COMANDOS DE REPETIÇÃO COM TESTE NO FINAL

Estruturas de repetição com teste de condição no final (pós-teste)

Outra solução para processamento repetitivo de um programa é usar o comando de repetição com teste no final, ou seja, executa-se a sequência de comandos a ser repetida e somente faz-se o teste da condição ao final. Esse comando, na linguagem C, é muito similar ao WHILE (enquanto), com uma pequena diferença que já elucidaremos.

- Em **Portugol** (pseudocódigo), esse comando é FACA... ENQUANTO.
- Na **linguagem C**, esse comando é o DO... WHILE.

Veja, a seguir, a sintaxe do comando de repetição com teste de condição no final em Portugol e em linguagem C:

Portugol

```
//Sintaxe geral do comando FAÇA... ENQUANTO
faca
{
   Sequência de comandos a ser repetida
}
enquanto (condicao)
// comando após a repetição
```

C

```
//Sintaxe geral do comando DO... WHILE
do
{
   Sequência de comandos a ser repetida
}
while (condição)
// comando após a repetição
```

Funcionamento do comando DO...WHILE

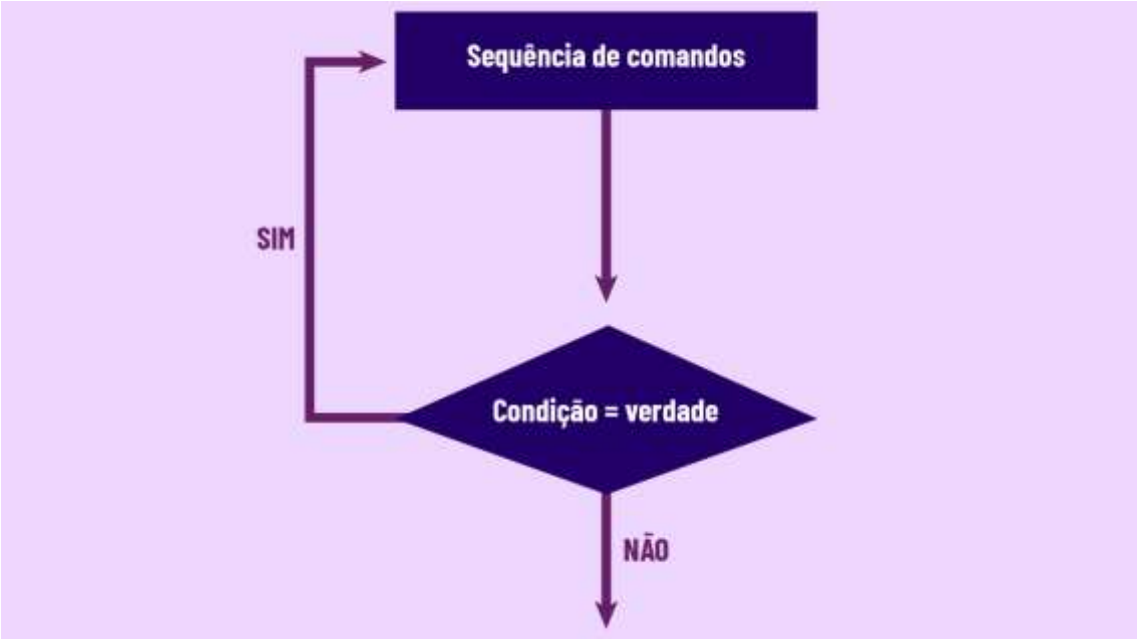
O comando DO... WHILE repete um bloco ou sequência de instruções enquanto uma condição for verdadeira. No momento em que a condição é

falsa, o controle do programa passa ao comando após a repetição (ao bloco que está sendo executado repetidas vezes).

Vejamos o funcionamento do comando DO... WHILE (acompanhe pela sintaxe geral demonstrada anteriormente):

- Executa a sequência de comandos a ser repetida.
- Se a condição é **verdadeira**, volta ao passo 1.
- Se a condição é falsa, vai para comando após repetição.

O fluxograma a seguir ilustra o funcionamento do comando de repetição com teste no final:



Podemos, então, concluir que:

1. A condição é avaliada (como verdadeira ou falsa) depois que a sequência de comandos a ser repetida é executada. O que significa que essa sequência será repetida pelo menos uma vez.
2. Enquanto a condição for verdadeira, a sequência de comandos a ser repetida é executada.
3. A sequência de comandos deixa de ser executada tão logo a condição seja falsa. A condição terá de ser falsa em algum momento, caso contrário a sequência de comandos será executada infinitamente, situação essa que chamamos de *loop*.

Colocando a teoria em prática

1º problema

Faça um programa que leia uma sequência de números inteiros, terminada em zero e mostre cada número lido (exceto o zero).

2º problema

Desenvolva um programa que leia uma sequência de números, podendo terminar com o número 0 ou 9. Para cada número lido (diferente de 0 ou 9), mostre seu sucessor caso o número seja par, ou seu antecessor se o número for ímpar.

Lógica: veja abaixo alguns exemplos de entrada com a lista terminando, no primeiro exemplo com 0 (zero) e no segundo exemplo com 9 (nove), e respectivos exemplos de saída. Optamos por usar o comando DO... WHILE, ou seja, precisamos de uma variável num para ler cada número da sequência.

Exemplo

Exemplo de entrada:
10 11 12 90 71 0
A saída do programa para a entrada acima seria: 11 10 13 91 70.
- 10 é par, mostra 11 (10+1 = sucessor).
- 11 é ímpar, mostra 10 (11-1=antecessor).
- 12 é par, mostra (12+1 = sucessor).
- 90 é par, mostra (90+1 = sucessor).
- 71 é ímpar, mostra (71-1=antecessor).

Determinamos a condição do comando DO... WHILE. A sequência pode ser encerrada com a leitura dos números 0 ou 9, logo, a repetição deve acontecer enquanto num for diferente de 0 e de 9, pois basta que uma das condições seja falsa para invalidar toda a condição e a sequência de repetição seja interrompida, uma vez que estamos usando o operador lógico E (*&& em C*).

Portugol

```
inteiro num

faca

{
```

```
}
enquanto(num!=0 && num!=9)
```

C

```
int num;
do
{
}
while(num!=0 && num!=9)
```

A leitura de valor para a variável num é realizada dentro da repetição, juntamente com o teste: se num é par (resto da divisão por 2 é igual a 0) ou se é ímpar (resto por 2 é diferente de 0). Esse teste é mutuamente exclusivo, de forma que podemos testar se num é par, aplicando a regra do par, ou se é ímpar, aplicando a regra do ímpar na cláusula senão, conforme mostrado a seguir. Devemos considerar ainda que o processamento de exibição do sucessor ou antecessor não deve acontecer quando for lido o conteúdo 0 ou 9 para a variável num.

Portugol

```
inteiro num
faca
{
   escreva ("Digite um número: ")
   leia (num)
   se (num!=0 && num!=9)
   {
      se (num%2 ==0)
      escreva ("Sucessor",num+1)
      senao
      escreva ("Antecessor",num-1)
   }
}
```

```
enquanto (num!=0 && num!=9);
```

C

```
int num
do
{
   printf ("Digite um número: ");
   scanf("%d",&num);
   if (num!=0 && num!=9)
   {
      if (num%2 ==0)
      printf ("Sucessor = %d\n\n ",num+1);
      else
      printf ("Antecessor = %d\n\n ",num-1);
   }
}
while (num!=0 && num!=9);
```

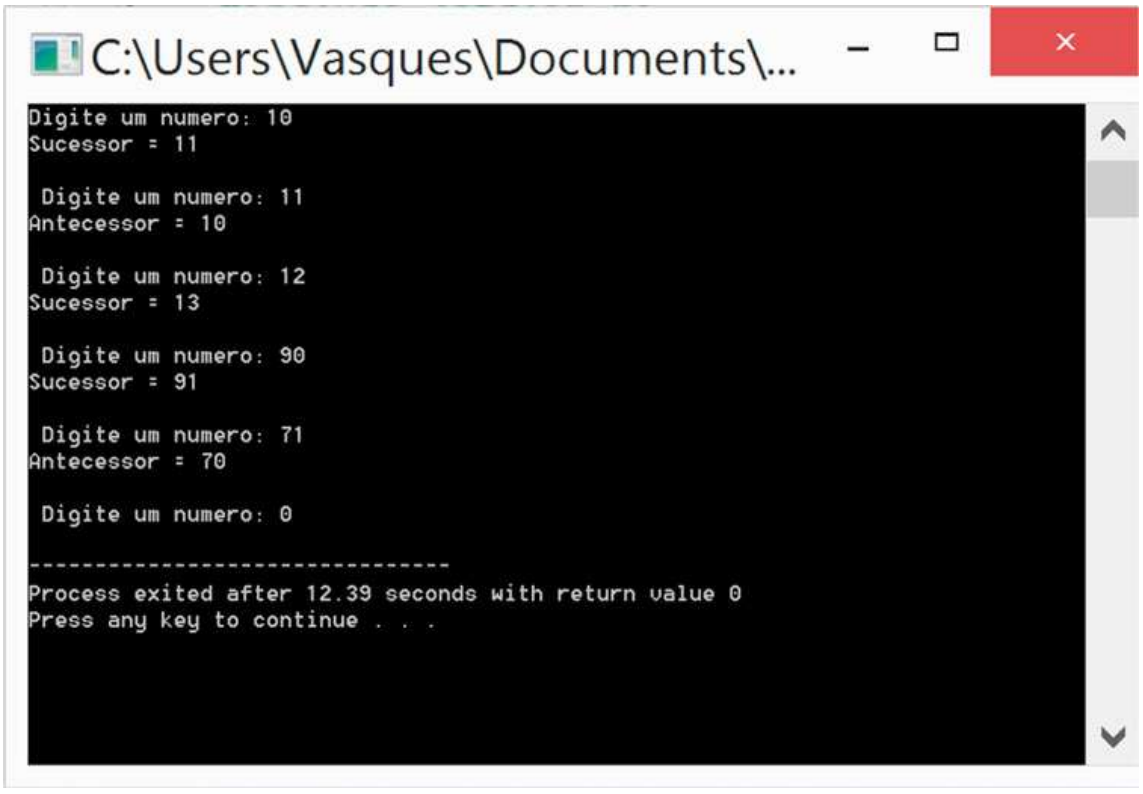
Agora, observe ao lado o código completo da solução usando o DO... WHILE na linguagem C:

```
// Código em Linguagem C


#include <stdio.h>
#include <stdlib.h>
int main()
{
   int num;
   do
   {
      printf ("Digite um número: \n");
      scanf("%d",&num);
```

```
        if (num!=0 && num!=9)
        {
            if (num%2 ==0)
                printf ("Sucessor = %d\n\n ",num+1);
            else
                printf ("Antecessor = %d\n\n ",num-1);
        }
    }
    while (num!=0 && num!=9);
    return 0;
}
```

Para finalizar esse problema, veja a imagem do resultado da execução do programa no Dev-C++ para a sequência de entrada: 10, 11, 12, 90, 71, 0.



Resultado da execução no ambiente Dev-C++.

3º problema

Desenvolva um programa que leia o salário bruto de 15 funcionários de uma empresa, calcule e exiba o salário líquido de cada funcionário. Lembre-se de

que o salário líquido é calculado abatendo o imposto do salário bruto, com base na tabela de imposto abaixo. Ao final, mostre o total de salários brutos, salários líquidos e impostos de todos os funcionários.

Faixa	Valor inicial	Valor final	% de imposto
1	R$ 0.00	R$ 999.00	10%
2	R$ 999.01	R$ 1.999,00	15%
3	R$ 1.999.01	R$ 9.999.00	20%
4	R$ 9.999,01	R$ 99.999.00	25%
5	Acima R$ 99.999,01	---	30%

Marcelo Vasques de Oliveira

Lógica: esse é um problema que será resolvido usando os 3 comandos de repetição. Vamos iniciar com o comando FOR (PARA). Variáveis reais (float) necessárias: salbruto, salliquido, imposto, totbruto, totliquido e totimposto, além da variável inteira para controlar a repetição.

1. Inicializar as variáveis totalizadoras.
2. Repetir 15 vezes (usando FOR):
    1. Ler salário bruto.
    2. Calcular o imposto (ninho de Ifs, aplicando cada porcentagem conforme salário).
    3. Contabilizar o somatório dos salários brutos, salários líquidos e impostos.
3. Exibir as variáveis totalizadoras.

Veja os códigos completos da solução na linguagem C usando, respectivamente, os comandos FOR, WHILE e DO... WHILE:

// Código em Linguagem C - Comando FOR

```
#include <stdio.h>
```

```
#include <stdlib.h>
int main()
{
   float salbruto, salliquido, imposto, totbruto=0, totliquido=0,totimposto=0;
   int contfunc;
   for (contfunc=1;contfunc<=5;contfunc++)
   {
      printf ("Digite o salário bruto: ");
      scanf("%f",&salbruto);
      if (salbruto >999)
         imposto = salbruto*0.10;
      else
         if (salbruto >1999)
         imposto = salbruto*0.15;
         else
            if (salbruto >9999)
            imposto = salbruto*0.20;
            else
               if (salbruto >99999)
               imposto = salbruto*0.25;
               else
               imposto = salbruto*0.30;
               salliquido = salbruto - imposto;
      printf ("Salário Liquido: %.2f \n",salliquido);
      totbruto = totbruto + salbruto;
      totliquido = totliquido + salliquido;
      totimposto = totimposto + imposto;
   }
   printf ("TOT salário bruto : %.2f \n",totbruto);
   printf ("TOT salário líquido : %.2f \n",totliquido);
   printf ("TOT imposto : %.2f \n",totimposto);
```

```
    return 0;
}

// Código em Linguagem C - Comando WHILE
#include <stdio.h>
#include <stdlib.h>
int main()
{
    float salbruto, salliquido, imposto, totbruto=0, totliquido=0, totimposto=0;
    int contfunc=1;
    while (contfunc<=15)
    {
        printf ("Digite o salário bruto: ");
        scanf("%f",&salbruto);
        if (salbruto >999)
            imposto = salbruto*0.10;
        else
            if (salbruto >1999)
            imposto = salbruto*0.15;
            else
                if (salbruto >9999)
                imposto = salbruto*0.20;
                else
                    if (salbruto >99999)
                    imposto = salbruto*0.25;
                    else
                    imposto = salbruto*0.30;
                    salliquido = salbruto - imposto;
        printf ("Salário Liquido: %.2f \n",salliquido);
        totbruto = totbruto + salbruto;
        totliquido = totliquido + salliquido;
```

```
        totimposto = totimposto + imposto;
        contfunc++;
    }
    printf ("TOT salário bruto : %.2f \n",totbruto);
    printf ("TOT salário líquido : %.2f \n",totliquido);
    printf ("TOT imposto : %.2f \n",totimposto);
    return 0;
}

// Código em Linguagem C - Comando DO WHILE
#include <stdio.h>
#include <stdlib.h>
int main()
{
    float salbruto, salliquido, imposto, totbruto=0, totliquido=0,totimposto=0;
    int contfunc=1;
    do
    {
        printf ("Digite o salário bruto: ");
        scanf("%f",&salbruto);
            if (salbruto >999)
            imposto = salbruto*0.10;
            else
                if (salbruto >1999)
                imposto = salbruto*0.15;
                else
                    if (salbruto >9999)
                    imposto = salbruto*0.20;
                    else
                        if (salbruto >99999)
                        imposto = salbruto*0.25;
```

```
                    else
                    imposto = salbruto*0.30;
                    salliquido = salbruto - imposto;
        printf ("Salário Liquido: %.2f \n",salliquido);
        totbruto = totbruto + salbruto;
        totliquido = totliquido + salliquido;
        totimposto = totimposto + imposto;
        contfunc++;
    } while (contfunc<=15);
    printf ("TOT salário bruto : %.2f \n",totbruto);
    printf ("TOT salário líquido : %.2f \n",totliquido);
    printf ("TOT imposto : %.2f \n",totimposto);
    return 0;
}
```