#### Estruturas de Repetição

**Introdução**

Programas ou aplicativos que fazem sentido são aqueles que substituem os humanos em atividades repetitivas, portando tediosas. Não faz sentido desenvolver um programa para fazer um cálculo, realizar uma única tarefa. Programas úteis executam vários cálculos, processam as mesmas ações sobre coleções de dados similares.

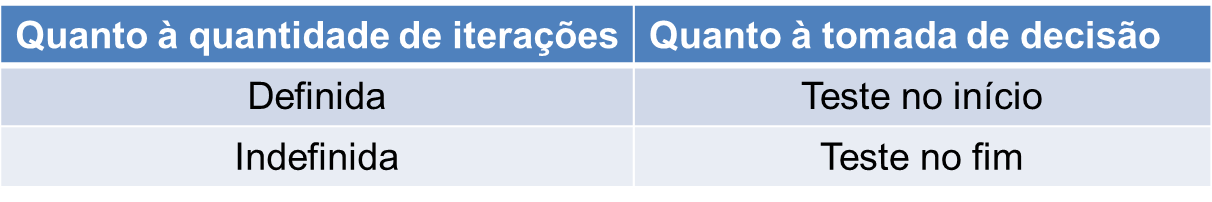
As estruturas de controle de fluxo de execução que permitem este tipo de processamentos são as ***estruturas de repetição***. Toda *estrutura de repetição* possui comportamentos padrão que devem ser observados para sua correta construção. Os elementos básicos de uma repetição são:

1. As ações a serem repetidas, que podem ser categorizadas em:
   1. Ações de processamento;
   2. Ações de controle
2. A decisão de quando encerrar a repetição

Para ilustrar o comportamento de repetições descrito acima, considere o problema de imprimir os números de 1 até 10 na tela do computador, um em cada linha. Este problema pode ser decomposto nas seguintes ações:

1. Iniciar um contador em 1; [*ação de controle*]
2. Enviar o conteúdo do contador para a tela e quebrar linha; [*ação de processamento*]
3. Incrementar o contador; [*ação de controle*]
4. Verificar se contador alcançou 10; [*decisão de encerrar*]

As estruturas de repetição podem ser classificadas segundo dois pontos de vista, o que justifica os diferentes comandos de repetição oferecidos pelas linguagens de programação. Estas classificações são 1) quanto à quantidade de iterações previstas em tempo de construção da repetição e 2) quanto ao teste para tomada de decisão de quanto encerrar a repetição. O quadro a seguir detalha estas duas classificações:



**O comando *for***

O comando *for* é uma estrutura de repetição que implementa somente quantidade definida de iterações e executa teste no inicio.

**Sintaxe**:

**for( inicialização; teste; incremento ) {**

**comandos;**

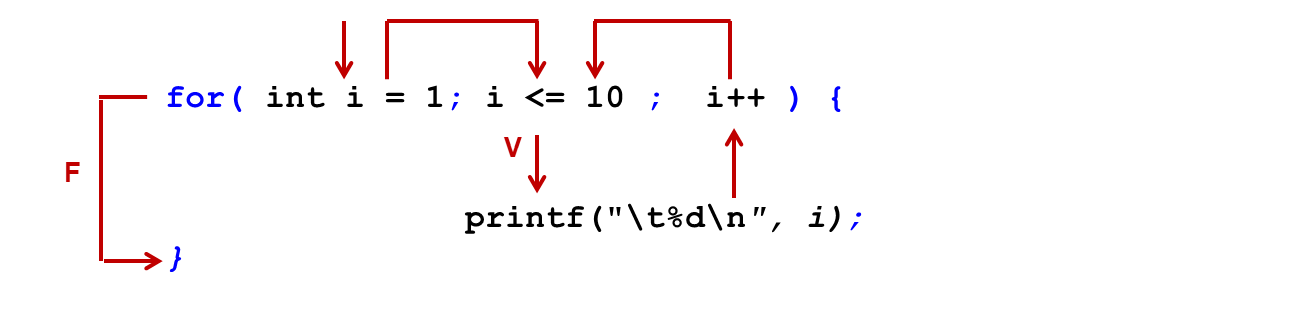
**}**

**Semântica:**

Na ***inicialização***, a *variável de controle* é iniciada, depois o ***teste*** é avaliado e sendo *verdadeiro*, executa o(s) comando(s) internos. Na sequência o ***incremento*** é feito e o teste novamente executado. E assim sucessivamente, até que teste seja avaliado *falso*, quando encerra o laço.

**Exemplo:**

Resolver o problema citado de imprimir os números de 1 até 10 na tela do computador, um em cada linha (as setas indicam a sequência de ações, conforme a semântica do comando ***for***):



**Outros exemplos:**

Listar números inteiros de 20 a 10, decrescente, na mesma linha:

**for(int k = 20; k >= 10; k--) {**

**printf("%3d, ", k);**

**}**

Listar letras maiúsculas de forma crescente e respectivo código *ASCII*, uma em cada linha:

**for(char c = ‘A’; c <= ‘Z’; c++) {**

**printf("\n %c = %d", c, c);**

**}**

Listar múltiplos de 5 entre 0 e 100:

**for(i = 0; i <= 100; i+=5)**

**printf("%3d, ", i);**

Listar anos que tiveram Copa do Mundo, desde 1950 até hoje:

**for(int ano = 1950; ano <= 2019; ano = ano+4)**

**printf("%4d,\n ", ano);**

**Exemplo de um programa completo aplicando comando *for*:**

Ler um número inteiro e calcular o seu fatorial:

**#include<stdio.h>**

**void main (){**

**int n, i;**

**long int fat;**

**printf("Digite um número inteiro positivo: ");**

**scanf("%d", &n);**

**for(fat = 1, i = n; i > 1; i--)**

**fat \*= i;**

**printf("\n Fatorial de %d = %d, ", n, fat);**

**}**

**O comando *while***

O comando *while* é uma estrutura de repetição que implementa quantidade definida ou indefinida de iterações e executa teste no inicio.

**Sintaxe:**

**while( condição ) {**

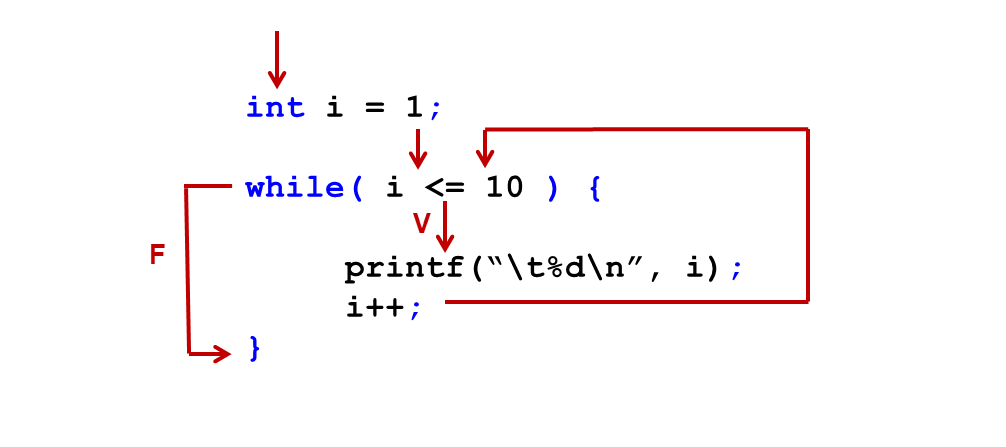
**comandos;**

**}**

**Semântica:**

A ***condição***é avaliada e sendo ***verdadeiro***, executa o(s) ***comando(s)***. Na sequência ***condição***é novamente avaliada e sendo ***verdadeiro***, executa o(s) ***comando(s)***. E assim sucessivamente, até que ***condição*** seja avaliada ***falso***, quando encerra o laço.

**Exemplo:**

****

**Outros exemplos:**

Listar números inteiros de 20 a 10, decrescente, na mesma linha:

**int k = 20;**

**while( k >= 10 ) {**

**printf(“%3d,”, k);**

**k--;**

**}**

Listar letras maiúsculas de forma crescente e respectivo código ASCII, uma em cada linha:

**char c = ‘A’;**

**while( c <= ‘Z’ ) {**

**printf(“\n %c = %d”, c, c);**

**c++;**

**}**

Listar múltiplos de 5 entre 0 e 100:

**i = 0;**

**while(i <= 100) {**

**printf(“%3d,”, i);**

**i += 5;**

**}**

Listar anos que tiveram Copa do Mundo, desde 1950 até hoje:

**int ano = 1950;**

**while( ano <= 2019 ) {**

**printf(“%4d\n,”, ano);**

**ano = ano + 4;**

**}**

**Exemplo de um programa completo aplicando comando *while*:**

Ler um número inteiro e calcular o seu fatorial:

**#include<stdio.h>**

**void main (){**

**int n, i;**

**long int fat = 1;**

**printf(“Digite um número inteiro positivo: “);**

**scanf(“%d”, &n);**

**i = n;**

**while( i > 1){**

**fat \*= i;**

**i--;**

**}**

**printf(“\n Fatorial de %d = %d,”, n, fat);**

**}**

**O comando *do...while***

O comando *do...while* é uma estrutura de repetição que implementa quantidade definida ou indefinida de iterações e provê teste no final.

**Sintaxe:**

**do {**

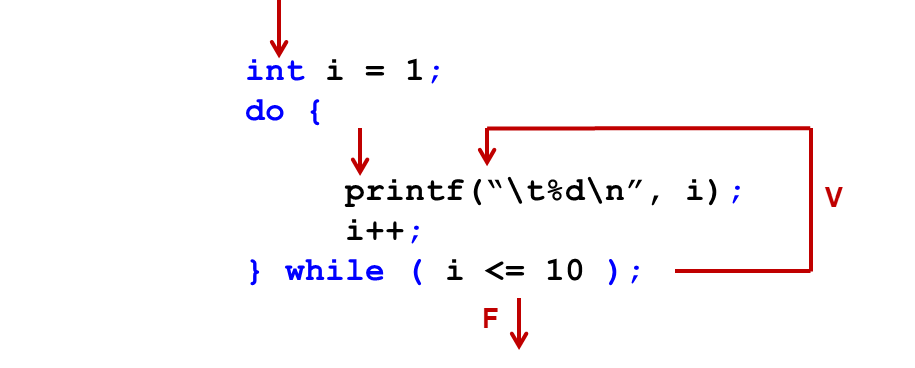
**comandos;**

**} while ( condição );**

**Semântica:**

O(s) ***comando(s)***são executados e ao final ***condição*** é avaliada e sendo ***verdadeiro***, retorna para executar o(s) ***comando(s)*** novamente. E assim sucessivamente, até que ***condição*** seja avaliada ***falso***, quando encerra o laço.

**Exemplo:**

****

**Outros exemplos:**

Listar números inteiros de 20 a 10, decrescente, na mesma linha:

**int k = 20;**

**do {**

**printf(“%3d,”, k);**

**k--;**

**} while( k >= 10 )**

Listar letras maiúsculas de forma crescente e respectivo código ASCII, uma em cada linha:

**char c = ‘A’;**

**do {**

**printf(“\n %c = %d”, c, c);**

**c++;**

**} while( c <= ‘Z’ )**

Listar múltiplos de 5 entre 0 e 100:

**i = 0;**

**do {**

**printf(“%3d,”, i);**

**i += 5;**

**} while( i <= 100 )**

Listar anos que tiveram Copa do Mundo, desde 1950 até hoje:

**int ano = 1950;**

**do {**

**printf(“%4d\n,”, ano);**

**ano = ano + 4;**

**} while( ano <= 2019 )**

**Exemplo de um programa completo aplicando comando *do...while*:**

Ler um número inteiro e calcular o seu fatorial:

**#include<stdio.h>**

**void main (){**

**int n, i;**

**long int fat = 1;**

**printf(“Digite um número inteiro positivo: “);**

**scanf(“%d”, &n);**

**i = n;**

**do {**

**fat \*= i;**

**i--;**

**} while( i > 1)**

**printf(“\n Fatorial de %d = %d,”, n, fat);**

**}**

**Usando estrutura *do...while* para consistir entrada:**

Ler o sexo de uma pessoa e garantir que será lido [ M / F ] :

char sexo;

do {

printf("\n Digite o sexo [ M / F ]: ");

sexo = toupper(getche());

} while( sexo != 'M' && sexo != 'F');

Ler um número inteiro entre 10 e 99 :

int num;

do {

printf("\n Digite um número inteiro de dois dígitos: ");

scanf("%d", &num);

} while( num < 10 || num > 99 );

**Exemplo Geral**

Programa que leia o sexo de dez pessoas, consistindo a entrada, e informe a quantidade de homens e quantidade de mulheres lidos:

#include<stdio.h>

void main (){

int i, qtH = 0, qtM = 0;

char sexo;

for(i = 1; i <= 10; i++){

do{

printf("\n Digite o sexo [ M / F ]: ");

sexo = toupper(getche());

} while( sexo != ‘M’ && sexo != ‘F’);

if (sexo == ‘M’)

qtH++;

else qtM++;

}

printf("\n Quantidade de homens = %d.", qtH);

printf("\n Quantidade de mulheres = %d.", qtM);

}

**Exercícios**

1) Faça o teste de mesa para os seguintes trechos de código:

void main (){

int j;

for(j = 1; j <= 10; j++)

printf(" %d, ", j);

}

-----------------------------------------------------

void main (){

for(int k = 50; k > 10; k -= 10)

printf(" %d, ", k);

}

-----------------------------------------------------

void main (){

for(int x = 1; x >= 10; x++)

printf(" %d, ", x);

}

-----------------------------------------------------

void main (){

for(int x = 1, int y = 4; x <= y; x +=2, y++)

printf(“\n %d, %d “, x, y);

}

-----------------------------------------------------

void main (){

int j = 0;

while(j <= 10){

printf(" %d, ", j);

j += 2;

}

}

-----------------------------------------------------

void main (){

int y = 0;

while(y <= 10){

printf(" %d, ", y);

y -= 1;

}

}

-----------------------------------------------------

void main (){

int i = 10, soma = 0;

do{

printf("\n %d, %d ", i, soma);

soma += i++;

} while(i <= 15);

}

-----------------------------------------------------

void main (){

int x = 0, y = 1;

while(x <= 5){

y \*= ++x;

printf(" %d, %d ", x, y);

}

}

-----------------------------------------------------

void main (){

for(int i = 10; i >= 0; i -= 3)

printf("\n %d ", i);

}

-----------------------------------------------------

1. Faça programa em C aplicando comando ***for*** para atender às seguintes especificações:
   * + 1. Mostrar os múltiplos de 10 entre 0 e 500, de forma decrescente.
       2. Calcular e mostrar a média de 10 números lidos.
       3. Ler a idade de 8 pessoas e mostrar a idade da mais velha.
       4. Ler peso e sexo de 5 pessoas e mostrar a mulher mais magra.
       5. Mostrar os números múltiplos de 3 e 8 entre 500 e 1000.
       6. Ler 5 números e mostrar seu produto.
       7. Mostrar os quadrados e cubos dos números entre 10 e 20.
       8. Mostrar a tabela de conversão de ºCelsius para Farenheit, entre 0 e 100 ºC – fórmula de conversão: F = 9/5C + 32.
     1. Faça um programa em C aplicando comando ***while*** ou ***do...while*** para atender às seguintes especificações:
        1. Calcular e mostrar a soma de vários numeros lidos, até que seja lido o zero.
        2. Ler a idade de várias pessoas, até que seja lido 0, e mostrar a mais nova (0 não deve ser considerado na identificação da pessoa mais nova).
        3. Ler vários numeros inteiros até que seja informado o zero, e mostrar a quantidade de números impares informados.
        4. Ler a idade e o sexo de um grupo de pessoas, até que seja informada idade = 0, e mostrar a maior idade entre os homens e a média de idade das mulheres. Consistir o sexo, aceitando apenas F e M.
        5. Repetidamente ler um número e, se ele for maior ou igual a zero, mostrar sua raiz quadrada, se for negativo, encerrar a execução. Cuidado, pois raiz quadrada de número negativo é indefinida.
        6. Ler uma quantidade qualquer (definida pelo usuário durante a execução) de notas entre 0 e 10 (não permitir notas fora desta faixa) e ao final mostrar a média da turma, a quantidade de notas maior ou igual a 7 e quantidade de notas menor que 5.
        7. Gerar um número aleatório entre 0 e 10 (pesquise e use a função ***rand*** da biblioteca ***stdlib.h***) e depois, repetidamente pede ao usuário para tentar adivinhar este número. A cada tentativa, informa ao usuário se o número sorteado é maior ou menor do que ele informou. No final informar quantas tentativas o usuário gastou para adivinhar.

1. **Desafio**:
2. Faça um programa que leia a altura (quantidade de linhas) e imprima um cone, como mostrado na figura abaixo.

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

**Dica**: observe que há uma relação direta entre a altura e os espaços em branco e quantidade de caracteres de cada linha.

Por exemplo, na figura acima, a altura é 6 (pirâmide com 6 linhas). E os espaços em branco são 5 na 1ª linha, 4 na 2ª, 3 na 3ª, 2 na 4ª, 1 na 5ª e 0 na 6ª linha. Ou seja, em cada linha a quantidade de espaços é dado por [altura – linha].

Já a relação entre linhas e ‘\*’ é: 1 na 1ª linha, 3 na 2ª, 5 na 3ª, 7 na 4ª, 9 na 5ª e 11 na 6ª linha. Ou seja, em cada linha a quantidade de ‘\*’ é [2\*linha – 1]

Pronto! Tá resolvido o problema! É só implementar em C.

1. Agora modifique o programa do item a) para gerar um losango, como mostrado abaixo.

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*

\*\*\*

\*