

COMANDOS DE REPETIÇÃO COM TESTE NO INÍCIO

Estruturas de repetição com teste de condição no início (pré-teste)

Existem determinados tipos de situações em que a solução com o comando FOR do C (ou PARA, em Portugal) não é a mais apropriada, como veremos a seguir.

1º exemplo

Desenvolva um programa que leia uma sequência de números inteiros terminada em 0 e mostre cada número lido (exceto o 0).

Lógica: o comando FOR do C não é indicado para resolver esse tipo de problema, posto que não sabemos quantos números serão lidos antes do 0. Pode até ser que a sequência tenha apenas o 0 e nada precise ser feito.

Não temos como precisar a quantidade exata de números que virão antes do 0, o que sinaliza o fim da sequência de números, diferentemente de todos os problemas de repetição vistos até aqui.

Sem saber a quantidade exata de números a serem processados, o uso do comando FOR não é adequado. Uma das soluções é usar o comando de repetição com teste no início, ou seja, o comando testa a condição antes de iniciar a sequência de comandos a ser repetida.

- Em Portugal (pseudocódigo), esse comando é o ENQUANTO.
- Na linguagem C, esse comando é o WHILE.

Demonstração

Vejamos a sintaxe em Portugal e em linguagem C:

Portugal

```
//Sintaxe geral do comando ENQUANTO
ENQUANTO (condição)
{
    Sequência de comandos a ser repetida
}
// comando após a repetição
```


Podemos, desse modo, chegar às seguintes conclusões:

1. A condição é avaliada (como verdadeira ou falsa) antes que a sequência de comandos a ser repetida seja executada. O que significa dizer que essa sequência pode não ser executada nenhuma vez.
2. Uma vez que a condição seja verdadeira, a sequência de comandos a ser repetida é executada.
3. Ao final de cada sequência de comandos a ser repetida, a condição é novamente testada.
4. A sequência de comandos deixa de ser executada tão logo a condição seja falsa. A condição terá de ser FALSA, em algum momento, pois caso contrário a sequência de comandos será executada infinitamente, situação que chamamos de *loop*.

Vamos resolver, passo a passo, o problema motivador para o comando WHILE. Como exemplo por exemplo desenvolver um programa que leia uma sequência de números inteiros terminada em zero e mostre cada número lido (exceto o zero).

Para solucionar esse problema, vamos construir a lógica do programa em cinco passos: Precisaremos de uma variável do tipo inteiro, que chamaremos de num.

Portugol
inteiro num

C
int num;

Entenderemos a condição: enquanto o número lido for diferente de 0, vamos mostrá-lo; a condição é: *num!=0* (num diferente de 0)

Portugol
enquanto (num!=0)

C
while (num!=0)

Antes de iniciar a repetição, devemos ter uma leitura prévia na variável num.

Para que possamos comparar o conteúdo da variável num com o valor 0, na primeira vez que a condição for testada, precisamos ter um valor já lido na variável num:

```
Portugol
inteiro num
leia(num)
enquanto (num!=0)
```

```
C
int num;
scanf("%d",&num);
while (num!=0)
```

Dentro da repetição, vamos estabelecer os comandos do processamento solicitado no enunciado, que, nesse caso, é apenas mostrar cada número lido:

```
Portugol
inteiro num
leia(num)
enquanto (num!=0)
{
    escreva (num)
}
```

```
C
int num;
scanf("%d",&num);
while (num!=0)
{
    printf ("O número lido foi = %d\n\n ",num);
}
```

Se o programa ficar como o anterior, o que acontecerá?

Irá exibir o primeiro número lido (antes do WHILE) indefinidamente, pois o conteúdo da variável num não será alterado na repetição.

Solução seria, então, inserir um comando de leitura de conteúdo para a variável num dentro da repetição, após o comando de exibir o número lido (mesmo comando scanf escrito antes do WHILE).

```

Portugol
inteiro num
leia(num)
enquanto (num!=0)
{
    escreva (num)
    leia (num)
}

```

```

C
int num;
scanf("%d",&num);
while (num!=0)
{
    printf ("O número lido foi = %d\n\n ",num);
    scanf("%d",&num);
}

```

A lógica está pronta!

Utilizaremos agora comandos de exibição para melhorar a interface do programa e a interação com o usuário ao executá-la. Vamos inserir duas vezes o comando printf, sendo um antes de cada scanf:

```

Portugol
inteiro num
leia(num)
escreva("Digite um número: ")
enquanto (num!=0)
{
    escreva (num)
    escreva("Digite um número: ")
    leia (num)
}

```

```

C
int num;
printf ("Digite um número: ");
scanf("%d",&num);
while (num!=0)
{

```

```

    printf ("O número lido foi = %d\n\n ",num);
    printf ("Digite um número: ");
    scanf("%d",&num);
}

```

Veja o programa completo em linguagem C a seguir:

```

C
#include <stdio.h>
#include <stdlib.h>
int main()
{
    int num;
    printf ("Digite um número: ");
    scanf("%d",&num);
    while (num!=0)
    {
        printf ("O número lido foi = %d\n\n ",num);
        printf ("Digite um número: ");
        scanf("%d",&num);
    }
    return 0;
}

```

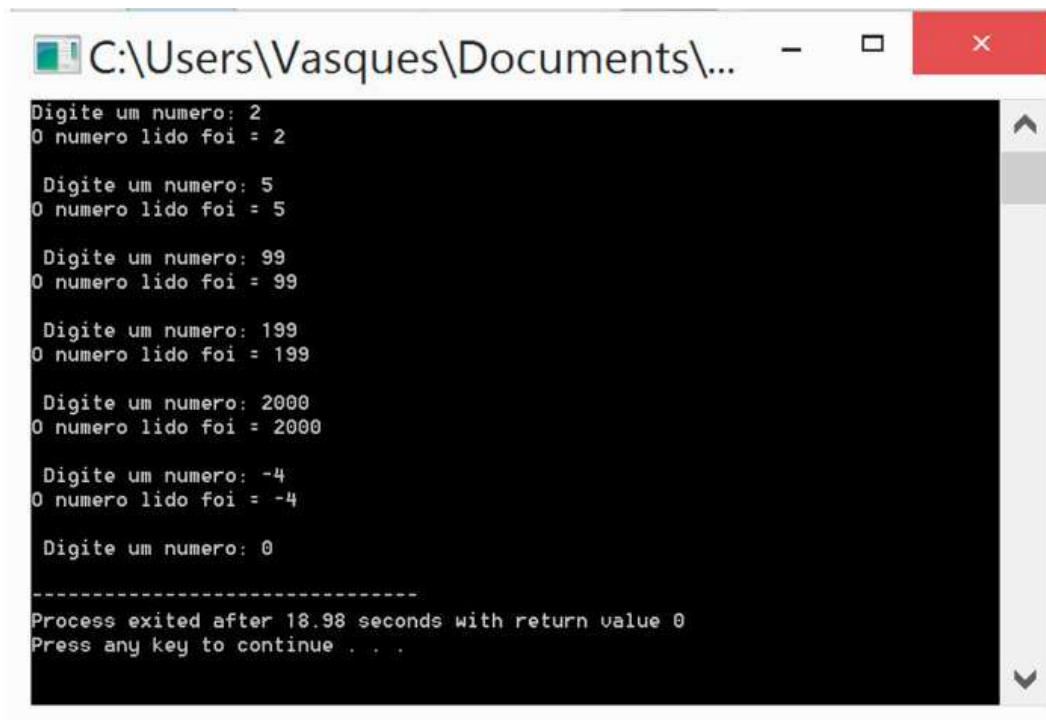
Sugerimos que você abra um novo projeto no Dev-C++, copie e cole o programa em C acima e execute-o com esta sequência de dados:

```

2
5
99
199
2999
-4
0

```

Veja como ficará a execução dessa sequência no respectivo código.



```
C:\Users\Vasques\Documents\...
Digite um numero: 2
0 numero lido foi = 2

Digite um numero: 5
0 numero lido foi = 5

Digite um numero: 99
0 numero lido foi = 99

Digite um numero: 199
0 numero lido foi = 199

Digite um numero: 2000
0 numero lido foi = 2000

Digite um numero: -4
0 numero lido foi = -4

Digite um numero: 0

-----
Process exited after 18.98 seconds with return value 0
Press any key to continue . . .
```

Colocando a teoria em prática

O comando WHILE permite que resolvamos problemas com repetições com número fixo e conhecido de vezes, assim como fizemos com o comando FOR?

Para sabermos a resposta da questão anterior, vejamos um exemplo de um problema dos programas vistos quando estudamos o comando FOR (PARA).

1º problema

Desenvolva um programa que leia 15 números inteiros e positivos e mostre o maior deles.

2º problema

Desenvolva um programa que leia 3 notas de 40 alunos, calcule e mostre a média aritmética e a situação de aprovação de cada aluno. Lembre-se de que apenas a média igual ou acima de 7 aprova o aluno.

//Linguagem C - Com o comando FOR

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
```

```

float nota1,nota2,nota3,media;
int contalunos;
for (contalunos=1;contalunos<=40;contalunos++)
{
    printf("Entre com a nota 1 do aluno: ");
    scanf("%f",&nota1);
    printf("Entre com a nota 2 do aluno: ");
    scanf("%f",&nota2);
    printf("Entre com a nota 3 do aluno: ");
    scanf("%f",&nota3);
    media=(nota1+nota2+nota3)/3;
    if (media>=7)
        printf("APROVADO com média %.2f \n\n",media);
    else
        printf("REPROVADO com média %.2f \n\n",media);
}
return 0;
}

```

C

// Código em Linguagem C - Com o comando WHILE

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    float nota1,nota2,nota3,media;
    int contalunos=1;
    while (contalunos<=6)
    {
        printf("Entre com a nota 1 do aluno: \n");
        scanf("%f",&nota1);
        printf("Entre com a nota 2 do aluno: \n");
        scanf("%f",&nota2);
        printf("Entre com a nota 3 do aluno: \n");
        scanf("%f",&nota3);
        media=(nota1+nota2+nota3)/3;
        if (media>=7)
            printf("APROVADO com média %.2f \n\n",media);
        else
            printf("REPROVADO com média %.2f \n\n",media);
        contalunos++;
    }
}

```



```
    return 0;
}
```

3º problema

Desenvolva um programa que leia uma sequência de letras (a... z) terminada em ponto (.) e que mostre quantas vezes cada vogal (a, e, i, o, u) apareceu na lista.

Lógica: vamos precisar de uma variável do tipo caracter (char em C) para armazenar cada letra lida; de uma variável para acumular cada vogal (conta, conte, conti, conto, contu) e inicializar cada uma com zero.

Portugol

caracter letra

inteiro conta=0, conte=0, conti=0, conto=0, contu=0

C

char letra;

int conta=0, conte=0, conti=0, conto=0, contu=0;

Como a condição está expressa em termos da variável de nome letra, precisamos garantir que nela haja conteúdo lido, o que nos leva a fazer uma leitura antes da repetição.

1. **Repetição:** usaremos WHILE (ENQUANTO).
2. **Condição:** enquanto o conteúdo da variável letra for diferente do ponto (.).

Portugol

caracter letra

inteiro conta=0, conte=0, conti=0, conto=0, contu=0

```
{
}
```

C

char letra;

int conta=0, conte=0, conti=0, conto=0, contu=0;

```
{
}
```

O processamento desse programa é contabilizar cada vogal, então temos de usar um comando de decisão (seleção) para que saibamos quando a letra for

uma das vogais. Como temos de contabilizar cada vogal, o comando mais adequado é o de seleção múltipla: SWITCH (ESCOLHA). Nosso interesse é apenas quando letra for igual a uma das vogais, pois incrementaremos cada uma das variáveis contadoras de vogal.

```
Portugol
escolha (letra)
{
  caso 'a':
    conta++; pare;
  caso 'e':
    conte++;pare;
  caso 'i':
    conti++;pare;
  caso 'o':
    conto++;pare;
  caso 'u':
    contu++;pare;
}
```

```
C
switch (letra)
{
  case 'a':
    conta++; break;
  case 'e':
    conte++;break;
  case 'i':
    conti++;break;
  case 'o':
    conto++;break;
  case 'u':
    contu++;break;
}
```

Agora temos de acrescentar a leitura de novo conteúdo para a variável letra dentro da repetição, pois, caso contrário, o valor lido antes da repetição será processado indefinidamente e mostrará cada contador de variável quando for lido o “.” (ponto) que determina o fim da sequência de letras, o final da repetição.

Portugol

caracter letra

inteiro conta=0, conte=0, contei=0, conto=0, contu=0

leia (letra)

enquanto (letra!= '.')

{

 escolha (letra)

 {

 caso 'a':

 conta++;pare;

 caso 'e':

 conte++;pare;

 caso 'i':

 conti++;pare;

 caso 'o':

 conto++;pare;

 caso 'u':

 contu++;pare;

 }

 leia (letra)

}

escreva (conta)

escreva (conte)

escreva (conti)

escreva (conto)

escreva (contu)

C

char letra;

int

int conta=0, conte=0, conti=0, conto=0, contu=0;

scanf("%c",&letra);

while (letra!='.')

{

 switch (letra)

 {

```

        case 'a':
            conta++;break;
        case 'e':
            conte++;break;
        case 'i':
            conti++;break;
        case 'o':
            conto++;break;
        case 'u':
            contu++;break;
    }
    scanf("%c",&letra);
}
printf("Total de a: %d \n",conta);
printf("Total de e: %d \n",conte);
printf("Total de i: %d \n",conti);
printf("Total de o: %d \n",conto);
printf("Total de u: %d \n",contu);

```

Agora veja o programa completo, escrito na linguagem C, acrescido do comando printf de interação com o usuário:

printf ("Digite uma letra minúscula (a..z) a cada linha e tecle ENTER: \n");

C

// Código em Linguagem C

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    char letra;
    int conta=0, conte=0, conti=0, conto=0, contu=0;

```

```
printf("Digite uma letra minúscula (a..z) a cada linha e tecle ENTER :
\n");
scanf("%c",&letra);
while (letra!='.')
{
    switch (letra)
    {
        case 'a':
            conta++;break;
        case 'e':
            conte++;break;
        case 'i':
            conti++;break;
        case 'o':
            conto++;break;
        case 'u':
            contu++;break;
    }
    scanf("%c",&letra);
}
printf("Total de a: %d \n",conta);
printf("Total de e: %d \n",conte);
printf("Total de i: %d \n",conti);
printf("Total de o: %d \n",conto);
printf("Total de u: %d \n",contu);
return 0;
}
```