

Estruturas de repetição

Prof.: Maiquel de Brito

20 de agosto de 2018

BLU3101 - Introdução à Informática para Automação

Departamento de Engenharias

UFSC Blumenau

1. Introdução
2. Iteração *while*
3. Iteração *do...while*
4. Iteração *for*
5. A instrução *Break*

Introdução

Exercício

Escreva um programa que imprima todos os números entre 1 e 5

Solução

```
1  int main(){  
2      printf("%d",1);  
3      printf("%d",2);  
4      printf("%d",3);  
5      printf("%d",4);  
6      printf("%d",5);  
7  }
```

Exercício

Escreva um programa que imprima todos os números entre 1 e 5.000.000

Solução

```
1  int main(){  
2      printf("%d",1);  
3      printf("%d",2);  
4      printf("%d",3);  
5      :  
6      printf("%d",5000000);  
7  }
```

Exercício

Escreva um programa que solicite um número ao usuário e depois imprima todos os numeros entre 1 e o número informado

Solução

```
1  int main(){
2      int i;
3      printf("Informe um número: ");
4      scanf("%d",&i);
5      ? ? ? ?
6  }
```

Às vezes é necessário programar a *repetição* de determinadas instruções

1. Para tornar a tarefa de escrita do programa mais eficiente

ex.: Escrever os números entre 1 e 1.000.000

2. Quando não se sabe em *design time* quantas repetições precisam ser feitas

ex.: Escrever os números entre 1 e um número informado pelo usuário

Estruturas de repetição

Estruturas de repetição

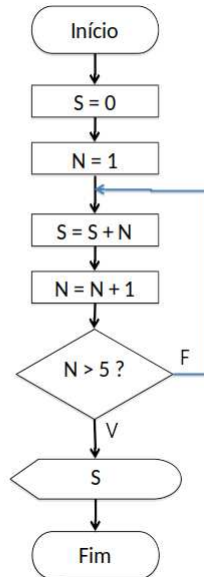
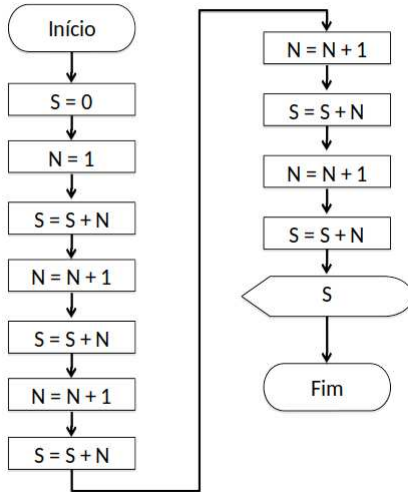
Possibilitam a repetição de um bloco de código. O número de vezes em que o bloco será repetido é controlado através de uma expressão lógica.

Estruturas de repetição na linguagem C

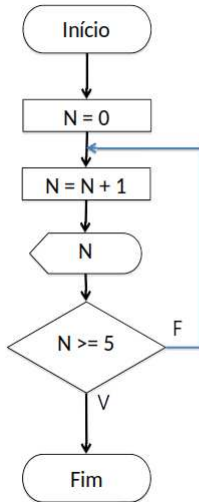
- `while`
- `do while`
- `for`

Algoritmos para somar
os 5 primeiros números
inteiros positivos

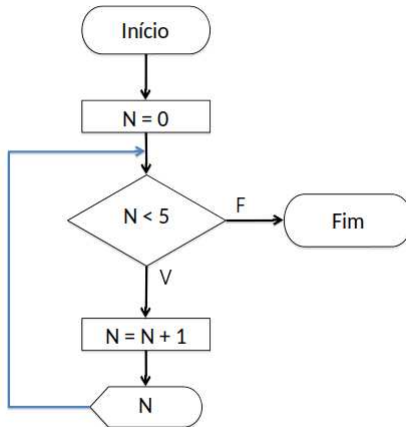
Fluxograma



Fluxograma

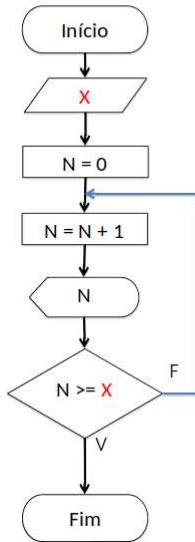


Algoritmos para mostrar os
inteiros positivos de 1 a 5

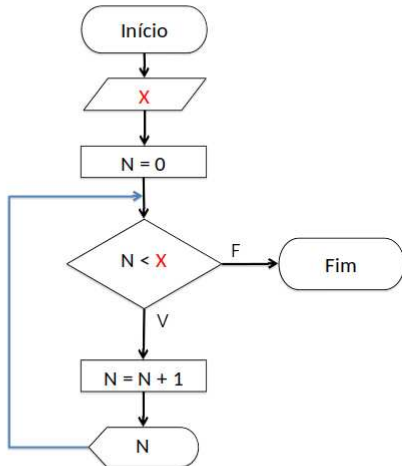


Existem diferenças?

Fluxograma



Algoritmos para mostrar os inteiros positivos de 1 a X (informado pelo usuário)



Existem Diferenças?

Estruturas de repetição em C

Diferentes estruturas expressam diferentes lógicas de repetição:

- `while` (enquanto - faça)
Teste condicional no início do bloco a ser executado
- `do while` (faça - enquanto)
Teste condicional no final do bloco a ser executado
- `for` (para - faça)
Número de repetições conhecido em *design time*

Iteração while

Iteração while

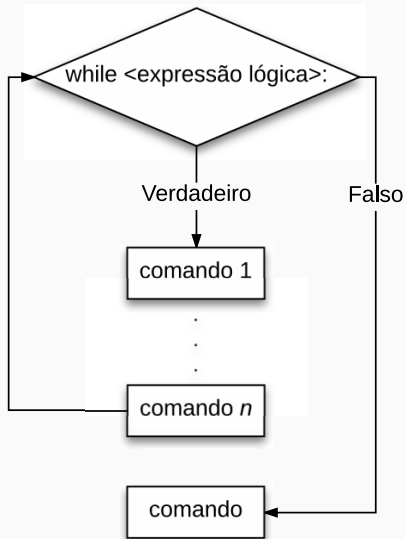
Possibilita repetir um bloco de comandos enquanto o resultado da expressão lógica for verdadeiro

Sintaxe: `while(expressão lógica){`
 sequência de comandos
`}`

Exemplo:

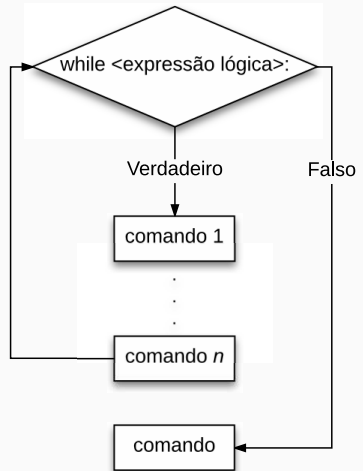
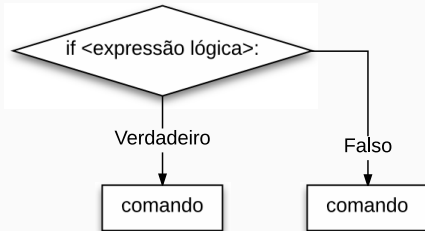
```
1  /* mostrar os valores de 1 a 5 */
2  int i = 0;
3  while(n<5){
4      n = n+1;
5      printf("%d",n);
6  }
```

Iteração - o loop While



Iteração - o loop While

Comparação *if* x *while*



Iteração - o loop While

O que será impresso por cada um dos programas abaixo?

Exemplo

```
1 int i=0;
2 while(i<=5){
3     printf("%d",i);
4     i = i+1;
5 }
```

Exemplo

```
1 int i=0;
2 while(i<=5){
3     printf("%d",i);
4
5 }
```

Exemplo

```
1 int i=10;
2 while(i<=5){
3     printf("%d",i);
4     i = i+1;
5 }
```

Iteração - o loop While

Passos essenciais em loops while

```
1 int i=0;           /*inicialização da variável de controle*/
2 while(i<=10){      /*checagem da variável de controle*/
3     printf("%d",i);
4     i = i+1;        /*atualização da variável de controle*/
5 }
```

1. Criar um programa que imprima todos os números de 1 a 100
2. Criar um programa que imprima todos os números de 1 a 10 em ordem decrescente
3. Criar um programa que imprima todos os números pares de 1 a 100

Iteração - o loop While

Loop definido

Sabe-se, em *design time*, quantas repetições ocorrerão

Exemplo

```
1 int i=1;
2 while(i<=5){
3     printf("%d",i);
4     i = i+1;
5 }
```

Loop indefinido

Algumas vezes não é possível saber, em *design time*, quantas repetições ocorrerão.

Exemplo:

Criar um programa que, continuamente, leia um número informado pelo usuário e que diga se o número é positivo (maior que zero) ou negativo (menor que zero). Para encerrar o programa, o usuário deve informar 0 (zero)

Iteração - loop indefinido

Criar um programa que, continuamente, leia um número informado pelo usuário e que diga se o número é positivo (maior que zero) ou negativo (menor que zero). Para encerrar o programa, o usuário deve informar 0 (zero)

```
1 int numero=1;
2 while(numero!=0){
3     /* ler o número */
4     printf("Digite um número: ");
5     scanf("%d",&numero)
6     /* verificar se é positivo ou negativo */
7     if(numero>0)
8         printf("positivo");
9     else
10         printf("negativo");
11 }
```

Iteração do...while

Iteração do...while

Semelhante ao `while`, porém a condição de execução do *loop* é verificada após a execução do bloco de código correspondente. O *loop* será executado pelo menos uma vez.

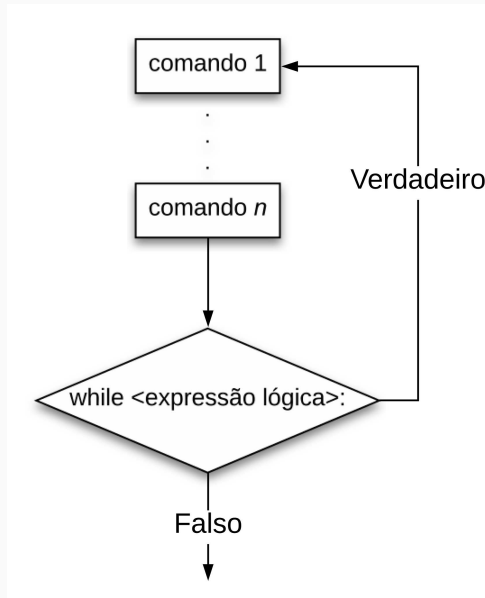
Sintaxe:

`do`

instruções

`while(expressão lógica) ;`

Iteração do...while



Iteração do...while

Exemplo

```
1      i = 100;  
2      do  
3          printf("%d",i);  
4          i++;  
5      while(i<10);
```

Iteração for

Iteração - o loop for

While é bom para *loops* indefinidos Para *loops* definidos, há os *loops* for

Exemplo

```
1 int i=0;
2 while(i<=10){
3     printf("%d",i);
4     i++;
5 }
```

Exemplo

```
1 int = 0;
2 for(i=0;i<11;i++){
3     printf("%d",i);
4
5 }
```

Iteração - o loop for

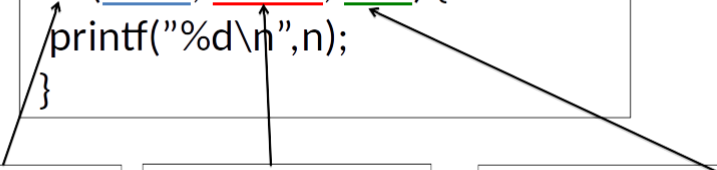
Possibilita repetir um bloco de comandos. O controle é feito através de variáveis definidas para o próprio laço. Normalmente empregado para situações em que se conhece o número de vezes que o bloco precisa ser executado.

Sintaxe:

```
for(inicialização; expressão lógica; incremento) {  
}
```

Iteração - o loop for

```
/* mostrar os valores de 1 a 5 */  
for (n = 1; n <= 5; n++) {  
    printf("%d\n",n);  
}
```



Inicialização
das variáveis
de controle
da repetição

Expressão lógica

Incremento das
variáveis

```
/* estrutura complexa */  
for (  
    a = 0, b = 0;  
    a * b < 1000 && a <= 30;  
    a = a + 1, b = b + 2  
) {  
    printf("a: %d b: %d\n",a,b);  
}
```

Fazer os seguintes exercícios usando *loops For*:

1. Criar um programa que imprima todos os números de 1 a 100
2. Criar um programa que imprima todos os números de 1 a 10 em ordem decrescente
3. Criar um programa que imprima todos os números pares de 1 a 100

A instrução Break

A instrução Break

A instrução break possibilita a interrupção de um laço de repetição. Quando a execução encontra uma instrução break a repetição é interrompida e a execução passa para a próxima instrução após o bloco de repetição.

A instrução Break

```
/* n terá valor 15 */  
int n = 0;  
while (n < 100) {  
    if (n == 15) {  
        break;  
    }  
    n = n + 1;  
}  
printf("%d\n",n);
```

```
/* n terá valor 15 */  
int n;  
for (n = 0;n < 100, n++) {  
    if (n == 15) {  
        break;  
    }  
}  
printf("%d\n",n);
```