

## Por que os Laços?

O propósito de um laço é conseguir que o Excel repita um fragmento de código um certo número de vezes. Quantas vezes o código será repetido pode ser especificada como um número fixo (p.ex. faça isto 10 vezes), ou como uma variável (p.ex. faça isto tantas vezes enquanto existirem linhas de dados).

Os laços podem ser construídos de muitas maneiras diferentes para se adaptar às diferentes circunstâncias. Frequentemente o mesmo resultado pode ser obtido de diferentes maneiras para se adaptar às suas preferências pessoais. Estes exercícios demonstram uma seleção de maneiras diferentes de se usar laços.

Existem duas espécies básicas de laços, ambas estão demonstradas aqui: os laços **Do...Loop** e **For...Next**. O código a ser repetido é colocado entre as palavras chaves.

Abra a pasta **Lacos no VBA.xls** e observe as quatro planilhas. Cada uma contém duas colunas de números (colunas A e B). A exigência é calcular uma média para os numerosos números em cada linha usando uma macro VBA.

Agora abra o Visual Basic Editor (**Alt+F11**) e dê uma olhada no código no *Módulo1*. Você verá várias macros diferentes. Nos exercícios seguintes, rode primeiro a macro e daí então leia o código e compreenda como ele fez isto e aquilo.

Você pode rodar as macros ou do Visual Basic Editor colocando o seu cursor na macro e pressionando a tecla **F5** key, ou do Excel abrindo a caixa de diálogo de Macros (**ALT+F8**) escolhendo a macro a ser rodada e clicando em **Executar**. É melhor rodar estas macros do Excel assim você pode vê-las funcionando.

### Exercício 1: Do... Loop Until...

O objetivo desta macro é rodar a coluna C enquanto for necessário colocando um cálculo em cada célula enquanto for necessário.

Na **Plan1** selecione célula **C2** e rode a macro **Laco1**.

Aqui está o código:

```
Sub Laco1()  
  
' Este laço roda até não existir nada na próxima coluna  
  
Do  
  
ActiveCell.FormulaR1C1 = "=Average(RC[-1],RC[-2])"  
  
ActiveCell.Offset(1, 0).Select  
  
Loop Until IsEmpty(ActiveCell.Offset(0, 1))  
  
End Sub
```

Esta macro coloca uma fórmula na active cell, e move-se para próxima célula abaixo. Ela usa **Loop Until** para dizer ao Excel ficar repetindo o código *até que* uma célula na coluna adjacente

(coluna D) esteja vazia. Em outras palavras, ela ficará repetindo enquanto existir algo na coluna D.

Delete os dados das células **C2:C20** e fique pronto para o próximo exercício

### Exercício 2: Do While... Loop

O objetivo desta macro é rodar a coluna C enquanto for necessário colocando um cálculo em cada célula enquanto for necessário.

Na **Plan1** selecione célula **C2** e rode a macro **Laco2**

Aqui está o código

```
Sub Laco2()  
  
' Este laço roda enquanto existir alguma coisa na próxima coluna  
  
Do While IsEmpty(ActiveCell.Offset(0, 1)) = False  
  
ActiveCell.FormulaR1C1 = "=Average(RC[-1],RC[-2])"  
  
ActiveCell.Offset(1, 0).Select  
  
Loop  
  
End Sub
```

Esta macro faz o mesmo trabalho que aquela última usando os mesmos parâmetros mas simplesmente expressando-os de uma maneira diferente. Em vez de repetir o código *Até* que alguma coisa ocorra, ela faz algo *Enquanto* algo for o caso. Ela usa o **Do While** para dizer ao Excel ficar repetindo o código *enquanto* existir algo na coluna adjacente em oposição ao *até* que não existir nada lá. A função **IsEmpty = False** significa "Não Está Vazia".

Delete os dados das células **C2:C20** e fique pronto para o próximo exercício

### Exercício 3: Do While Not... Loop

O objetivo desta macro é rodar a coluna C enquanto for necessário colocando o cálculo em cada célula enquanto for necessário.

Na **Plan1** selecione célula **C2** e rode a macro **Laco3**.

Aqui está o código:

```
Sub Laco3()  
  
' Este laço roda enquanto existir alguma coisa na próxima coluna  
  
Do While Not IsEmpty(ActiveCell.Offset(0, 1))  
  
ActiveCell.FormulaR1C1 = "=Average(RC[-1],RC[-2])"
```

```
ActiveCell.Offset(1, 0).Select
```

```
Loop
```

```
End Sub
```

Esta macro toma exatamente a mesma decisão que aquela última mas just expresses it numa maneira diferente. **IsEmpty = False** significa o mesmo que **Not IsEmpty**. Algumas vezes você não pode dizer o que você quer dizer de uma única maneira assim também o VBA oferece freqüentemente uma sintaxe alternativa.

Delete os dados das células **C2:C20** e fique pronto para o próximo exercício

#### Exercício 4: Incluindo uma declaração IF

O objetivo desta macro é como antes, mas sem trocar qualquer dado que possa já existir.

Mova-se para a **Plan2**, selecione célula **C2** e rode a macro **Laco4**.

Aqui está o código:

```
Sub Laco4()  
  
' Este laço roda enquanto existir alguma coisa na próxima coluna  
  
' Ela não calcula a media se existir algo na célula  
  
Do  
  
If IsEmpty(ActiveCell) Then  
  
    ActiveCell.FormulaR1C1 = "=Average(RC[-1],RC[-2])"  
  
End If  
  
ActiveCell.Offset(1, 0).Select  
  
Loop Until IsEmpty(ActiveCell.Offset(0, 1))  
  
End Sub
```

As macros anteriores não levam em conta qualquer conteúdo possível que poderia já existir na células na qual ela está efetuando os cálculos. Esta macro usa uma declaração IF que diz ao Excel para realizar o cálculo *somente se* a célula estiver vazia. Isto evita qualquer dado existente de ser sobrescrito. A linha dizendo ao Excel para se mover para a próxima célula está *do lado de fora* da declaração IF porque ela tem de fazer isto de qualquer maneira.

#### Exercício 5: Evitando Erros

Esta macro emprega a declaração IF num estágio mais avançado, e não tenta calcular uma media de células que estão vazias.

Primeiro observe o problema. Mova-se para a **Plan3**, selecione célula **C2** e rode a macro **Laco4**.

Note que devido a alguns dos pares de células nas colunas A e B estarem vazias, a função =AVERAGE lança um erro #DIV/0 (a função Average adiciona os números nas células daí então divide pelo número de números – se não existir qualquer número ela tenta dividir por zero e você não pode fazer isto!).

Delete os conteúdos das células **C2:C6** e **C12:C20**. Selecione a célula **C2** e rode a macro **Laco5**.

Aqui está o código:

```
Sub Loop5()  
  
' Este laço roda enquanto existir alguma coisa na próxima coluna  
  
' Ela não calcula a media se existir algo na célula  
  
' nem se não existirem dados para tirar a media (para evitar erros  
#DIV/0).  
  
Do  
  
    If IsEmpty(ActiveCell) Then  
  
        If IsEmpty(ActiveCell.Offset(0, -1)) And  
IsEmpty(ActiveCell.Offset(0, -2)) Then  
  
            ActiveCell.Value = ""  
  
        Else  
  
            ActiveCell.FormulaR1C1 = "=Average(RC[-1],RC[-2])"  
  
        End If  
  
    End If  
  
    ActiveCell.Offset(1, 0).Select  
  
Loop Until IsEmpty(ActiveCell.Offset(0, 1))  
  
End Sub
```

Note que desta vez não existe mensagem de erro devido ao Excel não ter tentado calcular as medias dos números que não existem.

Nesta macro existe uma segunda declaração IF *dentro* daquela uma que diz ao Excel fazer algo somente se a célula estiver vazia. Esta segunda declaração IF oferece ao excel uma escolha. Em vez de um simples If existe um If e um Else. Aqui está como o Excel lê suas instruções...

"Se a célula já tiver alguma coisa nela, vá para a próxima célula. Mas, se a célula estiver vazia, observe as células correspondentes nas colunas A e B e se elas estiverem ambas vazias, não escreva nada ("" ). Caso contrário, escreva a fórmula na célula. Daí então mova-se para a próxima célula."

### Exercício 6: For... Next Loop

Se você souber, ou conseguir que o VBE descubra, quantas vezes repetir um bloco de código você pode usar um laço **For... Next**.

Mova-se para a **Plan4**, selecione a célula **C2** e rode a macro **Laco6**.

Aqui está o código:

```
Sub Loop6()  
  
' Este laço se repete para um número fixo de vezes determinado pelo  
número de linhas  
  
' no intervalo  
  
    Dim i As Integer  
  
    For i = 1 To Selection.CurrentRegion.Rows.Count - 1  
  
        ActiveCell.FormulaR1C1 = "=Average(RC[-1],RC[-2])"  
  
        ActiveCell.Offset(1, 0).Select  
  
    Next i  
  
End Sub
```

Esta macro não faz uso de uma coluna de células adjacentes como aquele anterior fez para saber quando parar o laço. Em vez disto ela conta o número de linhas no intervalo atual de dados e usa o método **For... Next** para dizer ao Excel para laçar aquele número de vezes (menos um, porque quando o VBA conta ele parte do zero).

### Exercício 7: Obtendo a Referência De Algum Outro Lugar

Selecione a célula **G2** e rode a macro **Laco7**.

Aqui está o código:

```
Sub Laco7()  
  
' Este laço se repete para um número fixo de vezes obtendo sua  
referência de outro lugar  
  
    Dim i As Integer  
  
    Dim intRowCount As Integer
```

```
intRowCount = Range("A1").CurrentRegion.Rows.Count - 1

For i = 1 To intRowCount

    ActiveCell.FormulaR1C1 = "=Average(RC[-5],RC[-6])"

    ActiveCell.Offset(1, 0).Select

Next i

End Sub
```

Você pode conseguir a referência para o número de laços de qualquer lugar. Esta macro coloca um conjunto de calculos na coluna G para um número de vezes ditado pelo número de linhas no bloco de dados iniciando com a célula A1. A declaração **For... Next** foi simplificada um pouco declarando primeiro uma variável **intRowCount** e preenchendo-a, com a informação apropriada (quantas linhas no bloco por A1). Esta variável consegue ser usada na próxima linha em vez de uma longa linha de código. Isto é apenas um outro exemplo de se fazer o mesmo trabalho de uma maneira diferente.

Se você quiser construir um laço que sempre rode um bloco de código um número fixo de vezes, você poderá simplesmente usar uma expressão como:

```
For i = 1 To 23
```

### Exercício 8: Sobre Fazer Cálculos...

Nos exercícios anteriores colocamos um cálculo numa célula de planilha escrevendo uma função regular do Excel na célula (e deixando-a lá) da mesma forma que se você tiver digitado-a por si mesmo. A sintaxe para isto é:

```
ActiveCell.FormulaR1C1 = "DIGITE SUA FUNÇÃO AQUI"
```

Estas macros usaram:

```
ActiveCell.FormulaR1C1 = "=Average(RC[-5],RC[-6])"
```

Devido a este método colocar uma função na célula ao invés de um valor, seus resultados mudarão quando as células a que elas se referirem mudar, exatamente como funções regulares – devido elas serem funções regulares. O cálculo foi conseguido no Excel porque toda aquilo que a macro fez foi escrever a função.

Se você preferir, você pode conseguir que a macro faça o cálculo e apenas escreva o resultado na célula. O VBA tem seu próprio conjunto de funções mas infelizmente a AVERAGE não é uma delas. Entretanto, o VBA suporta muitas das funções mais comuns do Excel com o método **WorksheetFunction**.

Na **Plan1** selecione célula **C2** e rode a macro **Laco1**.

Dê uma olhada nas células que você acabou de preencher. Cada uma contém uma função, escrita pela macro.

Agora delete os conteúdos das células **C2:C20**, selecione célula **C2** e rode a macro **Loop8**.

Aqui está o código:

```
Sub Loop8()  
  
' Este laço faz os cálculo por si só e escreve o resultado em cada  
célula  
  
    Do  
  
        ActiveCell.Value = WorksheetFunction.Average(ActiveCell.Offset(0,  
-1).Value, _  
  
            ActiveCell.Offset(0, -2).Value)  
  
        ActiveCell.Offset(1, 0).Select  
  
    Loop Until IsEmpty(ActiveCell.Offset(0, 1))  
  
End Sub
```

Dê uma olhada nas células que você acabou de preencher. Desta vez não existe função, apenas o valor. Todos os calculos foram feitos pela macro que então escreveu o valor na célula.