



Aula 02

Minicurso de VBA
Luiz Alberto, professor

Fundamentos da Programação em VBA

Referenciando Linhas e Colunas

Rows("12:14").RowHeight=30

Linhas 12,13 e 14 terão altura 30

Range("16:16", "18:18, 20:20").RowHeight=30

Linhas 16, 18 e 20 serão alteradas

Columns("E:F").ColumnWidth=10

Colunas E e F terão largura 10

Range("H:H,J:J").ColumnWidth=10

Colunas H e J serão modificadas

Range(Columns(1), Columns(3)).ColumnWidth=5

Colunas A, B e C terão largura 5

Cells.Columns.AutoFit

Todas as colunas serão autoajustadas

4 Métodos para Encontrar a Última Linha

1. Utilize a propriedade **End** do objeto **Range**

Exemplo: `Range("K6").Value=Range("A4").End(xlDow).Row`

2. Utilize a propriedade **CurrentRegion** do objeto **Range**

Exemplo:

`Range("K10").Value=Range("A4").CurrentRegion.Rows.Count`

3. Utilize o método **SpecialCells** do objeto **Cells**

Exemplo:

`Range("K11").Value=Cells.SpecialCells(xlCellTypeLastCell).Row`

4. Utilize a propriedade **UsedRange** do objeto **Worksheet**

Exemplo:

`Range("K12").Value=Application.ActiveSheet.UsedRange.Rows.Count`



Variáveis e Tipos de Dados

- # Variáveis e porque utilizá-las
- # Tipos de dados e boas práticas
- # Trabalhando com variáveis
- # Escopo das variáveis

O Que São Variáveis?

Variáveis são **nomes** para **posições** de memória

Valores são atribuídos com o sinal de **=**
Exemplo: `titulo = Range("A1").Value`

Por que utilizar variáveis ?

Variáveis tornam o código mais **legível**

Variáveis **simplificam** a **manutenção** e permite a construção de códigos mais **complexos**.

Tipos de Dados para Variáveis

Data Type	Memory used	Range
Byte	1 byte	0 to 255
Integer	2 bytes	-32,768 to 32,768
Long	4 bytes	-2,147,483,648 to 2,147,483,647
Boolean	2 bytes	True / False
Double	8 bytes	Very large negative to positive range with high precision (also used for %)
String	1 byte per char	Depends on length
Object	4 bytes	Any object
Date	8 bytes	01,1,0100 to 12,31,9999
Currency	8 bytes	Very large negative to positive range up to 4 decimal places
Variant	16 bytes (more with characters)	Any value – can also hold values such as “Empty”, “Nothing” and “Null”

Declarando Variáveis, Vetores e Constantes

1

Declarando variáveis definindo o **tipo mais apropriado resulta em:**

1. Um código que executará mais rapidamente
2. Um código que estará menos sujeito a erros

Exemplo: Dim texto As String

Dim UltimaLinha As Long, PrimeiraLinha As Long

2

Um grupo de variáveis que tem um nome pode ser declarado como um **vetor.**

Exemplo: Dim meses(1 To 12) As String

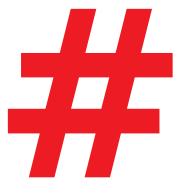
3

Se for necessário referenciar uma variável que nunca muda, use **constante.**

Exemplo: Const PI As Double = 3.1415

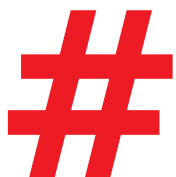
Dica: Utilize **Option Explicit** para minimizar os erros.

Utilizando Variáveis Objeto



Variáveis podem armazenar objetos. Objetos comuns são:

- | | | |
|--------------|---------|--------------|
| 1. Workbook | Dim wkb | As Workbook |
| 2. Worksheet | Dim wks | As Worksheet |
| 3. Range | Dim rng | As Range |



Para atribuir valores a variáveis objeto, é necessário utilizar o comando **SET.**

Exemplo: SET wkb = Workbooks.Add

Escopo de Variáveis



Procedimento ou função: variáveis que existem somente enquanto o procedimento é executado. A declaração (DIM) é feita no procedimento ou função.

```
Sub Procedimento()  
    Dim UltimaLinha As Long  
    ' - código  
End
```



Módulo: variáveis que são acessíveis por todos os procedimentos ou funções do módulo. A variável é declarada fora de qualquer procedimento ou função. Tipicamente abaixo do **Option Explicit**.

```
Option Explicit  
Dim UltimaLinha As Long  
Sub Procedimento()  
    ' - código  
End
```



Todos os módulos e procedimentos: variáveis que são acessíveis por TODOS os procedimentos e por TODOS os módulos. Utilize a palavra **Public** para declarar estas variáveis. Declaradas em qualquer módulo antes do primeiro procedimento.

```
Option Explicit  
Public Dim UltimaLinha As Long  
Sub Procedimento()  
    ' - código  
End
```

With..End With & Comandos de Decisão

- # With..End With simplifica a codificação
- # Tomando decisões com IF e Select Case

With..End

Os benefícios do With..End With são:

1. Escrita mais rápida do código
2. Código mais fácil de dar manutenção
3. Execução mais rápida do código

Sem With..End With

```
Set faixa = Range("A10", "A" & Cells(Rows.Count, 1).End(xlUp).Row)
faixa.Font.Name = "Arial"
faixa.Font.Size = 12
faixa.Font.Bold = True
```

Com..With..End With

```
Set faixa = Range("A10", "A" & Cells(Rows.Count, 1).End(xlUp).Row)
With faixa.Font
    .Name = "Arial"
    .Size = 12
    .Bold = True
End With
```

IF..Then

Tomando decisões no código

IF em 1 linha

```
If Range("B3").Value <> "" Then Range("C3").Value = Range("B3").Value
```

IF em 2 linhas

```
If Range("B3").Value <> "" Then  
    Range("C3").Value = Range("B3").Value  
    Range("D3").Value = Range("B3").Value  
+ 1  
End If
```

If..Then...Elseif..Else

```
Dim valor As Double  
valor = Range("A3").Value  
If valor < 100 Then  
    taxa = 2.50  
Elseif valor < 200 Then  
    taxa = 5.00  
Else  
    taxa = 7.50  
End If
```

Os operadores **AND** and **OR** são permitidos.

Select Case

Uma alternativa para muitos IFs

Tem-se uma **visão** melhor, mais **fácil** de dar manutenção e a execução mais **rápida**.

```
Select Case Range("B3").Value
  Case 1 To 200
    Range("C3").Value = "Bom"
  Case 0
    Range("C3").Value = ""
  Case is > 200
    Range("C3").Value = "Excelente"
  Case Else
    Range("C3").Value = "Ruim"
End If
```



Looping in VBA

controlando fluxo de execução

For..Next

Do Until / Do While Loop

For..Next

Laço de repetição simples, baseado em um contador.
Exit For encerra o laço.

```
Sub Laco_Simples()  
    Dim i As Long  
    Dim valor As Double  
    Dim ultimaLinha, contaLinha As Long  
    contaLinha = ActiveSheet.UsedRange.Rows.Count  
    ultimaLinha = ActiveSheet.UsedRange.Cells(contaLinha,1).Row  
    For i = 4 to ultimaLinha  
        valor = Range("F" & i).Value  
        If valor > 400 Then Range("F"& i).Value = value + 10  
        If valor < 0 Then Exit For  
    Next i  
End Sub
```

Do..Until / Do..While Loop

Do..While executa enquanto a condições especificada é verdadeira.

Do..Until executa o laço até que a condição seja verdadeira.

```
Sub Laco_Simples_com_Until()  
    Dim celulaInicial As Long  
    celulaInicial = 8  
    Do Until ActiveSheet.Range("A", celulaInicial).Value = ""  
        Range("B", &celulaInicial).Value = Range("A" & celulaInicial).Value  
        + 10  
        celulaInicial = celulaInicial + 1  
    Loop  
End Sub
```

```
Sub Laco_Simples_com_While()  
    Dim celulaInicial As Long  
    celulaInicial = 8  
    Do Until ActiveSheet.Range("A", celulaInicial).Value <> ""  
        Range("C", &celulaInicial).Value = Range("A" & celulaInicial).Value + 10  
        celulaInicial = celulaInicial + 1  
    Loop  
End Sub
```


Projeto#5

As instruções para a realização do Projeto#5 estão no arquivo Projeto_5.docx

Projeto#6

As instruções para a realização do Projeto#6 estão no arquivo Projeto_6.docx