Apêndice 2: Objetos

A2.1 Introdução

Este capítulo trata com alguns dos assuntos mais avançados do VBA. A primeira parte do capítulo introduz os objetos de planilha, declarações *With*, coleções, names, e o *pesquisador de objetos* também são discutidos.

A2.2 Objetos Worksheet: Uma Introdução

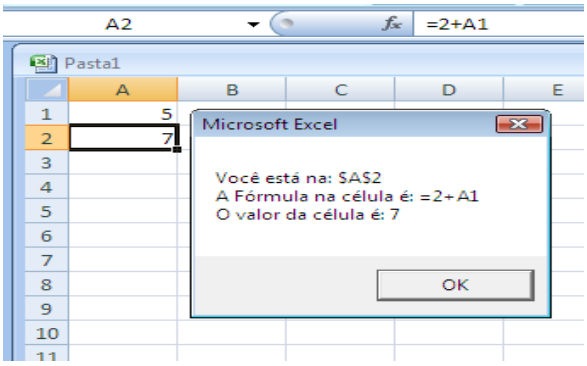
Objetos são os blocos de construção básicos do VBA. Embora você talvez não esteja ciente que está usando objetos, a maioria das coisas que você faz no VBA exige a manipulação de objetos. Podemos pensar num objeto como uma espécie de reservatório (container) com variáveis, funções, e subroutines dentro dele. Todos os componentes do Excel (pastas, planilhas, intervalos, e assim por diante) estão representados por um objeto na hierarquia de objeto do VBA (ver apêndice). Os *dados* do objeto são mantidos em variáveis especiais chamadas propriedades. Você pode acessar as propriedades usando o operador Dot (.). A macro seguinte usa a variável *objeto* ActiveCell do VBA e três de suas propriedades: Address, Formula, e Value:

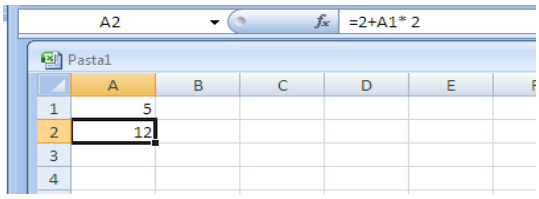
```
Sub ActivecellDemo()
Dim Temp
      MsgBox "Você está na: " & ActiveCell.Address _
            & Chr(13) & "A Fórmula na célula é: " _
            & ActiveCell.Formula & _
            Chr(13) & "O valor da célula é: " _
            & ActiveCell.Value
      ActiveCell.Formula = ActiveCell.Formula _
      & "* 2"
End Sub
```

Esta macro usa o *objeto* ActiveCell para fazer duas coisas:

1. Ela usa uma caixa de mensagem (message Box) para informar-lhe do conteúdo da célula A2. Este procedimento usa uma Message Box VBA e as propriedades ActiveCell.Formula e ActiveCell.Value.
2. Ela usa ActiveCell.Formula para mudar a fórmula na célula. Note a maneira como a fórmula é mudada: O último elemento na fórmula está multiplicado por 2. Assim, se você tem a fórmula `2 + A1` na célula A2, a nova fórmula será `2 + A1 * 2`. Se você rodar a macro novamente na mesma célula, a nova fórmula será `2 + A1 * 2 * 2`.

A macro produz os resultados seguintes:

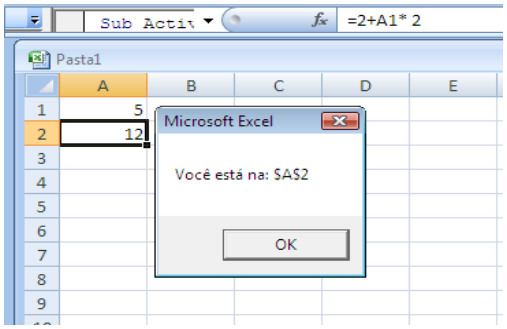


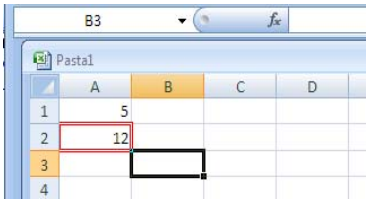


Métodos são funções contidas num objeto. Os métodos são usados para manipular o objeto. Como as propriedades, os métodos podem ser acessados usando o operador Dot (.). A linha divisória entre métodos e propriedades é algumas vezes muito confusa. A macro seguinte ativa os métodos BorderAround e Cells da ActiveCell:

```
Sub ActivecellDemo1()
     MsgBox "Você está na: " & ActiveCell.Address
     ActiveCell.BorderAround xlDouble, xlThick, 3
     ActiveCell.Cells(2, 2).Select
End Sub
```

E ela se parece com isto:





Note que a célula ativa na planilha moveu-se como resultado da última linha da macro.

A2.3 O Objeto Range

Os objetos têm tipos. Um dos tipos de objetos mais importantes no VBA é o Range. Uma célula de planilha e um intervalo de células são todos objetos do tipo Range. Por exemplo, um objeto variável VBA ActiveCell que encontramos na seção anterior é do tipo Range. Esta seção apresenta algumas das propriedades e métodos do objeto Range.

A2.3.1 Um intervalo como um Parâmetro para uma Função

Suponha que definimos o retorno sobre um ativo no período *t* como $r_t = \frac{\text{Preço}_t - \text{Preço}_{t-1}}{\text{Preço}_{t-1}}$, e a média do retorno de um ativo como $\bar{r} = \frac{1}{N}\sum_{t=1}^{N} r_t$. A *função* RetornoMedio aceita um intervalo coluna de preços de ativos como um parâmetro e calcula o *retorno médio do ativo*. Uma função auxiliar RetornoDoAtivo é usada para calcular *r* .

```
Function RetornoDoAtivo(P0 As Variant, P1 As _
        Variant) As Double
    RetornoDoAtivo = (P1 – P0) / P0
End Function
Function RetornoMedio(Rng As Range) As Double
    Dim NumeroDeLinhas As Integer
    Dim Precos As Variant
    Dim Temp As Double
    Dim i As Integer
    NumeroDeLinhas = Rng.Rows.Count
    Precos = Rng.Value
    Temp = 0
    For i = 2 To NumeroDeLinhas
        Temp = Temp + _
            RetornoDoAtivo (Precos(i – 1, 1), _
                Precos(i, 1))
    Next i
    RetornoMedio = Temp / (NumeroDeLinhas – 1)
End Function
```

Aqui está a função em ação:

	A	B	C
7	100.000	0,160	<--=RetornoMedio(A7:A12)
8	110.000	0,100	<--=RetornoMedio(A7:A9)
9	121.000	0,200	<--=RetornoMedio(A9:A12)
10	145.200		
11	174.240		
12	209.088		

Linhas de notas:

- `NumeroDeLinhas = Rng.Rows.Count`

O operador **Dot** é usado duas vezes. **Rng** é nosso objeto **Range**. **Rows** é a propriedade do objeto **Range**, assim **Rng.Rows** é um objeto do tipo **Collection** que representa todas as linhas no nosso intervalo. **Count** é uma propriedade do objeto do tipo **Collection** que armazena o número de membros na coleção, assim **Rng.Rows.Count** é uma variável que armazena o número de linhas no nosso intervalo.

- `Precos = Rng.Value`

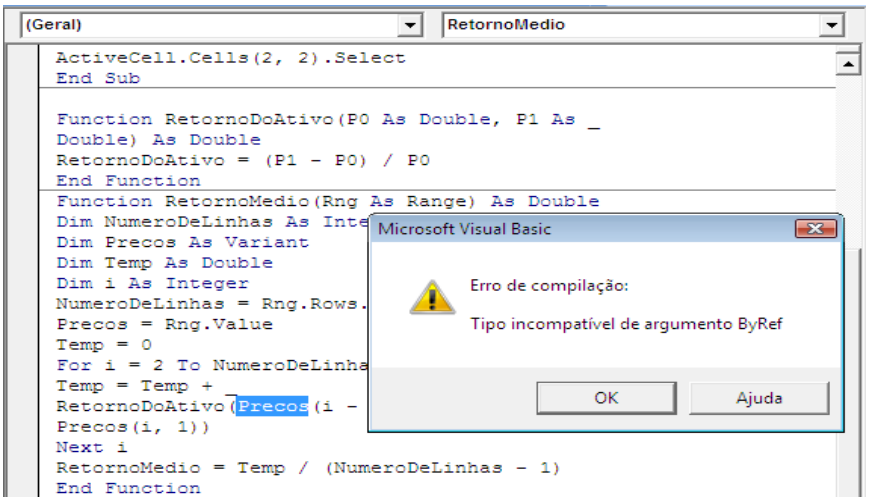
Value é uma propriedade do objeto **Range** contendo os valores de todas as células no intervalo. **Value** é do tipo **Variant**. Se o intervalo for de mais do que uma célula em tamanho, **Value** é um array bidimensional. O primeiro índice de **Value** é o índice de linha começando em 1, e o segundo índice é o índice de coluna começando em 1.

A2.3.2 Tipos de Considerações com Variants e Objetos como Parâmetros

Algumas vezes as variáveis **Variant** são digitadas depois da hora para o VBA usar o tipo de informação. O problema está ilustrado pela função **RetornoDoAtivo** na subseção anterior. Uma declaração melhor para a função **RetornoDoAtivo** teria sido

```
Function RetornoDoAtivo(P0 As Double, P1 As _
      Double) As Double
```

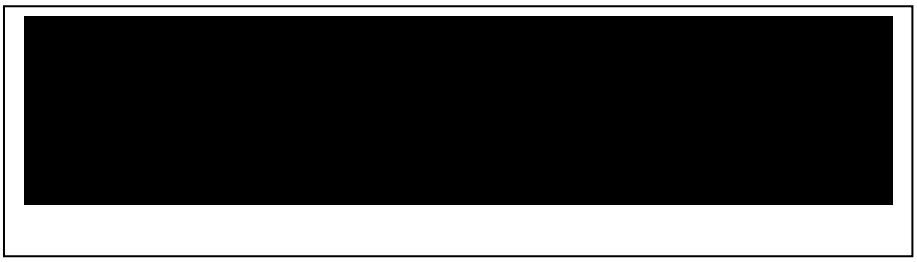
Infelizmente, quando tentamos usar a função, o VBA verifica a exatidão do nosso programa. Quando a verificação é realizada (logo antes da função ser usada), as variáveis **Precos** não é ainda um array de doubles. Como conseqüência o erro seguinte é emitido



A2.3.3 A Propriedade Item do Objeto Range

A propriedade **Item** do objeto **Range** permite-nos endereçar uma célula específica na planilha da qual o intervalo faz parte. Embora a **Item** seja referida como uma propriedade na documentação, ela realmente comporta-se duplamente como uma propriedade e um método. A função seguinte retorna a fórmula na célula três linhas abaixo e duas colunas para a esquerda do seu argumento:

```
Function DuasAEsquerdaTresParaBaixo(Rng As Range) As String
    DuasAEsquerdaTresParaBaixo = Rng.Item(4, 3).Formula
End Function
```



Note que Rng.Item(1, 1) é a célula no canto esquerdo superior do Rng.

A2.3.4 A Propriedade Range

A propriedade Range é aquela maneira de se acessar um intervalo numa planilha. Range é uma propriedade de muitos objetos do Excel.
Quando usada de maneira apropriada, como na próxima macro, Range é uma maneira curta de escrever ActiveSheet.Range.

```
Sub RangeDemo()
     Range("A1").Formula = 23
End Sub
```

Como esperado uma macro configurará a fórmula na célula "A1" da planilha ativa para 23. A próxima macro configura a fórmula
De cada célula no intervalo "A1:B2" da planilha ativa para 23.

```
Sub RangeDemo1()
     Range("A1:B2").Formula = 23
End Sub
```

A próxima macro configura a fórmula de cada célula no intervalo "A1:B2" da planilha ativa para 23, usando a alternativa chamada seqüência de Range. O primeiro argumento é a célula no canto esquerdo superior do intervalo. O segundo é a célula no canto direito inferior do intervalo.

```
Sub RangeDemo2()
     Range("A1", "B2").Formula = 23
End Sub
```

Range é também uma propriedade do objeto Range. O intervalo retornado pelo Range quando usado este modo é relativo ao objeto Range. A próxima macro configura a fórmula da célula "C2" da planilha ativa para 23.

```
Sub RangeDemo3()
     Range("B1").Range("B2").Formula = 23
End Sub
```

Observação: Range("B1") retorna o intervalo (ou célula) "B1" da planilha ativa. Range("B1").Range("B2") retorna a célula "B2" do intervalo que tem "B1" como o canto esquerdo superior. Em termos de planilha, Range("B1").Range("B2") retorna a célula "C2."

A próxima macro configura a fórmula de cada célula no intervalo "C2:D3" da planilha ativa para 23. A macro usa a célula "C2" como ponto de partida, tão bem quanto a alternativa chamada seqüência de Range.

```
Sub RangeDemo4()
     Range("C2").Range("A1", "B2").Formula = 23
End Sub
```

A2.4 A Declaração With

A declaração With permite você realizar uma série de declarações sobre um objeto especificado sem falar novamente o óbvio (o nome do objeto e seu pedigree, que pode ser muito longo). Se você tiver mais de uma propriedade para mudar ou mais de um método para usar para um único objeto, use a declaração With. A declaração With faz seus procedimentos statements rodarem mais rápidos e ajuda você evitar digitação repetitivas. A macro seguinte, um pouco artificial, configura algumas propriedades da fonte da célula no canto esquerdo superior da região atual da célula ativa. A fonte está configurada como sendo Arial, negrito, vermelha, e 15 pontos de tamanho.

```
Sub DemoSemWith ()
     ActiveCell.CurrentRegion.Range("A1").Font _
          .Bold = True
     ActiveCell.CurrentRegion.Range("A1").Font _
          .ColorIndex = 3
     ActiveCell.CurrentRegion.Range("A1").Font _
          .Name = "Arial"
     ActiveCell.CurrentRegion.Range("A1").Font _
          .Size = 15
End Sub
```

O que segue é masma macro usando a declaração With.

```
Sub DemoComWhit ()
     With ActiveCell.CurrentRegion.Range("A1").Font
          .Bold = True
          .ColorIndex = 3
          .Name = "Arial"
          .Size = 15
     End With
End Sub
```

Note o operador Dot(.) antes das propriedades na declaração With.

A2.5 Coleções

Uma Coleção é um conjunto ordenado de itens que podem ser referidos a elescomo uma unidade. Uma Coleção de objetos fornece uma maneira conveniente de se referir a um grupo relacionado de itens como um objeto único. Os ítens, ou membros, numa Coleção, precisam estar somente relacionados pelo fato que eles existem na Coleção. Os membros de uma Coleção não têm de compartilhar o mesmo tipo de dado.

Uma Coleção pode ser criada da mesma maneira que os outros objetos são criados. Os membros podem ser adicionados usando o método Add e removidos usando o método Remove. Membros específicos podem ser referidos usando-se um índice inteiro. O número de membros atualmente numa Coleção é disponível via o método Count. Nosso uso das Coleções está restrito ao uso do arsenal (bem numeroso) de Coleções que são partes do Modelo Objeto Excel, como a Rows Collection mencionada na seção anterior.

A2.5.1 A Declaração *For Each* em Uso com *Arrays* e *Coleções*

A declaração For Each é uma variação do laço For único para VBA. Esta declaração vem com dois sabores distintos.
A primeira variação usa a declaração para laçar um array como demonstrado na função seguinte:

```
Function SomaParaCada (Rng As Range) As Double
     Dim Elemento As Variant
     Dim Soma As Double
     Soma = 0
     For Each Elemento In Rng.Value
          Soma = Soma + Elemento
     Next Elemento
     ParaCadaSoma = Soma
End Function
```

	A	B	C	D	E
23	**SomaParaCada**				
24					
25	1	4	7	6	<--=SomaParaCada(A25:A27)
26	2	5	8	12	<--=SomaParaCada(A25:C25)
27	3	6	9	45	<--=SomaParaCada(A25:C27)

Pontos a serem notados:

- O membro atual do array está disponível às declarações dentro do corpo do laço através de variável laço (Elemento na função).
- A variável laço tem que ser do tipo Variant independente do tipo de array.
- Mudar o Elemento não será refletida no array atual (uma situação de leitura somente).
- Você não precisa saber o número de dimensões ou intervalo de índices para laçar o array. A função anterior funciona nos intervalos colunas (como na célula E25), intervalos linhas (como na célula E26), e intervalos retangulares (como na célula E27) da mesma maneira

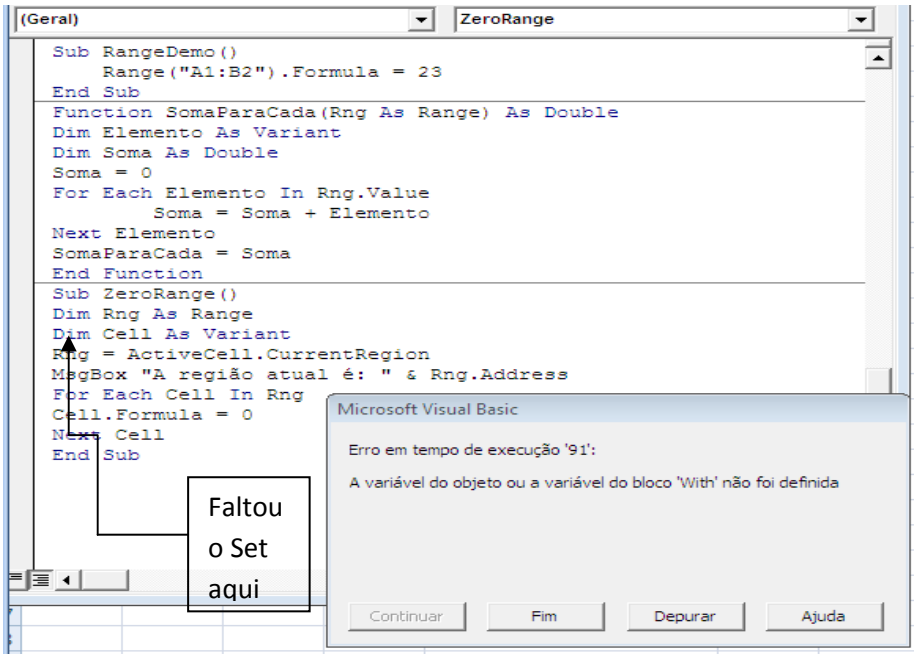
A2.5.2 A declaração *For Each* em Uso com *Coleções*

A segunda versão da declaração For Each laça as Coleções:

```
  Sub ZeroRange()
      Dim Rng As Range
      Dim Cell As Variant
      Set Rng = ActiveCell.CurrentRegion
      MsgBox "A região atual é: " & Rng. Address
      For Each Cell In Rng
          Cell.Formula = 0
      Next Cell
End Sub
```

Pontos a serem notados:

- Elemento é uma variável usada para interagir por *todos* os membros da coleção.
- Elemento tem que ser um dos tipos seguintes: Variant, Generic Object, ou o tipo específico de elemento que a Coleção é feita.
- Elementose refere ao membro atual da Coleção, e mudar o Element será refletido na Coleção.
- O uso da propriedade CurrentRegion do objeto Range. CurrentRegion é um objeto Range que representa a região atual. A região atual é um intervalo limitado por uma combinação de linhas e colunas em branco.
- O uso de declaração Set. Uma explicação completa está além do escopo deste livro. Para os nossos propósitos apenas assinalar a palavra Set para todas as tarefas do objeto. Se você não fizer, o que segue acontecerá:



A2.5.3 A Coleção Workbooks e o Objeto Workbook

Todas as pastas abertas atualmente são representadas por um objeto **Workbook** na **Coleção Workbooks**. A macro seguinte lista todas as pastas abertas numa coluna de células começando na célula ativa:

```
Sub ListarPastasAbertas()
    Dim i As Integer
    Dim Elemento As Workbook
          ActiveCell.Item(2, 1).Formula = _
          "Lista das Pastas abertas"
          ActiveCell.Item(4, 1).Formula = "Criada em:" _
          & FormatDateTime(Date, vbLongDate) & " Às: " _
          & FormatDateTime(Time, vbLongTime)
          With ActiveCell.Item(2, 1).Font
                           .Bold = True
                           .Name = "Arial"
                           .Size = 12
          End With
          i = 5
          For Each Elemento In Workbooks
          ActiveCell.Item(i, 1).Formula = _
          Elemento.FullName
          i = i + 1
          Next Elemento
 End Sub
```

Esta é a planilha após a macro ter sido rodada:

	A
59	**Lista das Pastas abertas**
60	
61	Criada em:quinta-feira, 31 de julho de 2008 Às: 14:08:37
62	Pasta1
63	D:\Contabilidade\Administração Financeira\Apostila\Cap02\Planilha de Custo de Produtos Vendidos1.xls
64	D:\Excel\Macros\Exemplo Macro Finanças em Excel #01.xls

Linhas de notas:

```
ActiveCell.Item(4, 1).Formula = _
    "Criada em:" & FormatDateTime(Date, _
    vbLongDate) & " Às: " & _
    FormatDateTime(Time, vbLongTime)
```

- A função **Date** retorna a data atual do sistema.
- A função **Time** retorna o tempo atual do sistema.
- A função **FormatDateTime** formata as variáveis **Date** e **Time** para mostrar.

```
For Each Elemento In Workbooks
  ActiveCell.Item(i, 1).Formula = _
        Elemento.FullName
  i = i + 1
Next Elemento
```

A declaração **For** laça por completo a **Coleção Workbooks**. Em cada interação o **Element** é um dos objetos da **Workbook** na **Coleção**. **FullName** é a propriedade do objeto **Workbook** contendo o caminho completo da workbook.

A próxima macro adiciona uma pasta à **Coleção Workbooks**:

```
Sub AdicionaPasta()
    Dim Pasta As Workbook
    Dim PastaAntiga As Workbook
    Set PastaAntiga = ActiveWorkbook
```

```
        Set Pasta = Workbooks.Add
        PastaAntiga.Activate
        MsgBox Pasta.FullName & " Adicionada"
    End Sub
```

Adicionando uma pasta à Coleção Workbooks torna a pasta adicionada mais recentemente a pasta ativa. Se quisermos permanecer onde estamos, precisamos reativar a velha pasta.

A2.5.4 A Coleção Worksheets e o Objeto Worksheet

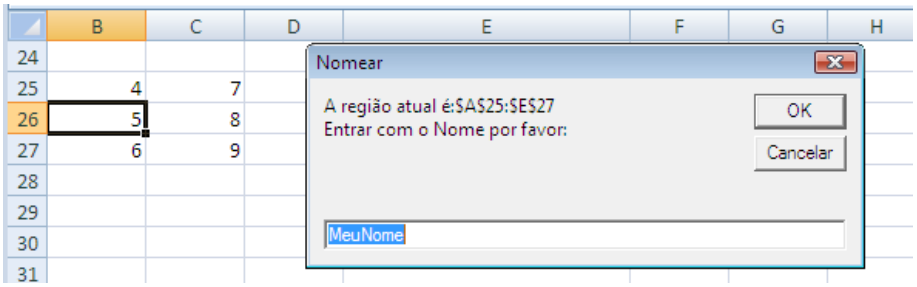
Todas as planilhas numa pasta são objetos Worksheet na Coleção Worksheets a qual é uma propriedade do objeto Workbook. Podemos usar a Coleção Worksheets sem um objeto como uma forma curta para ActiveWorkbook.Worksheets.

A2.6 Nomes

A2.6.1 Nomeando um Intervalo Usando uma Macro

A macro seguinte nomeia a região atual:

```
Sub NomeDaRegiaoAtual()
        Dim Rng As Range
        Dim MeuNome As String
        Set Rng = ActiveCell.CurrentRegion
        MeuNome = InputBox("A região atual é: " & Rng.Address _
& Chr(13) & "Entrar com o _& Nome por favor: ", "Nomear", "MeuNome")
        Names.Add Name:=MeuNome, RefersTo:="=" _
& Rng.Address
End Sub
```



A linha interessante numa macro é

```
    Names.Add Name :=MeuNome, RefersTo:="=" & _
           Rng.Address
```

Names é uma Coleção de todos os nomes na pasta ativa. Add é um método da Coleçã Names usado para adicionar membros à coleção. Usamos somente os primeiros dois parâmetros do método. O primeiro parâmetro MeuNome é o nome a adicionar à Coleção Names. O segundo parâmetro é uma string contendo o endereço que o name adicionado se refere, precedido pelo "=".

A2.6.2 Procurando por Nomes Definidos

A função seguinte procura por um name definido na pasta atual. A função retorna o valor Boolean "True" se o name é definido, e "False" se ele não faz parte da Coleção Names.

```
    Function ONomeE(Name As String) As Boolean
        Dim Elemento As Variant
        Dim Flag As Boolean
        Flag = False
        For Each Elemento In Names
            If Name = Elemento.Name Then
                Flag = True
                Exit For
```

```
            End If
        Next Elemento
        ONomeE = Flag
End Function
```

	B	C	D
37	**O Nome É em Ação**		
38			
39	**Nome**	**O Nome É**	
40	Beny	FALSO	<--=ONomeE(B40)
41	**MeuNome**	VERDADEIRO	<--=ONomeE(B41)
42	meunome	FALSO	<--=ONomeE(B42)
43	**MEUNOME**	FALSO	<--=ONomeE(B43)

Note que

- Os Names são case sensitive.
- O comando Exit For pode ser usado com o For Each.

A2.6.3 Referindo-se a um Intervalo Nomeado

A função seguinte calcula o *Retorno Médio* de um intervalo nomeado de preços de ativos:

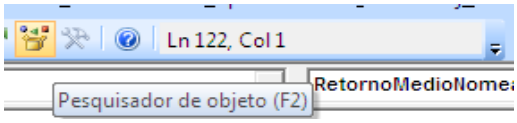
```
Function RetornoMedioNomeado(RangeName As String) _
As Double
Dim Rng As Range
Dim Precos As Variant
Dim Temp As Double
Dim i As Integer
Set Rng = Range(RangeName)
Precos = Rng.Value
Temp = 0
For i = 2 To UBound(Precos, 1)
Temp = Temp + (Precos(i, 1) - Precos(i - 1, 1)) / _
Precos(i - 1, 1)
Next i
RetornoMedioNomeado = Temp / (UBound _
(Precos, 1) - 1)
End Function
```

Note o novo modo de usar a propriedade Range introduzida numa:

```
Set Rng = Range(RangeName)
```

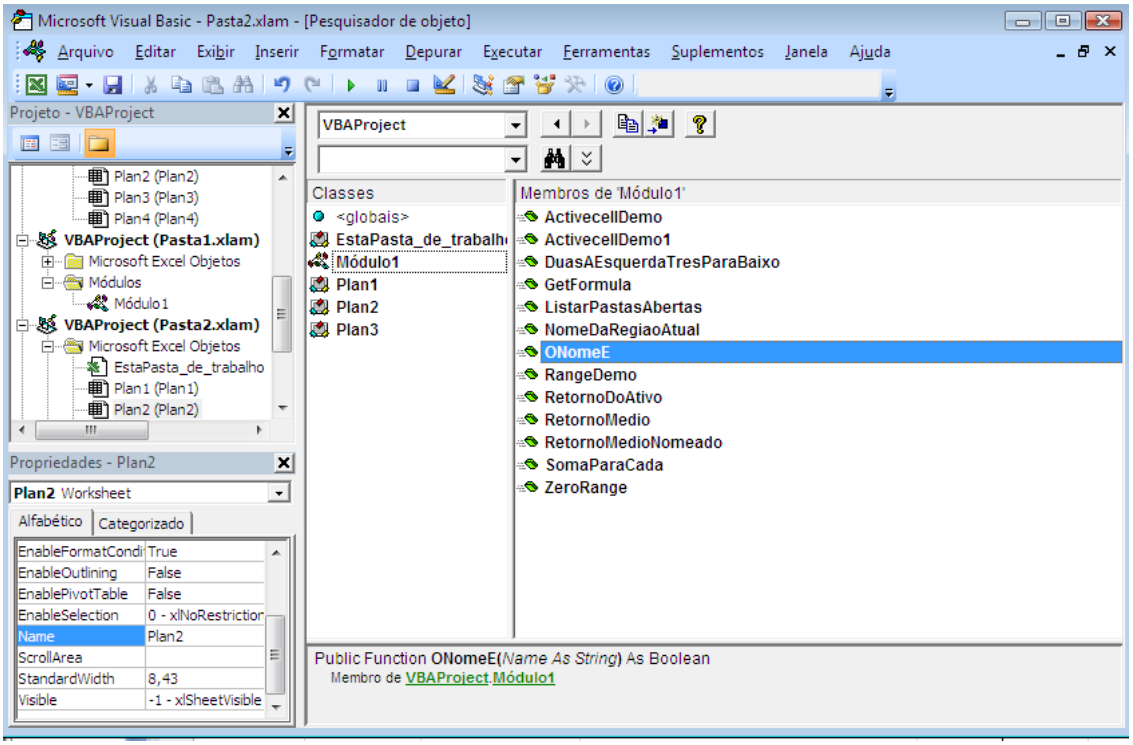
A2.7 Usando o Pesquisador de objetos (Object Browser)

O Pesquisador de objetos é uma maneira conveniente de se aprender sobre os diferentes objetos que estão disponíveis ao uso no VBA. O
pesquisador de objetos pode ser ativado pressionando seu botão na barra de ferramentas do VBA



Ou do menu (Exibir|Pesquisador de objetos...), ou ainda com a tecla de atalho F2.

A janela pesquisador de objetos é constituída de dois painéis, uma caixa de seleção, e uns poucos botões. Na caixa de seleção você pode escolher um conjunto mestre de objetos para observar (dois conjuntos mestres, VBA e Excel, estão sempre disponíveis). Selecione o conjunto mestre VBA para observar objetos que são internos quando se trabalha com o VBA. O conjunto mestre Excel lida com objetos que são específicos ao Excel. (Ele está em toda parte de um plano master Microsoft, sem intenção de trocadilhos, onde o VBA é um centro de muitas aplicações, não apenas o Excel.)



Uma vez selecionado o conjunto de objetos mestres, os outros dois painéis mudarão seus conteúdos para refletir o conjunto mestre selecionado. O painel da esquerda é uma lista de categorias (classificada em ordem alfabética ascendente). O painel da direita é uma lista de **Propriedades** ou **Métodos** atribuíveis ao objeto selecionado na painel da esquerda. Clicando no botão ponto de interrogação faz a tela ajuda apropriada ser mostrada. Clicando um nome no painel da direita (**ONomeE** na tela) mostrará a informação acerca dela no fundo da tela.

Exercícios

1. Suponha que você tenha uma planilha com uma série de números e fórmulas:

Suponha que você queira transformá-la nesta que segue:

Escrever uma macro que realize este propósito. Sua macro (aproximadamente será baseada na macro ActiveCellDemo da seção A2.2) deverá:

- Coloque um grupo de parênteses e multiplique o conteúdo das células por 100.
- Mova para baixo uma célulal (ver ActiveCellDemo1, seção A2.2).
- Pergunte se você quer repetir o processo. (Se "yes," ela deverá fazê-lo; se "no," a macro encerrará).

Nota: Os parênteses têm que vir após o "=." A função Right deverá ser usada para esta operação.

Você poderá querer se referir à seção A2.3 para mais informação on the função MsgBox and the values it returns.

2. Reescreva a macro do exercício 1 de modo que it deals corretamente with the end of the series. Um tratamento possível é não perguntar para repetir o processo quando se tratar com a última célula na série.

Para esta macro poderia ser útil pensar na última célula na série como a célula que cumpre o critério Cell.Item(2,1).Formula="" (ver seção A2.3).

3. Escreva uma macro que multiplique todas as células na região atual por 2.

4. Reescreva a macro do exercício 3 de modo que sua ação seja dependente dos conteúdos das células

- Se os conteúdos da célula forem fórmulas ele será trocado pela mesma fórmula multiplicado por 2.

- Se os conteúdos da célula forem números ele será trocado por um número igual to the número antigo multiplicado por 2.

- Sobre todas as outras células na região atual nada será feito.

Nota: Para os propósitos deste exercício uma fórmula é qualquer coisa começando com "=," e um número é qualquer coisa começando com os caracteres "0" a "9."

5. Reescreva a macro do exercício 4 de modo que ela use um outro método (o correto) para detetar a existência de uma fórmula na célula. Observe as diferentes propriedades do objeto Range no arquivo ajuda.

6. As anotações para as fórmulas de planilha neste livro foram feitas com uma macro. Por exemplo rodando uma macro Nesta planilha, "B16" sendo a célula ativa:

Produz a planilha seguinte:

Note que a coluna mudou de largura. Escreva uma macro para realizar a anotação. Se a célula imediatamente à direita da célula ativa não estiver vazia, uma macro deverá sobreescrevê-la somente após receber a confirmação do usuário.

7. O objeto Selection representa a seleção atual na planilha. Selection é geralmente, e para nossos propósitos sempre, um objeto Range. Reescreva a macro ido exercício 6 de modo que ele funcione num intervalo selecionado.

Note o seguinte:

- Se o intervalo selecionado é uma célula única ative a macro do exercício 6.

- Se o intervalo selecionado é um intervalo linha as anotações deverão ir para baixo do intervalo selecionado.

- Se o intervalo selecionado é mais do que uma coluna ou uma linha, a macro deverá abortar com uma mensagem apropriada.

8. As funções Array (Matriz) são funções que retornam com mais do que um valor. Por exemplo, a função de planilha TRANSPOR[1] retorna seu argumento girando por 90 graus, como demonstra a seguinte planilha:

As chaves não foram digitadas, mas foram adicionadas pelo Excel para indicar uma fórmula array. A macro seguinte criou a planilha anterior.

[1] Esta função embutida do Excel encontra-se na Categoria **Procura e referência**

```
Sub TranspoeMe()
     Range("E3:E6").FormulaArray = _
          "=Transpose(A3:D3)"
End Sub
```

A próxima macro é uma versão mais complicada que poderá tratar com qualquer tamanho ou lugar no intervalo de linha.

```
Sub TranspoeMeTambem()
     Dim L As Integer, C As Integer
     C = Selection.Columns.Count
     L = Selection.Rows.Count
     If C = 1 Then                                   'É uma Coluna
           MsgBox "Eu não faço Colunas"
     ElseIf L = 1 Then                              'É uma Linha
     Selection.Cells(1, C + 1).Range("A1:A" _
     & C). FormulaArray = "=Transpose(" & _
           Selection.Address(False, False) & ")"
           Else                                        ' O que ela é?
                 MsgBox "O que ela é?"
           End If
       End Sub
```

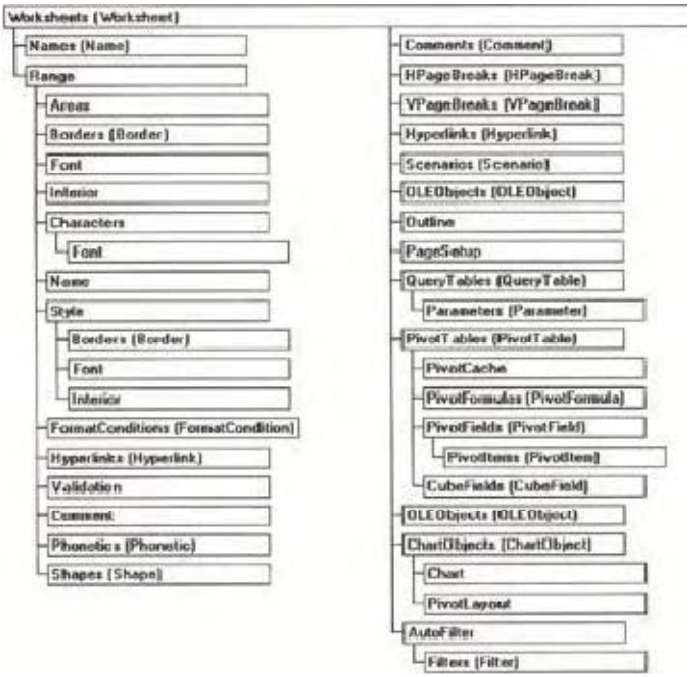
Reescreva o TranspoeMeTambem de modo que ele possa tratar com intervalos de colunas tão bem quanto intervalos de linhas.

9. Reescreva o TranspoeMeTambem do exercício 8 de modo que ele possa tratar com todos os intervalos a primeira vista.

Apêndice: Hierarquia do Objeto Excel

Objetos do Microsoft Excel

Microsoft Excel Objects (Worksheet)


Worksheets (Worksheet)
Names (Name)
Range
Areas
Borders (Border)
Font
Interior
Characters
Font
Name
Style
Borders (Border)
Font
Interior
FormatConditions (FormatCondition)
Hyperlinks (Hyperlink)
Validation
Comment
Phonetics (Phonetic)
Shapes (Shape)
Comments (Comment)
HPageBreaks (HPageBreak)
VPageBreaks (VPageBreak)
Hyperlinks (Hyperlink)
Scenarios (Scenario)
OLEObjects (OLEObject)
Outline
PageSetup
QueryTables (QueryTable)
Parameters (Parameter)
PivotTables (PivotTable)
PivotCache
PivotFormulas (PivotFormula)
PivotFields (PivotField)
PivotItems (PivotItem)
CubeFields (CubeField)
OLEObjects (OLEObject)
ChartObjects (ChartObject)
Chart
PivotLayout
AutoFilter
Filters (Filter)