

LINGUAGEM C: COMANDOS DE REPETIÇÃO

Prof. Humberto Razente

Sala 1B144

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 (por exemplo, $A > B$; $C == 3$; $\text{Letra} == \text{'a'}$)
 - Repetição com Teste no Início
 - Repetição com Teste no Final
 - Repetição Contada

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)

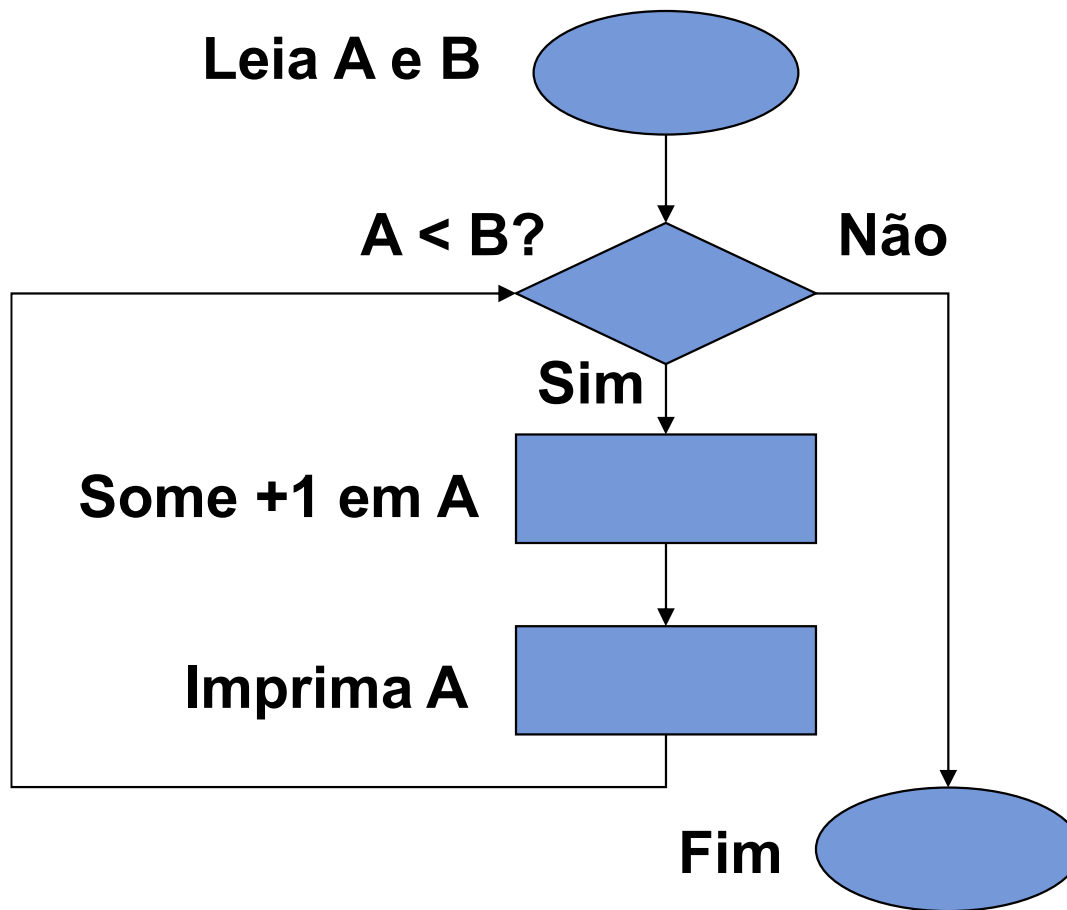
- Essa estrutura normalmente é denominada **laço** ou **loop**

FUNCIONAMENTO

- A condição da cláusula *enquanto* é testada
 - Se ela for verdadeira os comandos seguintes são executados em sequência 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, 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.



EXEMPLO – PSEUDO-CÓDIGO

Leia A;

Leia B;

Enquanto $A < B$

 A recebe $A + 1$;

 Imprima A;

Fim Enquanto

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

Condição errônea

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

Não muda valor

```
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.

```
Leia n;  
media recebe 0;  
n1 recebe 0;  
Enquanto (n1 < n)  
    Leia x;  
    media recebe media + x;  
    n1 ← n1 + 1;  
Fim enquanto  
Imprima media/n;
```

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 números 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 números 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

```
int main(){
    float val1, val2, val3, val4, val5, soma;

    printf("\nDigite o 1o. numero: ");
    scanf("%f", &val1);

    printf("\nDigite o 2o. numero: ");
    scanf("%f", &val2);

    printf("\nDigite o 3o. numero: ");
    scanf("%f", &val3);

    printf("\nDigite o 4o. numero: ");
    scanf("%f", &val4);

    printf("\nDigite o 5o. numero: ");
    scanf("%f", &val5);

    soma = val1 + val2 + val3 + val4 + val5;
    printf("\nO resultado da soma eh: %f", soma);

    return 0;
}
```

COMANDO WHILE - EXEMPLO

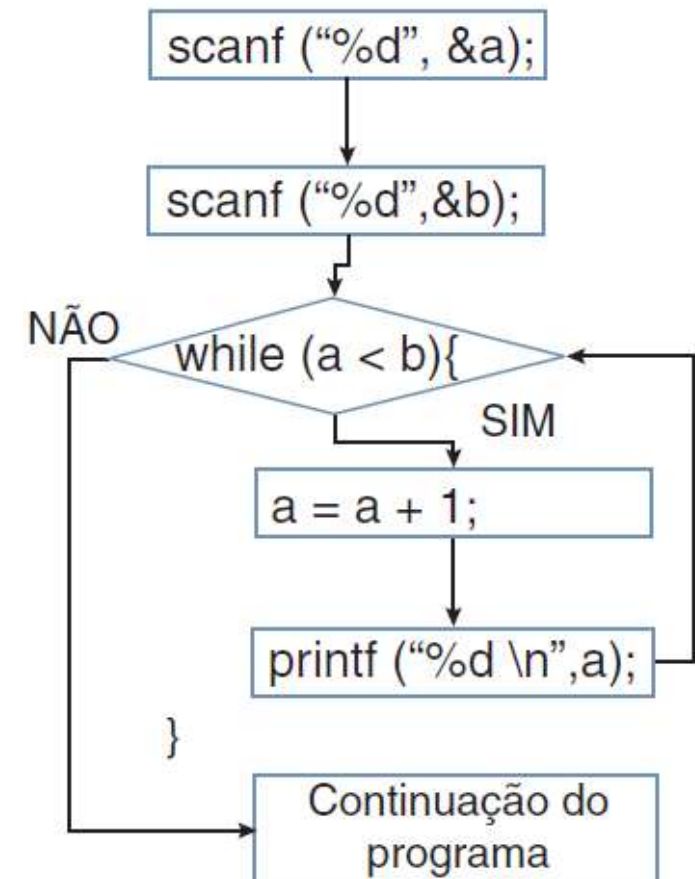
- Faça um programa para ler 5 números e mostrar o resultado da soma desses números

```
int main() {  
    float val, soma;  
    int contagem;  
    // inicializando o valor de soma  
    soma = 0; Acumulador  
    // inicializando o contador  
    contagem = 1;  
    while(contagem <= 5) {  
        printf("\nDigite o %do. numero: ", contagem);  
        scanf("%f", &val);  
        soma = soma + val; Acumula a soma a cada passo do loop  
        contagem = contagem + 1;  
    } Controla o número de execuções  
    printf("\nO resultado da soma eh: %.2f", soma);  
    return 0;  
}
```

COMANDO WHILE - EXEMPLO

- Imprimindo os números inteiros entre A e B

```
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;  
        printf("%d \n", a);  
    }  
  
    return 0;  
}
```



EXERCÍCIO

- Escreva, usando while, um programa para calcular a média de N números. O valor de N é dado pelo usuário.

```
int main() {  
    int n, n1, x;  
    float media = 0;  
    printf("Digite N:");  
    scanf("%d", &n);  
    n1 = 0;  
    while (n1 < n) {  
        printf("Digite X:");  
        scanf("%d", &x);  
        media = media + x;  
        n1 = n1 + 1;  
    }  
    printf("%f", media/n);  
  
    return 0;  
}
```


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

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

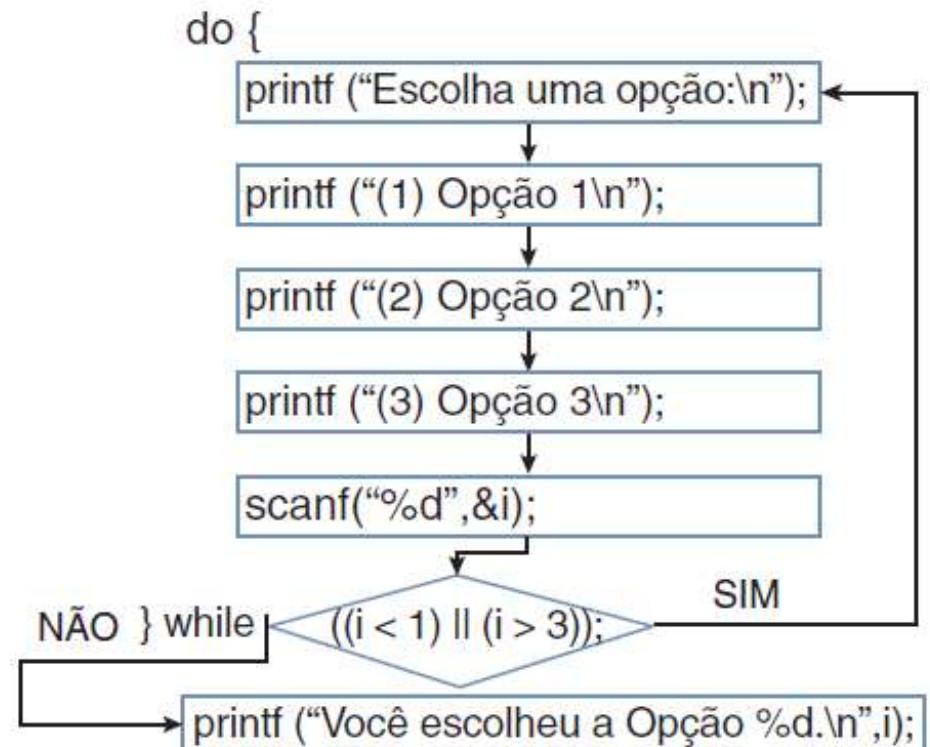
```
do {  
    sequência de comandos;  
} while (condição);
```

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 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;  
}
```

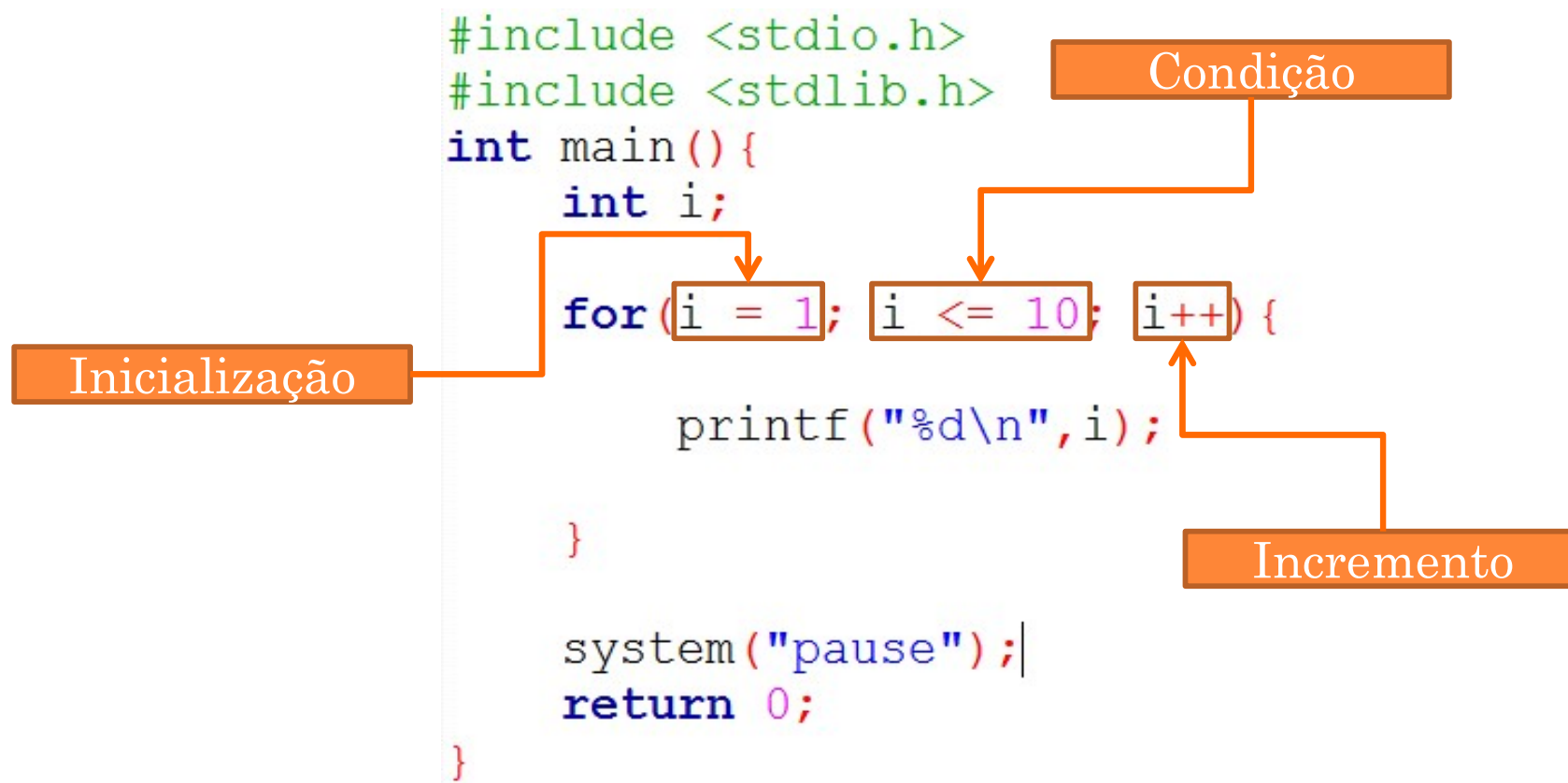
COMANDO FOR

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

- Exemplo: imprime os valores de 1 até 10



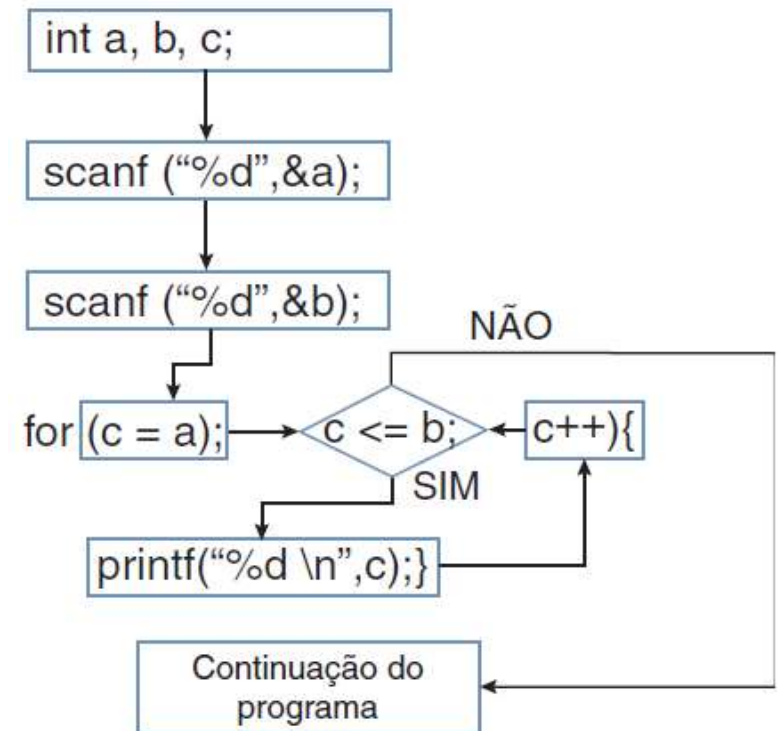
COMANDO FOR

- Comando **while**: repete uma sequência de comandos enquanto uma condição for verdadeira
- Comando **for**: repete uma sequência de comandos “N vezes”

EXEMPLO FOR

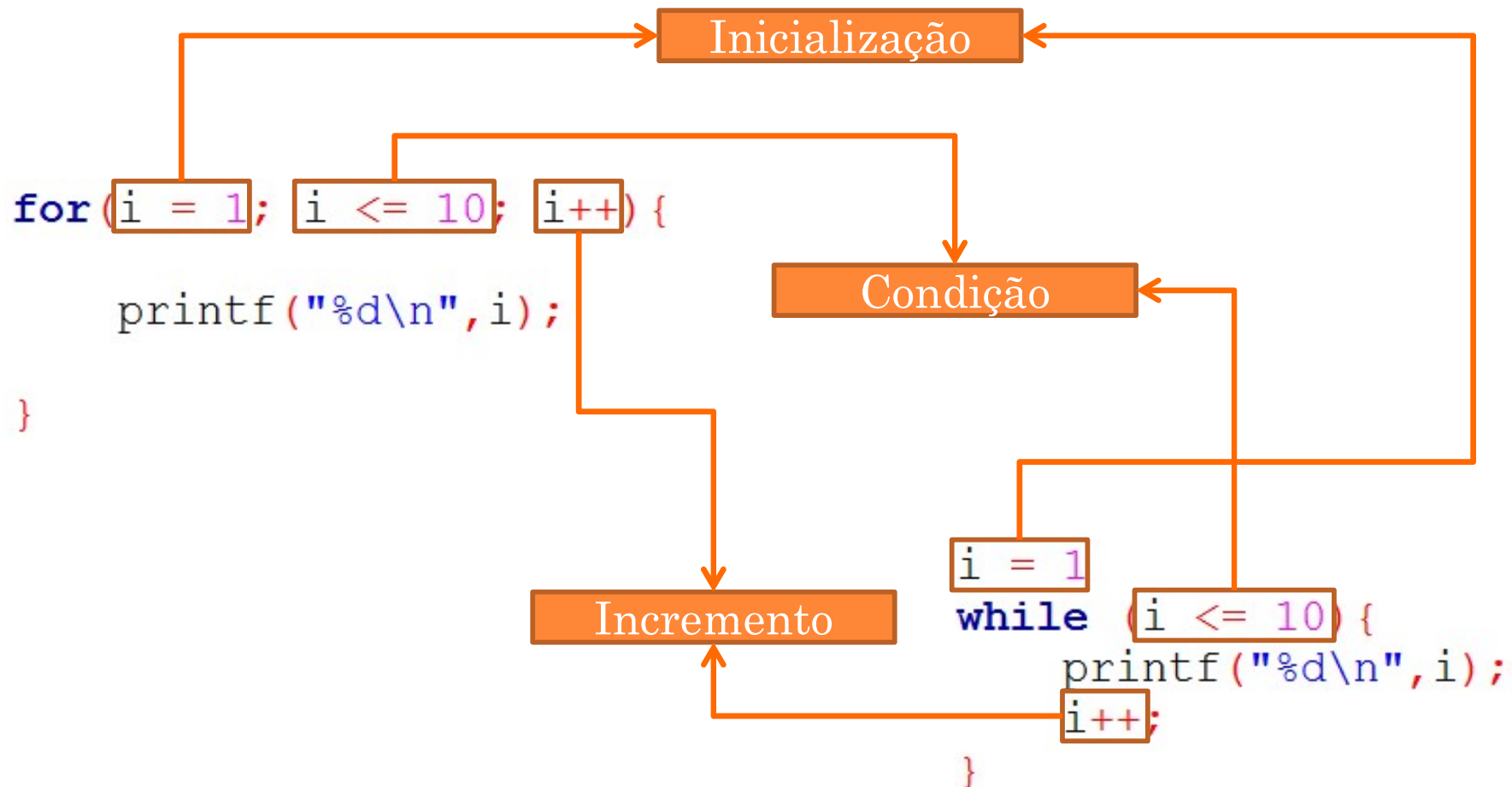
```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int a, b, c;
    printf("Digite o valor de a: ");
    scanf("%d", &a);
    printf("Digite o valor de b: ");
    scanf("%d", &b);
    for(c = a; c <= b; c++) {
        printf("%d \n", c);
    }

    return 0;
}
```



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;
}
```

EXERCÍCIO

- Escreva, usando *for*, um algoritmo para calcular a soma dos elementos de 1 a 10.

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

COMANDO BREAK

- Nós já vimos dois usos para o comando break: interrompendo os comandos switch. Ex.:

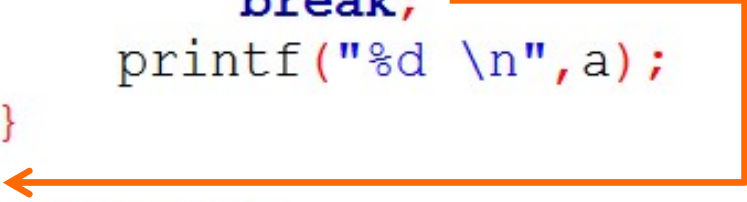
```
int num;  
scanf("%d", &num);  
switch(num) {  
    case 0: printf("Zero"); break;  
    case 1: printf("Um"); break;  
}
```

COMANDO BREAK

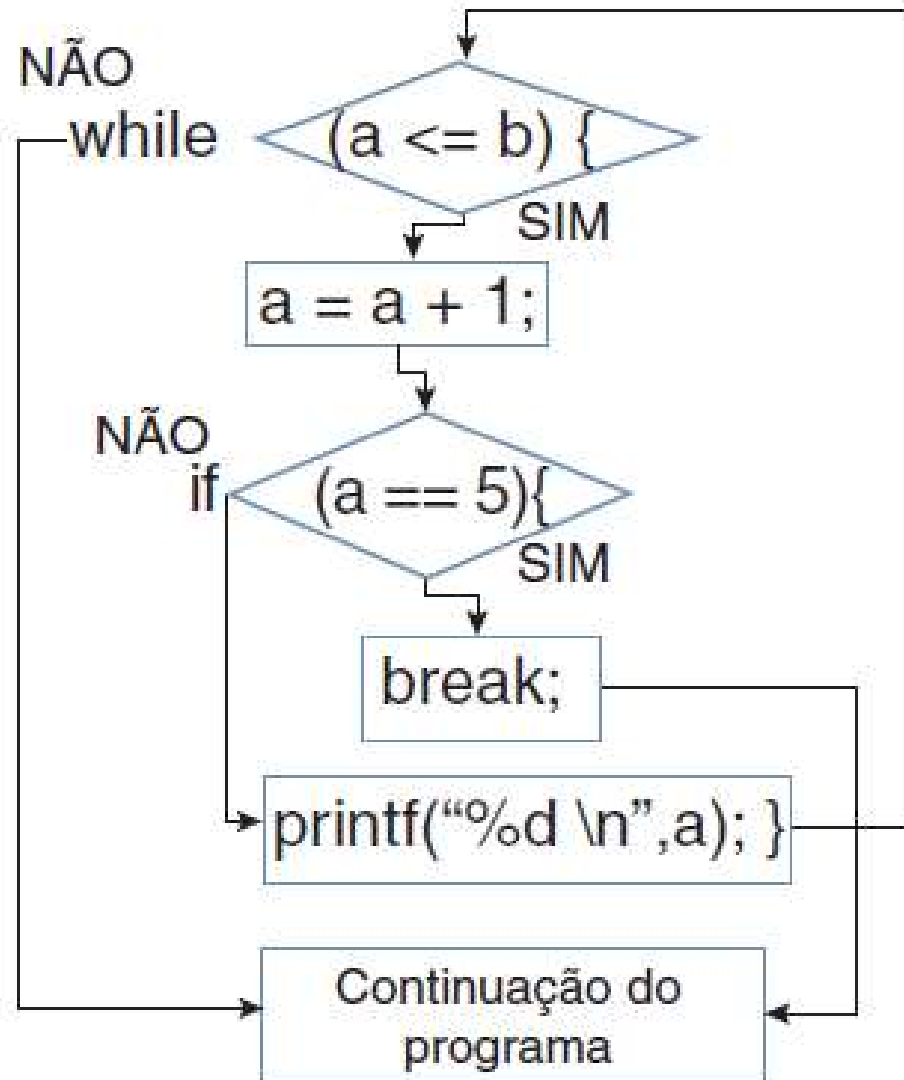
- Na verdade, o comando **break** serve para
 - quebrar a execução de um comando (como no caso do **switch**)
 - interromper a execução de qualquer *loop* (**for**, **while** ou **do-while**).
- O comando **break** é utilizado para terminar de forma abrupta uma repetição.
 - Por exemplo, se estivermos 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;
}
```

An orange rectangular box highlights the 'break;' statement and the subsequent 'printf' statement within the while loop. An orange arrow points from the 'break;' statement to the 'return 0;' statement, indicating that the loop is exited and the program proceeds to the return statement.

COMANDO BREAK




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

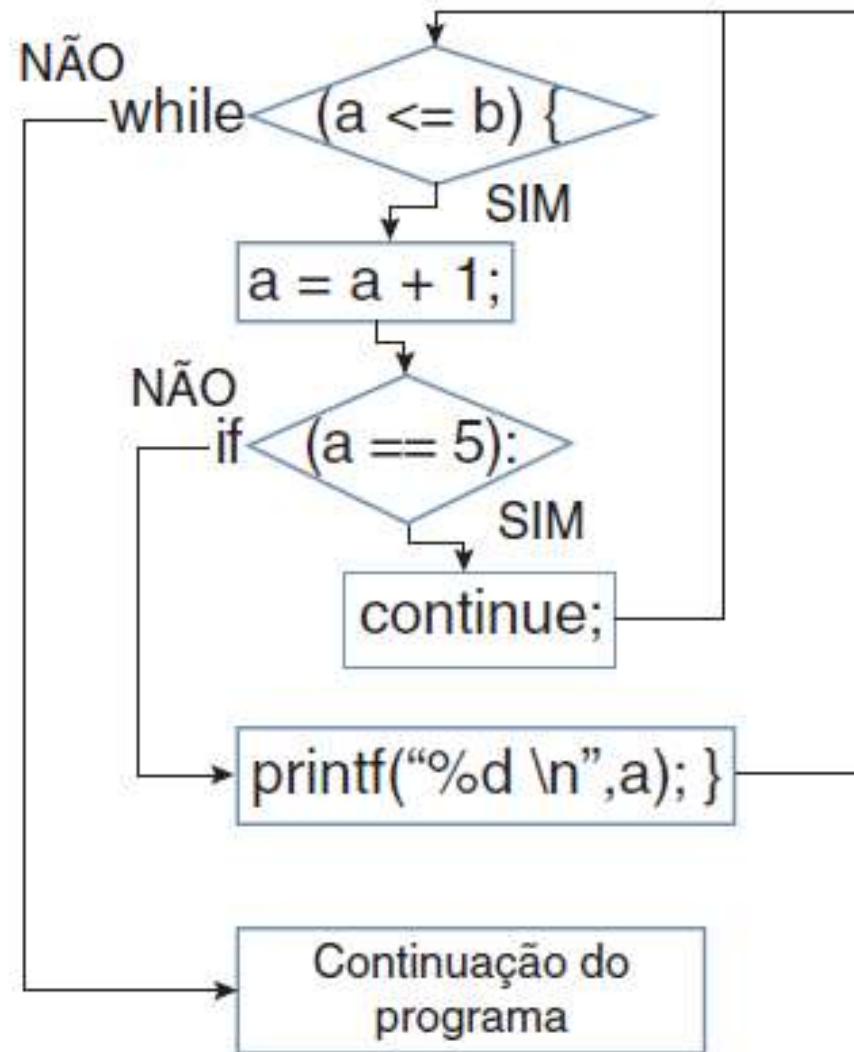
COMANDO CONTINUE

```
#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;
}
```



COMANDO CONTINUE



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

- O teorema da programação estruturada prova que a instrução **goto** não é necessária para escrever programas
 - Alguma combinação das três construções de programação (comandos sequenciais, condicionais e de repetição) são suficientes para executar qualquer cálculo.
 - Além disso, o uso de **goto** pode deixar o programa muitas vezes ilegível.

GOTO E LABEL

- Apesar de **banido** da prática de programação há décadas, **goto** ainda é incluído como parte da linguagem C:
 - 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;
}
```

MATERIAL COMPLEMENTAR

○ Vídeo Aulas

- Aula 18: Comando While
 - Aula 19: Comando For
 - Aula 20: Comando Do-While
 - Aula 21: Aninhamento de Repetições
 - Aula 22: Comando Break
 - Aula 23: Comando Continue
 - Aula 24: Comando Goto
-
- <https://programacaodescomplicada.wordpress.com/index/linguagem-c/>



LINGUAGEM C: COMANDOS DE REPETIÇÃO

41

Contém slides originais gentilmente
disponibilizados pelo Prof. André R. Backes (UFU)