Estruturas de Repetição em Linguagem de Programação em C

As estruturas de repetição, como o próprio nome já diz, são utilizadas para REPETIR determinados comandos do algoritmo. O número de vezes que esses comandos serão repetidos pode ser especificado pelo programador ou analisados de acordo com uma condição.

Looping com teste lógico no início (**while(**condição**)**).

A estrutura while é utilizada para se repetir comandos nos casos em que o número de repetições não é conhecido.

Os comandos vão sendo executados repetidamente até que uma condição específica seja satisfeita. Essa condição que interrompe a repetição dos comandos é conhecida como teste ou flag.

Sua sintaxe é:

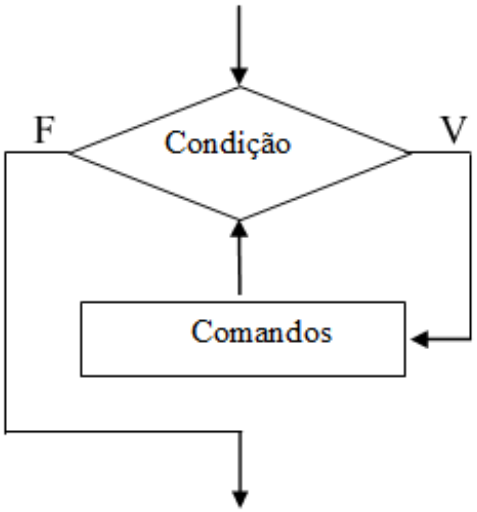
while (condição){

    comando 1

    ...

    comando n

} //fim while



Problema:

Escreva um algoritmo para calcular a média da idade de um grupo de 5 pessoas.

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

	setlocale(LC_ALL,"portuguese");
	int idade, c, somaidade;
	float media;

	c = 1; //variável de controle, sempre é do tipo inteiro essa variável.

	somaidade = 0;

	while (c <= 5){
		printf("Digite a idade da pessoa %d : \n",c);
		scanf("%d",&idade);
		somaidade = somaidade + idade;
		c = c + 1; // ou c++
	}

	media = (somaidade)/(c-1);
	printf("A média das idades é igual a %f ",media);
```

```
    return 0;
}
```

É preciso saber que a estrutura de repetição permite repetir diversas vezes determinado trechos do código. A estrutura de repetição minimiza o trabalho do desenvolvedor, evitando que digite longos trechos de códigos. Para saber quais trechos de códigos são considerados a repetições pelo uso de estruturas, basta identificar os padrões de comportamento. Então, padrões de comportamentos são aqueles trechos de códigos bastante parecidos.
Quando na sequencia de comandos necessita ser executado repetidas vezes, usamos uma estrutura de repetição. A estrutura de repetição também é conhecida como laço ou loop. Qualquer linguagem de programação possui comandos que implementam a estrutura de repetição.

Na linguagem de programação, possui:

while
do while
for

Esses três tipos serão detalhados no curso:

Comando WHILE permite executar instruções que são repetidas ‘enquanto’ determinada condição for verdadeira.

Sintaxe:

```
while (condição) {
    instrução1
    ...
    instrução n
}
```

Tudo que estiver dentro do bloco será executado enquanto a condição que estiver entre parentes for verdadeira.

Para exemplificar de uma forma simples, vamos analisar o código a seguir para imprimir 5 asterisco ‘ * ’ na tela.

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

int main() {

	setlocale(LC_ALL,"portuguese");
	printf(" * \n ");
	printf(" * \n ");
	printf(" * \n ");
	printf(" * \n ");
	printf(" * \n ");

	return 0;
}
```

Mas se fossem 1000 asteriscos ? Seria inviável escrever o comando printf( ) mil vezes.

A solução para esse problema é justamente o uso da estrutura de repetição. Vamos escrever o mesmo código usando o comando WHILE.

Algoritmo “exemplo_asterisco”

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

int main() {

	setlocale(LC_ALL,"portuguese");
	int c; //variável de controle inicio, sempre é do tipo inteiro (int)
	c = 1;  //iniciando a variável de controle

	while(c<=1000){
		printf(" * \n");
		c++;    //uso do operador de incremento  c = c + 1
	}


	return 0;
```

```
}
```

Uma observação importante, é sempre, modificar a variável de controle dentro do laço enquanto. Desta forma você vai garantir que em determinado momento a condição se torne FALSO, e assim, o laço será finalizado.
Para facilitar a didática, o exemplo a seguir imprime três “ * ” em vez de 1000.

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

int main() {

	setlocale(LC_ALL,"portuguese");
	int c; //variável de controle inicio, sempre é do tipo inteiro (int)
	c = 1;  //iniciando a variável de controle

	while(c<=3){
		printf(" * \n");
		c++;
	}


	return 0;
}
```

c	c <= 3
1	verdadeiro
2	verdadeiro
3	verdadeiro
4	falso

Observe que diferentemente de outros códigos esse programa não é sequencial. Pelos seguintes pontos:
1º) retorna para o ponto anterior para testar a condição.
2º) ele pula instruções, caso em que a condição é falsa.

No próximo exemplo, tem o uso do comando WHILE um pouco mais elaborado. Vamos analisar o problema da tabuada.
Escreva um algoritmo para mostrar na tela a tabuada do número n. Sendo n um número de 1 a 10.
Em outras palavras o algoritmo deve mostrar a tabuada do número que o usuário escolher.

Se o número for 3, o resultado será como mostrado a seguir:
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

Vamos aplicar a técnica para resolução de problemas:
entrada: um número (n)
saída: mostrar na tela as 10 linhas com a tabuada de n
processamento:
n * 1,
n * 2,
n * 3,
...
n * 9
n * 10

//programa "exemplo_Tabuada" em linguagem C

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

int main() {
```

```
	setlocale(LC_ALL,"portuguese");


	int c = 1, n; //variável de controle inicio, sempre é do tipo inteiro (int)
	printf("Digite um número: ");
	scanf("%d",&n);

	while(c<=10){

		printf("%d x %d = %d \n",n,c,n*c);
		c++;
	} //fim while

	return 0;
}
```

As variáveis que controlam as repetições de laços se chamam variáveis de controle, o que significa dizer que a variável c é uma variável de controle. A variável de controle deve ser sempre iniciada, e isso esta demostrado na linha antes da estrutura enquanto. O bloco do comando enquanto é executado varias vezes, até que a condição seja falsa. É necessário sempre inicializar a variável de controle. É necessário mudar o valor da variável de controle dentro do bloco repetição para não correr o risco de laços infinitos.

Exercícios de fixação:

1) Construir um programa que fique lendo números, até que seja digitado um número negativo. Após interromper as leituras, escrever quantos números foram lidos.

2) Escrever um programa que leia os valores dos salários de uma empresa. O programa deve interromper a leitura quando for digitado um salário menor ou igual a zero. Após efetuar as leituras, o programa deverá oferecer as seguintes informações.

a. A quantidade de salários;
b. A soma dos salários;
c. A média dos salários.

3) Construir um programa para ler os salários de uma empresa. O programa deve interromper a leitura quando for digitado um salário negativo. Após as leituras escrever:
a) A quantidade de salários lidos;
b) A soma dos salários;
c) A média dos salários;
d) O valor do maior salário;
e) O valor do menor salário.

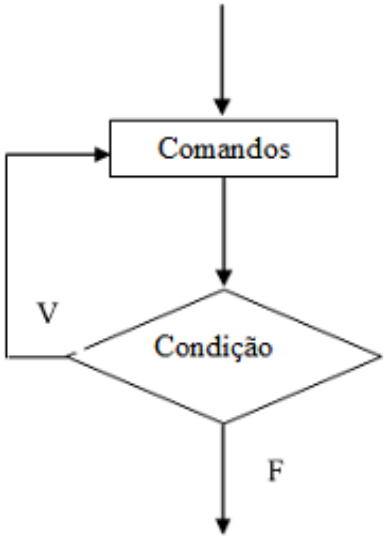
Looping com teste lógico no final do ... while(condição).

Caracteriza-se por uma estrutura que efetua um teste no final de um laço, verificando se é permitido ou não executar novamente o conjunto de comandos no interior do mesmo.
Deve-se existir um contador e esse necessita ser declarado e inicializado fora do laço.

É obrigatório que o contador mude seu valor dentro do laço, caso contrário, entra-se em laço infinito.

A condição do laço DO... WHILE fica no final, após o fechamento do bloco.
As instruções dentro do laço DO WHILE serão executadas, pelo menos, usa vez.

do while



Vamos analisar o problema da tabuada.
Escreva um algoritmo para mostrar na tela a tabuada do número n. Sendo n um número de 1 a 10.

Em **Linguagem C**

```
//programa  “exemploTabuada”

#include <stdio.h>
#include <locale.h>

int main() {

    setlocale(LC_ALL,"portuguese");


    int c = 1, n; //variável de controle inicio, sempre é do tipo inteiro (int)
    printf("Qual número você deseja gerar a tabuada: ");
```

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

    do{
        printf("%d x %d = %d \n",n,c,n*c);
        c++;
    } while(c<=10);

    return 0;
}
```

Programa que solicita para o usuário que digite um número inteiro e mostra esse valor ao quadrado, até o usuário digitar o valor zero ou negativo para sair do laço de repetição DO WHILE.

```
#include <locale.h>
#include <math.h> //Biblioteca math para utilizar a função pow()

int main() {

    setlocale(LC_ALL,"portuguese");


    int num, quadrado;
    printf("Digite um número: ");
    scanf("%d",&num);

    do{

        quadrado = pow(num,2); //função pow(), função que eleva ao
                    //quadrado o valor da variável num
        printf("O quadrado é %d \n",quadrado);
        printf("Digite um número: ");
        scanf("%d",&num);
```

```
    } while(num>0);


    return 0;
}
```

Estrutura de repetição **FOR**

Comando **FOR** em linguagem de programação, igual o **PARA** em português estruturado.

O comando FOR permite executar instruções que são repetidas enquanto determinada condição for verdadeira.
Observe o código da estrutura para linguagem C
Sintaxe:

```
para (<variavel = valorinicial>; <variavel = valorfinal>; <variavel><incremento> ) {
    bloco de instruções
}
```

A estrutura FOR, entre parêntese a variável de controle, condição e incremento da variável de controle.
Tudo que estiver no bloco de instruções, será executado enquanto a condição for verdadeira.
Observe que assim como o comando while, a condição do comando for fica dentro do parêntese. Observe que diferentemente do comando while a variável de controle é inicializada e finalizada dentro do parêntese. No comando for a variável de controle é inicializada antes do comando e modificada dentro do bloco.

Para exemplificar a funcionalidade do comando para vamos analisar o código para imprimir 5 asteriscos na tela.
Para a solução desse problema é justamente a estrutura de repetição. Para apresentar uma das soluções usando a estrutura de repetição, vamos escrever o mesmo código usando o comando PARA.

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

```
int main() {

	setlocale(LC_ALL,"portuguese");
	int i;
	for(i=1;i<=5;i++){
		printf("* \n");
	}

	return 0;
}
```

A variável i será a variável responsável para controlar o número de repetições do laço, por causa disso é chamada de variável de controle. Sempre iniciar a variável de controle entre parentes. Outro ponto importante é sempre modificar a variável de controle que em determinado momento a condição se torne falsa e assim finalizar a execução do laço.
É importante observar que a variável de controle (i), ela inicia em nesse exemplo com o valor 1, após verifica se i é menor ou igual a 5. Após isso executa o comando printf("*\n") e depois retorna e faz o incremento em 1, i++ que é igual a i = i + 1. Faz esse incremento em 1 e depois retorna verificar se o valor desse incremento é <=5, se for volta a executar o comando printf("* \n"), caso contrário sai da estrutura for.

Observe que diferente de outros códigos, esse programa não é sequencial pelos seguintes pontos: Ele retorna para as linhas anteriores para testar a condição e ele pula a instrução caso em que a condição é falsa.

Vamos analisar o problema da tabuada.
Escreva um algoritmo para mostrar na tela a tabuada do número n. Sendo n um número de 1 a 10.

Vamos aplicar a técnica para resolução de problemas:

Entrada: um número (n)
Saída: mostrar na tela as 10 linhas com a tabuada de n

Processamento:

	n * 1,

n * 2,
n * 3,
...
n * 9
n * 10

Solução em **linguagem C:**

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

int main() {

	setlocale(LC_ALL,"portuguese");
	int i, n;

	printf("Qual número você deseja gerar a tabuada: ");
	scanf("%d",&n);



	for(i=1;i<=10;i++){
		printf("%d x %d = %d \n",n,i,i * n);

	}

	return 0;
}
```

Comparações entre o comando WHILE com o comando FOR :

1 = inicialização de variáveis
2 = teste de condição
3 = incremento

int n,
 	
c = 1;
 	
printf(“Digite um número: ”)
scanf("%d",&n)
 	
while (c<=10) {
 	
printf("%d x %d = %d \n",n,i,i * n);
 	
c := c + 1
}
	int n, i;
	
	printf(“Digite um número: ”) ;
	scanf("%d",&n)
	
	for (i = 1; i<=10; i = i+1){
	printf("%d x %d = %d \n",n,i,i *
	n);
	}

O comando FOR parece mais “enxuto” pois permite em uma única.

A estrutura de repetição minimiza o número de linhas no código.

Exercícios de fixação:

1) Construir um programa que leia 10 números e após as leituras escreva:

a) O somatório dos números lidos;
b) O somatório dos números lidos maiores ou iguais a zero;
c) O somatório dos números lidos menores que zero.

2) Faça um programa que mostre a tabuada de um número digitado pelo usuário.

3) Apresentar o total da soma dos cem primeiros números inteiros (1+2+3+4+5+...97+98+100).

4) Elaborar um programa que efetue o cálculo da fatorial do valor 5 e apresente o resultado dessa operação.
Para entender o problema proposto, considere que, do ponto de vista matemático, fatorial é o produto dos números naturais desde 1 até o limite informado, neste caso 5. Assim sendo, a fatorial do valor 5, representada matematicamente como 5!, é a multiplicação de 1 x 2 x 3 x 4 x 5, que resulta no valor 120. Para efetuar essa operação, são necessárias duas variáveis, uma que controla as iterações do laço e outra que calcula a fatorial propriamente dito. Iniciar as variáveis cont e fat, ambas com valor inicial 1.

Tabela de cálculo da 5!			
cont	fat	fat := fat * cont	comentários
1	1	1	valor inicial das variáveis e fatorial
2	1	2	cálculo da fatorial com o contador em 2
3	2	6	cálculo da fatorial com o contador em 3
4	6	24	cálculo da fatorial com o contador em 4
5	24	120	cálculo da fatorial com o contador em 5