Estruturas de Repetição

Professora Andreia Machion

1

Estruturas de Repetição

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

2

Estruturas de Repetição

- Uma estrutura de repetição permite que uma sequência de instruções seja executada repetidamente, enquanto determinadas condições são satisfeitas.
- Essas condições são representadas por expressões lógicas como, por exemplo, A>B; C==3; Letra == 'a'
- Na linguagem C, temos 3 comandos de repetição
  - Por condição com teste no início
  - Por condição com teste no final
  - Contado com teste no início

3

Repetição por condição com teste no início

- Um conjunto de instruções pode ser repetido quando subordinado a uma condição como descrito abaixo

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).
- Condição
  - qualquer expressão que resulte em um valor do tipo lógico e que pode envolver operadores aritméticos, lógicos, relacionais e resultados de funções.
  - Ex:
    (x > 5)
    (n < 60 && n > 35)

4

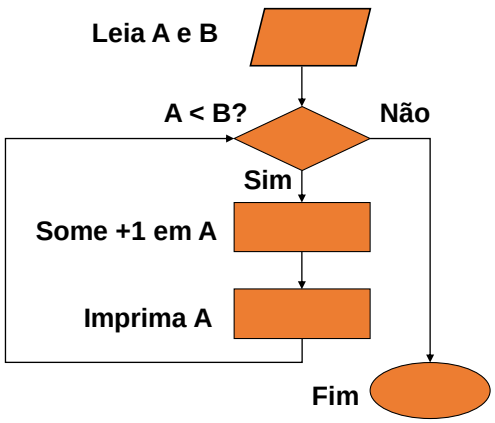
Dinâmica de funcionamento

- 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 (ou quando finalmente for), o fluxo do algoritmo é desviado para o primeiro comando após a cláusula ***fim enquanto***.

5

Repetição por Condição - Um conjunto de instruções é repetido enquanto atender uma condição.

Fluxograma



Pseudocódigo

```
Leia A e B;
Enquanto A < B
        A recebe A + 1;
        Imprima A;
Fim Enquanto
```

6

Um Loop Infinito ocorre quando cometemos algum erro ao especificar a condição lógica que controla a repetição ou esquecemos de algum comando dentro da iteração

Condição mal formulada

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

Esquecemos de atualizar o X

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

7

Comando while

- Equivale ao comando “enquanto” visto nos pseudocódigos
  - Repete a sequência de instruções enquanto a condição for verdadeira
  - **Repetição com Teste no Início**
- Esse comando possui a seguinte forma geral

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

8

Comando while – exemplo 1 – motivação

- Mostrar 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 já está ruim para limite 100, imagine para valores realmente grandes
- Precisamos de algo mais eficiente e inteligente!

9

Comando while – exemplo 1

- Mostrar 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

10

Contadores e acumuladores

11

Comando while – exemplo 2 – motivação

- Ler 5 números e mostrar o resultado da soma desses números

```c
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;
}
```

12

Comando while – exemplo 2

- 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

13

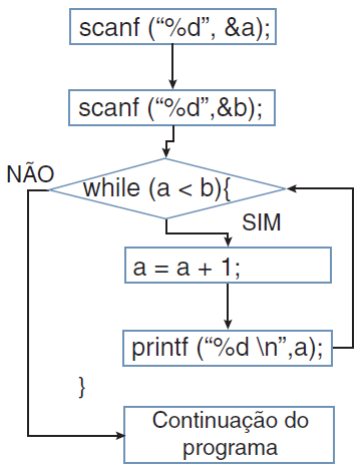
Comando while – exemplo 3

- 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;
}
```



14

Exercício: Escrever um programa para calcular a média de N números, usando while. O valor de N é dado pelo usuário.

```c
#include <stdio.h>
int main() {
    //variáveis
    int N, contador;
    float soma, media, val;
    //entrada inicial
    printf ("digite n: ");
    scanf ("%d", &N);
    //laço de repetição
    //valores iniciais
    soma = 0;
    contador = 1;
    //condição: valor final
    while (contador <= N) {
        printf ("digite o  %do valor: ", contador);
        scanf ("%f", &val);
        soma = soma + val;
        contador = contador + 1;
    }
    media = soma / N;
    //saída
    printf ("\nMedia = %.3f\n", media);
    return 0;
}
```

15

Comando do-while

- Comando **while**
  - utilizado para repetir um conjunto de comandos zero ou mais vezes
  - Repetição com teste no início
- Comando **do-while**
  - utilizado sempre que o bloco de comandos **deve ser executado ao menos uma vez**
  - **Repetição com teste no final**

16

Comando do-while

- Primeiro, executa instruções
- Depois avalia condição
  - se verdadeiro, reexecuta bloco de instruções
  - caso contrário, termina o laço

- Sua forma geral é (sempre termina com ponto e vírgula!)

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

17

Comando do-while – exemplo 1

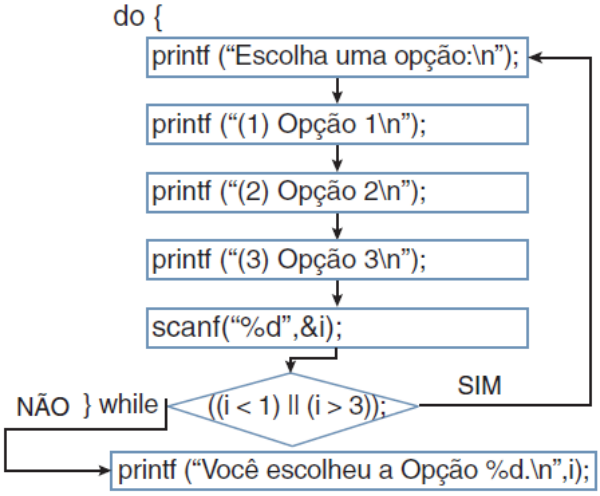
```
#include <stdio.h>
#include <stdlib.h>
int main(){
    int i;
    do{
        printf("Escolha uma opcao:\n");
        printf("(1) Opcao 1\n");
        printf("(2) Opcao 2\n");
        printf("(3) Opcao 3\n");
        scanf("%d",&i);

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

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

18

Comando do-while



19

Comando for

conveniente para laços contados

- O loop ou laço ***for*** é usado para repetir um bloco de instruções diversas vezes
  - Controle por contador
- Sua forma geral é

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

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.

20

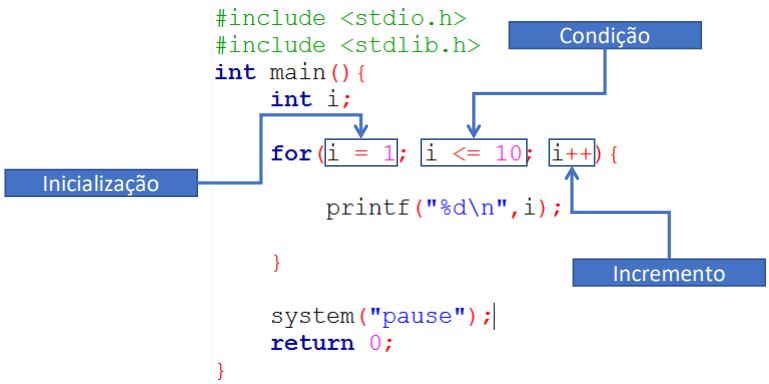
Comando for

- 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 à variável
  - Condição: especifica a condição para continuar no *loop*
    - seu valor final
  - Incremento: atualiza o valor da variável usada na contagem

21

Comando for

- Exemplo: imprime os valores de 1 a 10



22

for *x* while

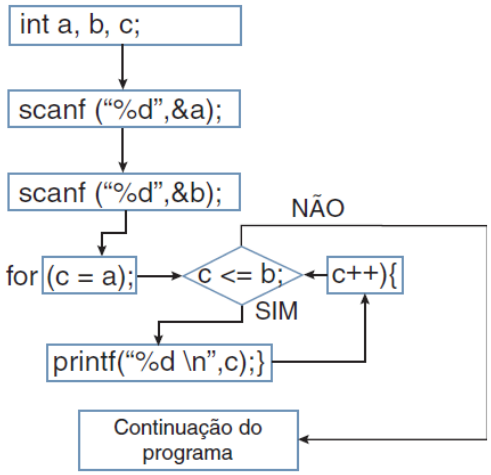
- Comando **while**: repete uma sequência de instruções enquanto uma condição for verdadeira.
- Comando **for**: repete uma sequência de instruções "**N** vezes".

Exemplo for – exemplo 2

```c
#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;
}
```

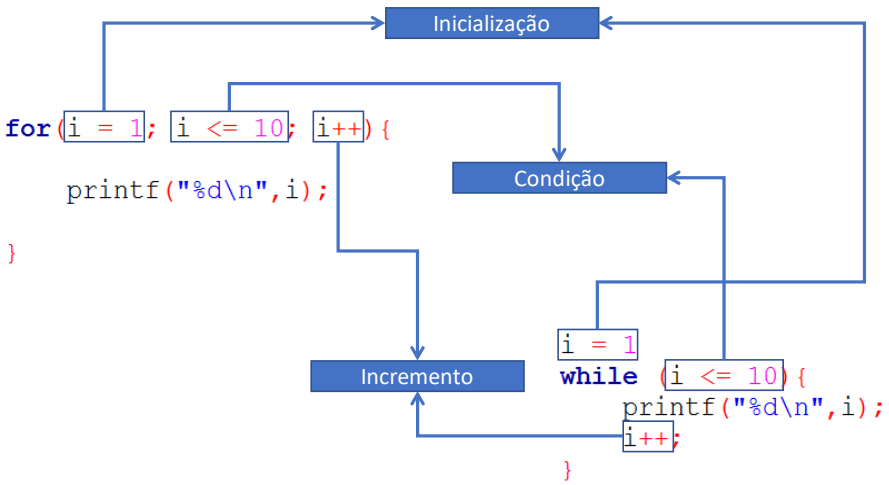
Exemplo for



25

for *x* while

- Exemplo: mostra os valores de 1 até 10



26

Comando for

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

```c
#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;
}
```

27

Comando for

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

```c
#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;
}
```

28

Comando for

- Cuidado: for sem incremento
  - omitir o incremento pode criar 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;
}
```

29

Exercício: Escrever um programa para calcular a média de N números, usando for. O valor de N é dado pelo usuário.

- Vamos fazer juntos!

30