BASIC-256 :: Lição 04

Repetição

Quase sempre precisamos realizar tarefas repetitivas em programação. Dizemos que as coisas são repetidas dentro de um laço, o laço de repetição. Em BASIC-256, o mais comum e utilizado é o laço `for-next`.

programa 16: Repetição com `for`

A maneira mais comum de se repetir algo é usando o comando `for`. Esse comando especifica um intervalo numérico (por exemplo, de 1 a 10) e faz com que uma variável qualquer (por exemplo, `i`) assuma cada valor inteiro no intervalo, um de cada vez. Enquanto isso, o programa executa as instruções contidas entre o `for` e o `next`. Veja o exemplo a seguir:

```
for i = 1 to 10
     print "Olá!"
next i
```

Esse programa imprime 10 vezes "Olá!". Como ele funciona?

Começamos atribuindo a `i` um intervalo de 1 a 10. Em inglês, `1 to 10`. No princípio, `i` vale 1. Enquanto `i` vale 1, o `print "Olá!"` é executado. Quando o computador encontra o comando `next i`, o valor de `i` passa a ser 2, e novamente o comando `print "Olá!"` é executado. Isso ocorre até que `i` tenha passado por todos os números inteiros do intervalo. Depois que `i` passou por todos os números do intervalo, o laço chega ao fim.

programa 17: Contando de 1 em 1

Você pode imprimir os valores que a variável `i` toma a cada iteração dentro do laço `for`**:**

```
for i = 1 to 10
     print i
next i
```

Em verdade, você pode utilizá-la como desejar, como veremos nos programas seguintes.

programa 18: Contando de 2 em 2

Observe a sutil diferença entre este e o programa anterior. Qual foi o efeito dessa diferença?

```
for i = 1 to 10
     print 2*i
next i
```

Como você faria para contar de 3 em 3? E como você faria para exibir apenas os números ímpares?

programa 19: Repetição com salto

Você pode "pular" números na repetição com o comando `step`:

```
for j = 0 to 10 step 2
     print j
next j
```

Refaça o programa 18 agora com o comando `step`.

programa 20: Contagem regressiva

Os saltos podem ser negativos também:

```
for t = 10 to 0 step -1
     print t
     pause 1.0
next t
print "Bum!"
```

Qual é a função do comando `pause`? Com você faria para obter uma contagem regressiva de 100 a 0, pulando de 10 em 10?

programa 21: Uma animação simples

Repetições são úteis para criarmos animações. Tente descobrir o que o programa a seguir faz, mesmo sem executá-lo:

```
color green
for i = 10 to 300 step 5
     circle i, 100, 10
     pause 0.01
     clg
next i
```

Você percebeu? Agora digite e execute esse programa, acrescentando mais um laço no final que faça a bolinha voltar.

programa 22: Repetindo escolhas

Podemos aninhar escolhas dentro de repetições. Ou seja, você pode combinar a estrutura `if...then` com a estrutura `for...next`, colocando uma dentro da outra. No programa seguinte, pedimos para o computador "jogar" 10 vezes uma moeda e imprimir na tela o resultado:

```
for i = 1 to 10
     jogada = rand
     if jogada >= 0.5 then
          print "Cara!"
     else
          print "Coroa!"
     end if
next i
```

O comando `rand` escolhe um número entre 0 e 1, e o atribui à variável `jogada`. Se `jogada` for maior ou igual a 0,5, então o programa imprime "`Cara!`". Se `jogada` for menor que 0,5, o programa imprime "`Coroa!`". E isso é feito 10 vezes.

Agora, no lugar de `print "Cara!"` escreva `print "Cara!";` (acrescente um ponto-e-vírgula no final), faça o mesmo com `print "Coroa!"` e veja o que acontece quando você executa o programa. O que faz um ponto-e-vírgula no final de um `print`?

programa 23: Escolhendo repetições

Você pode aninhar repetições dentro de escolhas. O que o seguinte programa faz?

```
input "Para cima ou para baixo? (Digite 1 para ir para cima e
       2 para baixo): ", opcao
if opcao = 1 then
     for i = 300 to 10 step -5
          circle 150, i, 10
          pause 0.01
          clg
     next i
else
     for i = 10 to 300 step 5
          circle 150, i, 10
          pause 0.01
          clg
     next i
end
```

Se você for atento, vai perceber que qualquer número diferente de 1 ocasionará a execução do segundo `for`. Por quê?

programa 24: Repetindo repetições

O programa a seguir desenha uma malha feita de pequenos círculos. Isso é possível através do aninhamento de dois laços `for`:

```
clg
for i = 50 to 250 step 10
     for j = 50 to 250 step 10
          circle i, j, 3
     next j
next i
```

Nem sempre, quando aninhamos escolhas e repetições umas dentro de outras, como nos programas acima, ficamos em uma situação confortável para entender o que está acontecendo. A dica é começar a ler o programa "de dentro para fora", entendendo o que faz cada parte "de dentro" para depois articulá-las com as partes "de fora".

No caso acima, começamos entendendo o que faz o `for` relativo à variável `j`, e depois o que faz o `for` da variável `i`. Mas, como notamos, o `for` de dentro contém as duas variáveis. Nesse caso, dê um valor arbitrário a `i`, que está "fora", e aja como se você fosse o computador, calculando cada valor.