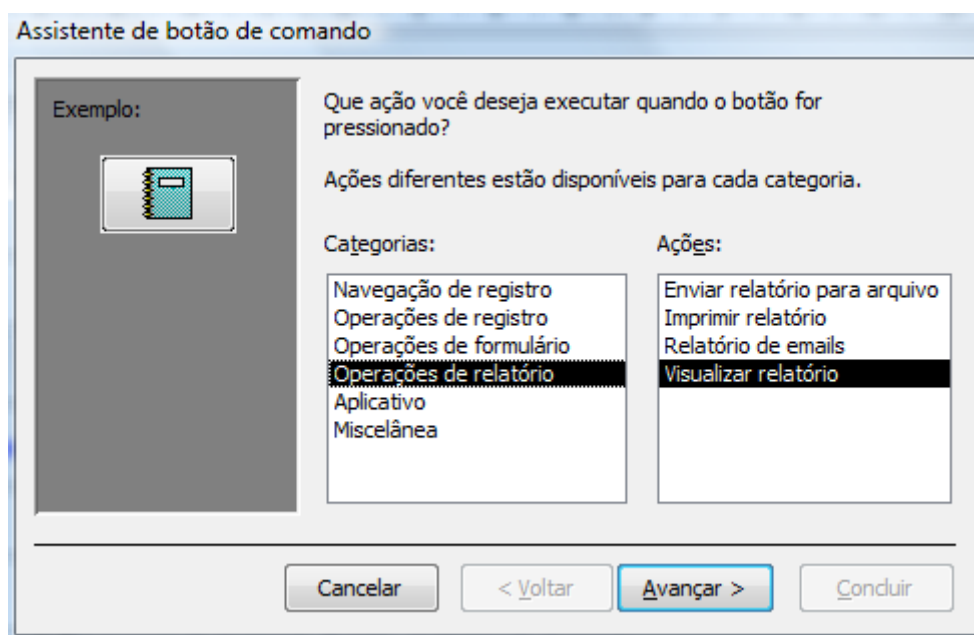


Relatório de veículos

modelo	ano	estado	valor
ASTRA GLS			
1999			
ASTRA GLS	1999	SP	R\$ 23.400,00
ASTRA GLS	1999	RJ	R\$ 24.100,00
ASTRA GLS	1999	RJ	R\$ 25.500,00
Resumo para 'ano' = 1999 (3 registros de detalhe)			
Soma			73000
2002			
ASTRA GLS	2002	SP	R\$ 31.400,00
ASTRA GLS	2002	SP	R\$ 32.500,00
Resumo para 'ano' = 2002 (2 registros de detalhe)			
Soma			63900
Resumo para 'modelo' = ASTRA GLS (5 registros de detalhe)			
Soma			136900
BLAZER STD			
1997			
BLAZER STD	1997	RJ	R\$ 17.500,00
Resumo para 'ano' = 1997 (1 registro de detalhe)			
Soma			17500

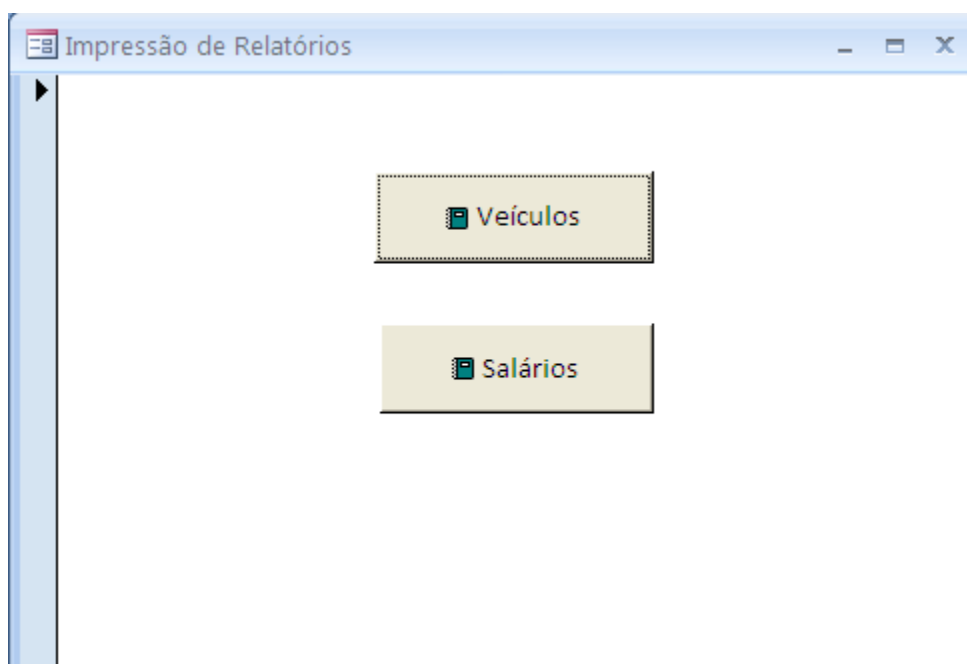
Obs.: Se o relatório não estiver adequado ao ser criado pelo assistente, dê-lhe um click com o botão direito do mouse e escolha “Modo Design”. Aí você pode configurá-lo como quiser.

Em seguida, crie um novo formulário contendo dois botões para apresentar os relatórios. Ao receber o click do mouse, cada botão exibe o relatório correspondente criado no item anterior. Utilize o “Assistente de botão de comando”, com o grupo de “Operações de relatório” e a ação “Visualizar relatório”. Veja o figura a seguir:



Escolha o relatório correspondente, coloque o nome adequado no botão, conclua a configuração e teste o botão.

O formulário deve ficar mais ou menos com a seguinte aparência:



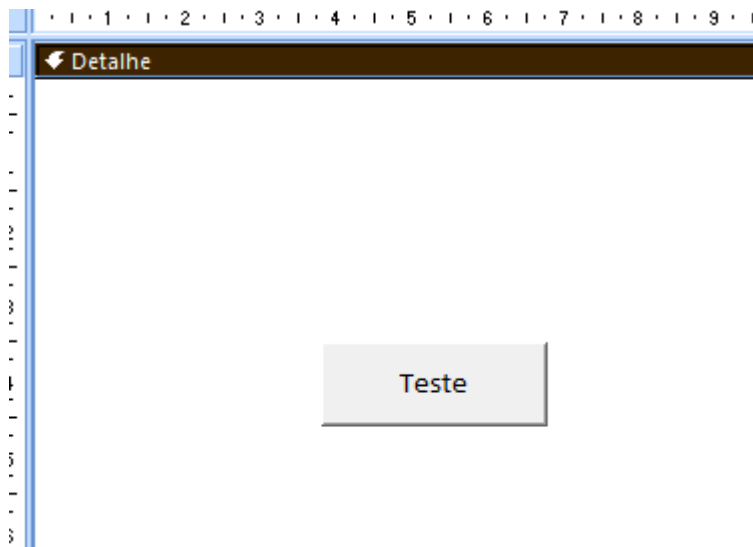
Faça um teste geral em seu sistema, que será finalizado na próxima aula.

Apresente-o ao professor.

Aula 5 de Access.

Abra o arquivo Revenda para fazer os acertos finais. Iremos implementar algumas melhorias no projeto, tais como não permitir o cadastramento de veículos em duplicidade ou de vendedores com mesmo código.

Para exemplificar isso, criaremos uma função baseada no livro “Programando em Microsoft Access com VBA, de Flávio Morgado”, em conjunto com o formulário abaixo. A função que utilizaremos consulta uma tabela de um banco de dados para verificar se um registro está ou não presente.



Option Compare Database
Option Explicit

```
Function achaReg() As Boolean
Dim db As Database
Dim rs As Recordset
Dim strcriterio As String
Dim strmsg As String
Dim strtitle As String
Dim procurado As String
Set db = CurrentDb()
Set rs = db.OpenRecordset("veiculos", dbOpenDynaset)
procurado = InputBox("Digite o veículo desejado...")
strcriterio = "modelo like '" + procurado + "'"
rs.FindFirst strcriterio
Do While Not rs.NoMatch
    strmsg = "modelo = " + rs!modelo
    strtitle = "sucesso"
    MsgBox strmsg, vbInformation, strtitle
    rs.FindNext strcriterio
    achaReg = True
rs.Close
```

```
Exit Function
Loop
If rs.NoMatch Then
    strmsg = "registro nao encontrado"
    strtitle = "fim do arquivo"
    MsgBox strmsg, vbInformation, strtitle
    achaReg = False
End If
rs.Close
End Function
```

```
Private Sub cmdTeste_Click()
    If achaReg Then
        MsgBox "Registro encontrado"
    Else
        MsgBox "Registro não encontrado"
    End If
End Sub
```

EXERCÍCIOS:

- 1-Faça a adequação da função acima e use-a para que não seja possível a inclusão na tabela “Veiculos”, de registros de veículos de mesma placa.
- 2-Faça a mesma coisa para a tabela “Vendedores”, para que não seja permitida a inclusão de vendedores com mesmo código.
- 3-Inclua mais um formulário de nome frmPrincipal, que será o formulário pelo qual a aplicação irá iniciar, quando abrirmos o arquivo “Revenda”.

Esse formulário deverá ter 4 botões, sendo:

- o primeiro botão exibe na forma modal, o formulário frmVeiculos (nome=cmdExibeFrmVeiculos).
- o segundo botão exibe modal, o formulário frmVendedor (nome=cmdExibeFrmVendedor)
- o terceiro botão exibe na forma modal o formulário frmExibeRelatorio.
- o quarto botão encerra o processamento (nome=cmdFim).

Obs.:

- I) Para exibir um formulário na forma modal, você deve configurar suas propriedades como: guia Outra / Janela restrita / Sim
- II) Se você precisar de uma variável global a todos os locais do projeto, declare-a dentro de um módulo, de acordo com a especificação a seguir:

```
Public var1 as integer ' Variável global ao projeto
Dim var2 as integer ' Variável local ao módulo
```

- 4-Inclua nesse formulário (propriedades: Formulário / formato / Imagem) a figura “Carrinho.jpg”, que deverá ficar mais ou menos como a próxima figura.

- 5-Inclua uma "hint" (dica) em cada botão desse formulário, informando a função de cada um. A propriedade é "Texto das dicas sobre controles".

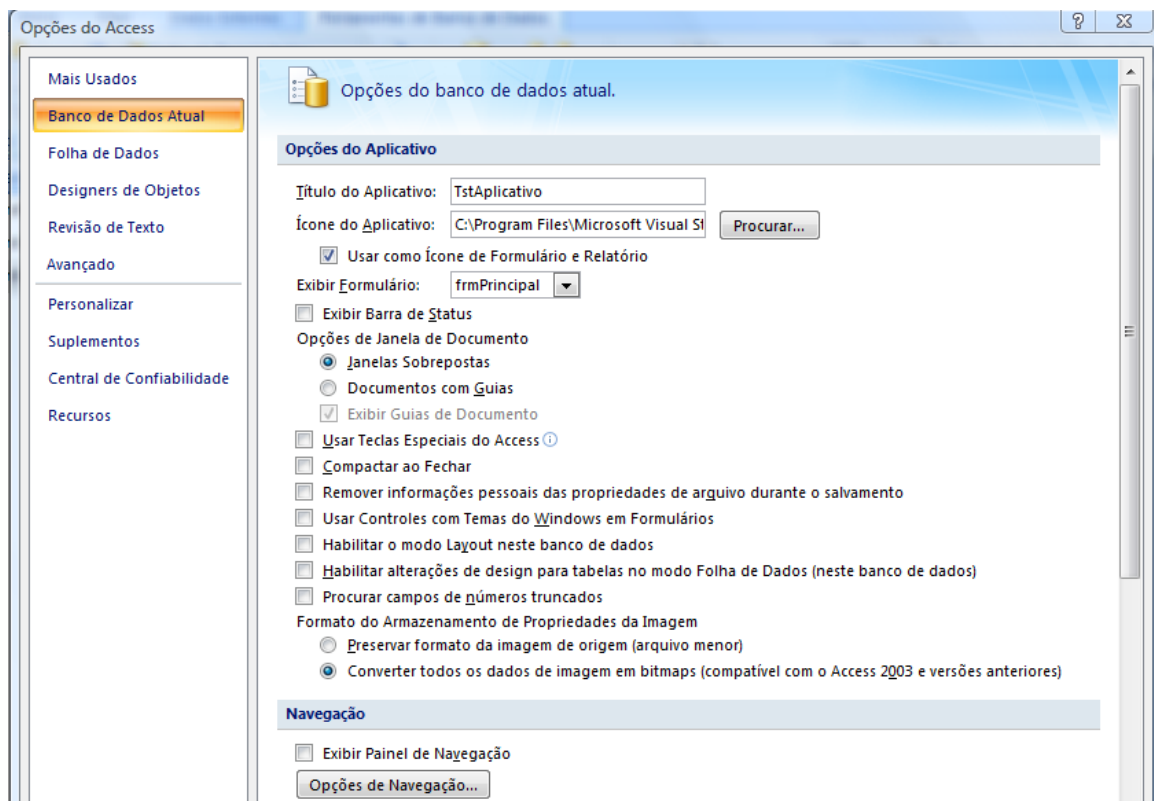


6- **Finalizando**, vamos configurar as condições de inicialização de nosso sistema.

- a) Para que o usuário não tenha acesso aos controles de formulário, selecione o formulário e configure a propriedade “Estilo da borda” para “Nenhum”. Configure também a propriedade “Seletores de registro” para “Não”
- b) Programar o evento Form_Load() do formulário, para que ao ser carregado, o mesmo fique maximizado:

```
Form_Load()  
    doCmd.Maximize  
End Sub
```

- c) Para que os relatórios fiquem visíveis (na frente do formulário que os chama), configurar suas propriedades PopUp Sim e Janela restrita Sim.
- d) Configure o Access para que o usuário não possa acessar seus recursos. Para isso, dê um click no botão Access / Opções do Access / Banco de Dados Atual, e configurar a janela apresentada de acordo com a figura a seguir.
- e) Atribuir um nome ao aplicativo, incluir um ícone no aplicativo, **desligar todas as caixas de seleção**, selecionar o formulário a iniciar a execução (frmPrincipal), dar um click no botão Ok e salvar o arquivo Revenda. Veja a janela a seguir - Note que a imagem mostra apenas uma parte das configurações, ao descer a tela, há outras opções que devem ser desligadas.



Feche todas as janelas, salve o arquivo e feche o Access. Abra o arquivo Revenda e note que só é possível o acesso aos formulários e aos relatórios que você criou. O usuário não tem acesso direto às tabelas e aos outros componentes do projeto.

Se você quiser acessar as tabelas ou outros recursos do Access, ao abrir o arquivo, pressione a tecla Shift e a mantenha pressionada até o arquivo abrir.

Faça um teste geral em seu sistema e verifique se tudo funciona a contento, antes de chamar o professor para avaliação.

Salve tudo e prepare-se para a apresentação final do projeto.

AUTOMATIZAÇÃO DE TAREFAS

O VBA também é utilizado em arquivos de script, ou seja, um arquivo texto com extensão ".vbs", contendo instruções que são interpretadas e executadas linha a linha, pelo programa "wscript.exe" do Windows. A execução ocorre ao darmos um duplo click no arquivo com extensão ".vbs".

Exemplo:

1) Clique com o botão direito do mouse na área de trabalho selecione Novo/Documento de Texto.

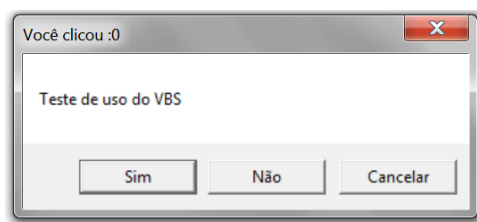
2) No novo documento, clique com o botão direito do mouse e selecione Editar.

3) Ao abrir o Bloco de Notas, digite o código abaixo (note que são instruções VBA).

```
dim variavel ' todas as variáveis em arquivos VBS são do tipo variant
variavel=0
do while variavel<4
    msgbox "Teste de uso do VBS",3,"Você clicou :" & variavel
    variavel=variavel + 1
loop
```

4) Salve o arquivo: Arquivo/Salvar como.../Tipo/Todos os arquivos/teste1.VBS/Salvar

5) Execute o arquivo dando-lhe um duplo click e confira o resultado com a figura abaixo:



Note durante a execução, o ícone do programa wscript.exe na barra de tarefas do Windows.

Vamos estragar esse programa, pois somos programadores inexperientes e ainda fazemos muitas bobagens...

Altere a instrução `variavel=variavel + 1` para `variave1=variavel + 1`

Se você não prestar bastante atenção, não vai achar o erro, pois o programa não acusa nada, só que não faz o que se espera dele. Salve-o e teste novamente.

Esse tipo de problema é resolvido com o uso da cláusula Option Explicit, que obriga o programador a declarar todas as variáveis. Nosso programa agora ficará assim:

```
Option Explicit
dim variavel ' todas as variáveis em arquivos VBS são do tipo variant
variavel=0
do while variavel<4
    msgbox "Teste de uso do VBS",3,"Você clicou :" & variavel
    variave1=variavel + 1
loop
```

Salve o programa e teste-o

Exercício1: Modifique o programa para ficar com a seguinte codificação:

```
option explicit ' Obriga o programador a declarar todas as variáveis
dim variavel ' todas as variáveis em arquivos VBS são do tipo variant
dim testeTecla
variavel=0
do while variavel<4
    testeTecla=msgbox ("Teste de uso do VBS",3,"Você clicou :" & variavel)
    variavel=variavel + 1
    msgbox testeTecla
loop
```

Note que no programa acima, a variável testeTecla recebe o valor retornado pela função msgbox, quando esta recebe o click do mouse em um de seus três botões.

Execute o script e verifique que ao ocorrer o click em um dos botões **Sim**, **Não** ou **Cancelar**, a função msgbox retorna um dos valores 6,7 ou 2, respectivamente

Exercício 2:

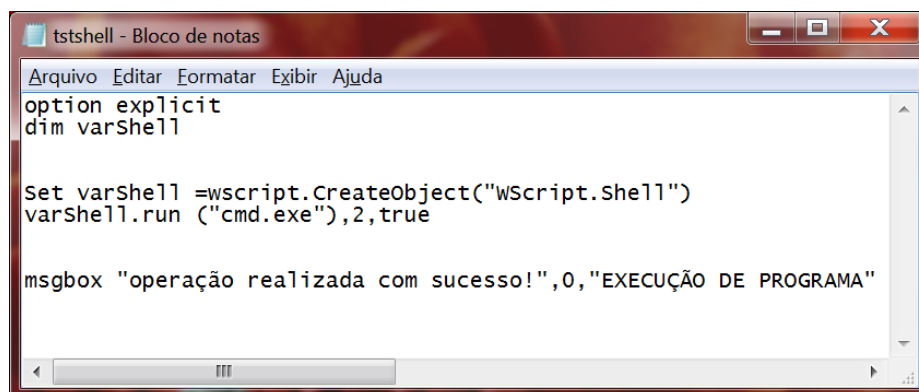
Modifique o programa para que ele continue contando os clicks apenas se recebê-los no botão **"Sim"**. Se o click for no botão **"Não"**, o programa deve informar que o click foi nesse botão e encerrar. Se o click for no botão **"Cancelar"**, o programa encerra sem nenhum aviso.

Obs.

1-A instrução que encerra o processamento de um script, é **wscript.quit**.

2-Tente fazer os exercícios. Se tiver dúvidas, há exercícios resolvidos no final da apostila.

Na figura abaixo vemos um exemplo de um script que executa o programa cmd.exe (abre o ambiente do ms-dos). O argumento 2 o deixa minimizado, e o argumento true significa que a execução do script é interrompida até que o programa que está sendo executado seja encerrado. Digite e teste o script abaixo.



Utilize o script que você testou acima para executar outro programa qualquer de sua preferência.

Exercício3:

Digitar e testar o script abaixo, que executa o Access e abre o arquivo "revenda.accdb" de seu projeto das aulas anteriores.

Obs.: Para executar o Access e abrir um arquivo ".accdb", use as instruções abaixo. O argumento "5" no final da instrução "varShell.run" significa que o programa será exibido em janela normal e o argumento "true" significa que, as **instruções do script** que vierem após a instrução que executou o Access, serão processadas somente após o Access ser encerrado. Se quisermos que o programa a ser executado fique maximizado, o argumento é o número "3".

```
option explicit
dim varShell      'Cria variável
Set varShell =wscript.CreateObject("WScript.Shell") 'Cria uma instância de
objeto
varShell.run
( ""msaccess.exe""C:\Users\Hamilton\Desktop\revenda.accdb"),5,true
```

Obs.:

- 1-Os nomes das pastas não podem conter espaço. Por exemplo, se o caminho for d:\meus documentos\... utilizar: d:\meusdo~1\... O nome abreviado da pasta com o "~" deve ter 8 caracteres, incluindo o "~".
- 2- Para maiores informações sobre ".run", consulte o site:
[http://msdn.microsoft.com/en-us/library/d5fk67ky\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/d5fk67ky(v=vs.85).aspx)

Complemento: Mude o argumento para "3" (p/ maximizar o Access na execução) e inclua algumas instruções após a chamada do Access. Verifique que com o argumento "true" essas instruções somente serão executadas quando encerrarmos o processamento do Access. Após esse teste, mude o argumento para "false" e verifique o resultado.

Exercício4:

Vamos testar mais um script visando avaliar algumas características especiais do wscript. Crie um novo arquivo de script de nome looping, contendo o código abaixo, analise e entenda cada linha de código. A seguir, execute o arquivo dando-lhe um duplo click:.

```
option explicit 'Obriga a declarar todas as variáveis
on error resume next 'Se ocorrer algum erro, pula p/ próxima instrução
dim varShell
wscript.Echo "O B AAAA! VAMOS BAGUNÇAR TUUUUDDOOO..." & chr(13) &
"pressione OK para zoar..."
Set varShell =wscript.CreateObject("WScript.Shell")
do
    wscript.sleep 200 'dorme por 200 milissegundos
    varShell.sendkeys "{NUMLOCK}"
    wscript.sleep 100
    varShell.sendkeys "{CAPSLOCK}"
```

```
wscript.sleep 100
varShell.sendkeys "{sCROLLLOCK}"
wscript.sleep 200
varShell.sendkeys "{sCROLLLOCK}"
wscript.sleep 100
varShell.sendkeys "{CAPSLOCK}"
wscript.sleep 100
varShell.sendkeys "{NUMLOCK}"
loop ' looping infinito...
wscript.Echo "testando!... vamos ver se sai do looping..."
```

Note o que acontece com as luzes de NumLock, CapsLock e ScrollLock e tente digitar algo aproveitável.

Tente parar a execução do script pelos métodos tradicionais: Alt f4, Ctrl Break, etc.

Obs.: Caso seu programa entre em looping, é necessário ativar o gerenciador de tarefas (Ctl+Alt+Del), acionar a guia "processos", selecionar o processo "wscript.exe" e clicar no botão "finalizar processo".

Exercício 5:

Criar o arquivo de script de nome backUpBd que faça uma cópia backup de um banco de dados. Esse procedimento costuma ser usado e pode ser muito útil para preservar suas informações em uma cópia anterior do banco de dados, antes que esse arquivo sofra alterações.

Se houver algum problema com o banco de dados a ser atualizado, como por exemplo ficar corrompido, você tem uma cópia dele antes da atualização.

Como exemplo, o código abaixo usa o arquivo "**testebd.accdb**" que se encontra no diretório raiz do drive "**d**" e faz uma cópia no diretório "c:\backup" com nome "anterior.accdb":

```
option explicit
dim objFSO, Wshs
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set Wshs = WScript.CreateObject("WScript.Shell")
objFSO.CopyFile "d:\testebd.accdb" , "c:\backup\anterior.accdb"
```

Adapte o script acima para o seu ambiente e teste-o.

Obs.:

1-Note que o script acima não cria novas pastas, apenas copia o arquivo em pastas existentes. Para criar pastas, faremos outro exercício.

2-Se o arquivo a gravar já existir, este será sobrescrito.

Para complementar este exercício:

a) Transforme o arquivo script em somente leitura

- b) Crie um atalho para executá-lo, com ícone significativo.

Exercício 6:

Criar um script que irá automatizar algumas tarefas para acionar o banco de dados "revenda.accdb" de seu projeto das aulas anteriores. Esse script deverá fazer o seguinte:

- a) Ao iniciar deve exibir uma caixa de mensagem com três botões: Sim, Não, Cancela. Nessa caixa, além de dar boas vindas ao usuário deverá constar as informações a seguir.
- b) Deverá informar ao usuário que caso ele vá fazer atualizações no banco de dados, deverá clicar no botão "Sim", que o backup será feito automaticamente.
- c) Se for fazer apenas consultas, clicar no botão "**Não**", que não será feito backup.
- d) Se o usuário quer simplesmente sair, deverá clicar em "**Cancela**".
- e) Após o click em um dos botões "**Sim**" ou "**Não**", o script deve executar o Access com o arquivo revenda.accdb, no modo maximizado.
- f) Ao encerrar a execução do Access, o script deve verificar se houve a criação do backup e alertar o usuário para tomar as providências necessárias p/ a preservação desse backup.
- g) Em seguida, o script deve exibir mensagem do sucesso ou não da operação.
- h) Gravar o arquivo de script no modo "**Somente Leitura**", criar um atalho para sua execução e colocar nesse atalho, um ícone representativo das operações das tarefas a realizar.

Complemento:

Analise e teste o script abaixo - Ele cria uma pasta de nome "copia" no drive "d" e uma subpasta de nome composto pelo ano, mês, dia, hora, minuto e segundo. Teste o script e o inclua no programa do exercício 6, para que o arquivo backup do banco de dados não seja sobrescrito. Com esse complemento, cada vez que for gerado um novo backup, uma nova pasta será criada p/ gravar o banco de dados anterior, que será preservado.

```
option explicit
dim varShell,varData, frase
Set varShell =wscript.CreateObject("WScript.Shell")
varData=year(now) & month(now) & day(now) & hour(now) & minute(now) &
second(now)
msgbox varData
frase="cmd /k CD D:\& mkdir d:\copia\" & vardata
msgbox frase
varShell.run frase,1,true ' Cria no drive "d" uma pasta de nome "copia" e sub pasta
"yyyymmddhhmmss"
msgbox "operação realizada com sucesso!",0,"EXECUÇÃO DE PROGRAMA"
```

A partir do script acima alterar o exercício 6 para que o backup do banco de dados Revenda seja sempre gravado em uma nova pasta. Dessa maneira, todos os backups estarão identificados pela data de gravação e não mais ocorrerá o problema de sobreposição.

Obs.: Se quiser fechar a tela do DOS, basta incluir a instrução " & exit" no string frase, ou seja, o conteúdo da variável frase ficará assim:

```
frase="cmd /k CD D:\& mkdir d:\copia\" & vardata & " & exit"
```

Apresentar o script ao professor para avaliação.

PROJETO FINAL DA DISCIPLINA

O projeto constitui-se na confecção de um sistema em Access para controlar as notas e faltas dos alunos da disciplina "Programação em Microinformática".

O sistema deve ser encabeçado por um formulário principal cujo acesso se dá através de uma senha, que o usuário pode tentar acertar por até três vezes.

O sistema deverá ter os módulos ALUNOS e RELATÓRIOS.

O módulo ALUNOS deverá ter as seguintes funcionalidades: Inclusão, Alteração e Exclusão.

O módulo RELATÓRIOS deverá exibir dois tipos de relatórios: de notas e de faltas. Deve haver duas possibilidades de relatório de nota: por ordem alfabética de aluno e por ordem decrescente de nota.

Descrição sucinta sobre os itens solicitados acima:

INCLUSÃO: Solicita o número de matrícula do aluno, verifica o dígito de controle e verifica se o aluno já existe no cadastro. Se não existir e se o dígito for correto, o aluno é cadastrado.

ALTERAÇÃO: Solicita o número de matrícula, verifica o dígito e verifica se o aluno está no banco de dados. Abre campos para modificação.

Obs.: deve haver dois tipos distintos de modificação: Dados Cadastrais e de notas. Quando houver modificações em notas deve gravar a data (do sistema) em que as notas foram modificadas.

EXCLUSÃO: Idem anterior, mais solicitação de confirmação da exclusão;

RELATÓRIOS:

NOTAS – Exibe todas as notas e o conceito de cada aluno, de acordo com o critério de avaliação. Deve exibir a nota numérica e o conceito em forma de letra, de acordo com o critério de avaliação especificado no Plano de Ensino. O Conceito deve ser calculado através de uma função.

FALTAS – Informa o total de faltas e se o aluno está reprovado por faltas.

Capriche nas funcionalidades e no visual!!!.

Obs.: Aqui você é o analista que projeta e implementa o sistema e deve, ao mesmo tempo, colocar-se no lugar do usuário. Faça um sistema intuitivo e amigável, e que não possibilite ao usuário cometer erros. Por exemplo, se um componente tal como um botão ou uma caixa de texto não tiverem função em determinada fase de funcionamento do sistema, então eles não deveriam estar lá. Pense na possibilidade de desabilitá-los ou torná-los invisíveis, até que se precise deles. Outro exemplo seria o usuário estar alterando os dados cadastrais de um aluno e ter à disposição o botão excluir. Seria um contra-senso.

A previsão de confecção deste projeto é de quatro semanas a e vale de 0 a 40 pontos.

Ao final desse tempo, o projeto deverá ser apresentado ao professor e será entregue em CD ou DVD para avaliação.

Importante:

A execução do programa inicial do projeto deve ser feita através de um script que automatize tarefas importantes visando a preservação do sistema, o auxílio ao usuário ou outras que você julgue importantes ou interessantes.

*** Na apostila "Scripts Interessantes" você pode encontrar exemplos de scripts elaborados e utilizados por alunos de turmas anteriores.

SUGESTÕES INICIAIS QUE PODERÃO AJUDAR NA CONFEÇÃO DO TRABALHO:

- 1) Importar através do Access o arquivo fornecido pelo professor "Sua_Turma_PrgMicro.xls" e trabalhar esse arquivo.

Obs.: A importação pode ser feita abrindo o Access e criando um novo banco de dados (Banco de Dados em Branco). Em seguida, selecione a guia "Dados Externos" e dê um click no ícone Excel. Localize a planilha que quer importar e dê um click em OK. Siga o assistente para a importação e após o arquivo ser importado, configure a estrutura do banco de dados a seu gosto.

- 2) Segundo informações do CEI (Centro de Informática do CEETEPS), o cálculo de dígito de controle pelo módulo 11 é realizada de acordo com o código abaixo. A rotina do CEI faz o processamento do número de matrícula em forma de texto e não numérica, como fizemos em nossos exemplos anteriores. Isso é útil quando usamos números grandes, com mais de 10 dígitos.

Rotina para cálculo do dígito de verificação do número de matrícula dos alunos da FATEC-SP:

```
Public Function Checa_Matricula(txtMatricula)
    Dim lx As Integer, Soma As Integer, Dígito As Integer
    For lx = 1 To 7
        If Not IsNumeric(Mid(txtMatricula, lx, 1)) Then
            MsgBox "Matrícula contém caracter não numérico", VB_Question
            lx = 8
            Exit Function
        End If
    Next lx
    Soma = 0
    '*****
    ' Soma com pesos 7,6,5,4,3,2,1 para cada algarismo da Matrícula
    '*****
    For lx = 1 To 7
        Soma = Soma + (Val(Mid(txtMatricula, lx, 1)) * (8 - lx))
    Next lx
    '*****
    ' Obtem resto da divisão de Soma por 11
    '*****
    Dígito = Soma Mod 11

    If Dígito > 1 Then
```



```
Dígito = 11 - Dígito
End If
If Dígito <> Right(txtMatricula, 1) Then
    MsgBox "Dígito de verificação diferente do informado.", vb_Question
End If
End Function
```

Obs.: Note que o processo de cálculo é o mesmo do módulo 11 porém, no cálculo de dígito do número de matrícula do aluno da FATEC SP, o número de menor ordem é multiplicado por 1 e não por 2, como no método usual.

Exemplo: 0810698-2

0	8	1	0	6	9	8
x	x	x	x	x	x	x
7	6	5	4	3	2	1

$0 + 48 + 5 + 0 + 18 + 18 + 8 = 97$

$97 / 11 = 8 \text{ resto } 9$ Neste caso dígito será $11 - 9 = 2$

- 3) Para confecção do projeto, o código abaixo poderá ser útil na codificação do tratador do evento click do botão “Localizar Registro”. Ao localizar o registro procurado, uma cópia de seus campos será exibida no formulário.

```
Private Sub cmdLocalizar_Click()
    Dim codMatr As Variant
    Dim codigolido As Long
    codMatr = InputBox(" Digite a Matrícula!", " I n f o r m a r M a t r i c u l a ")
    If codMatr <> Null Or codMatr <> "" Then

        codigolido = Val(codMatr)
        If Checa_Matricula(codigolido) Then

            If achaMatricula(codMatr) Then

                'Encontrar o registro que coincide com o controle
                Dim rs As Object
                Set rs = Me.Recordset.Clone
                rs.FindFirst "[Matrícula]=" & codMatr & ""
                If Not rs.EOF Then Me.Bookmark = rs.Bookmark
            Else

                ' Nao encontrou a matricula no BD
                MsgBox "Matrícula não encontrada..."
            End If
        End If
    End If
End Sub
```

```
Function achaMatricula(codMatr) As Boolean
    Dim db As Database
    Dim rs As Recordset
    Dim strcriterio As String
    Dim procurado As String
    Set db = CurrentDb()
    Set rs = db.OpenRecordset("Alunos", dbOpenDynaset)
    strcriterio = "Matrícula like '" + codMatr + "'"
    rs.FindFirst strcriterio
    Do While Not rs.NoMatch
        rs.FindNext strcriterio
        achaMatricula = True
        rs.Close
        Exit Function
    Loop
    If rs.NoMatch Then
        achaMatricula = False
    End If
    rs.Close
End Function
```

COMPLEMENTO – Alguns programas do projeto Access das aulas e dos exercícios de script:

Programa associado ao evento click do botão cmdVender, do frmVeiculos (c/ função)

```
Private Sub cmdVender_Click()
    Dim codigostr As String
    Dim codigolido As Long
    Dim codigo As Long
    Dim digitolido As Integer

    On Error GoTo ExibeErro:

    codigostr = InputBox(" Digite o código do vendedor: ", "  V e n d a   d e   V e í c u l o")

    ' verifica se foram digitados só numeros.

    If Len(codigostr) > 10 Then
        MsgBox "Código com mais de 10 dígitos - redigite!"
        Exit Sub
    End If

    If Not IsNumeric(codigostr) Then
        MsgBox "Número inválido! Digite apenas números...", vbCritical, "ATENÇÃO"
        Exit Sub
    End If

    codigolido = Val(codigostr) 'converte o codigo p/ numero
    digitolido = codigolido Mod 10 'separa o digito lido
    codigolido = Int(codigolido / 10) 'tira o digito do codigo - trunca

    If digitolido = CalculaDC(codigolido) Then 'Chamada da função VerificaDC
        'o digito bateu...
        codvendedor = Val(codigostr) ' codvendedor é campo do BD

        'A instrução abaixo salva o registro corrente - grava o codigo do vendedor.

        DoCmd.RunCommand acCmdRefresh 'Grava registro corrente

        MsgBox "Venda efetuada com sucesso!"
        cmdVender.Caption = "Vendido"
        cmdProximo.SetFocus
        cmdVender.Enabled = False
    Else
        ' No else, o digito não bateu...
        MsgBox "Dígito não bate - redigite!", vbCritical, "A T E N Ç Ã O"
        Exit Sub
    End If
    Exit Sub
ExibeErro:

    MsgBox Error(Err)

End Sub
```

Programas Principais do formulário “frmVendedor”:

```
Private Sub cmdOk_Click()
```

```
DoCmd.RunCommand acCmdRefresh 'atualiza o banco de dados
```

```
txtcodvendedor.Enabled = False
```

```
txtNome.Enabled = False 'desabilita as caixas de texto
```

```
txtEndereco.Enabled = False
```

```
txtEstadoCivil.Enabled = False
```

```
txtSalarioFixo.Enabled = False
```

```
cmdInsereVendedor.Enabled = True 'habilita os botões
```

```
cmdProximo.Enabled = True
```

```
cmdAnterior.Enabled = True
```

```
End Sub
```

```
Private Sub Form_Load()
```

```
txtcodvendedor.Enabled = False
```

```
txtNome.Enabled = False
```

```
txtEndereco.Enabled = False
```

```
txtEstadoCivil.Enabled = False
```

```
txtSalarioFixo.Enabled = False
```

```
End Sub
```

```
Private Sub cmdInsereVendedor_Click()
```

```
'On Error GoTo Err_cmdInsereVendedor_Click
```

```
Dim novocodstr1 As String
```

```
Dim novocodstr2 As String
```

```
Dim novocodigo As Long
```

```
Dim digitocalc As Integer
```

```
novocodstr1 = InputBox("Digite o código do vendedor:")
```

```
If Len(novocodstr1) > 9 Then 'Verifica se o código tem até 9 dígitos
```

```
MsgBox "O código deve ter no máximo 9 números", vbCritical, "A T E N Ç Ã O"
```

```
Exit Sub
```

```
End If
```

```
If Not IsNumeric(novocodstr1) Then 'Verifica se só tem números
```

```
MsgBox "Número inválido! Digite apenas números...", vbCritical, "ATENÇÃO"
```

```
Exit Sub
```

```
End If
```

```
novocodstr2 = InputBox("R E D I G I T E o código do vendedor para verificação...")
```

```
If novocodstr1 <> novocodstr2 Then
```

```
MsgBox "Código inválido... tente novamente!", vbCritical, "A T E N Ç Ã O"
```

```
Exit Sub
```

```
End If
```

```
novocodigo = Val(novocodstr1)
```

```
digitocalc = CalculaDC(novocodigo) 'calculo do dígito de controle p/ o novo código
```

```
novocodigo = novocodigo * 10 + digitocalc 'incorpora o dc ao código
```

```
DoCmd.GoToRecord , , acNewRec 'insere um registro novo
```

```
codvendedor = novocodigo
```

```
MsgBox "Digite os campos para cadastramento do vendedor e pressione o botão Ok!"
```

```
txtNome.Enabled = True
```

```
txtEndereco.Enabled = True
```

```
txtEstadoCivil.Enabled = True
```

```
txtSalarioFixo.Enabled = True
```

```
txtNome.SetFocus
```

```
cmdInsereVendedor.Enabled = False
```

```
cmdProximo.Enabled = False
```

```
cmdAnterior.Enabled = False
```

```
Exit Sub
```

```
Exit_cmdInsereVendedor_Click:
```

```
Exit Sub
```

```
Err_cmdInsereVendedor_Click:
```

```
MsgBox Err.Description
```

```
Resume Exit_cmdInsereVendedor_Click
```

```
End Sub
```

Função CalculaDC

```
Function CalculaDC(novocodigo As Long) As Integer
```

```
Dim codigo As Long
```

```
Dim digito As Integer
```

```
Dim soma As Integer
```

```
Dim mult As Integer
```

```
codigo = novocodigo
```

```
mult = 2
```

```
soma = 0
```

```
While codigo > 0
```

```
    soma = soma + (codigo Mod 10) * mult
```

```
    mult = mult + 1
```

```
    codigo = Int(codigo / 10)
```

```
Wend
```

```
digito = soma Mod 11
```

```
If digito > 1 Then digito = 11 - digito
```

```
CalculaDC = digito 'Retorna o valor do digito calculado
```

```
End Function
```

SCRIPTS

Programas solicitados nos exercícios:

Exercício 2:

```
option explicit
dim variavel ' todas as variáveis em arquivos VBS são do tipo variant
dim testetecla
variavel=0
do while variavel<4
    testetecla=msgbox ("Teste de uso do VBS - Continua o teste???",3,"Você clicou
:" & variavel)
    variavel=variavel + 1
    if testetecla = 7 then
        msgbox "você clicou Não..."
        wscript.quit
    end if
    if testetecla =2 then wscript.quit
loop
```

Exercício 6:

```
option explicit
dim testetecla ,objFSO, Wshs, usrProfile, varShell

testetecla=msgbox ("Sr. usuário: bem vindo ao acesso a banco de dados. Clique em Sim
para gravar um arquivo de backup.",3,"BOAS VINDAS AO SISTEMA REVENDA")

if testetecla=2 then wscript.quit
if testetecla = 6 then
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    Set Wshs = WScript.CreateObject("WScript.Shell")
    usrProfile = Wshs.ExpandEnvironmentStrings("%UserProfile%")
    objFSO.CopyFile "d:\testebd.accdb" , "c:\backup\anterior.accdb"
    msgbox "Arquivo de backup gravado..."

end if

Set varShell =wscript.CreateObject("WScript.Shell") 'Cria uma instância de objeto
varShell.run ("""msaccess.exe""C:\Users\Hamilton\Desktop\revenda.accdb"),3,true

if testetecla=6 then wscript.Echo "Sr. Usuário... foi gravado o arquivo backup ... favor
providenciar!"
msgbox "operação realizada com sucesso!",0,"Sistema banco de Dados"
```

Exercício 6 com complemento

```
option explicit
dim testetecla ,objFSO, Wshs, usrProfile, varShell, vardata, frase, novaSubPasta

testetecla=msgbox ("Sr. usuário: bem vindo ao acesso a banco de dados. Clique em
Sim para gravar um arquivo de backup.",3,"BOAS VINDAS AO SISTEMA REVENDA")

Set varShell =wscript.CreateObject("WScript.Shell") 'Cria uma instância de objeto
```

```
if testetecla=2 then wscript.quit
if testetecla = 6 then ' o arquivo backup será criado...

    varData=year(now) & month(now) & day(now) & hour(now) & minute(now) &
second(now)

    msgbox varData

    frase="cmd /k CD D:\& mkdir d:\copia\" & vardata ' monta string criação nova
pasta
    msgbox frase
    varShell.run frase,1,true ' Cria no drive "d" uma pasta de nome "copia" e sub
pasta "yyyymmddhhmmss"
        ' Fará backup do arquivo

    novaSubPasta="d:\copia\" & varData & "\anterior.accdb"
    msgbox novaSubPasta
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    Set Wshs = WScript.CreateObject("WScript.Shell")
    usrProfile = Wshs.ExpandEnvironmentStrings("%UserProfile%")
    objFSO.CopyFile "d:\testebd.accdb" , novaSubPasta
    msgbox "Arquivo de backup gravado..."

end if

        'execução do Access
varShell.run (""msaccess.exe""C:\Users\Hamilton\Desktop\revenda.accdb"),3,true

if testetecla=6 then wscript.Echo "Sr. Usuário... foi gravado o arquivo backup ... favor
providenciar!"
    msgbox "operação realizada com sucesso!",0,"Sistema banco de Dados"
```

Bibliografia

MORGADO, Flávio. Programando em Microsoft Access com VBA, Rio de Janeiro, Ciência Moderna, 2008.

No desenvolvimento dos scripts da aula "AUTOMATIZAÇÃO DE TAREFAS", foram consultados os seguintes endereços web:

<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/wsconWSHBasics.asp>

[http://msdn.microsoft.com/en-us/library/d5fk67ky\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/d5fk67ky(v=vs.85).aspx)
http://www.macoratti.net/wsh_1.htm