

# Comandos de Repetiçao

## Switch-case

## Goto

SSC0304

Prof. Maurício A Dias  
macdiaspae@gmail.com

## ESTRUTURAS DE REPETIÇÃO

- Uma estrutura de repetição permite que uma sequência de comandos seja executada repetidamente, enquanto determinadas condições são satisfeitas.
- Essas condições são representadas por expressões lógicas (como, por exemplo,  $A > B$ ;  $C == 3$ ;  $\text{Letra} == 'a'$ )
  - Repetição com Teste no Início
  - Repetição com Teste no Final
  - Repetição Contada

## ESTRUTURAS DE REPETIÇÃO

- O real poder dos computadores está na sua habilidade para repetir uma operação ou uma serie de operações muitas vezes.
- Este repetição chamada **laços** (loop) é um dos conceitos básicos da programação estruturada

## REPETIÇÃO POR CONDIÇÃO

- Um conjunto de comandos de um algoritmo pode ser repetido quando subordinado a uma condição:

```
enquanto condição faça  
    comandos;  
fim enquanto
```

- De acordo com a condição, os comandos serão repetidos zero (se falso) ou mais vezes (enquanto a condição for verdadeira).
  - Essa estrutura normalmente é denominada **laço** ou **loop**

## REPETIÇÃO POR CONDIÇÃO

### ○ Condição

- qualquer expressão que resulte em um valor do tipo lógico e pode envolver operadores aritméticos, lógicos, relacionais e resultados de funções.

- Ex:

$x > 5$

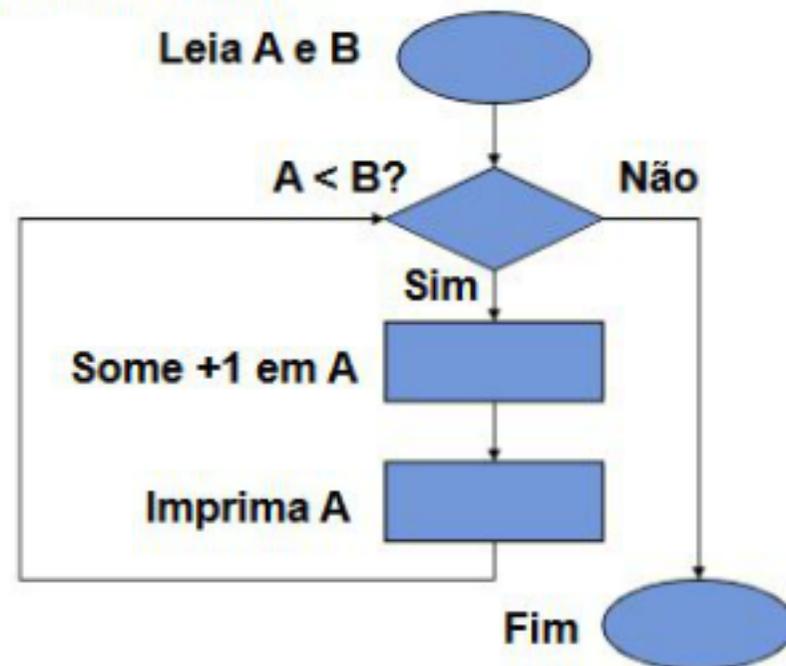
$(N < 60) \ \&\& \ (N > 35)$

## FUNCIONAMENTO

- A condição da cláusula *enquanto* é testada.
  - Se ela for verdadeira os comandos seguintes são executados em seqüência como em qualquer algoritmo, até a cláusula *fim enquanto*.
  - O fluxo nesse ponto é desviado de volta para a cláusula *enquanto* e o processo se repete.
  - Se a condição for falsa (ou quando finalmente for), o fluxo do algoritmo é desviado para o primeiro comando após a cláusula *fim enquanto*.

## REPETIÇÃO POR CONDIÇÃO

- Relembrando em fluxogramas
  - Um processo pode ser repetido até atender ou não uma condição.



## LOOP INFINITO

- Um loop ou laço infinito ocorre quando cometemos algum erro
  - ao especificar a condição lógica que controla a repetição
  - ou por esquecer de algum comando dentro da iteração.

## LOOP INFINITO

```
X recebe 4;  
enquanto (X < 5) faça  
    X recebe X - 1;  
    Imprima X;  
fim enquanto
```

```
X recebe 4;  
enquanto (X < 5) faça  
    Imprima X;  
fim enquanto
```

## EXERCÍCIO

- Escreva, em pseudo-código, o algoritmo para calcular a média de  $N$  números

## COMANDO WHILE

- Equivale ao comando “enquanto” utilizado nos pseudo-códigos.
  - Repete a sequência de comandos enquanto a condição for verdadeira.
  - **Repetição com Teste no Início**
- Esse comando possui a seguinte forma geral:

```
while (condição) {  
    sequência de comandos;  
}
```

## COMANDO WHILE - EXEMPLO

- Faça um programa que mostra na tela os número de 1 a 100

```
int main() {  
    // programa que mostra na tela números de 1 ate 100  
    printf(" 1 2 3 4 .... ");  
    return 0;  
}
```

- A solução acima é inviável para valores grandes. Precisamos de algo mais eficiente e inteligente

## COMANDO WHILE - EXEMPLO

- Faça um programa que mostra na tela os número de 1 a 100

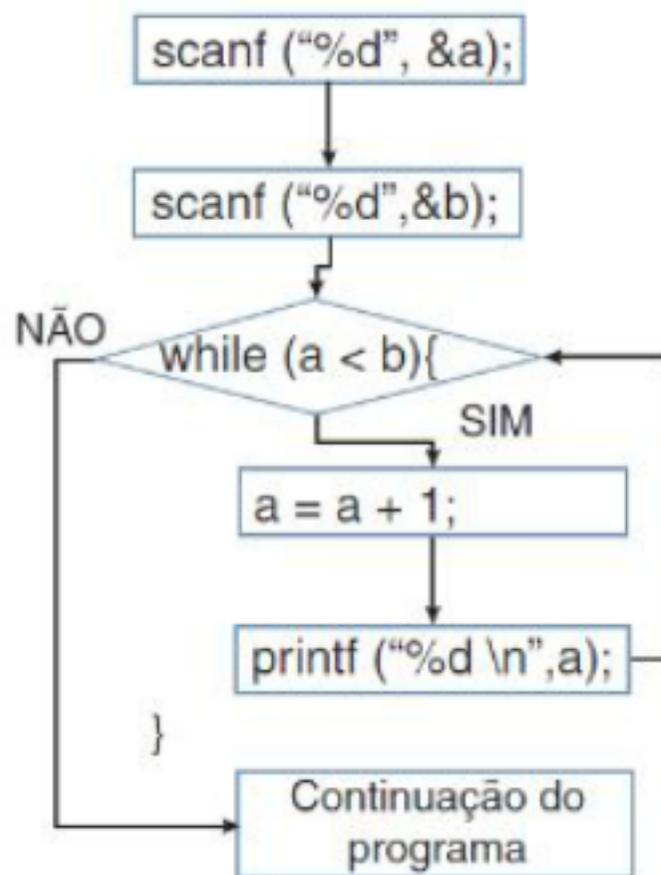
```
int main() {  
    // programa que mostra na tela números de 1 ate 100  
    int numero;  
    numero = 1; Inicializa o contador  
    while(numero <= 100) {  
        printf("%d", numero);  
        numero = numero + 1; Incrementa o contador  
    }  
    return 0;  
}
```

- Observe que a variável **numero** é usada como um **contador**, ou seja, vai contar quantas vezes o loop será executado

## COMANDO WHILE - EXEMPLO

- Faça um programa para ler 5 números e mostrar o resultado da soma desses números

## COMANDO WHILE - EXEMPLO



## COMANDO DO-WHILE

- Comando **while**: é utilizado para repetir um conjunto de comandos zero ou mais vezes.
  - Repetição com Teste no Início
- Comando **do-while**: é utilizado sempre que o bloco de comandos **deve ser executado ao menos uma vez**.
  - Repetição com Teste no Final

## COMANDO DO-WHILE

- o executa comandos
- o avalia condição:
  - se verdadeiro, re-executa bloco de comandos
  - caso contrário, termina o laço
- o Sua forma geral é (sempre termina com ponto e vírgula!)

```
do {  
    sequência de comandos;  
} while (condição);
```

## COMANDO DO-WHILE

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i;
    do{
        printf("Escolha uma opcao:\n");
        printf("(1) Opcao 1\n");
        printf("(2) Opcao 2\n");
        printf("(3) Opcao 3\n");
        scanf("%d", &i);

    }while((i < 1) || (i > 3));

    system("pause");
    return 0;
}
```

## COMANDO DO-WHILE

do {

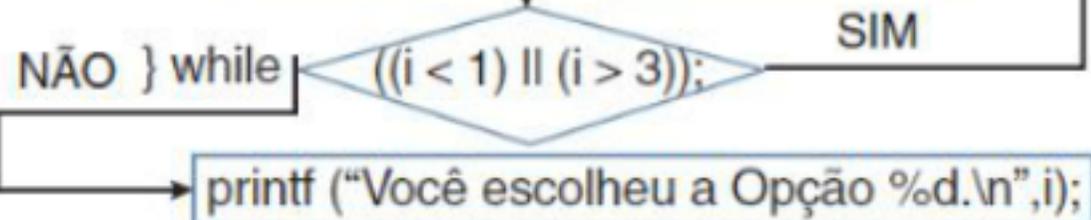
```
printf ("Escolha uma opção:\n");
```

```
printf ("(1) Opção 1\n");
```

```
printf ("(2) Opção 2\n");
```

```
printf ("(3) Opção 3\n");
```

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



## COMANDO FOR

- O loop ou laço *for* é usado para repetir um comando, ou bloco de comandos, diversas vezes
  - Maior controle sobre o loop.
- Sua forma geral é

```
for(inicialização; condição; incremento) {  
    sequência de comandos;  
}
```

## COMANDO FOR

1. **inicialização:** iniciar variáveis (contador).
2. **condição:** avalia a condição. Se verdadeiro, executa comandos do bloco, senão encerra laço.
3. **incremento:** ao término do bloco de comandos, incrementa o valor do contador
4. repete o processo até que a **condição** seja falsa.

```
for(inicialização; condição; incremento) {  
    sequência de comandos;  
}
```

## COMANDO FOR

- Em geral, utilizamos o comando **for** quando precisamos ir de um valor inicial até um valor final.
- Para tanto, utilizamos uma variável para a realizar a contagem
  - Exemplo: `int i;`
- Nas etapas do comando **for**
  - Inicialização: atribuímos o valor inicial a variável
  - Condição: especifica a condição para continuar no *loop*
    - Exemplo: seu valor final
  - Incremento: atualiza o valor da variável usada na contagem

- Exemplo: imprime os valores de 1 até 10

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int i;
    for (i = 1; i <= 10; i++) {
        printf("%d\n", i);
    }
    system("pause");
    return 0;
}
```

Inicialização

Condição

Incremento

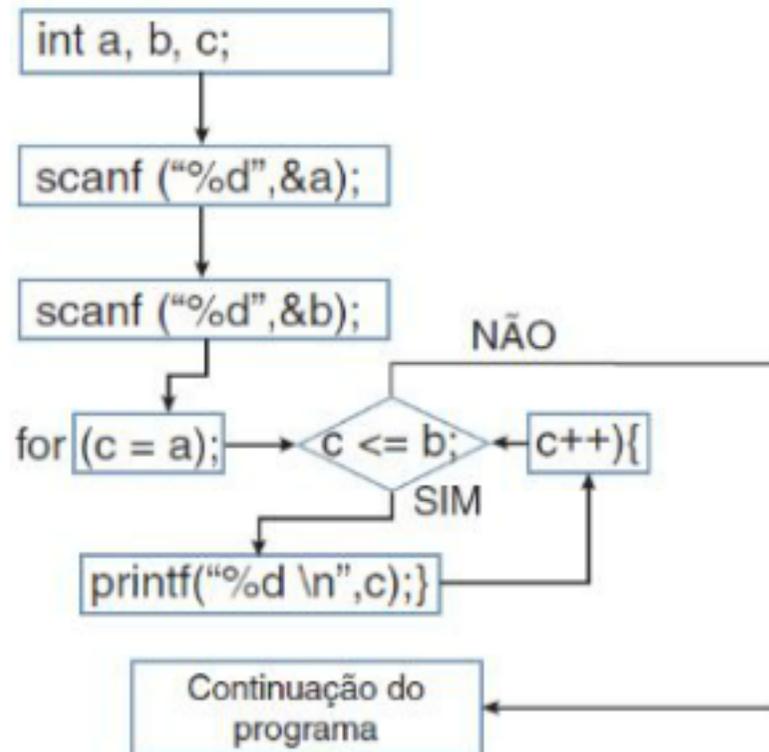
## COMANDO FOR

- Comando **while**: repete uma seqüência de comandos enquanto uma condição for verdadeira.
- Comando **for**: repete uma seqüência de comandos "N vezes".

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a, b, c;
    printf("Digite o valor de a: ");
    scanf("%d", &a);
    printf("Digite o valor de b: ");
    scanf("%d", &b);
    for(c = a; c <= b; c++) {
        printf("%d \n", c);
    }

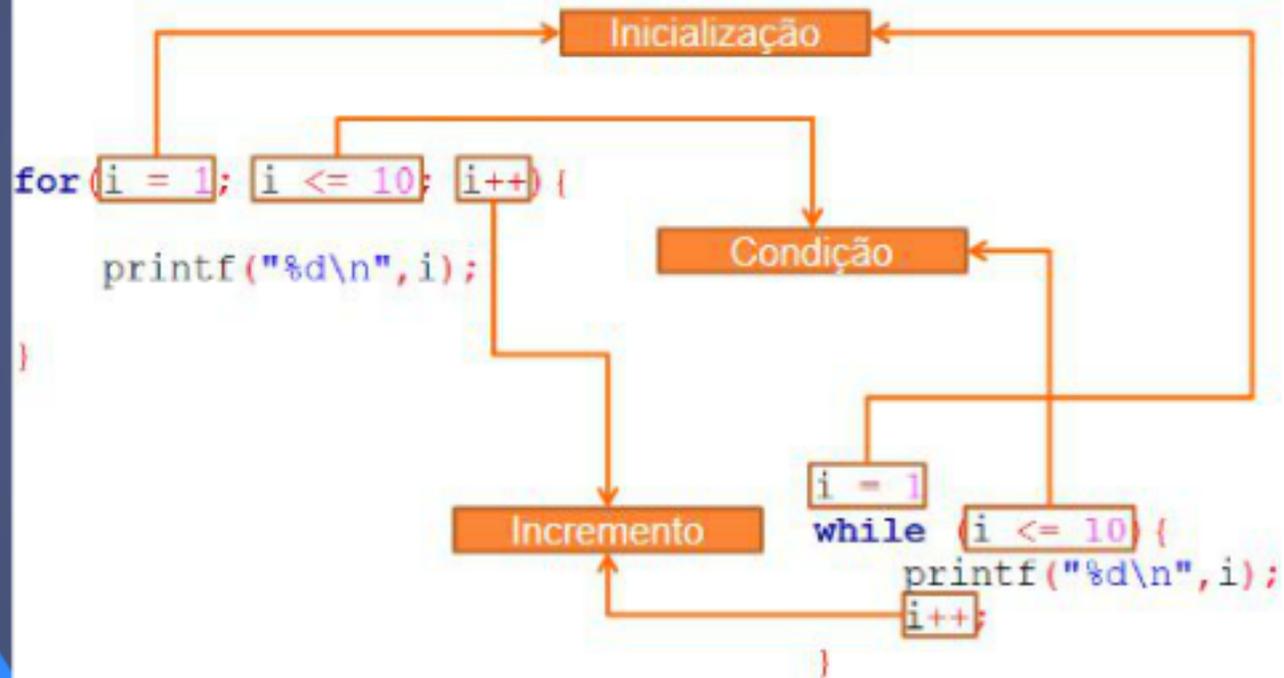
    return 0;
}
```

## EXEMPLO FOR



## FOR VERSUS WHILE

- Exemplo: mostra os valores de 1 até 10



## COMANDO FOR

- Podemos omitir qualquer um de seus elementos
  - inicialização, condição ou incremento.
- Ex.: **for** sem inicialização

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int a,b,c;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    for (; a <= b; a++){
        printf("%d \n",a);
    }
    system("pause");
    return 0;
}
```

## COMANDO FOR

- Cuidado: for sem condição
  - omitir a condição cria um laço infinito;
  - condição será sempre verdadeira.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a,b,c;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    //o comando for abaixo é um laço infinito
    for (c = a; ; c++){
        printf("%d \n",c);
    }
    system("pause");
    return 0;
}
```

## COMANDO FOR

- Cuidado: for sem incremento
  - omitir o incremento cria um laço infinito;
  - Incremento pode ser feito nos comandos.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a,b,c;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    for (c = a; c <= b; ){
        printf("%d \n",c);
        c++;
    }
    system("pause");
    return 0;
}
```

## O comando `switch`

- O comando `switch` teste sucessivamente o valor de uma expressão contra uma lista de constantes inteiras ou caracteres. Quando o valor coincide, os comandos associados àquela constante são executados.
- Sintaxe: `switch(expressão constante) {`
  - `case constante1:`
    - `instruções; break;`
  - `case constante2:`
    - `instruções; break;`
  - `default:`
    - `instruções;``}`

- O comando **break** causa uma saída imediata do **switch**. Se não existir um comando **break** seguindo as instruções de um caso, o programa segue executando todas as instruções dos casos abaixo.
- Se nenhum caso for satisfeito e existir um caso **default**, a execução começará nele, senão o programa processará as instruções seguintes ao bloco **switch**. Um **default** é opcional.

```
#include <stdio.h>
main()
{
    float num1, num2;
    char op;
    printf("\nDigite uma expressão no formato: num op num: ");
    scanf("%f %c %f", &num1, &op, &num2);
    switch( op ) {
        case '+':
            printf(" = %f", num1+num2); break;
        case '-':
            printf(" = %f", num1-num2); break;
        case '*':
            printf(" = %f", num1*num2); break;
        case '/':
            printf(" = %f", num1/num2); break;
        default:
            printf("\nOperador desconhecido");
    }
}
```

```
#include <stdio.h>
main()
{
    float num1, num2;
    char op;
    printf("\nDigite uma expressão no formato: num op num: ");
    scanf("%f %c %f", &num1, &op, &num2);
    switch( op ) {
        case '+':
            printf(" = %f", num1+num2); break;
        case '-':
            printf(" = %f", num1-num2); break;
        case '*':
        case 'x':
            printf(" = %f", num1*num2); break;
        case '/':
            printf(" = %f", num1/num2); break;
        default:
            printf("\nOperador desconhecido");
    }
}
```

```
switch( regioao )
{
    case 'l':
        printf("\nOs vendedores sao: Rafael, Joao e Mariana\n");
        printf("Informe a primeira letra do nome do vendedor: ");
        vendedor = getche();
        switch( vendedor )
        {
            case 'r': printf("\nVendas: R$ 10.000,00\n"); break;
            case 'j': printf("\nVendas: R$ 12.000,00\n"); break;
            case 'm': printf("\nVendas: R$ 14.000,00\n");
        }
        break;
}
```

## COMANDO BREAK

- o comando **break** serve para
  - quebrar a execução de um comando (como no caso do **switch**)
  - interromper a execução de qualquer *loop* (**for**, **while** ou **do-while**).
- O comando **break** é utilizado para terminar de forma abrupta uma repetição. Por exemplo, se estivermos dentro de uma repetição e um determinado resultado ocorrer, o programa deverá sair da repetição e continuar na primeira linha seguinte a ela

## COMANDO BREAK

```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int a,b;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    while (a <= b){
        a = a + 1;
        if(a == 5)
            break;
        printf("%d \n",a);
    }

    return 0;
}
```

## COMANDO CONTINUE

- Comando **continue**
  - Diferente do comando **break**, só funciona dentro do *loop*;
  - “Pula” essa iteração do *loop*.
- Quando o comando **continue** é executado, os comandos restantes da repetição são ignorados. O programa volta a testar a condição do laço para saber se o mesmo deve ser executado novamente ou não;

## COMANDO CONTINUE

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a,b;
    printf("Digite o valor de a: ");
    scanf("%d",&a);
    printf("Digite o valor de b: ");
    scanf("%d",&b);
    while (a <= b) {
        a = a + 1;
        if(a == 5)
            continue;
        printf("%d \n",a);
    }

    return 0;
}
```

## GOTO E LABEL

- É um salto condicional (**goto**) para um local especificado.
- Este local é determinado por uma palavra chave no código (**label**).
  - Este local pode ser a frente ou atrás no programa, mas deve ser dentro da mesma função.

- Forma geral:

```
palavra_chave:  
goto palavra_chave;
```



## GOTO E LABEL

- Apesar de banido da prática de programação, pode ser útil em determinadas circunstâncias.
  - Ex: sair de dentro de laços aninhados.

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int i, j, k;
    for(i = 0; i < 5; i++)
        for(j = 0; j < 5; j++)
            for(k = 0; k < 5; k++)
                if(i == 2 && j == 3 && k == 1)
                    goto fim;
                else
                    printf("Posicao [%d,%d,%d]\n", i, j, k);

    fim://label
    printf("Fim do programa\n");

    return 0;
}
```



Comandos de Repetiçao  
Switch-case  
Goto

SSC0304

Prof. Maurício A Dias  
macdiaspae@gmail.com