

Capítulo 7 - Loops básicos!

Até aqui aprendemos sobre escrever dados na tela e ler informações que o usuário digita, aprendemos o que são variáveis e constantes, aprendemos um pouco mais sobre operadores (aritméticos, lógicos e relacionais) e também aprendemos a controlar o fluxo de um algoritmo. Através de estruturas de decisões e de seleção decidimos para onde o fluxo do nosso algoritmo deve seguir.

Com o que aprendemos até agora já dá pra fazer muita coisa com programação!

Mas agora vamos aprender um recurso MUITO usado na programação: Os LOOPS. Entender como funcionam os LOOPS na programação fará você mudar a forma de pensar em algoritmos.

O que é LOOP?

Lembra quando você aprendeu a fazer multiplicação?

O(A) professor(a) deve ter te ensinado a fazer várias somas. Certo?

Por exemplo ...

$$4 * 5 = 4 + 4 + 4 + 4 + 4$$

Nosso(a) professor(a), nos ensinou a fazer um loop!

Em programação, LOOP é uma instrução para o programa repetir tarefas.

No algoritmo da multiplicação, nós somamos o primeiro valor X vezes, sendo X o segundo valor.

Os loops são muito utilizados no mundo da programação. Eles vêm em 3 sabores: ENQUANTO-FAÇA, REPITA-ATÉ e PARA-FAÇA.

Neste capítulo vamos estudar os dois primeiros: ENQUANTO-FAÇA e REPITA-ATÉ.

Estrutura de repetição ENQUANTO-FAÇA

O funcionamento da estrutura de repetição ENQUANTO-FAÇA (em inglês WHILE-DO) é tão simples quanto o SE-ENTÃO-SENÃO. A diferença é que os passos dentro deste bloco são repetidos enquanto a expressão booleana resultar VERDADEIRO.

Obs: Lembrando os tipos de dados do capítulo 3, o tipo de dados booleano só pode assumir dois valores: VERDADEIRO ou FALSO.

Voltando ao ENQUANTO ... Vejamos como ficaria o pseudo-código desta estrutura:

ENQUANTO <expressão booleana> FAÇA

 <instruções a serem executadas enquanto a expressão booleana resultar em VERDADEIRO>

FIM-ENQUANTO

Também chamamos esta estrutura de repetição de *loop pré-testado*, pois a expressão booleana é verificada antes da primeira execução. Se inicialmente ela já resultar em FALSO, as instruções que estão dentro do bloco não são executadas nenhuma vez.

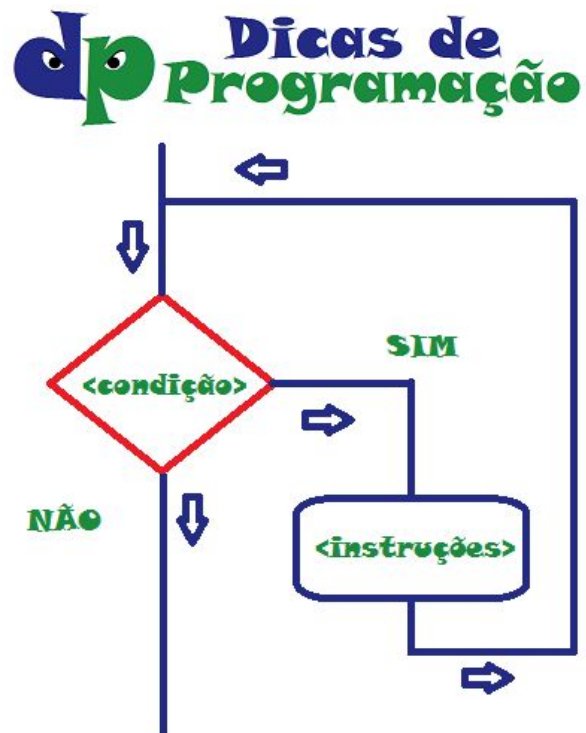
Este é o fluxograma desta estrutura de repetição. Repare que testamos a condição antes de entrar no LOOP:

Hora de praticar!

Para aprender programação, nada melhor que praticar! Vamos ver um exemplo de LOOP com a estrutura ENQUANTO-FAÇA, utilizando a ferramenta VisuAlg.

Vamos fazer um algoritmo para somar valores até o usuário digitar o valor 0. Ou seja, vamos somar todos os valores que o usuário digitar, porém quando ele digitar 0 o "loop" acaba, a cada iteração do loop vamos apresentar o resultado atual da soma.

```
algoritmo "SomaEnquantoValorDiferenteDe0"  
var
```



```
valorDigitado : REAL
soma : REAL
inicio

    soma := 0
    ESCREVA ("Digite um valor para a soma: ")
    LEIA (valorDigitado)

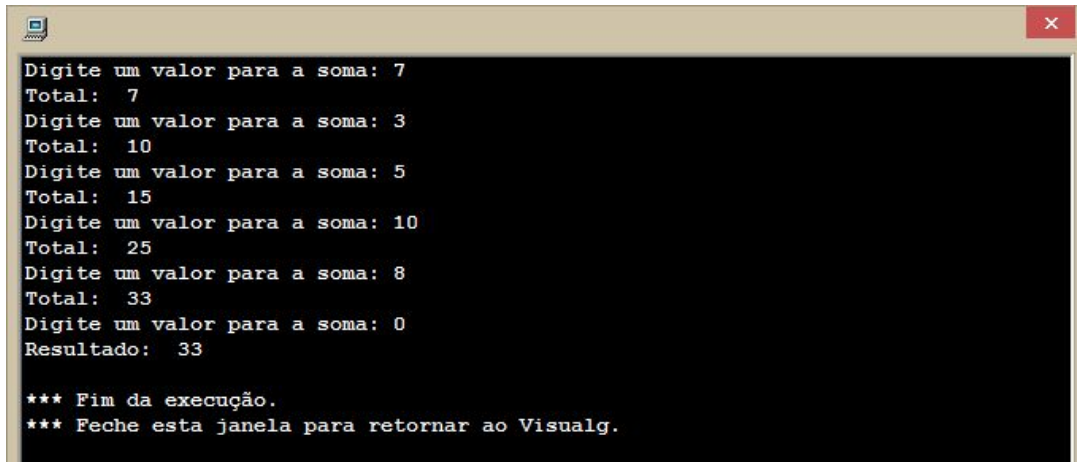
    ENQUANTO valorDigitado <> 0 FACA
        soma := soma + valorDigitado
        ESCREVAL ("Total: ", soma)
        ESCREVA ("Digite um valor para a soma: ")
        LEIA (valorDigitado)
    FIMENQUANTO

    ESCREVAL ("Resultado: ", soma)

finalgoritmo
```

Obs. A função ESCREVAL quebra a linha (como um ENTER) no final.

O resultado deste algoritmo é algo assim:



```
Digite um valor para a soma: 7
Total: 7
Digite um valor para a soma: 3
Total: 10
Digite um valor para a soma: 5
Total: 15
Digite um valor para a soma: 10
Total: 25
Digite um valor para a soma: 8
Total: 33
Digite um valor para a soma: 0
Resultado: 33

*** Fim da execução.
*** Feche esta janela para retornar ao Visualg.
```

Estrutura de repetição REPITA-ATÉ

Acho que você já deve imaginar como é esta estrutura né? Não!? Fácil!

Lembra que eu disse que a estrutura ENQUANTO-FAÇA é conhecida como *loop pré-testado*. Então, a estrutura REPITA-ATÉ (REPEAT-UNTIL em inglês) é o contrário. Ela é um LOOP *pós-testado*. Isso significa que a verificação para repetir o LOOP é testada no final do bloco.

Este é o pseudo-código do REPITA-ATÉ:

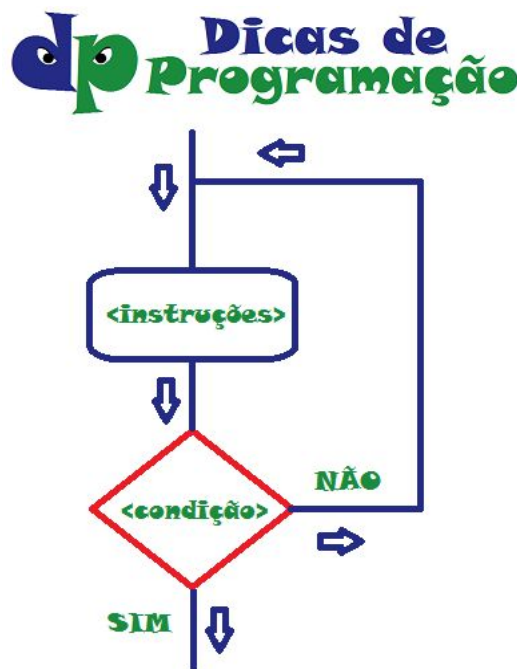
REPITA

<instruções a serem executadas repetidamente até a expressão booleana retornar VERDADEIRO>

ATÉ <expressão booleana>

Uma coisa muito importante a se notar é que além de ser pós-testada, esta estrutura testa o contrário do ENQUANTO. Ou seja, na estrutura REPITA-ATÉ, as instruções do bloco são executadas repetidamente enquanto a expressão booleana resultar FALSO. A partir do momento que a expressão booleana resultar VERDADEIRO, o fluxo do algoritmo sairá do LOOP.

Veja o funcionamento no fluxograma:



Não sei se você também percebeu, enquanto na estrutura ENQUANTO-FAÇA o bloco do LOOP pode não ser executado nenhuma vez, na estrutura REPITA-ATÉ o bloco é executado pelo menos uma vez.

Hora de praticar!

Que tal fazer o mesmo exercício que fizemos acima com a estrutura ENQUANTO-FAÇA, mas desta vez utilizando a estrutura REPITA-ATÉ? Vamos ver como ficaria?

```
algoritmo "SomaAteValorIgualA0"
var
    valorDigitado : REAL
    soma : REAL
inicio
    soma := 0

    REPITA
        ESCREVA ("Digite um valor para a soma: ")
        LEIA (valorDigitado)
        soma := soma + valorDigitado
        ESCREVAL ("Total: ", soma)
    ATÉ valorDigitado = 0

finalgoritmo
```

Algumas diferenças ...

Se você prestar atenção, vai perceber que na estrutura ENQUANTO-FAÇA tivemos que repetir uma parte do código antes do LOOP e dentro do LOOP. Repetimos a seguinte parte:

```
    ESCREVA ("Digite um valor para a soma: ")
    LEIA (valorDigitado)
```

Isso aconteceu porque a estrutura ENQUANTO-FAÇA é pré-testada. Não daria pra testar se o usuário digitou o valor 0 se ele ainda não tivesse digitado valor nenhum.

Na estrutura REPITA-ATÉ não precisamos escrever essas duas linhas duas vezes, pois ela é pós-testada.

Ah! Outra coisa que também não pode ser deixada de lado é que agora o teste de verificação do LOOP mudou de (valorDigitado <> 0) na estrutura ENQUANTO, para (valorDigitado = 0) na estrutura REPITA-ATÉ.

Você saberia explicar por quê? Pense um pouco e responda por si mesmo. O resultado deste algoritmo é o mesmo do anterior.

Conclusão

Percebemos que é possível utilizar qualquer uma das duas estruturas para implementar LOOPS, porém cada uma é mais apropriada dependendo do problema. Neste problema em particular, a estrutura REPITA-ATÉ se mostrou mais apropriada. Uma vez que nesta estrutura não é necessário repetir um pedaço do código.

A decisão de qual estrutura utilizar entre as duas, sempre será tomada observando a diferença entre PRÉ-TESTADA e PÓS-TESTADA. Fora isso é gosto pessoal (ou requisito do chefe para padronizar o código).

Aprenda muito bem os LOOPS! As estruturas de repetição são muito utilizadas em desenvolvimento de softwares. Entender como elas funcionam é muito importante para resolver problemas que precisam executar tarefas repetidas vezes. Acredite, existem muitos!

Para praticar a utilização da estrutura ENQUANTO, um exercício!

Lembra da multiplicação do começo do capítulo? Quero que você faça um algoritmo para calcular multiplicação através de somas consecutivas, para facilitar assumo que os dois fatores da multiplicação são positivos.

Agora uma dica bônus ...

As linguagens de programação são diferentes umas das outras, mas no fundo a lógica de programação é a mesma (quase sempre). Por exemplo. Na linguagem JAVA, não existe a estrutura REPITA-ATÉ. Mas existe a DO-WHILE, ou seja FAÇA-ENQUANTO. Esta também é pós-testada, mas o teste da condição não é o contrário da WHILE-DO. Pelo motivo óbvio. FAÇA-ENQUANTO (o teste der verdadeiro). Gostou da dica? Agora é com você! Faça o algoritmo para calcular a multiplicação através de somas.