



SUMÁRIO

CAPÍTULO 01 - INTRODUÇÃO AO VBA.....	4
1.1. Sobre a Apostila	4
1.2. Sobre o VBA.....	4
1.3. Características da Linguagem.....	4
1.4. Primeiro Contato com o VBA (Gravador de Macros).....	4
Guia Desenvolvedor.....	4
Usando o Gravador de Macros.....	5
Salvando as Macros.....	8
Abrindo arquivos contendo Macros	8
Alterando o nível de segurança da macro	9
Visualizando o código da macro (Visual Basic Editor – VBE)	10
Analizando uma macro pelo VBE	12
Excluindo Macros.....	15
Executando a Macro de outras formas	16
Usando o gravador de macro como auxílio ao VBE.....	20
CAPÍTULO 2 – CONCEITOS BÁSICOS DE PROGRAMAÇÃO VBA.....	22
2.1. Referências a Células	22
Instrução Range e Cells.....	23
2.2. Variáveis.....	23
2.3. Operadores Aritméticos e Funções Matemáticas	27
Operadores Aritméticos	27
Funções Matemáticas	28
2.4. Manipulando Strings.....	28
Concatenando Valores	28
Funções de String	29
2.5. Manipulando Data e Hora	30
Funções de Data e Hora	31
2.6. Fazendo Conversão de Tipo	32
CAPÍTULO 3 – USANDO ESTRUTURAS DE CONTROLE.....	33
3.1. Estruturas Condicionais	33
Operadores Condicionais	33
Operadores Lógicos.....	34
If...Then... End If.....	35
If...Then...Else...End If	35
Estruturas Encadeadas	36
Usando Elseif.....	36
Usando If Composto	37
Select Case...Case...End Select.....	37
Select Case...Case...Case Else...End Select	38
3.2. Estruturas de Repetição.....	39
For...Next	41
Do While...Loop (Loop Condicional)	42
Do Until...Loop (Loop Condicional)	43
While...Wend (Loop Condicional)	43
Do...Loop While / Do ...Loop Until (Loop Condicional)	43
For...Each.....	44
Combinando Estruturas condicionais e Repetição	44
Interrompendo fluxo de processamento: Exit Do, Exit For e Exit Sub.....	45
3.3. Verificando Tipo	47
3.4. Comparando Strings	48





VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

CAPÍTULO 4 – REGISTROS E MATRIZES.....	50
Matrizes.....	50
Sobre Matrizes:	50
Alterando o Índice da Matriz:	51
Redimensionando o Índice da Matriz (Matrizes Dinâmicas):	51
Preservando os valores de uma Matriz (Matrizes Dinâmicas):	52
Registros	53
 CAPÍTULO 5 – MANIPULANDO CÉLULAS.....	55
Seleção Estática de Células	55
Seleção Simples:	56
Seleção de um conjunto de células:	56
Seleção alternada de células:	56
Seleção Dinâmica de Células	56
Seleção de uma área de células:	57
Seleção o final de uma área de células:	57
Seleção conjunto de células em uma área de células:	58
Manipulando Valores da Célula	59
 CAPÍTULO 6 – FORMATANDO CÉLULAS.....	61
Formatando Valores.....	61
Formatando Células.....	61
Outras alternativas para alterar a cor de células:	62
Aplicando Bordas em Células da Planilha.....	63
Excluindo Conteúdo das Células	64
Excluindo e Inserindo Células na Planilha.....	65
Inserindo Células na Planilha:	65
Excluindo Células na Planilha:	65
Alinhando Conteúdo em Células	66
Alinhamento Horizontal:	66
Alinhamento Vertical:	67
Ajustando Espaçamento de Células	67
Mesclando Células.....	68
Usando Conjuntos.....	68
 CAPÍTULO 7 – SUBROTINAS E FUNÇÕES	69
Construindo Sub-rotinas	69
Criando Sub-rotinas:	69
Chamando Sub-Rotinas:	69
Passando Valores a Sub-Rotinas (ByVal):	71
Passando Valores opcionais a Sub-Rotinas (Optional):	71
Construindo funções (Métodos)	72
Criando Funções:	72
Passando Valores a Funções (ByVal):	72
Usando Referências públicas e privadas (Heranças de Objetos)	73
 CAPÍTULO 8 – USANDO INTERFACES	74
Janelas Modais	74
Formulários	76
Componentes de um Useform	78
Criando um Formulário	79
Configurando componentes do Formulário	93
<input type="checkbox"/> Encerrando um Formulário	105
<input type="checkbox"/> Carregando um Userform:	105
 CAPÍTULO 9 – MANIPULANDO OBJETOS E EVENTOS	106
9.1. Visão Geral de Objetos	106
9.2. Objeto Application	107





Conhecendo alguns Métodos e Propriedades do Objeto Application	107
9.3. Objeto Workbook.....	108
Conhecendo alguns Métodos, Propriedades e Conjuntos do Objeto Workbook	108
9.4. Objeto Worksheet.....	110
Conhecendo alguns Métodos, Propriedades e Conjuntos do Objeto Worksheet	110
9.5. Visão Geral de Eventos.....	110
9.6. Aplicação Prática para o Uso de Objetos	111
Criando um Encerramento para o Sistema de Registro de Vendas.....	112
Abrindo o Formulário Registro de Vendas ao Abrir a Pasta de Trabalho	113
Fechando a Pasta de Trabalho e Salvando os Dados da Planilha	113
 CAPÍTULO 10 – CRIANDO GRÁFICOS	114
Criando uma Planilha para a Origem de Dados do Gráfico	114
Criando um Gráfico Simples	116
Criando um Gráfico mais Detalhado.....	118
Modificando a Aparência do Gráfico.....	120
Excluindo o Gráfico Criado	121
 CAPÍTULO 11 – IMPRESSÃO DE DADOS	122
Demarcando a Área de Impressão	122
Controlando a Saída de Impressão	123
Ajustando a Orientação de Página para a Impressão	124
Centralizando o Alinhamento de Impressão	125
Configurando Cabeçalho e Rodapé na Impressão.....	126
Ajustando o Tamanho da Área de Impressão	128
 CAPÍTULO 12 – FILTRANDO DADOS	129
Criando o Formulário de Filtragem	129
Programando o Formulário de Filtragem.....	130
Programando a Execução da Filtragem pelo Formulário	132
 CAPÍTULO 13 – CRIANDO TABELA E GRÁFICOS DINÂMICOS	134
Criando um Formulário de Entrada de Dados para a Tabela Dinâmica.....	134
Programando o Formulário para criar o Relatório da Tabela Dinâmica	136
 CAPÍTULO 14 – CLASSIFICANDO DADOS	145
 CAPÍTULO 15 – CRIANDO SUBTOTAL	148
 CAPÍTULO 16 – TRATANDO E DEPURANDO ERROS	151
Executando Tratamento de Erros	151
Tipos de Erros Gerados por um Software	151
Instruções de Tratamento de Erros do VBA	152
Depurando Código	153
Principais Comandos de Depuração do VBE	154
 CAPÍTULO 17 – TRANSFERINDO DADOS ENTRE EXCEL E O ACCESS	156
Abordagem ao ADO:.....	157
Visão geral de um banco de dados	158
Habilitando componentes ADO:.....	159
Criando uma conexão com um banco de dados:	160
Consultando com um banco de dados:	161
Adicionando novos registros ao banco de dados:	163
Alterando registros do banco de dados:	164
Excluindo registros do banco de dados:	165
Protegendo o contra falhas de conexão:	166



CAPÍTULO 01 - INTRODUÇÃO AO VBA

1.1. Sobre a Apostila

Esta apostila foi confeccionada por profissionais da área tendo como objetivo principal um aprendizado eficiente do aluno em **Visual Basic Application (VBA)** dando as bases para programar aplicações automatizadas no programa **Microsoft Excel 2013**.

1.2. Sobre o VBA

O VBA é a principal linguagem de programação adotada pela Microsoft para desenvolver aplicações em programas do Microsoft Office.

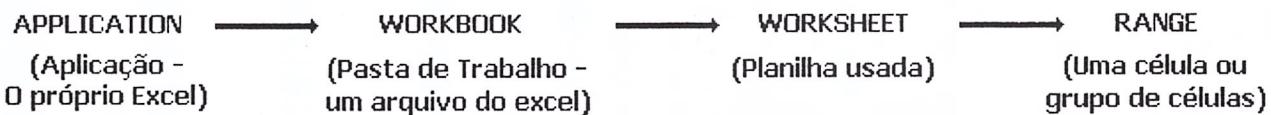
Embora o VBA tenha semelhança com o VB (Visual Basic) se diferencia dele pois existe particularidades ao tratar os aplicativos Microsoft Office tendo sua curva de aprendizado de acordo com o nível de conhecimento e dedicação a programação da linguagem.

1.3. Características da Linguagem

A principal característica na programação VBA é programar códigos dedicados a automatizar aplicativos Office, diferente da programação VB tradicional que tem como foco gerar códigos executáveis independentes de aplicação.

Outra característica em destaque é o conceito de programação Orientado a Objetos (POO) onde a manipulação de componentes dos aplicativos segue um princípio hierárquico:

Visão geral da estrutura de objetos manipulados pelo VBA no Excel



1.4. Primeiro Contato com o VBA (Gravador de Macros)

Geralmente o usuário comum do Excel, utilizando qualquer versão desse software, irá se deparar com a possibilidade de utilizar o gravador de macros para repetir as ações gravadas.

O Gravador de Macros é uma forma intuitiva de programação para leigos, porém estudaremos mais a fundo sobre esse recurso observando as possibilidades que ele nos pode trazer.

Guia Desenvolvedor

O primeiro procedimento a ser adotado é ativar a guia **Desenvolvedor**, por padrão ela vem desativada precisando ser ativada manualmente pelo usuário.





VBA

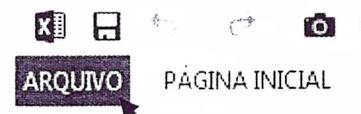
MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

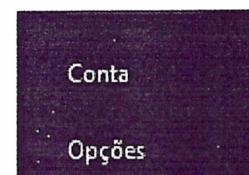


Siga os procedimentos abaixo para habilitá-la:

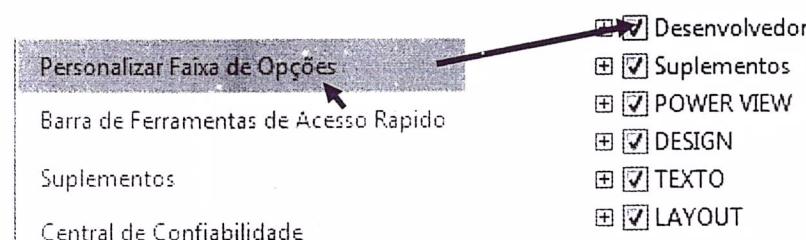
1º) Com o programa Microsoft Excel 2013 aberto, dê um clique no Botão Arquivo:



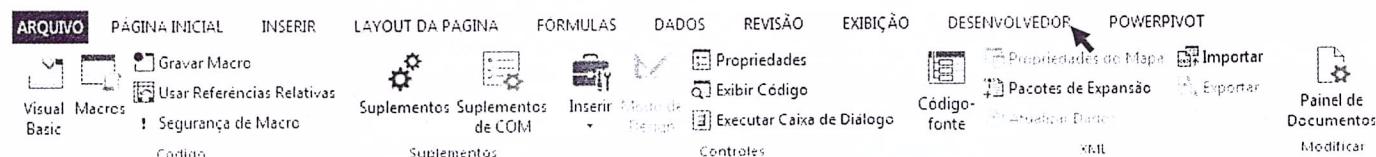
2º) Com o menu Office aberto, clique no botão Opções do Excel:



3º) Clique na caixa de seleção Mostrar guia Desenvolvedor na Faixa de Opções:



4º) Clique em Ok, a guia Desenvolvedor já poderá ser acessada na faixa de opções.



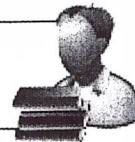
Usando o Gravador de Macros

Para gravar uma macro é relativamente simples, siga os passos a seguir:

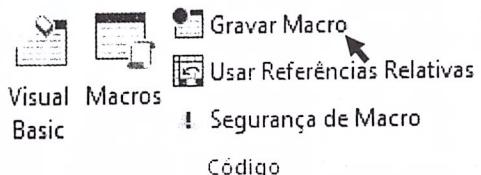
1º) Construa a planilha abaixo:

	A	B	C	D
1	Região	Data	Produto	Vendas
2	RJ	15/5/2010	A	1234
3	RJ	20/5/2010	B	2345
4	RJ	10/5/2010	C	4234
5	SP	20/5/2010	A	4221
6	SP	15/5/2010	B	4564
7	SP	10/5/2010	C	1775
8	BA	15/5/2010	A	3221
9	BA	20/5/2010	B	3674
10	BA	10/5/2010	C	1223

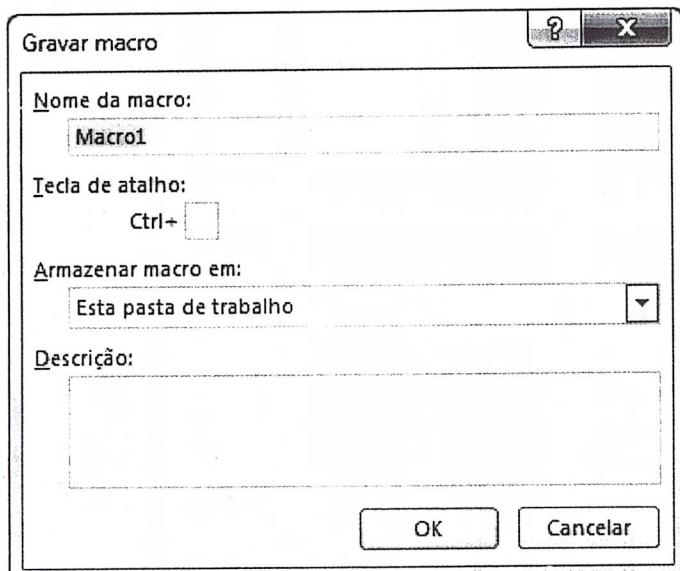




2º) Inicie o gravador de macros clicando em **Gravar macro**:



3º) Na caixa de diálogo Gravar Macro definimos a criação da macro:



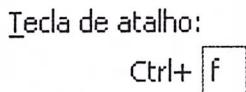
4º) Na caixa **Nome da macro** inserimos um nome para macro, conforme abaixo:



Observação quanto aos nomes de macro:

- 1) Caso não nomeie a macro um nome será adotado automaticamente;
- 2) Não é permitido nomear a macro com numeração inicial no início do nome;
- 3) Espaços não são permitidos no nome da macro;
- 4) Caracteres especiais também não serão permitidos.

5º) Insira uma tecla de atalho para permitir a execução da macro:



Observação quanto às teclas de atalho:

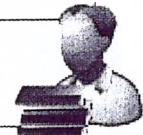
1. A tecla de atalho é opcional;
2. Para execução da tecla de atalho é digitada a letra em conjunto a tecla <CTRL>;
3. Caso exista conflito com alguma tecla de atalho usada no Excel é associada também a tecla <SHIFT>.





VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

6º) Na lista **Armazenar macro em**, selecionamos a pasta de trabalho no qual será armazenada a macro:

Armazenar macro em:

Esta pasta de trabalho

Existem três opções que podem ser escolhidas:

Esta pasta de trabalho: Salva a macro juntamente com o arquivo gravado.

Pasta de trabalho pessoal de macros: A macro ficará disponível para todos os arquivos criados no Excel, será criada uma pasta de trabalho pessoal de macros oculta (**Personal.xlsb**) e salva a macro nessa pasta de trabalho.

Nova pasta de trabalho: É uma macro temporária que só funciona no momento de sua criação, é pouco utilizada.

7º) Na caixa Descrição digitaremos um comentário sobre a macro, é opcional e não afeta o funcionamento da mesma:

Descrição:

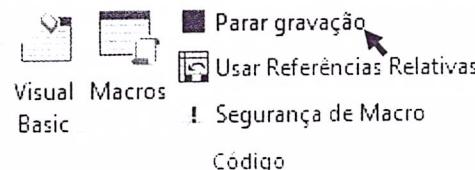
Esta mácro modifica o visual da planilha.

8º) Clique em **Ok** para iniciar a gravação da macro;

9º) Selecione a planilha e aplique as modificações abaixo:

	A	B	C	D
1	Região	Data	Produto	Vendas
2	RJ	15/05/10	A	1234
3	RJ	20/05/10	B	2345
4	RJ	10/05/10	C	4234
5	SP	20/05/10	A	4221
6	SP	15/05/10	B	4564
7	SP	10/05/10	C	1775
8	BA	15/05/10	A	3221
9	BA	20/05/10	B	3674
10	BA	10/05/10	C	1223

10º) Interrompa a gravação da macro clicando em **Parar gravação** na guia Desenvolvedor:



11º) Teste a macro removendo toda a formatação da planilha e executando a macro pela usando a tecla de atalho;

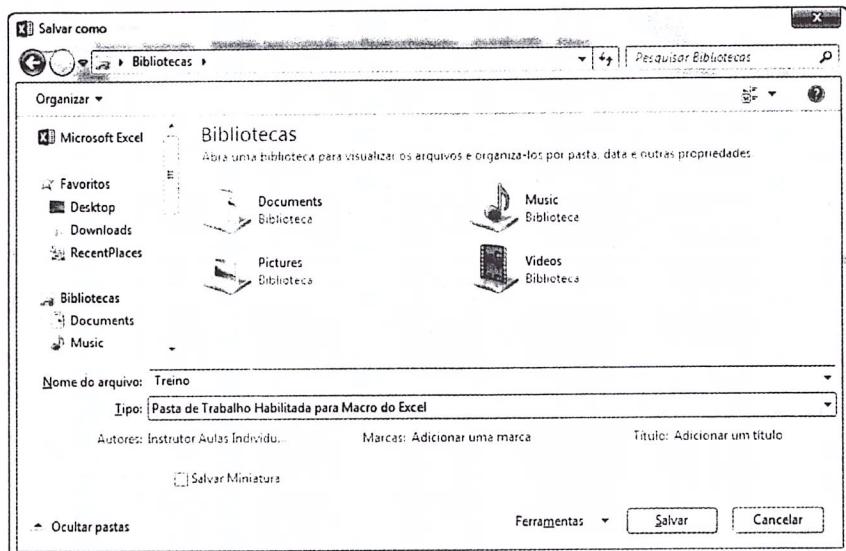




Salvando as Macros

Após criar as macros devemos salvar as pastas de trabalho que possuem macros, no Excel 2013 todos os arquivos que contém macros apresentarão extensão **XLSM** e os arquivos sem macro apresentarão extensão **XSLX**.

1º) Salve o arquivo criado escolhendo o tipo **Pasta de trabalho habilitada para Macro do Excel (*.xlsm)**:



2º) Arquivo gerado após a gravação, o ícone será diferenciado de um arquivo **XSLX**.



Nota: Ainda é possível criar arquivos XLS (válido nas versões do Excel 97 a 2003), atenção deve ser tomada nesse caso pois será perdido acesso aos novos recursos de programação do Excel 2013.

3º) Feche o Excel.

Abrindo arquivos contendo Macros

A partir da versão 2013 o Excel passou a verificar a execução de macros impondo normas de segurança verificando a confiabilidade da macro.

1º) Abra o arquivo **Treino.xlsm** e veja se mostrou a mensagem abaixo, para poder executar a macro clique no botão **Opções**:

! AVISO DE SEGURANÇA As macros foram desabilitadas.

Habilitar Conteúdo

x



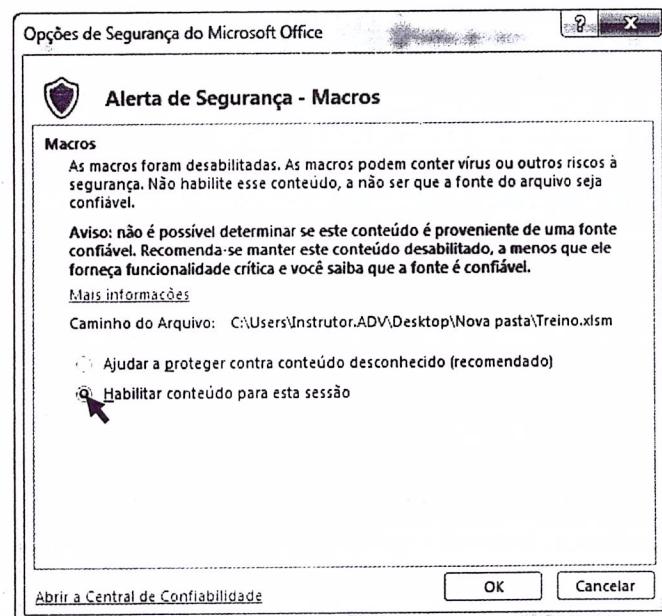


VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

2º) Para poder executar a macro selecione a opção Habilitar este conteúdo e clique em OK:

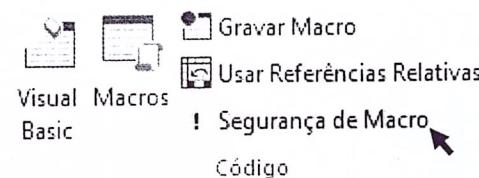


Essa verificação ocorre porque a Microsoft passou a adotar o esquema de Assinaturas Digitais que podem ser compradas pelo usuário, muito útil se for arquivos utilizados entre empresas diferentes.

Alterando o nível de segurança da macro

Se você é um usuário que cria suas próprias macros e não se importa em excutá-las sem medo podemos alterar o nível de segurança da macro.

1º) Na guia Desenvolvedor clicamos em Segurança de Macro:



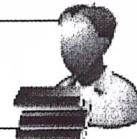
2º) No painel a seguir é exibido 4 Níveis de Segurança da macro:

Configurações de Macro

Para macros em documentos que não estão em um local confiável:

- Desabilitar todas as macros sem notificação
- Desabilitar todas as macros com notificação
- Desabilitar todas as macros, exceto as digitalmente assinadas
- Habilitar todas as macros (não recomendado; códigos possivelmente perigosos podem ser executados)



**Observação sobre os níveis de segurança da macro:**

Desabilitar todas as macros sem notificação: Desabilita todas as macros por completo;

Desabilitar todas as macros com notificação: Permite que o usuário decida quando deve ou não permitir executar uma macro (opção padrão);

Desabilitar todas as macros, exceto as assinadas digitalmente: Permite executar somente macros assinadas digitalmente.

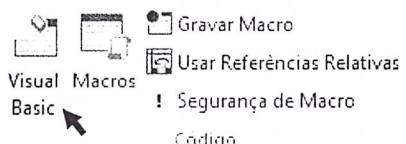
Habilitar todas as macros: Essa opção permite execução de qualquer macro ao abrir o arquivo no Excel ficando por sua conta e risco, para o nossos estudos o ideal será que essa opção esteja habilitada.

3º) Selecione a última opção e clique em **OK** para desabilitar a verificação de macros.

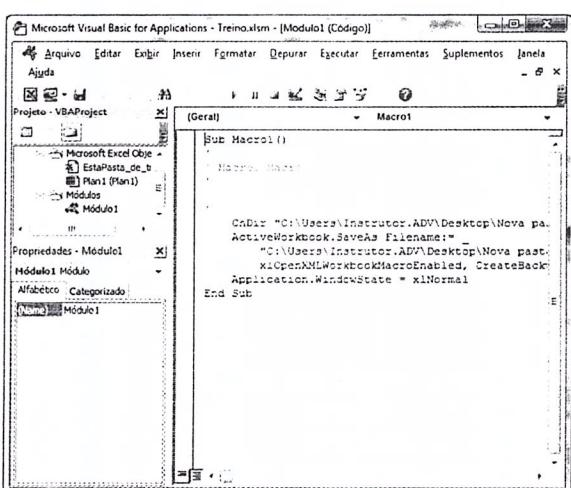
Visualizando o código da macro (Visual Basic Editor – VBE)

Nosso próximo passo é visualizar o código gerado pelo gravador de macros utilizando o editor de macros do VBA, chamado de VBE.

1º) Na guia **Desenvolvedor** clique no botão Visual Basic:



2º) Será exibido a janela do VBE:

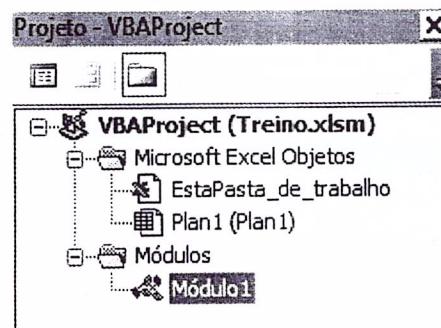


3º) Existem três janelas principais no editor VBE, são elas:





- Janela Project Explorer: Exibe a relação de componentes em uso relacionadas à macro.



O **VBAProject** está relacionado ao arquivo em que a macro está gravada, existem duas pastas internas associadas que podem ser expandidas clicando no sinal de mais (+).

A pasta **Microsoft Excel Objetos**, podem conter código específico a determinado objeto, podendo ser:

Esta pasta de trabalho: contém módulos relativos ao próprio arquivo (Workbook).

Plan1, Plan*, ...: contém módulos de programação associados a planilhas específicas (Worksheet).

Formulários: armazena formulários para interagir com o usuário.

Módulos: Armazena módulos de programação gerados pelo gravador de macros.

Módulos de classe: permite criar módulos que possam ser utilizados por qualquer programador.

- Janela Módulo: Armazena todo o código VBA, no nosso caso o código gerado pela macro.

```
(Geral) Formatação
Sub Formatação()
    ' Formatação Macro
    ' Esta mácro modifica o visual da planilha.

    ' Atalho do teclado: Ctrl+f

    Range("A1:D10").Select
    With Selection
        .HorizontalAlignment = xlCenter
        .VerticalAlignment = xlBottom
        .WrapText = False
        .Orientation = 0
        .AddIndent = False
        .IndentLevel = 0
        .ShrinkToFit = False
        .ReadingOrder = xlContext
        .MergeCells = False
    End With
    Range("B2:B10").Select
    Selection.NumberFormat = "dd/mm/yy;@"
    Range("A1:D10").Select
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    With Selection.Borders(xlEdgeLeft)
```





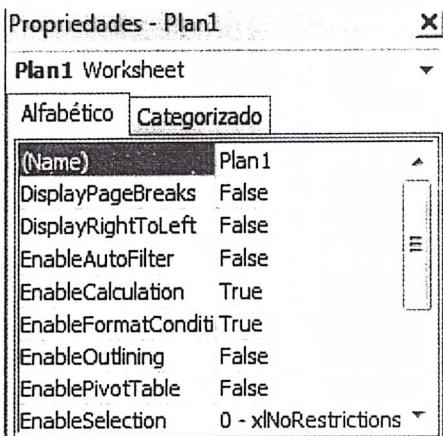
VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno



- **Janela Propriedades:** Permite editar a propriedade de vários componentes variando de acordo com o objeto selecionado.

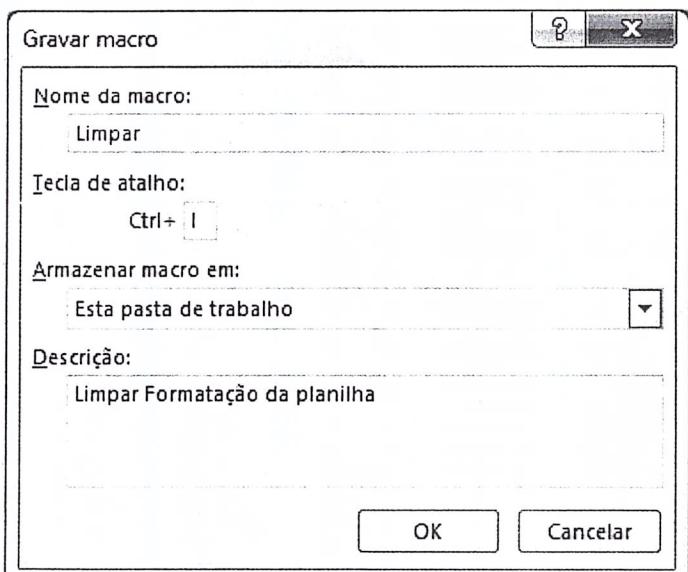


Analizando uma macro pelo VBE

Conforme já foi visto, todas as ações realizadas pelo usuário podem ser registradas pelo gravador de macros.

Crie e grave uma nova macro seguindo as orientações abaixo:

- 1º) Grave uma nova macro conforme abaixo:



- 2º) Selecione toda a planilha e retire todos os formatos aplicados:



- 3º) Pare a gravação da macro;





4º) Abra o VBE e verifique se ficou gravado o código da macro dentro do módulo, provavelmente foi criado um Módulo2:

```
Microsoft Visual Basic for Application...
Arquivo Editar Exibir Inserir Formatar
Depurar Executar Ferramentas Suplementos
Janela Ajuda
(Geral) Limpar
Sub Limp...
' Limpar Macro
' Limpar Formatação da planilha
'
' Atalho do teclado: Ctrl+I
'
Range("A1:D10").Select
Selection.ClearFormats
End Sub
```

Observação sobre a macro Limpar():

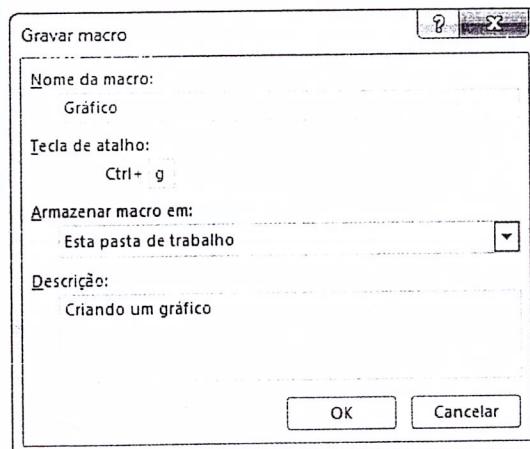
Se você notar nessa macro foi gerado menos código e todos os comandos escritos equivalem aos comandos executados:

1. O comando **Range("A1:D10").Select**, está relacionado a todas as células selecionadas da planilha.
2. O comando **Selection.ClearFormats** significa a exclusão de toda a formatação da planilha.

Embora o código seja bastante simples, é possível ter uma idéia do que está sendo executado no VBA.

5º) Feche a janela do VBE;

6º) Grave uma outra macro para realizar a criação de um gráfico:





7º) Selecione as regiões de células das colunas A, B e D da planilha:

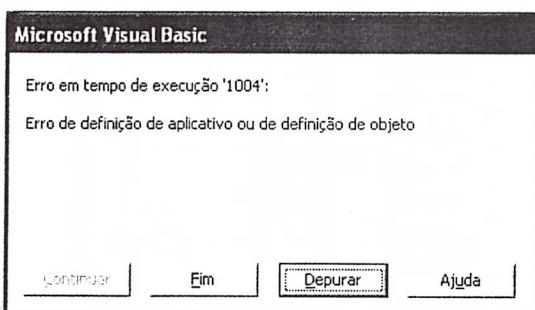
	A	B	C	D
1	Região	Data	Produto	Vendas
2	RJ	15/05/10	A	1234
3	RJ	20/05/10	B	2345
4	RJ	10/05/10	C	4234
5	SP	20/05/10	A	4221
6	SP	15/05/10	B	4564
7	SP	10/05/10	C	1775
8	BA	15/05/10	A	3221
9	BA	20/05/10	B	3674
10	BA	10/05/10	C	1228

8º) Crie o gráfico abaixo:



9º) Exclua o gráfico e tente a macro pela tecla de atalho;

10º) Atenção, tivemos um erro!!!



11º) Porque ocorreu um erro? Não gravamos os passos para criar um gráfico?



VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

12º) Clique no botão depurar e será indicado em amarelo o erro no código:

```
Sub Gráfico()
    ' Gráfico Macro
    ' Criando um gráfico
    ' Atalho do teclado: Ctrl+g

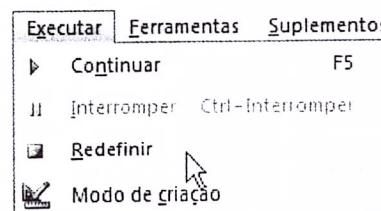
    Range("A1:B4,D1:D4").Select
    Range("D4").Activate
    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.SetSourceData Source:=Range("'Plan1'!$A$1")
    ActiveChart.ChartType = xlColumnClustered
End Sub
```

Nota: Esse tipo de erro é chamado **Erro de tempo de execução**, ou seja, embora tenham sido feitos todos os passos corretamente pelo usuário o gravador de macro interpretou errado e gerou um resultado inesperado.

Quando esses erros ocorrem o código entre imediatamente em **modo de depuração** aguardando correção do erro.

Não se preocupe, esses tipos de erros já são esperados do gravador pois ele não é perfeito, em nosso estudo, aprenderemos a fundo como gerar um código eficiente e funcional.

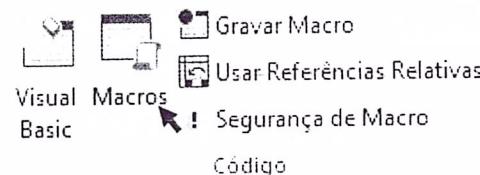
13º) Para interromper o modo de depuração, na janela do VBE clique no menu Executar e clique em Redefinir;



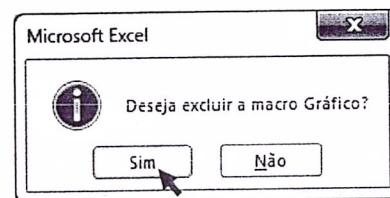
14º) Feche o editor do VBE.

Excluindo Macros

1º) Na guia Desenvolvedor clique no botão Macro:



2º) Clique em Sim para confirmar a exclusão da macro:





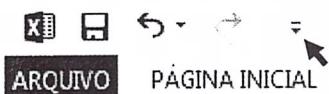
Nota: Só serão exibidas as macros a serem excluídas somente se forem criadas pelo gravador de macros.

Executando a Macro de outras formas

Podemos executar a macro de diferentes maneiras além da tecla de atalho, são elas:

- **Barra de ferramentas acesso rápido:** Podemos associar uma macro a botão na barra de ferramentas de acesso rápido.

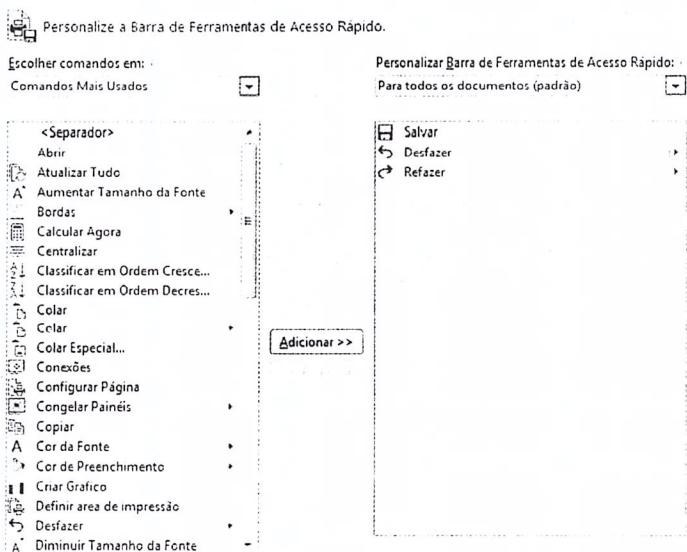
1º) Clique no botão lateral Personalizar Barras de ferramentas de acesso rápido;



2º) Clicamos em seguida na lista a opção Mais comandos;



3º) Na janela exibida escolhemos os comandos de Macros e adicionamos os dois comandos que criamos a barra de ferramentas:





4º Podemos alterar os ícones dos comandos de macro selecionando a macro e clicando no botão **Modificar**:

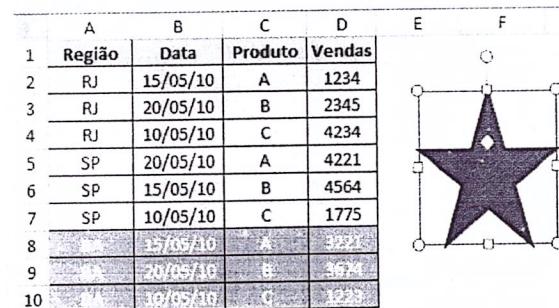


5º Após as alterações clique em Ok e a macro já estará disponível para execução pelo usuário.

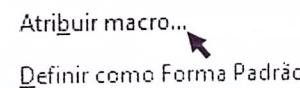


- **Atribuir macro a uma forma ou imagem:** Caso sua planilha disponha de alguma forma ou imagem podemos também atribuir uma macro.

1º Selecione uma forma gráfica na planilha:

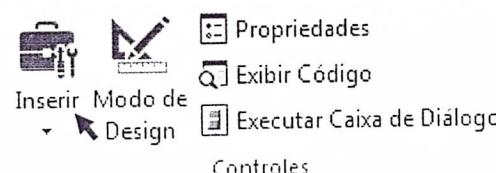


2º Clique com o botão direito na forma e clique na opção Atribuir Macro:



- **Atribuir macro a um controle de formulário:** Poderemos também inserir componentes de formulário que são muito mais eficientes que os meios anteriores.

1º Na guia Desenvolvedor clique no botão Inserir:





2º) Será exibida duas categorias de controles: formulário e outra ActiveX (veremos seu uso mais adiante):

Controles de Formulário



Controles ActiveX



3º) Clique no controle de formulário botão:

Controles de Formulário

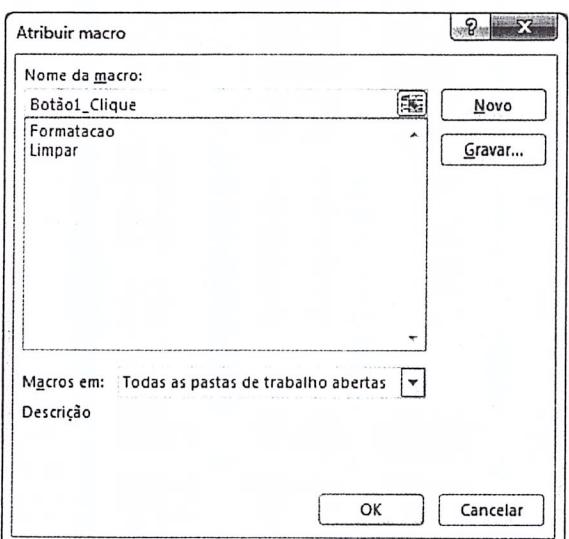


Botão (Controle de Formulário)

Controles ActiveX



4º) Em seguida, Clique na planilha e selecione o módulo de macro que será executado pelo botão:



5º) Clique em OK, para testar o botão clique na planilha e em seguida clique no botão para ver o resultado.

Criando módulos no VBE

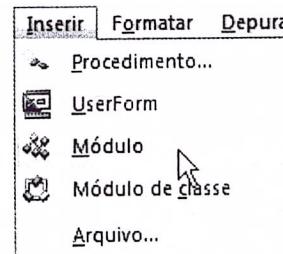
Durante nosso estudo vimos à possibilidade de realizarmos programação do VBA pelo gravador de macros, mas como pode perceber ele é limitado e existe o risco de gerar códigos errados, a partir de agora criaremos nossos próprios códigos diretamente no VBE.

1º) Abra o editor VBE;

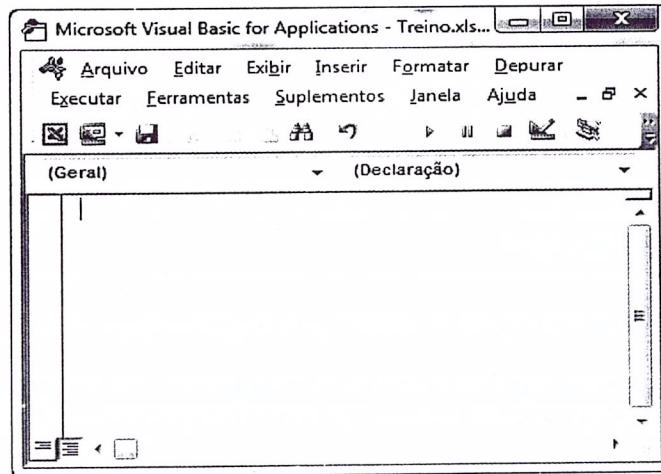




2º) Clique no menu Inserir e clique em Módulo;



3º) Será criada a janela de um módulo em branco:



4º) Para começar a inserir os códigos no módulos digite as linhas de instruções abaixo:

```
Sub CalculaTotal()  
End Sub
```

Nota: Essas linhas significam que será construída uma “**Sub-rotina**”, uma sub-rotina tem como característica armazenar todo o código a ser executado pela macro.

5º) Digitamos as linhas de código abaixo dentro das linhas da sub-rotina:

```
Sub CalculaTotal()  
  
    Range("C11") = "Total"  
    Range("D11").FormulaLocal = "=soma(D2:D10)"  
  
End Sub
```

6º) Aparentemente essas linhas não apresentaram nenhum sentido para você, para descobrirmos o que elas fazem, clique na linha inicial do programa, vá teclando a tecla F8 (tecla de função), uma barra amarela irá percorrer o código até a última linha do código:

```
Sub CalculaTotal()  
    Range("C11") = "Total"  
    Range("D11").FormulaLocal = "=soma(D2:D10)"  
End Sub
```





VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno



Nota: A tecla de função F8 é muito utilizada quando queremos depurar o código!

7º) Feche o VBE e observe o que aconteceu no final da planilha:

	A	B	C	D
1	Região	Data	Produto	Vendas
2	RJ	15/05/10	A	1234
3	RJ	20/05/10	B	2345
4	RJ	10/05/10	C	4234
5	SP	20/05/10	A	4221
6	SP	15/05/10	B	4564
7	SP	10/05/10	C	1775
8	PA	15/05/10	A	1775
9	PA	20/05/10	B	1775
10	PA	10/05/10	C	1775
11		Total		26491

Nota: Observe que foi feito duas coisas, na célula C11 foi escrito a palavra Total e na célula D11 surgiu uma função de SOMA.

8º) Olhando o resultado final nota-se um resultado simples gerado pelo código VBA digitado;

9º) Para facilitar sua futura compreensão podemos inserir "Comentários", um comentário é uma nota colocada dentro de um código para lembrar ao programador do que se trata. Insira comentários conforme abaixo:

```
Sub CalculaTotal()

    ' Escreve a palavra TOTAL na célula C11
    Range("C11") = "Total"

    ' Escreve a função SOMA na célula D11
    Range("D11").FormulaLocal = "=soma(D2:D10)"

End Sub
```

Nota: Os comentários são ignorados durante o processamento de um código.

Usando o gravador de macro como auxílio ao VBE

Embora o gravador de macros possa parecer limitado a princípio, esse recurso poderá ser útil na criação de códigos no VBE podendo servir como uma espécie de "cola de código", ou seja, se você não lembrar exatamente dos códigos utilizados, então, o gravador poderá ser útil nesse caso.

1º) Crie a planilha abaixo para facilitar seu entendimento;

	A	B	C	D	E
1	Vendedor	Janeiro	Fevereiro	março	Total
2	Marcos	R\$ 2.000,00	R\$ 3.000,00	R\$ 4.000,00	
3	Ana	R\$ 1.500,00	R\$ 2.600,00	R\$ 7.400,00	
4	Pedro	R\$ 3.000,00	R\$ 2.500,00	R\$ 1.200,00	
5	Paulo	R\$ 1.100,00	R\$ 1.300,00	R\$ 1.500,00	
6	Ricardo	R\$ 2.600,00	R\$ 2.000,00	R\$ 1.999,00	
7	Renato	R\$ 1.200,00	R\$ 1.400,00	R\$ 1.700,00	
8	Roberto	R\$ 3.600,00	R\$ 1.200,00	R\$ 2.600,00	





VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

2º) Abra o VBE e Insira um módulo em branco;

3º) Digite o código abaixo dentro do módulo:

```
Sub total()
    ' Realiza a soma da Range de células de B2 à D2
    Range("E2").FormulaLocal = "=SOMA(B2:D2)"

    ' Formata o valor no formato MOEDA
    Range("E2").Style = "Currency"
End Sub
```

4º) Execute o código e verifique se você obteve o resultado abaixo na planilha:

	A	B	C	D	E
1	Vendedor	Janeiro	Fevereiro	março	Total
2	Marcos	R\$ 2.000,00	R\$ 3.000,00	R\$ 4.000,00	R\$ 9.000,00
3	Ana	R\$ 1.500,00	R\$ 2.600,00	R\$ 7.400,00	
4	Pedro	R\$ 3.000,00	R\$ 2.500,00	R\$ 1.200,00	
5	Paulo	R\$ 1.100,00	R\$ 1.300,00	R\$ 1.500,00	
6	Ricardo	R\$ 2.600,00	R\$ 2.000,00	R\$ 1.999,00	
7	Renato	R\$ 1.200,00	R\$ 1.400,00	R\$ 1.700,00	
8	Roberto	R\$ 3.600,00	R\$ 1.200,00	R\$ 2.600,00	

Nota: Observe que a macro atende ao cálculo do resultado para uma linha, se quisermos que seja executado o cálculo para outras linhas e não sabemos como criar o código?

A resposta é: **Gravador de Macros.**

5º) Usando o gravador de macros faça os seguintes procedimentos:

- Inicie a gravação de uma Macro chamada **procedimento**;
- Selecione a **célula E2**;
- Dê um duplo clique na **alça de preenchimento** para que sejam calculados as demais linhas da planilha;
- Encerra a gravação da macro.

6º) Após esse passos gravados na macro procedimento tente visualizar o código abaixo:

```
Sub procedimento()
    ' procedimento Macro
    ' ...

    Range("E2").Select
    Selection.AutoFill Destination:=Range("E2:E8")
    Range("E2:E8").Select
End Sub
```

7º) Observe que na macro que criamos somente as linhas abaixo nos interessam pois são as que realmente executam o código;

```
Range("E2").Select
Selection.AutoFill Destination:=Range("E2:E8")
```



ADV – Curso de Informática Telefone: (21) 2210-1180
 Av. Treze de Maio, 23 – 8º andar – Centro – Rio de Janeiro - RJ
<http://www.cursoadv.com.br/>

Página 21



VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

8º) Copie esse trecho de código para a macro que criamos, veja o resultado abaixo:

```
Sub total()
    'Realiza a soma da Range de células de B2 à D2
    Range("E2").FormulaLocal = "=SOMA(B2:D2)"

    'Formata o valor no formato MOEDA
    Range("E2").Style = "Currency"

    'Seleção da célula que contém a fórmula de cálculo
    Range("E2").Select

    'Preenche o restante das células da mesma coluna
    'com a fórmula copiada
    Selection.AutoFill Destination:=Range("E2:E8")
End Sub
```

Nota: Observe que estão sendo também inseridos os comentários para ficar claro o que cada linha executa dentro do código.

9º) Execute o código de novo e veja se foi obtido o resultado esperado na planilha:

	A	B	C	D	E
1	Vendedor	Janeiro	Fevereiro	março	Total
2	Marcos	R\$ 2.000,00	R\$ 3.000,00	R\$ 4.000,00	R\$ 9.000,00
3	Ana	R\$ 1.500,00	R\$ 2.600,00	R\$ 7.400,00	R\$ 11.500,00
4	Pedro	R\$ 3.000,00	R\$ 2.500,00	R\$ 1.200,00	R\$ 6.700,00
5	Paulo	R\$ 1.100,00	R\$ 1.300,00	R\$ 1.500,00	R\$ 3.900,00
6	Ricardo	R\$ 2.600,00	R\$ 2.000,00	R\$ 1.999,00	R\$ 6.599,00
7	Renato	R\$ 1.200,00	R\$ 1.400,00	R\$ 1.700,00	R\$ 4.300,00
8	Roberto	R\$ 3.600,00	R\$ 1.200,00	R\$ 2.600,00	R\$ 7.400,00

CAPÍTULO 2 – CONCEITOS BÁSICOS DE PROGRAMAÇÃO VBA

Vimos que no estudo básico dos códigos gerados com macros parecem que são estáticos, ou seja, os códigos gerados não parecem desempenhar muita funcionalidade até agora.

Aprenderemos nesse tópico alguns recursos básicos de programação que facilitarão a confecção de códigos cuja lógica iremos desenvolver na maioria dos programas no VBA.

2.1. Referências a Células

Praticamente em todos os códigos desenvolvidos em VBA fazemos referência a Células pois nessa linguagem a única maneira do código se comunicar com o mundo exterior será criando referências usando dois tipos de instruções: RANGE e CELLS.





Instrução Range e Cells

- Instrução RANGE: Cria uma referência nominal ao endereço da célula. Observe a planilha abaixo:

	A	B	C	D	E
1					
2		Produto	Valor	Quantidade	Total
3	Refrigerante	R\$ 1,30	5		

Se quisermos exibir o total do cálculo da multiplicação do valor com a quantidade bastaria clicar na célula E3 e inserir a fórmula cálculo =C3 * D3, nesse cálculo resultará no valor 6,5 (R\$ 6,50).

Programando o mesmo cálculo através da programação VBA usando a instrução RANGE ficaria assim:

```
Sub calculo()
    Range("E3") = Range("C3") * Range("D3")
End Sub
```

Observe que embora pareça estranho, esse pequeno programa realiza a mesma tarefa que um cálculo convencional realizado diretamente na célula onde cada célula é referenciada pelo seu endereço, assim como em uma fórmula de cálculo convencional.

- Instrução 'CELLS (y,x)': Cria uma referência numérica ao endereço da célula sendo esse endereço tratado como uma coordenada numérica "x" para a coluna e uma coordenada numérica "y" para a linha.

Poderíamos escrever o mesmo código anterior na seguinte forma:

```
Sub calculo()
    Cells(3, 5) = Cells(3, 3) * Cells(3, 4)
End Sub
```

Onde a coluna B = 5, a coluna C = 3 e a coluna D = 4.

A instrução Cells realiza um tipo de referência chamada **matricial** que, embora pareça a princípio nada fácil de usar, é útil pois ao adotar valores no lugar de letras possibilita criar referências rápidas e simples com células.

2.2. Variáveis

A variável é um recurso utilizado em todas as linguagens de programação para reservar espaço na memória (alocação), o VBA não é exceção.

Durante a execução de um código (processamento) valores precisam ser armazenados para serem utilizados em um momento certo.





VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

Por exemplo, suponha que o programa quando executado recebe dois valores, realiza um cálculo e retorna o resultado desse cálculo em um determinado momento. Veja a representação do uso das variáveis abaixo:

Variável	Valores
A	12
B	2

Cálculo	Resultado
Soma (A + B)	14
Subtração (A - B)	10
Multiplicação (A x B)	24
Divisão (A ÷ B)	6

Procedimentos para declarar variáveis:

- **Declaração da Variável:** Informamos que o código usará variável;
- **Identificadores:** Identificamos essas variáveis com um nome para que o código possa reconhecê-la;
- **Tipo de dado:** Informamos que tipo de valor essa variável irá armazenar (texto, valor, booleano, etc);
- **Atribuição de valores:** Armazenamos o valor na variável.

Código abaixo foi modificado para usar variáveis

Código Original (Sem variáveis):

```
Sub calculo()
    Cells(3, 5) = Cells(3, 3) * Cells(3, 4)
End Sub
```

Código Original (Com variáveis):

```
Sub teste()
    'Declaração de variáveis
    Dim A, B, Resultado As Single

    'Atribuição de valores as variáveis
    A = Cells(3, 3)
    B = Cells(3, 4)
    Resultado = A * B

    'Retornando valor da variável
    Cells(3, 5) = Resultado

End Sub
```

Na declaração das variáveis usamos a instrução 'Dim', o tipo é precedido pela cláusula 'As'. Repare também que na atribuição de valores usamos o sinal de igual (=) ou operador de atribuição para indicar recebimento de valores pela variável.

Embora aparentemente mais complexo, o código usando variáveis apresenta a praticidade de ser adaptável, e quanto mais complexo for o código maior será utilizado variáveis.

Regras para se declarar nomes de variáveis (Identificadores):

Regras de declaração de variáveis:

1º Regra: Nomes de variáveis devem começar exclusivamente com letras.

2º Regra: O nome deve conter letras, números, caracteres de sublinhado (nunca espaço).

3º Regra: Não devem exceder 40 caracteres.

4º Regra: Não devem ser utilizadas palavras reservadas pela linguagem.





Abaixo é exibida uma tabela contendo os tipos de dados que podemos definir para variáveis

Tipo de dados	Tamanho de armazenamento	Intervalo
Byte (inteiro simples)	1 byte	de 0 a 255
Boolean (booleano ou lógico)	2 bytes	True ou False
Integer (inteiro)	2 bytes	de -32.768 a 32.767
Long (número inteiro longo)	4 bytes	de -2.147.483.648 a 2.147.483.647
Single (número decimal de precisão simples)	4 bytes	de -3,402823E38 a -1,401298E-45 para valores negativos; de 1,401298E-45 a 3,402823E38 para valores positivos
Double (número decimal de dupla precisão)	8 bytes	de -1,79769313486231E308 a -4,94065645841247E-324 para valores negativos; de 4,94065645841247E-324 a 1,79769313486232E308 para valores positivos.
Currency (valores monetários)	8 bytes	de -922.337.203.685.477,5808 a 922.337.203.685.477,5807
Decimal	14 bytes	+/-79.228.162.514.264.337.593.543.950.335 sem vírgula decimal; +/-7,9228162514264337593543950335 com 28 casas decimais à direita; o menor número diferente de zero é +/- 0,00000000000000000000000000000001.
Date	8 bytes	De 1 de janeiro de 100 a 31 de dezembro de 9999
String (comprimento variável)	10 bytes + comprimento da seqüência	De 0 a aproximadamente 2 bilhões
String (comprimento fixo)	Comprimento da seqüência	De 1 a aproximadamente 65.400
Variant (com números)	16 bytes	Qualquer valor numérico até o intervalo de um Double
Variant (com caracteres)	22 bytes + comprimento da seqüência	O mesmo intervalo de String de comprimento variável
Objeto	4 Bytes	Qualquer referência a objeto

Nota: Além do **tipo de dado** definir que informação será armazenada pela variável também será definindo o quanto de **espaço** essa informação irá ocupar na **memória**.

Declarando Variáveis usando Sufixos:

Tipo de dado	Sufixo alternativo	Exemplos equivalentes
Integer	%	Dim Valor As Integer (OU) Dim Valor%
Long	&&	Dim Valor As Long (OU) Dim Valor&&
Single	!	Dim Valor As Single (OU) Dim Valor!
Double	#	Dim Valor As Double (OU) Dim Valor#
Currency	@	Dim Valor As Currency (OU) Dim Valor@
String	\$	Dim Valor As String (OU) Dim Valor\$

Nota: A declaração usando sufixos simplifica a declaração de variáveis, embora a declaração convencional seja a mais utilizada.

Declarando variáveis Globais e Locais:

- Variáveis Globais:** Existe durante toda a execução do código e mantêm esses valores após a execução do código, todas as macros poderão acessá-las. A variável global é declarada fora do código da macro.

Exemplo de código com variável global:

- 1º) Construa a planilha abaixo:

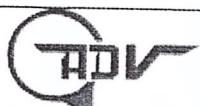
A	B	C	D	E	F
1					
2	Mês	Jan	Fev	Mar	Abr
3	Valor	R\$ 1.000,00	R\$ 1.300,00	R\$ 1.200,00	R\$ 1.500,00
4					
5				CALCULAR	
6				Resultado	
7					RESULTADO
8					

- 2º) Construa os módulos abaixo e atribua um módulo a cada botão:

```
'Declarando variáveis globais
Dim val1, val2, val3, val4 As Currency
Sub calcular()
    'Atribuindo valores as variáveis
    val1 = Range("C3")
    val2 = Range("D3")
    val3 = Range("E3")
    val4 = Range("F3")
End Sub

Sub resultado()
    'Declarando variável local
    Dim resultado As Currency
    resultado = val1 + val2 + val3 + val4

    'Retornando o valor do cálculo
    Range("F6") = resultado
End Sub
```





3º) Primeiramente clique no botão **Calcular** e depois no botão **Resultado**:

	A	B	C	D	E	F
1						
2		Mês	Jan	Fev	Mar	Abr
3		Valor	R\$ 1.000,00	R\$ 1.300,00	R\$ 1.200,00	R\$ 1.500,00
4					CALCULAR	
5						Resultado R\$ 5.000,00
6						RESULTADO
7						
8						

4º) Se você clicar inicialmente nesses botões ao contrário será exibido o resultado zero;

5º) Após clicar no botão calcular será exibido sempre o mesmo resultado a não ser que sejam fornecidos novos valores.

- **Variáveis Locais:** Ao contrário das variáveis globais as variáveis locais só mantêm o valor durante a execução do código do módulo.

Alterando a localização das variáveis locais para dentro do módulo Calcular()

```
Sub calcular()
    'Declarando variáveis globais
    Dim val1, val2, val3, val4 As Currency

    'Atribuindo valores as variáveis
    val1 = Range("C3")
    val2 = Range("D3")
    val3 = Range("E3")
    val4 = Range("F3")
End Sub
```

Observe que o módulo **Resultado()** não exibe mais o resultado do cálculo.

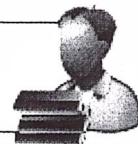
2.3. Operadores Aritméticos e Funções Matemáticas

Operadores Aritméticos

Toda linguagem de programação dispõe de operadores aritméticos para realizar operações matemáticas podendo ser simples ou complexas:

Operador	Função	Exemplo
+	Soma dois números	4 + 5 (Resultado: 9)
-	Subtrai dois números	4 - 6 (Resultado: -2)
*	Multiplica dois valores	4 * 6 (Resultado: 24)
/	Divide dois números e retorna um resultado decimal	4 / 6 (Resultado: 0,6666667)
\	Divide dois números e retorna somente a parte inteira	4 \ 6 (Resultado: 0)
Mod	Retorna o resto de uma divisão	6 Mod 2 (Resultado: 0)
^	Eleva um número a potência de um expoente	2 ^ 4 (Resultado: 16)
()	Usando parênteses para alterar a sequência de cálculo	Cálculo com e sem parênteses: (1+2+3+4)*6 => (Resultado: 60) 1+2+3+4*6 => (Resultado: 30)





Funções Matemáticas

São utilizadas em conjunto com valores matemáticos com objetivo de gerar operações matemáticas complexas:

Função	Descrição	Exemplo
Int	Retorna a parte inteira de um valor.	Int(9.5) (Resultado: 9)
Round	Arredonda a casa decimal de um valor.	Round (3.545,2) (Resultado: 3.54)
Rnd	Retorna um valor aleatório.	Rnd() * 6 (Resultado: \approx 4,23)
Sqr	Obtém a raiz quadrada de um valor	Sqr(9) (Resultado: 3)

Exemplo de um código que retorna a raiz quadrática de um valor:

1º) Construa a planilha abaixo:

	A	B	C	D
1				
2	A	B	C	
3	-1	2	6	
4				
5	Resultado de X:			
6	Resultado de X':			
7				
8	CALCULAR EQ. 2º GRAU			
9				

2º) Construa o módulo abaixo:

```
Sub equ_2grau()
    Dim a, b, c As Integer
    Dim x, y As Integer
    a = Cells(3, 2)
    b = Cells(3, 3)
    c = Cells(3, 4)

    x = Round(Sqr((b ^ 2) - (4 * a * c)) / (2 * a), 2)
    y = Round(-Sqr((b ^ 2) - (4 * a * c)) / (2 * a), 2)

    Cells(5, 4) = x
    Cells(6, 4) = y
End Sub
```

3º) Atribua esse função ao botão de comando e execute-a em seguida;

4º) O Resultado é mostrado abaixo:

Resultado de X:	-2,65
Resultado de X':	3

2.4. Manipulando Strings

Concatenando Valores

No VBA é possível unir valores numéricos com strings através do operadores de concatenação "&" ou "+" ao utilizar, todo valor concatenado com string se torna parte da string. Abaixo exemplo de um código que concatena:





1º) Crie o módulo abaixo:

```
Sub idade()
    Dim nome As String
    Dim ano, idade As Integer

    nome = Range("C2")
    ano = Range("C3")
    idade = 2010 - ano
    Range("C4") = nome & ", você têm " & idade & " anos"
End Sub
```

2º) Crie a planilha abaixo e associe o módulo ao botão:

	A	B	C
1			
2	Nome	Paulo	
3	Ano de Nascimento	1975	
4	Idade		
5			
6			EXIBIR IDADE

3º) Clique no botão e veja se exibiu a mensagem abaixo:

Idade	Paulo, você têm 35 anos
-------	-------------------------

Funções de String

O VBA possui um número grande de funções para manipulação de strings, as strings abrangem um número relativamente alto de informações como, por exemplo, números telefônicos e de CPF, nomes de pessoas, endereços, e etc.

Abaixo é exibida uma tabela com algumas das principais funções de string do VBA

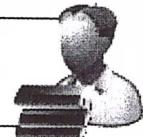
Função	Descrição	Exemplo
Len	Retorna a quantidade numérica de caracteres que formam a string. Sintaxe: Len(String)	Len("Casa") => [Resultado: 4]
Mid	Retorna parte de uma string. Sintaxe: Mid(String, posição, nº caracteres)	Mid("Casa", 2, 3) => [Resultado: "asa"] Mid("Casa", 1, 1) => [Resultado: "a"]
InStr	Retorna a posição numérica de um caractere dentro de uma string. Sintaxe: InStr(String, caracterer)	InStr("Casa", "s") => [Resultado: 3] InStr("Casa", "C") => [Resultado: 1]
Left	Retorna parte de uma string a partir da esquerda. Sintaxe: Left(String, nº caracteres)	Left("Casa", 1) => [Resultado: "C"] Left("Casa", 2) => [Resultado: "Ca"] Left("Casa", 3) => [Resultado: "Cas"]
Rigth	Retorna parte de uma string a partir da direita. Sintaxe: Left(String, nº caracteres)	Rigth("Casa", 1) => [Resultado: "a"] Rigth("Casa", 2) => [Resultado: "sa"] Rigth("Casa", 3) => [Resultado: "asa"]
UCase	Converter os caracteres de uma string em maiúscula (caixa alta) Sintaxe: UCase(String)	UCase("Casa") => [Resultado: "CASA"] UCase("casa") => [Resultado: "CASA"] UCase("cASA") => [Resultado: "CASA"]
LCase	Converte os caracteres de uma string em caixa baixa (minúscula) Sintaxe: LCase(String)	LCase("Casa") => [Resultado: "casa"] LCase("casa") => [Resultado: "casa"] LCase("cASA") => [Resultado: "casa"]





VBA

MICROSOFT EXCEL 2013 COM VBA

Apostila do Aluno

Exemplo de código usando funções de manipulação de strings:

1º) Crie a planilha abaixo:

	A	B	C	D	E	F
1						
2		Dados:	Ana Paula Guimarães, 20, Touro			
3						
4		Nome:				
5		Idade:				
6		Signo:				
7						
8					EXTRAIR DADOS	
9						

2º) Crie o módulo de código abaixo:

```

Sub dados()
    Dim dados, nome, idade, signo As String
    Dim posidade, separador As Integer

    dados = Cells(2, 3) 'Obtem a string a ser analizada

    separador = InStr(dados, ",") 'Obtem a posição da primeira vírgula
    posidade = InStr(dados, "20") 'Obtem a posição da escrita da idade

    nome = Left(dados, separador - 1) 'Extrai o nome da pessoa
    idade = Mid(dados, posidade, 2) 'Extrai a idade da pessoa
    signo = Right(dados, 5) 'Extrai o signo da pessoa

    Range("C4") = UCase(nome) 'Exibe o nome (caixa alta)
    Range("C5") = UCase(idade & " anos") 'Exibe a idade (caixa alta)
    Range("C6") = UCase(signo) 'Exibe o signo (caixa alta)

End Sub

```

3º) Insira o código no botão de comando, execute o código e obtenha o resultado abaixo:

	A	B	C	D	E	F
1						
2		Dados:	Ana Paula Guimarães, 20, Touro			
3						
4		Nome:	ANA PAULA GUIMARÃES			
5		Idade:	20 ANOS			
6		Signo:	TOURO			
7						
8					EXTRAIR DADOS	
9						

2.5. Manipulando Data e Hora

Data e Hora são informações relativas a tempo que o VBA permite manipular como dados através de funções, essas informações podem também estar relacionadas ao tempo real gerado pelo relógio do seu computador.





Funções de Data e Hora

Abaixo são exibidas algumas das principais funções de manipulação de data e hora no VBA:

Função	Descrição	Exemplo
Now	Retorna a Data e a Hora atuais do sistema. <u>Sintaxe:</u> Now	Now [Resultado: 13/05/2010 23:14:35]
Date	Retorna a Data atual do sistema <u>Sintaxe:</u> Date	Date [Resultado: 13/05/2010]
Time	Retorna a Hora atual do sistema <u>Sintaxe:</u> Time	Time [Resultado: 23:14:35]
Day	Retorna um valor de 1 a 31 em relação ao dia da data. <u>Sintaxe:</u> Day(Data)	Day("12/05/2010") [Resultado: 12]
Month	Retorna um valor de 1 a 12 em relação ao dia da data. <u>Sintaxe:</u> Month(Data)	Day("12/05/2010") [Resultado: 5]
Year	Retorna o valor real do ano da data. <u>Sintaxe:</u> Year(Data)	Day("12/05/2010") [Resultado: 2010]
Weekday	Retorna o número inteiro entre 1(domingo) e 7 (sábado) que representa o dia da semana <u>Sintaxe:</u> Weekday(Data)	Weekday("12/05/2010") [Resultado: 7]
Hour	Retorna um valor de 0 a 23 representando a hora da expressão. <u>Sintaxe:</u> Hour(hora)	Weekday("23:32:45") [Resultado: 23]
Minute	Retorna um valor de 0 a 59 representando o minuto da expressão. <u>Sintaxe:</u> Minute(hora)	Minute("23:32:45") [Resultado: 32]
Second	Retorna um valor de 0 a 59 representando o segundo da expressão. <u>Sintaxe:</u> Second(hora)	Second("23:32:45") [Resultado: 45]

Exemplo de código usando funções de manipulação de Datas:

1º Crie a planilha abaixo:

	A	B	C
1			
2	Data de Nascimento:	13/9/1964	
3	Data Atual:		
4			
5	Ano Atual:		
6	Ano que Nasceu:		
7			
8	Idade:		





2º) Crie o módulo de código abaixo:

```
Sub CalcularIdade()
    Range("C3") = Date 'Exibe a data atual
    Range("C5") = Year(Range("C3")) 'Exibe o ano da data atual
    Range("C6") = Year(Range("C2")) 'Exibe o ano da data de nascimento
    Range("C8") = Range("C5") - Range("C6") & " anos" 'Exibe a idade
End Sub
```

3º) Insira o código no botão de comando, execute o código e obtenha o resultado abaixo:

	A	B	C
1			
2	Data de Nascimento:	13/9/1964	
3	Data Atual:	11/9/2010	
4			
5	Ano Atual:	2010	
6	Ano que Nasceu:	1964	
7			
8	Idade:	46 anos	
9			
10	CALCULANDO A IDADE		
11			

2.6. Fazendo Conversão de Tipo

A conversão de tipo é um recurso que a linguagem utiliza para converter texto em valor e vice-versa, existem funções próprias para se converter tipo no Excel, abaixo é exibido uma lista das principais funções de conversão:

Função	Descrição
CBool	Retorna um valor booleano. Sintaxe: CBool (expressão)
CByte	Retorna um valor byte. Sintaxe: CByte (expressão)
CCur	Retorna um valor monetário. Sintaxe: CCur (expressão)
CDate	Retorna um valor data. Sintaxe: CDate (expressão)
CDbl	Retorna um valor double. Sintaxe: CDbl (expressão)
CInt	Retorna um valor inteiro. Sintaxe: CInt (expressão)
CLng	Retorna um valor inteiro longo. Sintaxe: CLng (expressão)
CSng	Retorna um valor single. Sintaxe: CSng (expressão)
CStr	Retorna um valor string. Sintaxe: CStr (expressão)
CVar	Retorna um valor variant. Sintaxe: CVar (expressão)