



DESCRIÇÃO

Análise da sintaxe e do funcionamento das categorias de comandos de repetição — com variável de controle (número fixo e conhecido de vezes), com teste de condição no início (pré-teste) e no final (pós-teste) —, apontando a mais adequada para a solução do problema.

PROPÓSITO

Aplicar três diferentes estruturas (comandos) de repetição, disponíveis na maioria das linguagens de programação, permitindo que o programa possa repetir um bloco ou sequência de instruções, fundamental nas soluções algorítmicas para processar um conjunto repetido de dados.

PREPARAÇÃO

Antes de iniciar seu estudo, instale os programas **Portugol Studio** e **Dev-C++**, pois são fundamentais para o acompanhamento do tema.

OBJETIVOS

MÓDULO 1

Validar linhas de código a partir de uma solicitação de comando FOR em Portugol e na linguagem C

MÓDULO 2

Identificar conceitos e assertivas relacionados aos comandos de repetição com teste no início

MÓDULO 3

Identificar conceitos e assertivas relacionados aos comandos de repetição com teste no final

MÓDULO 1

⦿ Validar linhas de código a partir de uma solicitação de comando FOR em Portugol e na linguagem C

ESTRUTURAS DE REPETIÇÃO COM VARIÁVEL DE CONTROLE

Os trechos de algoritmos a seguir processam um conjunto de dados único para obter os resultados desejados.

1º Exemplo: ler 3 notas de um aluno, calcular e exibir a média aritmética dessas notas.

// Código em Portugol

```
funcao inicio ()  
{  
  Real nota1, nota2, nota3, media  
  escreva (" nota 1 = ")  
  leia (nota1)  
  escreva (" nota 2 = ")  
  leia (nota2)  
  escreva (" nota 3 = ")  
  leia (nota3)  
  media = (nota1+nota2+nota3)/3  
  escreva (" media = ",media)  
}
```

// Código em Linguagem C

```
int main ()  
{  
  float nota1, nota2, nota3, media  
  printf (" nota 1 = ");  
  scanf ("%f", &nota1);  
  printf (" nota 2 = ");  
  scanf ("%f", &nota2);  
  printf (" nota 3 = ");  
  scanf ("%f", &nota3);  
  media = (nota1+nota2+nota3)/3;
```

```
printf ("media = %f",media);  
}
```

2º Exemplo: ler 3 notas de um aluno, calcular e mostrar a média aritmética e exibir se o aluno foi aprovado ou não (média igual ou acima de 7 aprova o aluno).

Observe que a única diferença desta solução e a do exemplo anterior são **as duas últimas linhas**:

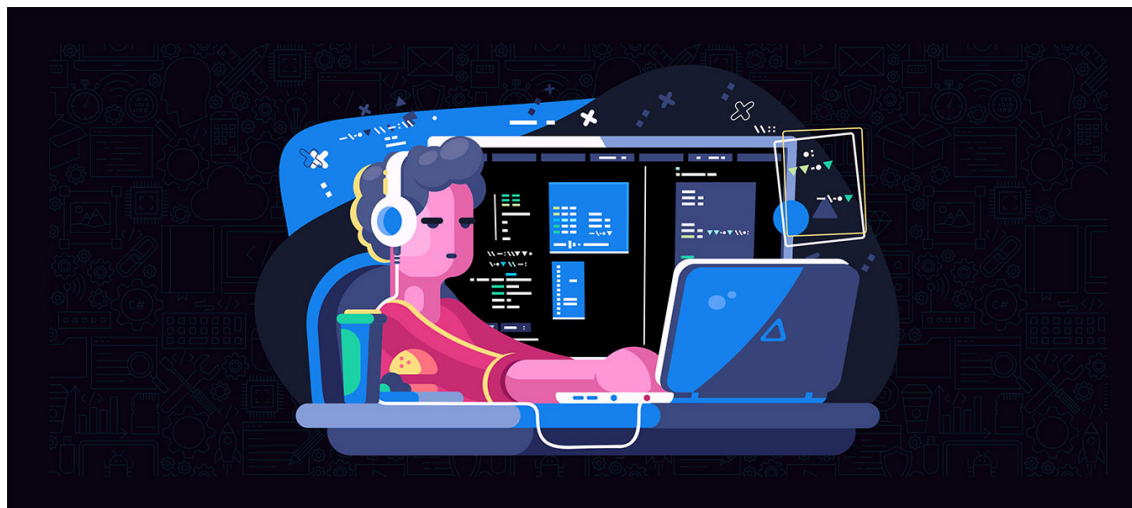
// Código em Portugol

```
funcao inicio ()  
{  
Real nota1, nota2, nota3, media  
escreva (" nota 1 = ")  
leia (nota1)  
escreva (" nota 2 = ")  
leia (nota2)  
escreva (" nota 3 = ")  
leia (nota3)  
media = (nota1+nota2+nota3)/3  
escreva (" media = ",media)  
se (media>=7) escreva (" , Aluno Aprovado")  
senao escreva (" , Aluno Reprovado")  
}
```

// Código em Linguagem C

```
int main ()  
{  
float nota1, nota2, nota3, media  
printf (" nota 1 = ");  
scanf ("%f", &nota1);  
printf (" nota 2 = ");  
scanf ("%f", &nota2);  
printf (" nota 3 = ");  
scanf ("%f", &nota3);  
media = (nota1+nota2+nota3)/3;
```

```
printf ("media = %f",media);  
if (media>=7) printf (" , Aluno Aprovado");  
else printf (" , Aluno Reprovado");  
}
```



Vimos dois exemplos que processam três notas de um único aluno. Mas e se quiséssemos calcular e mostrar a média e a situação de aprovação de 40 alunos de uma turma? Temos duas soluções de acordo com o que sabemos até o momento:

1

Executar o programa 40 vezes;

2

Usar 120 variáveis: 3 variáveis para as notas de cada um dos 40 alunos ($3 \times 40 = 120$).

VOCÊ CONSIDERA VIÁVEL ALGUMA DESSAS SOLUÇÕES? E SE FOSSEM 100, 300 OU 1.000 ALUNOS?

A sequência de comandos para ler, calcular e mostrar a média e situação de um aluno é a mesma para 100 ou qualquer outra quantidade de alunos, concorda? **Como devemos proceder então?**

REPETINDO MANUALMENTE OS COMANDOS

COMENTÁRIO

Os **comandos de repetição** nos permitem repetir, quantas vezes desejarmos, uma sequência ou bloco de comandos.

DEMONSTRAÇÃO

Vamos ver um exemplo da utilização dos comandos de repetição.

Objetivo: ler 3 notas de 40 alunos, calcular e mostrar a média aritmética e a situação de aprovação de cada aluno (média igual ou acima de 7 aprova o aluno).

LÓGICA:

Para resolver esse problema, precisamos aprender as estruturas de repetição, também chamadas de comandos de repetição ou de iteração. Dessa forma, somos capazes de repetir uma sequência ou bloco de comandos a quantidade de vezes que precisamos.

Neste exemplo, precisamos repetir 40 vezes a mesma sequência de comandos do programa do exemplo anterior, pois o procedimento de cálculo da média de um aluno é o mesmo para cada um dos 40 alunos.

Uma das possíveis soluções para a repetição de uma sequência de comandos, em um programa, é o uso do comando classificado como estrutura de repetição com variável de controle, que possibilita a repetição de uma sequência ou bloco de comandos um número fixo e conhecido de vezes.

É o mais indicado quando sabemos previamente, pelo enunciado do problema, o número de vezes que a repetição vai acontecer. Neste exemplo a premissa é que devemos processar notas de uma turma com 40 alunos. Sabemos, portanto, que vamos repetir o procedimento (sequência ou bloco de comandos) para calcular a média, mostrá-la e exibir a situação do aluno 40 vezes: uma vez para cada um dos 40 alunos que fazem parte da turma.

- Em Portugol (pseudocódigo), o comando é o PARA.

- Na linguagem C, o comando é o FOR.

A seguir, apresentamos a sintaxe geral do comando de repetição com variável de controle em Portugol e em C:

// Código em Portugol

// Sintaxe geral do comando PARA

PARA (inicialização; condição; incremento_decremento)

{

Bloco ou sequência de comandos a ser repetida

}

// comando após a repetição

// Código em Linguagem C

// Sintaxe geral do comando FOR

FOR (inicialização; condição; incremento_decremento)

{

Bloco ou sequência de comandos a ser repetida

}

// comando após a repetição

O comando é composto de 3 partes:

INICIALIZAÇÃO

Valor inicial da variável de controle. Essa ação é executada, uma única vez, ao iniciar o comando.

CONDIÇÃO

Expressão relacional (retorna um valor verdadeiro ou falso) associada à variável de controle. A condição é avaliada antes repetição: se for verdade, a repetição ocorre; se for falsa, a repetição não ocorre e o fluxo do programa vai para o **comando após a repetição**.

INCREMENTO_DECREMENTO

Aumento ou diminuição do valor da variável de controle ao fim da **sequência de comandos a ser repetida**. O incremento ou decremento pode ser de qualquer valor, conforme a solução desejada.

FUNCIONAMENTO DO COMANDO FOR

Essa estrutura de repetição usa uma variável que controla cada vez que o bloco ou sequência de comandos será repetido. Chamamos de laço cada ciclo de repetição da sequência de comandos.

1

Essa variável de controle recebe o valor inicial, definido na inicialização.

2

O valor da variável de controle é comparado com a condição, que define o fim da repetição.

3

Se a condição for verdadeira:

Primeiro passo: o bloco, ou sequência de comandos, a ser repetido é executado;

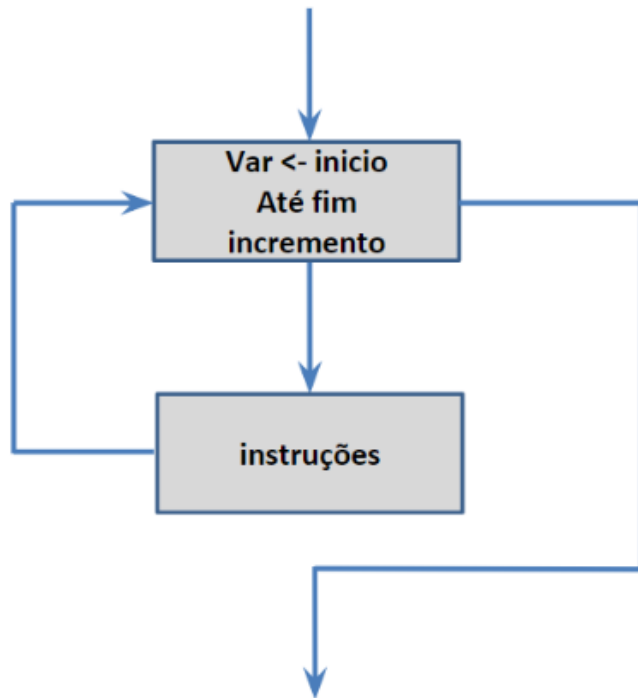
Segundo passo: o valor da variável de controle é alterado (incrementado ou decrementado), conforme o problema, em incremento_decremento;

Terceiro passo: volta-se ao segundo passo.

4

Se a condição for falsa, a sequência de comandos a ser repetida é interrompida e o controle do código é passado ao comando após repetição.

O fluxograma a seguir ilustra o funcionamento da sintaxe geral do comando FOR:



O exemplo a seguir demonstra o uso dessa estrutura de repetição em Portugol e na linguagem C:

// Código em Portugol

```
PARA (cont=1;cont<=10;cont=cont+1)
{
  Sequência de comandos a ser repetida
}
```

ONDE:

Variável de controle: cont, do tipo inteiro

Inicialização: cont=1

Condição: cont<=10

Incremento_decremento: cont=cont+1

// Código em Linguagem C

```
FOR (cont=1;cont<=10;cont=cont+1)
{
Sequência de comandos a ser repetida
}
```

ONDE:

Variável de controle: cont, do tipo inteiro

Inicialização: cont=1

Condição: cont<=10

Incremento_decremento: cont=cont+1

Veja algumas situações em que podemos aplicar esse comando:

MOSTRAR OS 10 PRIMEIROS NÚMEROS INTEIROS E POSITIVOS EM ORDEM CRESCENTE

// Código em Portugol

```
PARA (cont=1;cont<=10;cont=cont+1)
{
escreva (cont, "\n")
}
```

// Código em Linguagem C

```
FOR (cont=1;cont<=10;cont=cont+1)
{
printf ("%d\n",cont)
}
```

MOSTRAR OS 10 PRIMEIROS NÚMEROS INTEIROS E POSITIVOS EM ORDEM DECRESCENTE

// Código em Portugol

```
PARA (cont=10;cont>=1;cont=cont-1)
{
    escreva (cont, "\n")
}
```

// Código em Linguagem C

```
FOR (cont=10;cont>=1;cont=cont-1)
{
    printf ("%d\n",cont)
}
```

MOSTRAR OS NÚMEROS PARES ENTRE 1 E 10 (INCLUSIVE)

// Código em Portugol


```
PARA (cont=2;cont<=10;cont=cont+2)
{
    escreva (cont, "\n")
}
```

// Código em Linguagem C

```
FOR (cont=2;cont<=10;cont=cont+2)
{
    printf ("%d\n",cont)
}
```

MOSTRAR TODAS AS DEZENAS ENTRE 0 E 100 (INCLUSIVE), EM ORDEM CRESCENTE

// Código em Portugol

 PARA (cont=10;cont<=100;cont=cont+10)

```
{
escreva (cont, "\n")
}
```

// Código em Linguagem C

```
FOR (cont=10;cont<=100;cont=cont+10)
{
printf ("%d\n",cont)
}
```

MOSTRAR TODAS AS CENTENAS ENTRE 0 E 1000 (INCLUSIVE), EM ORDEM DECRESCENTE

// Código em Portugol

```
PARA (cont=1000;cont>=100;cont=cont-100)
{
escreva (cont, "\n")
}
```

// Código em Linguagem C

```
FOR (cont=1000;cont>=100;cont=cont-100)
{
printf ("%d\n",cont)
}
```

MOSTRAR A SOMA DOS NÚMEROS INTEIROS E POSITIVOS ENTRE 1 E 10

// Código em Portugol

```
Soma=0
PARA (cont=1;cont<=10;cont=cont+1)
{
Soma=soma+cont
}
```

```
escreva (cont, "\n")
```

```
// Código em Linguagem C
```

```
Soma=0;
FOR (cont=1;cont<=10;cont=cont+1)
{
Soma=soma+cont;
}
printf ("A soma dos números entre 1 e 10 =%d\n",soma);
```

MOSTRAR A MÉDIA ARITMÉTICA DOS NÚMEROS INTEIROS E POSITIVOS ENTRE 1 E 10

```
// Código em Portugol
```

```
Soma=0
PARA (cont=1;cont<=10;cont++)
{
Soma=soma+cont
}
Média=soma/10
escreva (media, "\n")
```

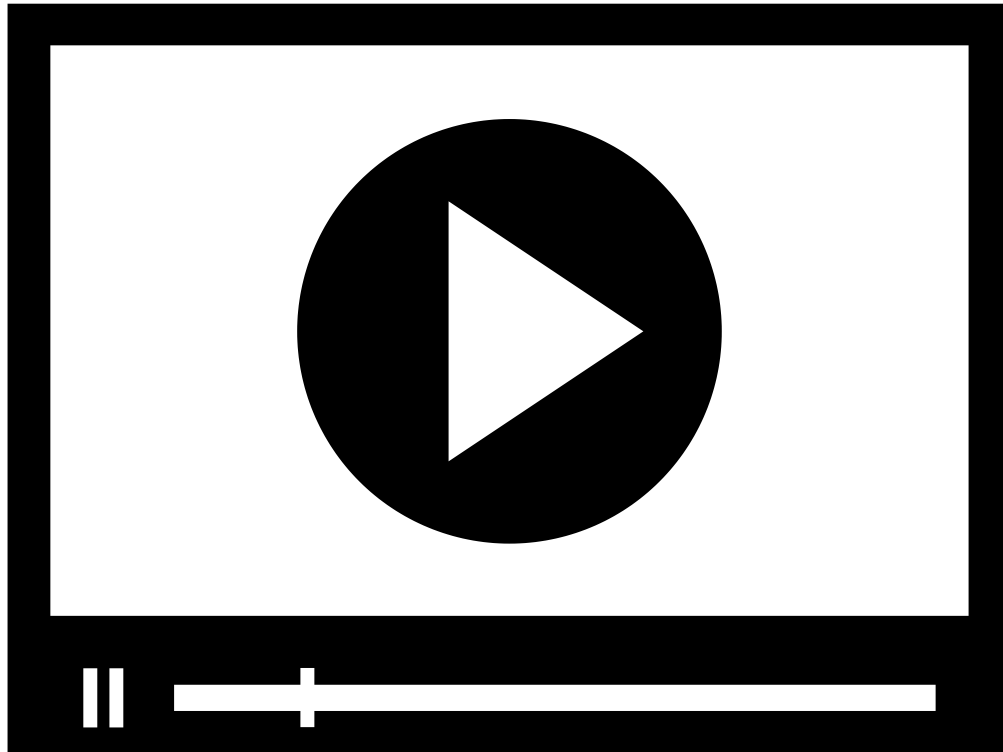
```
// Código em Linguagem C
```

```
Soma=0;
FOR (cont=1;cont<=10;cont++)
{
soma=soma+cont;
}
media=soma/10;
printf ("A média dos números entre 1 e 10= %f\n", media);
```

TEORIA NA PRÁTICA

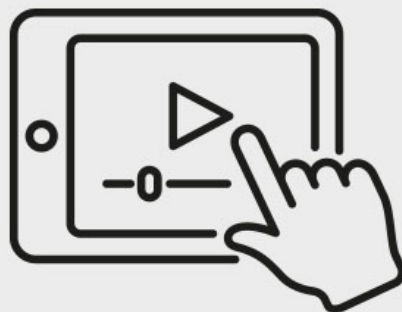
Selecionamos alguns problemas para você resolver usando algoritmos em Portugol e na linguagem C. Vamos ver?

1º problema: desenvolva um programa que leia um número e o mostre 20 vezes.



Neste vídeo, iremos resolver o primeiro problema juntos e teremos a ajuda de Marcelo Vasques, mestre em Computação Aplicada e Automação:

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



**AGORA TENTE RESOLVER OS PROBLEMAS ABAIXO
NO AMBIENTE DEV-C++.**

VOCÊ PODERÁ COMPARAR A SUA RESPOSTA NO BOTÃO DE SOLUÇÃO.

2º problema: desenvolva um programa que leia 15 números inteiros e positivos e mostre o maior deles.

LÓGICA:

Precisamos de 3 variáveis do tipo inteiro (int) para armazenar cada número a ser lido, o maior dos números e controlar a repetição.

1

Inicializar a variável maior com zero.

2

Repetir 15 vezes (comando de repetição PARA):

Ler o número (comando de entrada de dados);

Se o número for superior à variável maior, ela recebe o conteúdo do número lido.

3

Exibir o conteúdo da variável maior (comando de exibição de dados).

SOLUÇÃO

// Código em Portugol

programa

{

funcao inicio()

{

inteiro num,cont,maior;

maior=0

para (cont=1;cont<=15;cont=cont+1)

```
{
escreva ("Digite um número: ")
leia (num)
se num> maior
{
maior=num
}
{
escreva ("O maior dos números lidos = ",maior)
}
}
```

// Código em Linguagem C

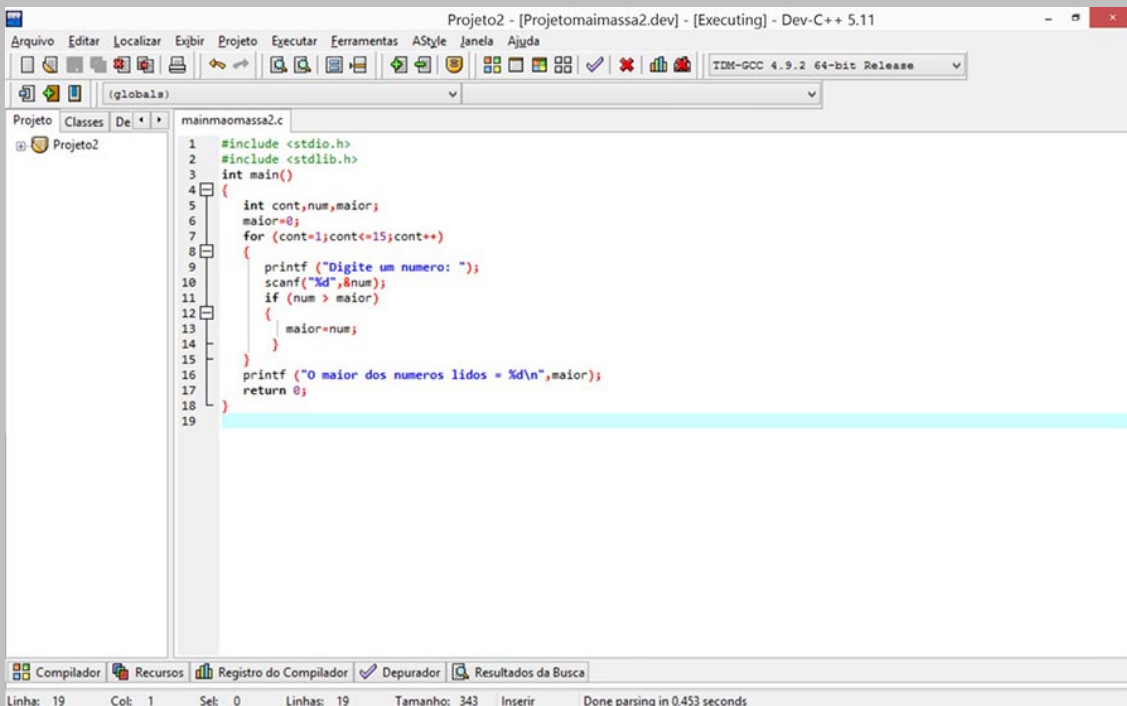
```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int cont,num,maior;
maior=0;
for (cont=1;cont<=15;cont++)
{
printf ("Digite um número: ");
scanf("%d",&num);
if (num > maior)
{
maior=num;
}
}
printf ("O maior dos números lidos = %d\n",maior); return 0;
}
```

E, para finalizarmos este problema:

Veja o **ambiente da ferramenta Dev-C++ com o código na linguagem C**;

Veja a tela com o **resultado da execução no ambiente Dev-C++**.

AMBIENTE DA FERRAMENTA DEV-C++ COM O CÓDIGO NA LINGUAGEM C

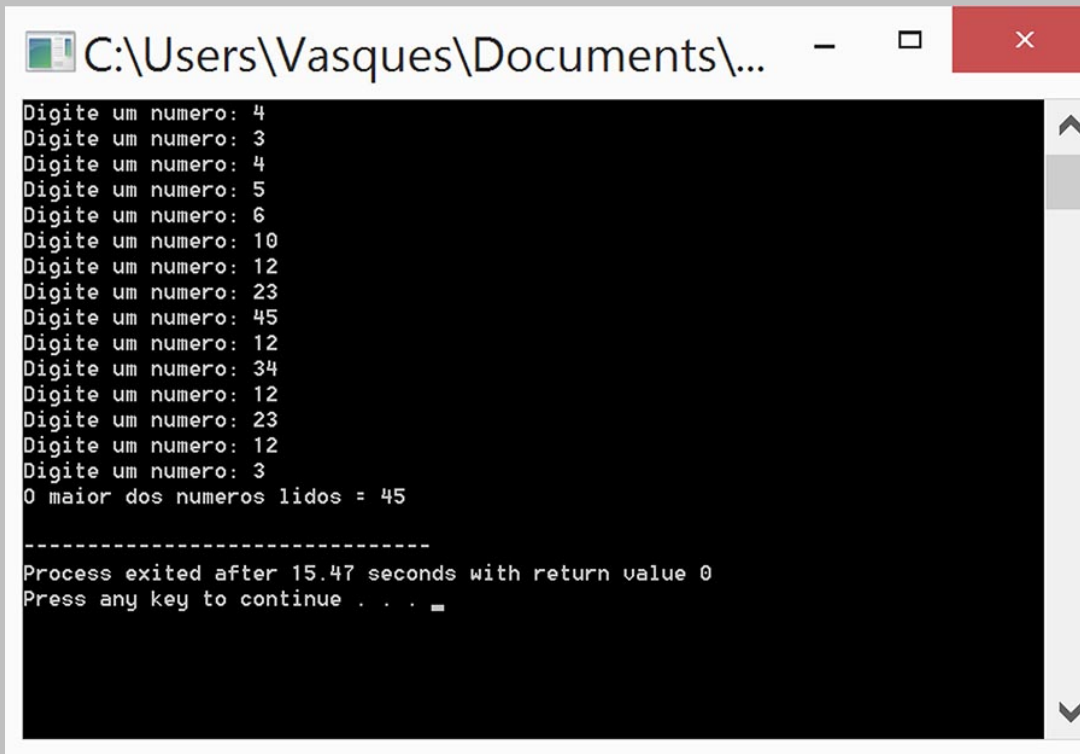


The screenshot shows the Dev-C++ IDE interface. The title bar reads "Projeto2 - [Projetomaimassa2.dev] - [Executing] - Dev-C++ 5.11". The menu bar includes "Arquivo", "Editar", "Localizar", "Exibir", "Projeto", "Executar", "Ferramentas", "AStyle", "Janela", and "Ajuda". The toolbar contains various icons for file operations, execution, and debugging. The "Projeto" pane on the left shows a project named "Projeto2". The main editor displays the file "mainmaomassa2.c" with the following C code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main()
4 {
5     int cont,num,maior;
6     maior=0;
7     for (cont=1;cont<=15;cont++)
8     {
9         printf ("Digite um numero: ");
10        scanf ("%d",&num);
11        if (num > maior)
12        {
13            maior=num;
14        }
15    }
16    printf ("O maior dos numeros lidos = %d\n",maior);
17    return 0;
18 }
19
```

The status bar at the bottom shows "Linha: 19", "Col: 1", "Sel: 0", "Linhas: 19", "Tamanho: 343", "Inserir", and "Done parsing in 0.453 seconds".

RESULTADO DA EXECUÇÃO NO AMBIENTE DEV-C++



```
C:\Users\Vasques\Documents\...
Digite um numero: 4
Digite um numero: 3
Digite um numero: 4
Digite um numero: 5
Digite um numero: 6
Digite um numero: 10
Digite um numero: 12
Digite um numero: 23
Digite um numero: 45
Digite um numero: 12
Digite um numero: 34
Digite um numero: 12
Digite um numero: 23
Digite um numero: 12
Digite um numero: 3
0 maior dos numeros lidos = 45

-----
Process exited after 15.47 seconds with return value 0
Press any key to continue . . .
```

3º problema: desenvolva um programa que leia o salário de 10 funcionários de uma empresa, calcule e mostre o maior salário e a média salarial da empresa.

LÓGICA:

Precisamos de 4 variáveis do tipo real (float), para armazenar cada salário a ser lido, o maior salário, a soma salarial — para calcular a média — e média salarial, além de uma variável inteira (int) para controlar a repetição.

1

Inicializar com zero as variáveis: maior e soma.

2

Repetir 10 vezes (comando de repetição PARA):

Ler o salário do funcionário (comando de entrada de dados);

Se o salário for superior à variável maior, ela recebe o conteúdo do salário lido;

Acumular a soma dos salários na variável soma.

3

Calcular a média salarial, dividindo a soma dos salários por 10 (total de funcionários).

4

Exibir o conteúdo das variáveis maior e média (comando de exibição de dados).

SOLUÇÃO

// Código em Portugol

```
programa
{
funcao inicio()
{
inteiro cont
real salario,media,soma,maior;
maior=0 soma=0;
para (cont=1;cont<=10;cont++)
{
escreva ("Digite o salário do funcionário: ");
leia (salario);
soma=soma+salario;
se (salario > maior)
{
maior=salario;
}
}
media=soma/10;
escreva ("O maior salário da empresa e = ",maior);
escreva ("A média salarial da empresa e = ",media);
}
}
```

// Código em Linguagem C

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

```
#include <stdlib.h>

int main()
{
    int cont;

    float salario,media,soma,maior;
    maior=0; soma=0;
    for (cont=1;cont<=10;cont++)
    {
        printf ("Digite o salário do funcionário: ");
        scanf("%f",&salario);
        soma=soma+salario;
        if (salario > maior)
        {
            maior=salario;
        }
    }
    media=soma/10;
    printf ("O maior salário da empresa e = %.2f \n",maior);
    printf ("A média salarial da empresa e = %.2f \n",media);
    return 0;
}
```

4º problema: desenvolva um programa que leia 3 notas de 40 alunos, calcule e mostre a média aritmética e a situação de aprovação de cada um deles. Lembre-se que apenas a média igual ou acima de 7 aprova o aluno.

LÓGICA:

Precisamos de 3 variáveis do tipo real (float) para armazenar as notas de cada aluno, uma variável real para armazenar a média das notas e uma variável inteira (int) para controlar a repetição.

1

Repetir 40 vezes (comando de repetição PARA):

Ler *nota1* , *nota2* e *nota3* de cada aluno (comando de entrada de dados):

Calcular a média do aluno: $(nota1+nota2+nota3) / 3$;

Se a média do aluno for ≥ 7 , exibir *aluno aprovado* e sua média; senão, exibir *aluno reprovado* e sua média.

SOLUÇÃO

// Código em Portugol

```
programa
{
funcao inicio()
{
real nota1,nota2,nota3,media
inteiro contalunos
para (contalunos=1;contalunos<=40;contalunos++)
{
escreva("Entre com a nota 1 do aluno: ")
leia (nota1)
escreva("Entre com a nota 2 do aluno: ")
leia (nota2)
escreva("Entre com a nota 3 do aluno: ")
leia (nota3)
media=(nota1+nota2+nota3)/3
se (media>=7)
{
escreva("Aluno APROVADO com média :",media)
}
senao
{
printf("Aluno REPROVADO com média : ",media)
}
}
```

```
}
```

```
}
```

// Código em Linguagem C

```
#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",&a1);
```

```
printf("Entre com a nota 2 do aluno: ");
```

```
scanf("%f",&a2);
```

```
printf("Entre com a nota 3 do aluno: ");
```

```
scanf("%f",&a3);
```

```
media=(nota1+nota2+nota3)/3
```

```
if (media>=7)
```

```
{
```

```
printf("Aluno APROVADO com média : %.2f",media);
```

```
}
```

```
else
```

```
{
```

```
printf("Aluno REPROVADO com média : %.2f",media);
```

```
}
```

```
}
```

```
return 0;
```

```
}
```

5º problema: desenvolva um programa que leia, inicialmente, a porcentagem de reajuste dos salários dos funcionários de uma empresa. Na sequência, deve ler o salário de cada um dos 50 funcionários, calcular e mostrar o novo salário reajustado, aplicando a porcentagem de ajuste sobre os respectivos salários atuais. Ao final, o maior salário reajustado da empresa deve ser apresentado na tela.

LÓGICA:

Precisamos de 4 variáveis do tipo real (float) para armazenar a porcentagem de reajuste, cada salário a ser lido, cada salário a ser reajustado e o maior salário reajustado, além de uma variável inteira (int) para controlar a repetição.

1

Ler a porcentagem de reajuste (comando de entrada de dados).

2

Inicializar com zero a variável *maiorsal* .

3

Repetir 40 vezes (comando de repetição PARA):

Ler o salário do funcionário (comando de entrada de dados);

Calcular o salário reajustado, aplicando a porcentagem de aumento lido;

Se *salarioreajustado* > *maiorsalario*

maiorsalario = *salarioreajustado*

4

Exibir o conteúdo da variável *maiorsalario* .

SOLUÇÃO

// Código em Portugol

programa

{

funcao inicio()

{

real percreaj, salario, salarioreaj, maiorsal

inteiro cont

maiorsal=0

```

escreva("Percentual de reajuste salarial: ")
leia (percreaj);
para (cont=1;cont<=50;cont++)
{
    escreva("Informe o salário do funcionário: ");
    leia (salario)
    salarioreaj = salario+ (salario*percreaj/100)
    escreva("O salário reajustado e : ",salarioreaj)
    se (salarioreaj>maiorsal)
        maiorsal=salarioreaj
}
escreva("O maior salário reajustado e : ",maiorsal);
}
}

```

// Código em Linguagem C

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    float percreaj,salario,salarioreaj,maiorsal;
    int cont;
    maiorsal=0;
    printf("Percentual de reajuste salarial:");
    scanf ("%f",&percreaj);
    for (cont=1;cont<=50;cont++)
    {
        printf("Informe o salário do funcionário:");
        scanf("%f",&salario);
        salarioreaj=salario+(salario*percreaj/100);
        printf("O salário reajustado e : %.2f \n\n",salarioreaj);
        if (salarioreaj>maiorsal)
            maiorsal=salarioreaj;
    }
    printf("O maior salário reajustado e : %.2f",maiorsal);
}

```



```
return 0;  
}
```

6º problema: desenvolva um programa que leia um número N e, em seguida, uma lista de N números inteiros. Este programa também deve calcular e mostrar a soma dos números pares e dos números ímpares da lista.

LÓGICA:

Precisamos de 4 variáveis inteiras (int) para armazenar o número N, cada número da lista de N números, a soma dos pares e a soma dos ímpares, além da variável para controle da repetição da leitura e processamento dos N números.

1

Ler o número N (comando de entrada de dados).

2

Inicializar as variáveis contadoras: *somapar* e *somaimpar* .

3

Repetir N vezes:

Ler o número da lista;

Se o resto da divisão do número da lista por 2 = 0;

Somapar = *somapar* + número da lista

Senão *somaimpar* = *somaimpar* + número da lista

4

Exibir o conteúdo das variáveis *somapar* e *somaimpar* .

SOLUÇÃO

// Código em Portugol

programa

```

{
funcao inicio()
{
inteiro cont, n, num, somapar, somaimpar
somapar=0
somaimpar=0
escreva("Digite a quantidade de números da lista: "); leia(n);
para (cont=1; cont<=n; cont++)
{
escreva ("Digite um número: ");
leia(num);
se (num%2==0)
somapar=somapar+num;
senao
somaimpar=somaimpar+num;
}
escreva ("A soma dos números pares =",somapar);
escreva ("A soma dos números ímpares =",somaimpar);
}
}

```

// Código em Linguagem C

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
int cont,n,num,somapar,somaimpar;
somapar=0;
somaimpar=0;
printf ("Digite a quantidade de números da lista: ");
scanf("%d",&n);
for (cont=1;cont<=n;cont++)
{
printf ("Digite um número: ");
scanf("%d",&num);

```

```
if (num%2==0)
somapar=somapar+num;
else
somaimpar=somaimpar+num;
}
printf ("A soma dos números pares = %d\n",somapar);
printf ("A soma dos números ímpares = %d\n",somaimpar);
return 0;
}
```

E, para finalizarmos este problema:

Veja a imagem do **resultado da execução desse programa no Dev-C++**.

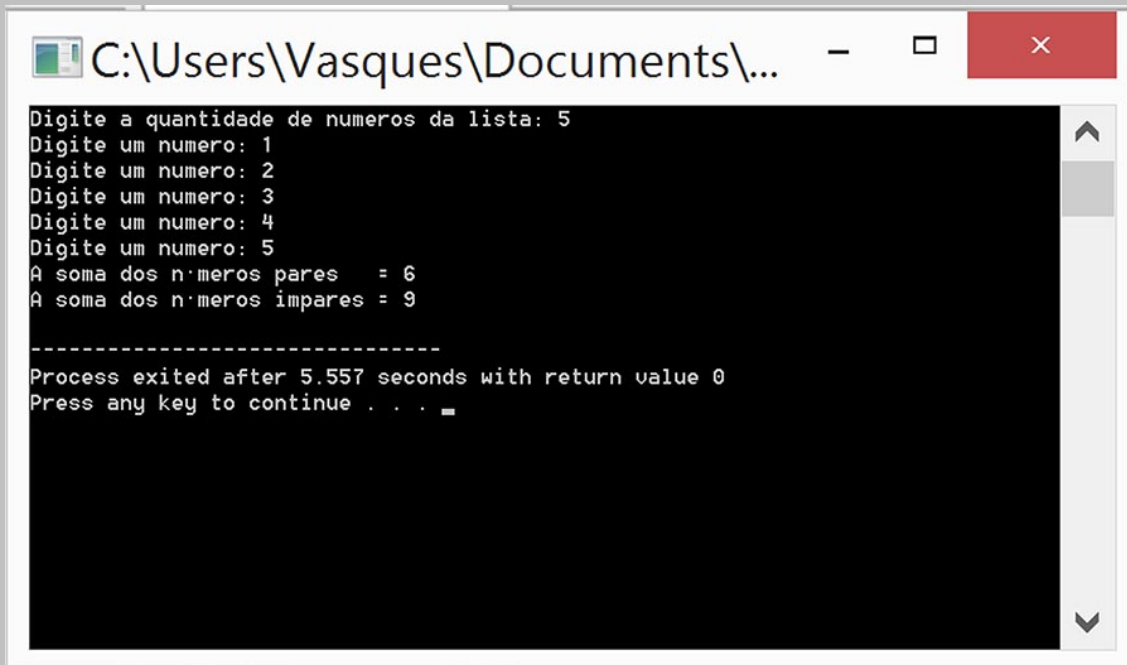
RESULTADO DA EXECUÇÃO NO AMBIENTE DEV-C++

O primeiro dado é a quantidade de números da lista: 5

Na sequência, os 5 números: 1 2 3 4 5

A soma dos pares: $2 + 4 = 6$

A soma dos ímpares: $1 + 3 + 5 = 9$



```
C:\Users\Vasques\Documents\...
Digite a quantidade de numeros da lista: 5
Digite um numero: 1
Digite um numero: 2
Digite um numero: 3
Digite um numero: 4
Digite um numero: 5
A soma dos n'meros pares = 6
A soma dos n'meros impares = 9
-----
Process exited after 5.557 seconds with return value 0
Press any key to continue . . .
```

VEM QUE EU TE EXPLICO!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Motivação para uso do comando FOR

Funcionamento do comando FOR

VERIFICANDO O APRENDIZADO

1. ASSINALE A OPÇÃO QUE APRESENTA CORRETAMENTE A ESTRUTURA DO COMANDO FOR, PARA MOSTRAR OS NÚMEROS PARES DE 2 A 2002 (INCLUSIVE), EM ORDEM DECRESCENTE:

A) for cont=2002, cont>=2, cont=cont-2

☐ **B)** for (cont=2; cont<=2002; cont=cont+2)

C) for (cont=2002; cont>=2; cont=cont-2)

D) for (cont=2000; cont>2; cont=cont-2)

2. ASSINALE A OPÇÃO QUE APRESENTA O TRECHO DE CÓDIGO CORRETO EM PORTUGOL PARA MOSTRAR A SOMA DOS NÚMEROS COMPREENDIDOS ENTRE 1 E 121 (INCLUSIVE):

A) soma=0

para (cont=1; cont<=121; cont++)

{

soma = soma + cont

{

mostre(soma)

B) para (cont=121; cont>=1; cont--)

{

soma = soma + cont

{

mostre(soma)

C) para (cont=1; cont<=121; cont++)

{

soma = soma + cont

{

mostre(soma)

D) soma=0

para (cont=1; cont<=121; cont--)

{

soma = soma + cont

{

mostre(soma)

GABARITO

1. Assinale a opção que apresenta corretamente a estrutura do comando FOR, para mostrar os números pares de 2 a 2002 (inclusive), em ordem decrescente:

A alternativa "C " está correta.

Como é em ordem decrescente, o valor inicial da variável que controla a repetição é **2002**; (**cont=2002**). A condição é (**cont>=2**) enquanto cont for maior ou igual a 2, vai processar. O decremento, já que vamos começar com valor maior que a condição, será de 2, pois estamos processando números pares. Assim o correto é: **for (cont=2002;cont>=2;cont=cont-2)**.

2. Assinale a opção que apresenta o trecho de código correto em Portugol para mostrar a soma dos números compreendidos entre 1 e 121 (inclusive):

A alternativa "A " está correta.

A **alternativa B** está errada, pois não inicializa a variável soma e decrementa a variável de controle que foi iniciada com 1; a **alternativa C** está errada, pois não inicializa a variável soma e incrementa a variável de controle que foi iniciada com 121; já a **alternativa D** está errada, pois decrementa a variável de controle que foi inicializada com 1.

MÓDULO 2

⦿ Identificar conceitos e assertivas relacionados aos 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 Portugol) não é a mais apropriada, como veremos a seguir.

1º Exemplo: desenvolver 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 Portugol (pseudocódigo), este comando é o ENQUANTO.
- Na linguagem C, este comando é o WHILE.

DEMONSTRAÇÃO

Vejamos a sintaxe em Portugol e em Linguagem C:

// Código em Portugol

//Sintaxe geral do comando ENQUANTO

ENQUANTO (condição)

{

Sequência de comandos a ser repetida

}

// comando após a repetição

// Código em Linguagem C

```
//Sintaxe geral do comando WHILE
```

```
WHILE (condição)
```

```
{
```

```
Sequência de comandos a ser repetida
```

```
}
```

```
// comando após a repetição
```

FUNCIONAMENTO DO COMANDO WHILE

O comando 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 WHILE (acompanhe pela sintaxe geral demonstrada acima):

1

A condição é testada;

Caso condição = *VERDADEIRA* :

Executar a sequência de comandos a ser repetida;

Voltar ao teste da condição (primeira ação).

Caso condição = *FALSA* :

Ir para comando após repetição.

O fluxograma representa a lógica a ser seguida pelo comando de repetição com teste no início:

Podemos, então, concluir que:

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.

Uma vez que a condição seja verdadeira, a sequência de comandos a ser repetida é executada.

2

3

Ao final de cada sequência de comandos a ser repetida, a condição é novamente testada.

A sequência de comandos deixa de ser executada tão logo a condição seja falsa. A condição terá que ser FALSA, em algum momento, pois caso contrário a sequência de comandos será executada infinitamente, situação que chamamos de loop.

4

Vamos resolver, passo a passo, o problema motivador para o comando WHILE:

EXEMPLO: DESENVOLVER UM PROGRAMA QUE LEIA UMA SEQUÊNCIA DE NÚMEROS INTEIROS TERMINADA EM ZERO E MOSTRE CADA NÚMERO LIDO (EXCETO O ZERO).

Vamos construir a lógica do programa em cinco passos:

PRECISAREMOS DE UMA VARIÁVEL DO TIPO INTEIRO, QUE CHAMAREMOS DE *NUM*

// Código em Portugol

inteiro num

// Código em Linguagem 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)

// Código em Portugol

enquanto (num!=0)

// Código em Linguagem 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:

// Código em Portugol

inteiro num

leia(num)

enquanto (num!=0)

// Código em Linguagem C

int num;

scanf("%d",&num);

while (num!=0)

DENTRO DA REPETIÇÃO, VAMOS ESTABELEECER OS COMANDOS DO PROCESSAMENTO SOLICITADO NO

ENUNCIADO, QUE, NESSE CASO, É APENAS MOSTRAR CADA NÚMERO LIDO

// Código em Portugol

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

// Código em Linguagem 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 VAI ACONTECER?

Vai ficar exibindo 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).

// Código em Portugol

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

```
leia (num)
```

```
}
```

```
// Código em Linguagem 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* :

```
// Código em Portugol
```

```
inteiro num
```

```
leia(num)
```

```
escreva ("Digite um número: ")
```

```
enquanto (num!=0)
```

```
{
```

```
escreva (num)
```

```
escreva ("Digite um número: ")
```

```
leia (num)
```

```
}
```

```
// Código em Linguagem 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:

// Código em Linguagem C

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

SUGESTÃO

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

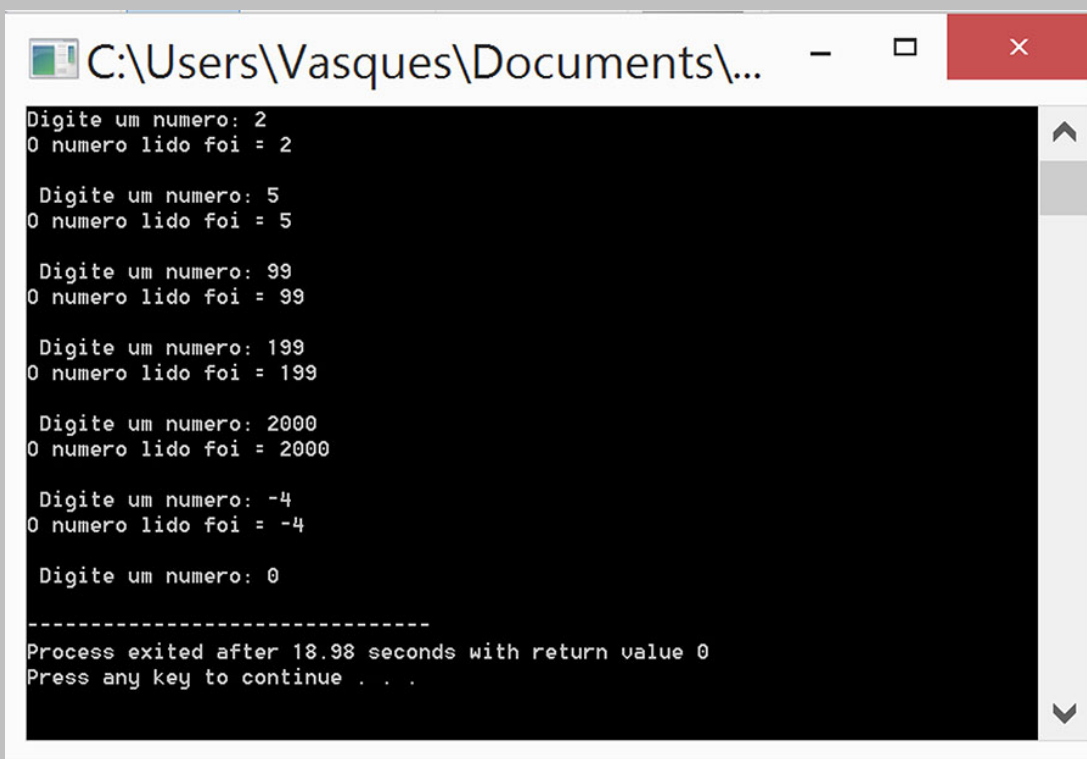
2999

-4

0

Veja como ficará a execução dessa sequência no respectivo **código**.

RESULTADO DA EXECUÇÃO DESSE PROGRAMA NO DEV-C++



```
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 . . .
```

TEORIA NA PRÁTICA

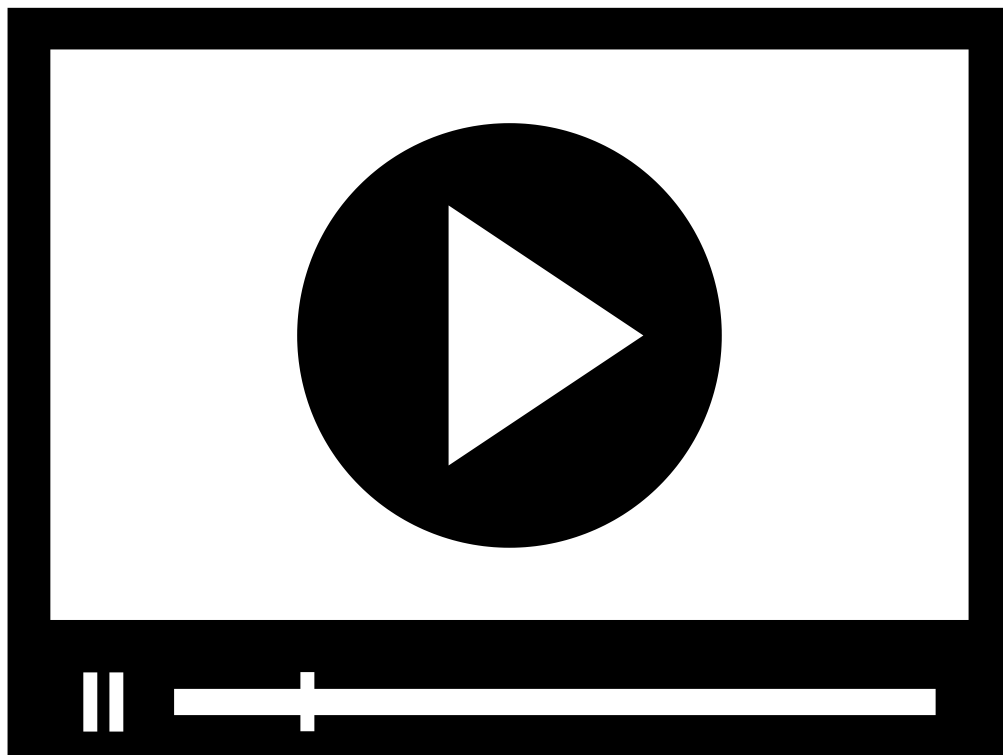
O comando WHILE permite que resolvamos problemas com repetições com número fixo e conhecido de vezes, assim como fizemos com o comando FOR?

RESOLUÇÃO

COMENTÁRIO

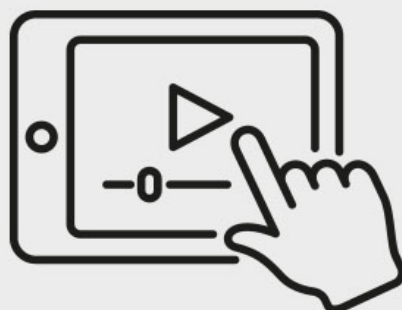
Vejamos um exemplo de um dos programas vistos quando estudamos o comando FOR (PARA).

1º problema: desenvolva um programa que leia 15 números inteiros e positivos e mostre o maior deles.



Neste vídeo, iremos resolver o primeiro problema juntos e teremos a ajuda de Marcelo Vasques, mestre em Computação Aplicada e Automação:

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



AGORA TENTE RESOLVER OS PROBLEMAS ABAIXO NO AMBIENTE DEV-C++.

VOCÊ PODERÁ COMPARAR A SUA RESPOSTA NO BOTÃO DE SOLUÇÃO.

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 que apenas a média igual ou acima de 7 aprova o aluno.

SOLUÇÃO

// Código em Linguagem C - Com o comando FOR

```
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;
}
```

// Código em Linguagem C - Com o comando WHILE


```

int main()
{
float nota1,nota2,nota3,media;
int contalunos=1;
while (contalunos<=40)
{
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);
contalunos++;
}
return 0;
}

```

3º problema: desenvolver 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:

1

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.

// Código em Portugol

caracter letra

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

// Código em Linguagem C

char letra;

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

2

Repetição: usaremos WHILE (ENQUANTO);

Condição: enquanto o conteúdo da variável letra for diferente do ponto (.).

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.

// Código em Portugol

caracter letra

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

{
}

// Código em Linguagem C

char letra;

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

{
}

3

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

// Código em Portugol

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

// Código em Linguagem C

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

4

Agora temos que 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.

// Código em 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)
```

// Código em Linguagem C

```
char letra;
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 a: %d \n",conte);
printf("Total de a: %d \n",conti);
printf("Total de a: %d \n",conto);
printf("Total de a: %d \n",contu);

```

SOLUÇÃO

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");

// 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;
}
```

VEM QUE EU TE EXPLICO!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Motivação para uso do comando com teste no início

Funcionamento do comando com teste no início

VERIFICANDO O APRENDIZADO

1. AVALIE CADA ASSERTIVA A SEGUIR, REFERENTES AOS COMANDOS WHILE E FOR, DA LINGUAGEM C E ASSINALE A ÚNICA CORRETA:

- A) O comando WHILE repete até que a condição seja **falsa**.
- B) O comando WHILE e FOR podem ser aplicados exatamente aos mesmos problemas, não havendo distinção entre eles.
- C) Sendo o teste da condição no início da repetição, a sequência de comandos a ser repetida sempre será executada.
- D) Para problemas onde a quantidade de vezes é conhecida, não podemos usar o comando WHILE, apenas o FOR.

2. CONSIDERE FAZER UM PROGRAMA EM C QUE LEIA UMA SEQUÊNCIA DE NÚMEROS INTEIROS TERMINADA EM 9 OU 99. ASSINALE A OPÇÃO QUE MOSTRA CORRETAMENTE A EXPRESSÃO DA CONDIÇÃO DO COMANDO ENQUANTO PARA RESOLVER O PROBLEMA.

- A) while (num<>9 e num<> 99)
- B) while (num != 9 && num!=99)
- C) while (num =9 !! num = 99)
- D) while (num <= 99)

GABARITO

1. Avalie cada assertiva a seguir, referentes aos comandos WHILE e FOR, da Linguagem C e assinale a única correta:

A alternativa "A " está correta.

A **alternativa B** está incorreta porque existem problemas aplicáveis ao ENQUANTO para os quais o FOR não é adequado; a **alternativa C** é incorreta pois basta que a condição seja falsa já no primeiro laço da repetição para que ela não seja executada; já a **alternativa D** é incorreta porque nesse tipo de problema, ambos os comandos podem ser usados.

2. Considere fazer um programa em C que leia uma sequência de números inteiros terminada em 9 ou 99. Assinale a opção que mostra corretamente a expressão da condição do comando ENQUANTO para resolver o problema.

A alternativa "B " está correta.

ENQUANTO número, FOR diferente de 9 e de 99:

```
while (num != 9 && num!=00)
```

diferente é !=

E = &&

MÓDULO 3

⊙ **Identificar conceitos e assertivas relacionados aos 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), este comando é **FACA... ENQUANTO**.

Na **Linguagem C**, este comando é o DO... WHILE.

DEMONSTRAÇÃO

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

// Código em Portugol

//Sintaxe geral do comando FAÇA... ENQUANTO

faca

{

Sequência de comandos a ser repetida

}

enquanto (condicao)

// comando após a repetição

// Código em Linguagem C

//Sintaxe geral do comando DO... WHILE

do

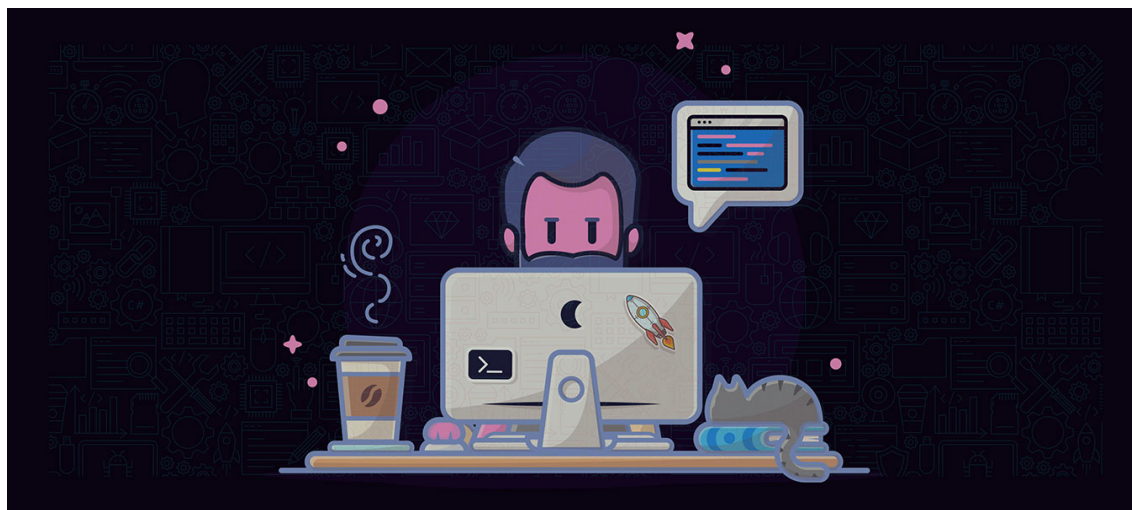
{

Sequência de comandos a ser repetida

}

while (condicao)

// 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):

1

Executa a sequência de comandos a ser repetida.

2

Se a condição é verdadeira, volta ao passo 1.

3

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.

Enquanto a condição for verdadeira, a sequência de comandos a ser repetida é executada.

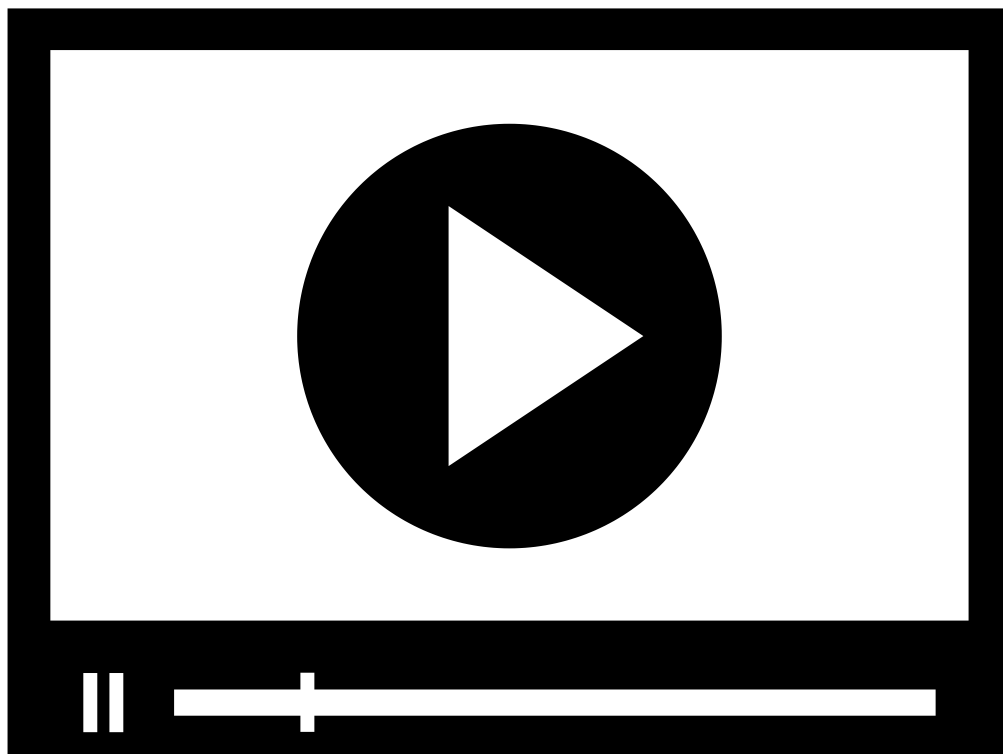
2

3

A sequência de comandos deixa de ser executada, tão logo a condição seja falsa. A condição terá que ser falsa em algum momento, pois caso contrário a sequência de comandos será executada infinitamente, situação essa que chamamos de loop.

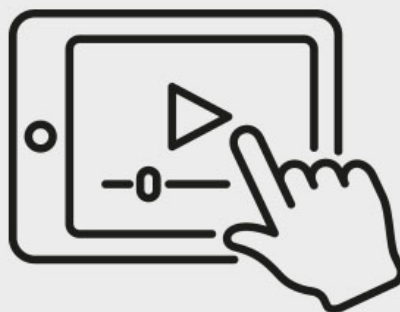
TEORIA NA PRÁTICA

1º problema: leia cinco números inteiros e positivos e mostre o maior deles.



Neste vídeo, iremos resolver o primeiro problema juntos e teremos a ajuda de Marcelo Vasques, mestre em Computação Aplicada e Automação:

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



+ SAIBA MAIS

Qualquer que seja o problema, sempre poderemos optar pela solução com WHILE ou DO... WHILE e teremos apenas essas duas diferenças citadas? Leia a nossa **resposta** completa.

AGORA TENTE RESOLVER OS PROBLEMAS ABAIXO NO AMBIENTE DEV-C++.

VOCÊ PODERÁ COMPARAR A SUA RESPOSTA NO BOTÃO DE SOLUÇÃO.

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 aqui 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.

1

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).

// Código em Portugol

```
inteiro num
```

```
faca
```

```
{
```

```
}
```

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

// Código em Linguagem C

```
int num
```

```
do
```

```
{
```

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

2

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* .

// Código em 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);
```

// Código em Linguagem 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);
```

SOLUÇÃO

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: ");  
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;  
}
```

E, para finalizarmos este problema:

Veja a imagem do **resultado da execução do programa no Dev-C++** para a sequência de entrada: 10, 11, 12, 90, 71, 0.

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 ($10+1 = \text{sucessor}$)

11 é ímpar, mostra 10 ($11-1=\text{antecessor}$)

12 é par, mostra ($12+1 = \text{sucessor}$)

90 é par, mostra ($90+1 = \text{sucessor}$)

71 é ímpar, mostra ($71-1=\text{antecessor}$)

RESULTADO DA EXECUÇÃO DO PROGRAMA NO DEV-C++


```
C:\Users\Vasques\Documents\...
Digite um numero: 10
Sucessor = 11

  Digite um numero: 11
Antecessor = 10

  Digite um numero: 12
Sucessor = 13

  Digite um numero: 90
Sucessor = 91

  Digite um numero: 71
Antecessor = 70

  Digite um numero: 0

-----
Process exited after 12.39 seconds with return value 0
Press any key to continue . . .
```

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

TABELA DE IMPOSTO

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%

Atenção! Para visualizaçãocompleta da tabela utilize a rolagem horizontal

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):

Ler salário bruto;

Calcular o imposto (ninho de lfs, aplicando cada porcentagem conforme salário);

Contabilizar o somatório dos salários brutos, salários líquidos e impostos.

3

Exibir as variáveis totalizadoras.

SOLUÇÃO

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<=15;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 Liquidado: %.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 salário líquido : %.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 salário líquido : %.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++;
    }
    printf ("TOT salário bruto : %.2f \n",totbruto);
    printf ("TOT salário líquido : %.2f \n",totliquido);
    printf ("TOT salário líquido : %.2f \n",totimposto);
```

```
return 0;  
}
```

VEM QUE EU TE EXPLICO!

Os vídeos a seguir abordam os assuntos mais relevantes do conteúdo que você acabou de estudar.

Motivação para uso do comando com teste no final

Funcionamento do comando com teste no final

Comparação entre os 3 comandos de repetição

VERIFICANDO O APRENDIZADO

1. AVALIE AS AFIRMATIVAS A SEGUIR COM RELAÇÃO AOS COMANDOS DO... WHILE, WHILE E FOR DA LINGUAGEM C E ASSINALE A ALTERNATIVA INCORRETA:

A) O comando DO... WHILE repete a sequência de comandos até que a condição seja verdadeira.

B) O comando DO... WHILE, por fazer o teste da condição no final do laço que se repete, sempre vai executar ao menos uma vez a sequência de comandos a ser repetida.

C) No comando DO... WHILE, o teste da condição é feito ao final do laço da repetição e no comando WHILE, o teste é feito no início do laço.

D) Para problemas nos quais conhecemos o número de vezes que a sequência de comandos será repetida, os 3 comandos (FOR, WHILE, DO... WHILE) podem ser usados, sendo o comando FOR o mais adequado.

2. CONSIDERANDO O TRECHO DE CÓDIGO ABAIXO COM O COMANDO FOR, ASSINALE O SEU EQUIVALENTE USANDO O COMANDO DO... WHILE:

FOR (CONT=100; CONT>=1; CONT--)
PRINTF(“CONTADOR=”,CONT);

A) cont=100;
do
{
printf (“contador=”, cont);
cont--;
}
while (cont>=1);

B) cont=100;
do
{
printf (“contador=”, cont);
cont--;
}
while (cont>1);

C) cont=100;
do
{
printf (“contador=”, cont);
}
while (cont>1);

D) cont=100;
do
{
printf (“contador=”, cont);
cont--;
}
while (cont<=1);

GABARITO

1. Avalie as afirmativas a seguir com relação aos comandos DO... WHILE, WHILE e FOR da Linguagem C e assinale a alternativa incorreta:

A alternativa "A " está correta.

O comando DO... WHILE repete enquanto a condição for **verdadeira**, ou até que ela seja **falsa**.

2. Considerando o trecho de código abaixo com o comando FOR, assinale o seu equivalente usando o comando DO... WHILE:

```
for (cont=100; cont>=1; cont--)  
printf("contador=",cont);
```

A alternativa "A " está correta.

Na **alternativa B**, a condição está errada, diferente do trecho do FOR; na **alternativa C**, falta o decremento da variável de controle **cont=**; já na **alternativa D**, a condição com sinal de comparação está invertido e difere do trecho do FOR.

CONCLUSÃO

CONSIDERAÇÕES FINAIS

As três estruturas de repetição podem ser usadas, indiscriminadamente, em todo problema que demandar repetições de instruções no processamento dos dados, ou seja, sempre que for exigido que um bloco de comandos seja repetido. Todavia, existirá sempre a estrutura mais adequada, conforme a especificidade do problema.

Para ouvir um *podcast* sobre o assunto, acesse a versão online deste conteúdo.



Podcast disponível em:

REFERÊNCIAS

ANDRADE, M. C. **Algoritmos**. Rio de Janeiro: SESES, 2014. 128 p.

ASCENCIO, A. F. G.; CAMPOS, E. A. V. **Fundamentos da programação de computadores: Algoritmos, Pascal, C/C++ e Java**. 3. ed. São Paulo: Pearson Education, 2009.

FORBELLONE, A. L. V; EBERSPACHER, H. **Lógica de programação**. 3. ed. São Paulo: Makron Books, 2005.

MANZANO, J.A.N.G.; OLIVEIRA, J. F.; **Algoritmos** - Lógica para Desenvolvimento de Programação de Computadores. São Paulo: Erica (Saraiva), 2016.

SOFFNER, R. **Algoritmos e Programação em Linguagem C**. ed. 1. São Paulo: Saraiva, 2013.

EXPLORE+

Busque o **Online C++ Compiler** do editor GDB para compilar um programa em C, caso não tenha o Dev-C++ instalado

CONTEUDISTA

Marcelo Vasques de Oliveira

 **CURRÍCULO LATTES**