



Estruturas de repetição

Prof. Marcelo Vasques de Oliveira

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 uma 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

Validação de linhas de código

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

Módulo 2

Comandos de repetição com teste no início

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

Módulo 3

Comandos de repetição com teste no final

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



Introdução

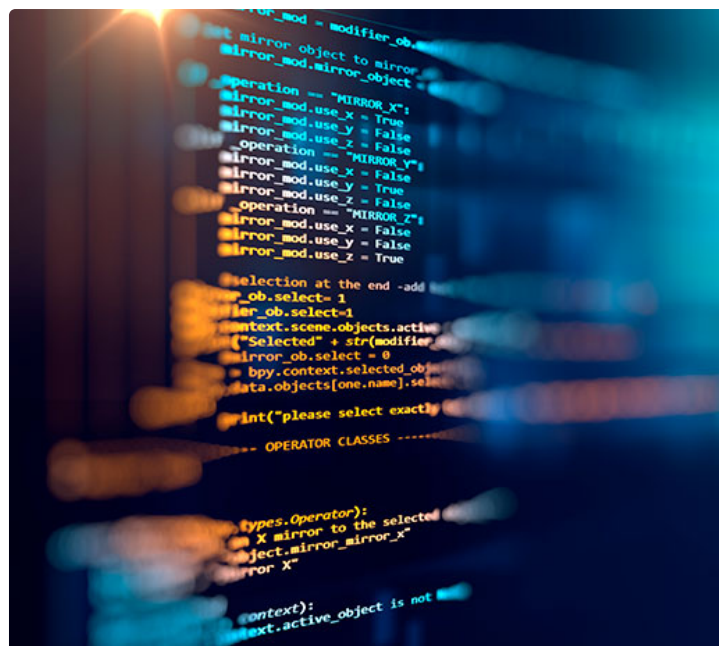
Este estudo refere-se às estruturas (comandos) de repetição, usadas pelas linguagens de programação para permitir que um bloco ou sequência de comandos (instruções) possam ser repetidos, em um programa de computador. Aplicaremos essas estruturas usando a linguagem de programação C e Portugal.

Os comandos ou estruturas de repetição (ou iteração) estão disponíveis em 3 categorias na maioria das linguagens de programação (em C, existem as 3 categorias):

- 1) Comando de repetição com variável de controle (número fixo e conhecido de vezes).
- 2) Comando de repetição com teste de condição no início (pré-teste).
- 3) Comando de repetição com teste de condição no final (pós-teste).

Veremos também a sintaxe e o funcionamento de cada uma dessas categorias de comandos de repetição.

Dependendo do problema e do tipo de repetição necessária, uma das categorias acima tende a se adequar. Mas existem casos em que mais de uma categoria, ou até mesmo as 3, atende à solução. Nessas situações, prevalecerá a preferência do programador, considerando que cada categoria de comando tem vantagens e desvantagens e que o uso inadequado de uma categoria implica complexidade de entendimento do código, o que o deixa mais suscetível a erros.



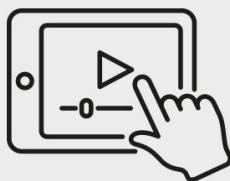
1 - Validação de linhas de código

Ao final deste módulo, você será capaz de 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

Neste vídeo, vamos explorar a estrutura de repetição e as variáveis de controle em C. Aprenderemos como utilizar os laços de repetição, como `for`, `while` e `do-while`, para executar blocos de código de forma iterativa. Além disso, discutiremos o papel das variáveis de controle na iteração eficiente.

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



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.

Portugol



C



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**:

Portugol



C



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:

- Executar o programa 40 vezes.
- 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 igual para 100 ou qualquer outra quantidade de alunos, concorda? Como devemos proceder então?

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 na quantidade de vezes que precisamos.

Nesse 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 igual 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 em 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. Nesse 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:

Portugol



C



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 da 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

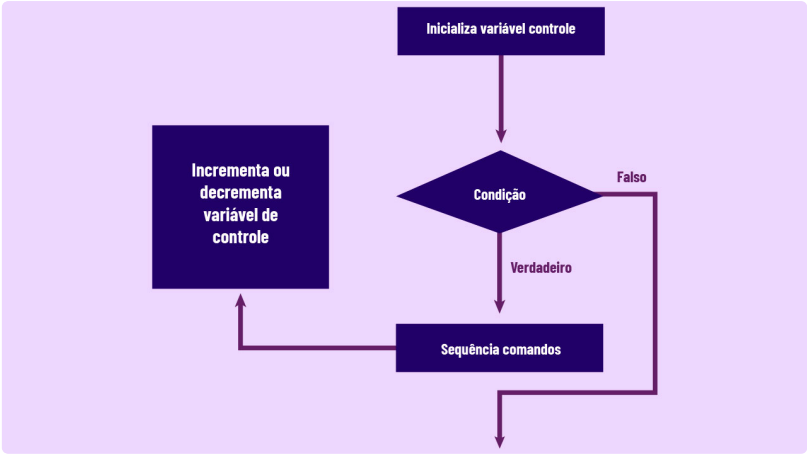
Aumento ou diminuição do valor da variável de controle ao fim da sequência de comandos a ser repetida. O incremento ou decremento pode ser de qualquer valor, conforme a solução desejada.

Funcionamento do comando FOR

Essa estrutura de repetição usa uma variável que controla cada vez que o bloco ou a 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:
 - 3.1. **Primeiro passo:** o bloco, ou sequência de comandos, a ser repetido é executado.
 - 3.2. **Segundo passo:** o valor da variável de controle é alterado (incrementado ou decrementado), conforme o problema, em incremento_decremento.
 - 3.3. **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.

Observe o fluxograma que ilustra o funcionamento da sintaxe geral do comando FOR:



Agora veja o uso dessa estrutura de repetição em Portugol e na linguagem C:

Portugol



C



Em Portugal:

- Variável de controle: cont, do tipo inteiro.
- Inicialização: cont=1.
- Condição: cont<=10.
- Incremento_decremento: cont=cont+1.

Na linguagem C:

- 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 Portugal



C



Mostrar os 10 primeiros números inteiros e positivos em ordem decrescente:

Código Em Portugol



C



Mostrar os números pares entre 1 e 10 (inclusive):

Código Em Portugol



C



Mostrar todas as dezenas entre 0 e 100 (inclusive), em ordem crescente:

Código Em Portugol



C



Mostrar todas as centenas entre 0 e 1000 (inclusive), em ordem decrescente:

Código Em Portugol



C



Mostrar a soma dos números inteiros e positivos entre 1 e 10:

Código Em Portugol



C



Mostrar a média aritmética dos números inteiros e positivos entre 1 e 10:

Código Em Portugol



C



Colocando a teoria em prática

Para praticarmos, selecionamos alguns problemas para você resolver usando algoritmos em Portugol e na linguagem C.

1º problema

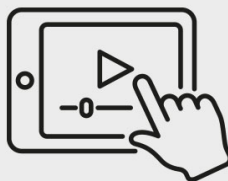
Desenvolva um programa que leia um número e o mostre 20 vezes.



Resolução do primeiro problema

Neste vídeo, iremos resolver o primeiro problema de maneira didática.

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 20 vezes (comando de repetição PARA):
 - 2.1. Ler o número (comando de entrada de dados).
 - 2.2. 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).

Veja a seguir a solução do problema:

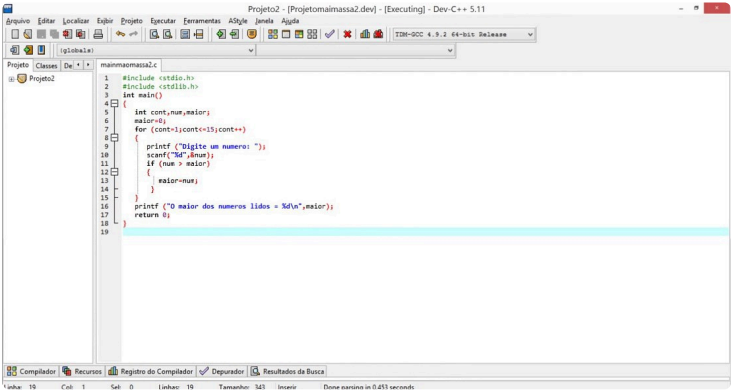
Portugal



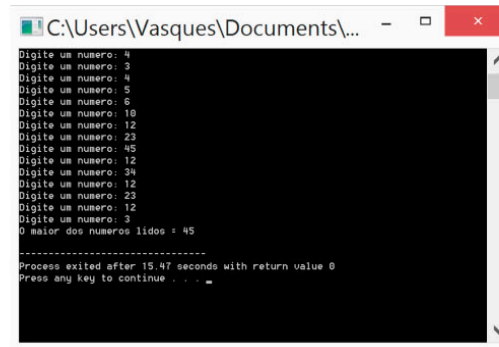
C



Para finalizar esse problema, veja o ambiente da ferramenta Dev-C++ com o código na linguagem C e a tela com o resultado da execução no ambiente Dev-C++.



Ambiente da ferramenta Dev-C++ com o código na linguagem C



```
C:\Users\Vasques\Documents\... - _ x
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
O maior dos numeros lidos = 45
-----
Process exited after 15.47 seconds with return value 0
Press any key to continue . . . _
```

Resultado da execução no ambiente Dev-C++

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 a 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):
 - 2.1. Ler o salário do funcionário (comando de entrada de dados).
 - 2.2. Se o salário for superior à variável maior, ela recebe o conteúdo do salário lido.
 - 2.3. 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).

A seguir vemos a solução:



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 de 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):
 - 1.1. Ler nota1, nota2 e nota3 de cada aluno (comando de entrada de dados).
 - 1.2. Calcular a média do aluno: $(\text{nota1} + \text{nota2} + \text{nota3}) / 3$.
 - 1.3. Se a média do aluno for ≥ 7 , exibir aluno aprovado e sua média; senão, exibir aluno reprovado e sua média.

Portugol



C



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).
4. Ler o salário do funcionário (comando de entrada de dados).
5. Calcular o salário reajustado, aplicando a porcentagem de aumento lido.
6. Se *salarioreajustado* > *maiorsalario*
maiorsalario = *salarioreajustado*.
7. Exibir o conteúdo da variável *maiorsalario*.

Portugol



C



6º problema

Desenvolva um programa que leia um número N e, em seguida, uma lista de N números inteiros. Esse 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:
 - 3.1. Ler o número da lista.
 - 3.2. Se o resto da divisão do número da lista por 2 = 0
 $Somapar = somapar + \text{número da lista}$
Senão $somaimpar = somaimpar + \text{número da lista}$.
4. Exibir o conteúdo das variáveis *somapar* e *somaimpar*.

Portugol



C



Para finalizar esse problema, veja o resultado da execução desse programa no 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: $1 + 3 + 5 = 9$.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 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)
- E for (cont==2000; cont>2; cont=cont-2)

Parabéns! 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).

Questão 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)
```

E

```
soma == 0
para (cont ==1; cont<=121;cont--)
{
    soma = soma+cont
    {
        mostre(soma)
```

Parabéns! 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.



2 - Comandos de repetição com teste no início

Ao final deste módulo, você será capaz de 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 Português) 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



C



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:

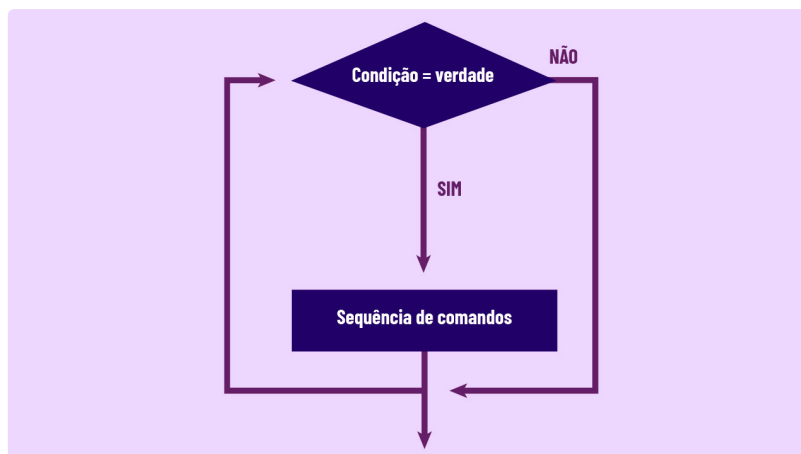
1.1. Executar a sequência de comandos a ser repetida.

1.2. Voltar ao teste da condição (primeira ação).

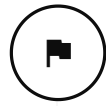
2. Caso condição = FALSA:

2.1. Ir para comando após repetição.

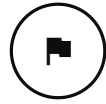
O fluxograma abaixo representa a lógica a ser seguida pelo comando de repetição com teste no início:



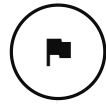
Podemos, desse modo, chegar às seguintes conclusões:



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.



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 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



C



Entenderemos a condição: enquanto o número lido for diferente de 0, vamos mostrá-lo; a condição é: $num \neq 0$ (num diferente de 0)

Portugol



C



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



C



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

Portugol



C



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



C



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



C



Veja o programa completo em linguagem C a seguir:

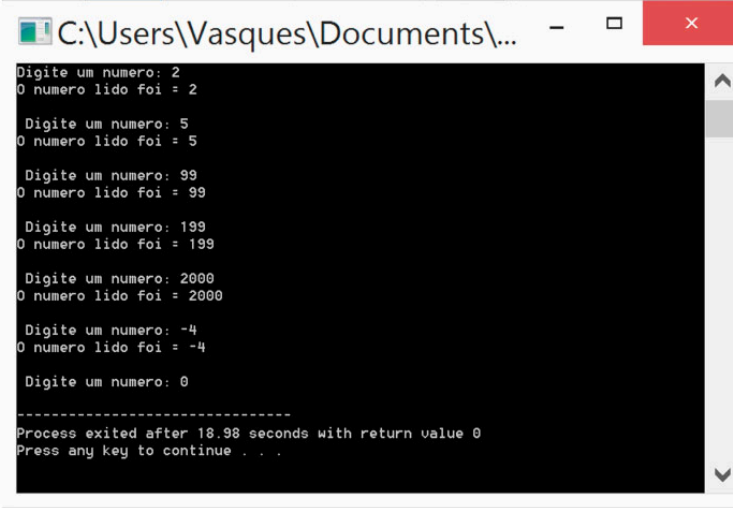
C



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.

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\Users\Vasques\Documents\...' and standard window controls. The command prompt has a black background with white text. It shows a series of prompts 'Digite um numero:' followed by user input and the program's response 'O numero lido foi:'. The inputs are 2, 5, 99, 199, 2000, -4, and 0. After the last input, a separator line '-----' is shown, followed by the message 'Process exited after 18.98 seconds with return value 0' and 'Press any key to continue . . .'.

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

Digite um numero: 5
O numero lido foi: 5

Digite um numero: 99
O numero lido foi: 99

Digite um numero: 199
O numero lido foi: 199

Digite um numero: 2000
O numero lido foi: 2000

Digite um numero: -4
O 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.



Resolução de Problema de estrutura de repetição

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

C



C



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 carácter (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



C



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



C



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



C



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



C



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



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Avalie cada assertiva a seguir referente 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.
- E** O comando WHILE é o substituto natural do comando FOR.

Parabéns! 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.

Questão 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)

E While (num == 99)

Parabéns! A alternativa B está correta.

ENQUANTO número, FOR diferente de 9 e de 99:

```
while (num != 9 && num!=00)
diferente é !=
E = &&
```



3 - Comandos de repetição com teste no final

Ao final deste módulo, você será capaz de 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), 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



C



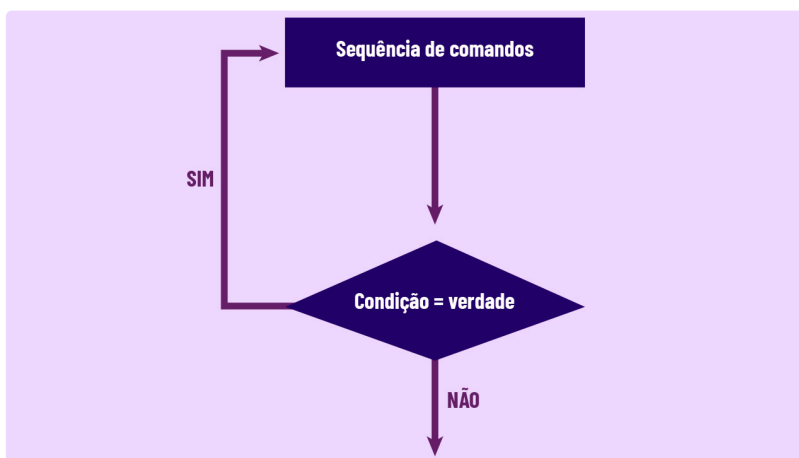
Funcionamento do comando DO...WHILE

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).

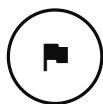
Vejamos o funcionamento do comando DO... WHILE (acompanhe pela sintaxe geral demonstrada anteriormente):

- 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.

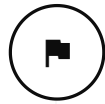
O fluxograma a seguir ilustra o funcionamento do comando de repetição com teste no final:



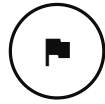
Podemos, então, concluir que:



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.



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

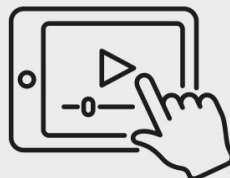
Faça um programa que leia uma sequência de números inteiros, terminada em zero e mostre cada número lido (exceto o zero).



Resolução do problema de comandos de repetição com teste no início

Neste vídeo, iremos resolver o primeiro problema juntos e de forma didática.

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



Qualquer que seja o problema, sempre poderemos optar pela solução com WHILE ou DO... WHILE e teremos apenas essas duas diferenças citadas? Confira aqui a nossa [resposta](#) completa para essa pergunta. Em seguida, tente resolver os problemas abaixo no ambiente Dev-C++.

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 ($10+1 = \text{sucessor}$).
- 11 é ímpar, mostra 10 ($11-1 = \text{antecessor}$).
- 12 é par, mostra 13 ($12+1 = \text{sucessor}$).
- 90 é par, mostra 91 ($90+1 = \text{sucessor}$).
- 71 é ímpar, mostra 70 ($71-1 = \text{antecessor}$).

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



C



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.

Portugal



C

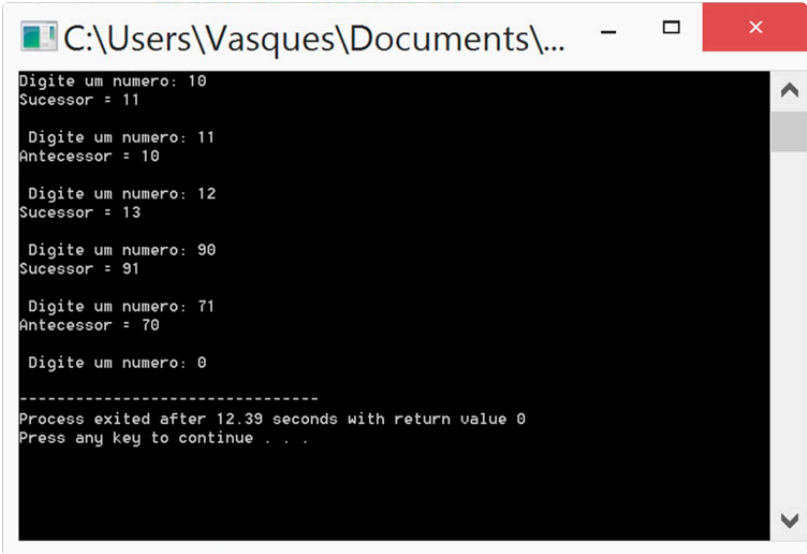


Agora, observe ao lado o código completo da solução usando o DO...
WHILE na linguagem C:

C



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
1	R\$ 0.00	R\$ 999.00
2	R\$ 999.01	R\$ 1.999,00
3	R\$ 1.999.01	R\$ 9.999.00
4	R\$ 9.999,01	R\$ 99.999.00
5	Acima R\$ 99.999,01	---

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):
 - 2.1. Ler salário bruto.
 - 2.2. Calcular o imposto (ninho de ifs, aplicando cada porcentagem conforme salário).
 - 2.3. 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:

C



C



C



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 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.
- E** Os comandos FOR, WHILE e DO...WHILE podem ser intercambiados livremente.

Parabéns! A alternativa A está correta.

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

Questão 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);
```

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



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

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

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

```
cont == 100;
do
{
E printf("contador=", cont);
 cont--;
}
while(cont<=1)
```

Parabéns! 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.

## Considerações finais

As três estruturas de repetição estudadas 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.



### Podcast

Para encerrar, ouça um breve resumo do conteúdo estudado.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.



## Explore +

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

Leia o artigo **Estruturas de repetição: C++**, da plataforma DEVMEDIA.

## 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.



### Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.



Download material

O que você achou do conteúdo?



Relatar problema