INSTITUTO FEDERAL DE MATO GROSSO
CAMPUS RONDONÓPOLIS
CURSO SUPERIOR DE ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Estrutura de Dados e Análise de Algoritmos

Prof. Daniel Domingos Alves

daniel.alves@ifmt.edu.br

Estruturas de repetição em C
07/11/2024

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 lógica (como, por exemplo, A>B; C==3; Letra == ‘a’)
  - Repetição com Teste no Início
  - Repetição com Teste no Final
  - Repetição Contada

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
    int numero;
    numero = 1;
    while(numero <= 100){
        printf("%d", numero);
        numero = numero + 1;
    }
    return 0;
}
```

Inicializa o contador

Incrementa o contador

- 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 val, soma;
    int contagem;
    // inicializando o valor de soma
    soma = 0;
    // inicializando o contador
    contagem = 1;
    while(contagem <= 5){
        printf("\nDigite o %do. numero: ", contagem);
        scanf("%f", &val);
        soma = soma + val;
        contagem = contagem + 1;
    }
    printf("\nO resultado da soma eh: %.2f", soma);
    return 0;
}
```

Acumulador

Acumula a soma a cada passo do loop

Controla o número de execuções

COMANDO WHILE - EXEMPLO

- Imprimindo os números 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;
}
```

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

```c
#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.

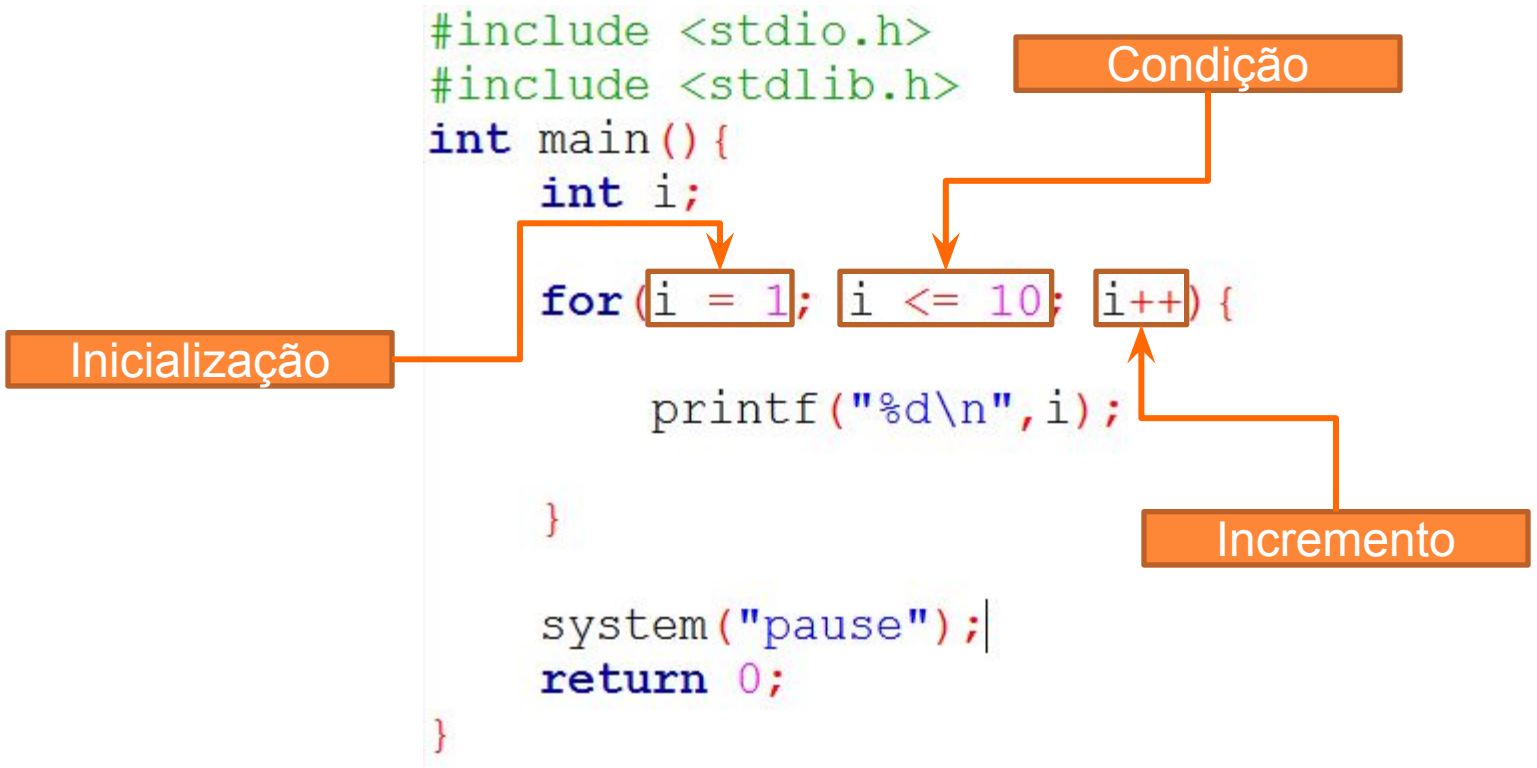
```
for(inicialização; condição; incremento){
    sequência de comandos;
}
```

Comando for

- 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

Comando for

- Exemplo: imprime os valores de 1 até 10



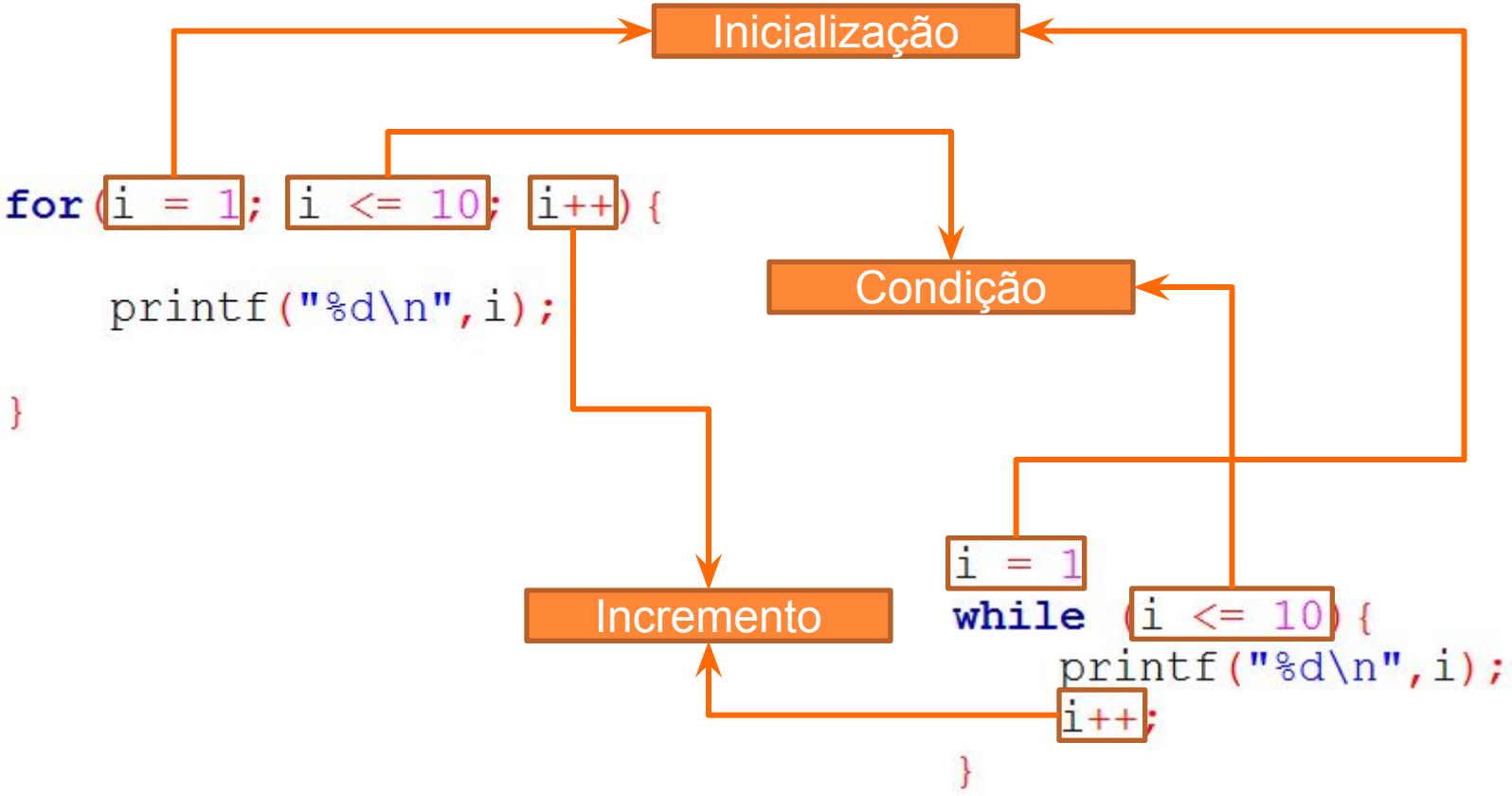
Comando for

- Comando **while**: repete uma sequência de comandos enquanto uma condição for verdadeira.

- Comando **for**: repete uma sequência de comandos “N vezes”.

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.

EXERCÍCIO

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

```c
#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

- 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

```c
#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

```c
#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;
}
```