Introdução à Programação #12

Teste Lógico *do while*

Os testes lógicos de repetição são, como já deve saber, extremamente úteis em inúmeras situações pois permitem-nos reutilizar código. Recentemente abordámos o teste lógico de repetição *while* e hoje iremos abordar um outro semelhante: o *do while*. A sua sintaxe é a seguinte:

```
1.  do
2.  {
3.     //código a ser repetido
4.  } while (condição);
```

Ao contrário do que acontece com o teste lógico *while,* no *do while,* o código é executado **primeiro** e só de seguida é que se confirma a veracidade da condição. Se a condição for verdadeira, então o código é executado mais uma vez, caso não o seja, já não será executado.

Este facto faz com que o que código contido dentro do teste lógico seja executado **pelo menos uma vez**. Veja então o seguinte exemplo de uma simples calculadora com o código documentado.

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.    /*
6.     * Variável que irá determinar se executamos
7.     * o código da calculadora de novo ou não.
8.     */
9.    int calcular;
10.
11.   do {
12.
13.     char operacao;
14.     float num1,
15.         num2;
16.
17.     //Limpeza do buffer
18.     fflush(stdin);
19.     __fpurge(stdin);
20.
21.     printf("Escolha a operação [+ - * / ]: ");
22.     scanf("%c",&operacao);
23.
24.     printf("Insira o primeiro número: ");
25.     scanf("%f",&num1);
26.
27.     printf("Insira o segundo número: ");
28.     scanf("%f",&num2);
29.
30.     switch( operacao )
31.     {
32.       case '+':
33.         printf("%.2f + %.2f = %.2f\n", num1, num2, num1 + num2);
34.         break;
35.
36.       case '-':
37.         printf("%.2f - %.2f = %.2f\n", num1, num2, num1 - num2);
38.         break;
39.
```

```
40.       case '*':
41.         printf("%.2f * %.2f = %.2f\n", num1, num2, num1 * num2);
42.         break;
43.
44.       case '/':
45.         printf("%.2f / %.2f = %.2f\n", num1, num2, num1 / num2);
46.         break;
47.
48.       default:
49.         printf("Você digitou uma operação invalida.\n");
50.         break;
51.     }
52.
53.     printf("Insira 0 para sair ou 1 \n");
54.     scanf("%d", &calcular);
55.
56.   } while (calcular);
57.   /*
58.    * Se "calcular" for diferente de 0 (falso), o código
59.    * da calculadora será executado novamente.
60.    *
61.    * Para terminar o código basta então digitar 0 quando
62.    * for pedido.
63.    */
64.
65.   return 0;
66.
67. }
```

Teste Lógico *for*

Outro teste lógico de repetição que é muito importante é o *for* e é muito útil quando, por exemplo, precisamos de inicializar uma variável a que deve ser incrementado um número. Veja a sintaxe:

```
1.  for(inicio_do_loop ; condição ; termino_de_cada_iteração)
2.  {
3.   //código a ser executado
4.  }
```

Vamos agora ver um pequeno exemplo comparando a sintaxe com um teste lógico *while*. O código seguinte utiliza o último *loop* mencionado para imprimir os números de 0 a 100:

```
1.  #include <stdio.h>
2.
3.  int main()
4.  {
5.      int num;
6.      num = 0;
7.
8.      while (num <= 100) {
9.          printf("%d\n", num);
10.         num++;
11.     }
12.
13.     return 0;
14. }
```

Abaixo pode encontrar o seu equivalente com o teste de repetição *for*:

```c
#include <stdio.h>

int main()
{
    int num;

    for(num = 0; num <=100; num++) {
        printf("%d\n", num);
    }

    return 0;
}
```

Apesar de parecer apenas uma pequena redução de duas linhas, com códigos mais complexos poderão ser poupadas mais linhas e este *loop* também melhora a legibilidade do código.

http://pplware.sapo.pt/tutoriais/vamos-programar-introducao-a-programacao-12