

# ESTRUTURA DE REPETIÇÃO

# Programação estruturada

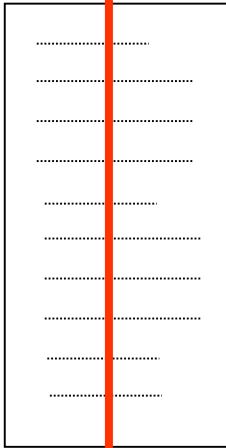
2

- Define que os programas podem ser resumidos em conjuntos de três estruturas diferentes:
  - ▣ Estrutura sequencial.
  - ▣ Estrutura de decisão.
  - ▣ Estrutura de repetição.
  
- Cada estrutura define a sequência e a quantidade de vezes que as instruções serão executadas.

# Programação estruturada

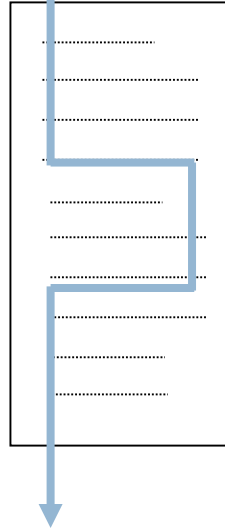
3

fluxo  
de execução  
sequencial



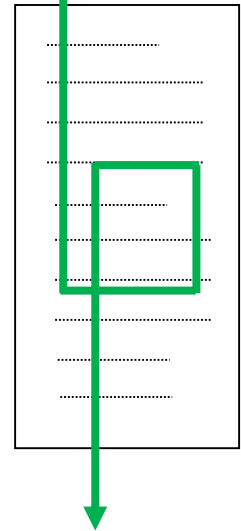
os comandos são executados um após o outro sequencialmente do início ({} até o final {})

fluxo  
de execução  
com decisão



os comandos são executados dependendo do valor de uma condição, ou expressão lógica:  
if, if else e switch case

fluxo  
de execução  
com repetição



os comandos são executados, de forma repetida, um determinado número de vezes:  
for, while e do while

# Estrutura sequencial

4

- As instruções são executadas da primeira até a última, na ordem que foram descritas.
  - ▣ Cima para baixo, esquerda para direita, respeitando regra de precedência em operação matemática.
- **Todas as instruções do código serão executadas.**

# Estrutura sequencial

5

```
#include <stdio.h>

int main(void)
{
    int a, b;
    int r1;
    float r4;

    printf("Informe o valor de a: ");
    scanf("%d", &a);
    printf("Informe o valor de b: ");
    scanf("%d", &b);

    r1 = a + b;
    printf("Soma de %d e %d = %d\n", a, b, r1);

    r4 = ((float) a) / ((float) b);
    printf("Divisao de %d e %d = %.2f\n", a, b, r4);

    return 0;
}
```

# Estrutura de decisão

6

- Determina se um conjunto de instruções será ou não executado, dependendo de um teste lógico.
- Testes lógicos são realizados entre valores (variáveis e constantes) através de *operadores relacionais* e *operadores lógicos*.
  - ▣ O resultado de testes lógicos é representado por apenas dois valores:
    - Verdadeiro: 1;
    - Falso: 0;

# Estrutura de decisão

7

```
int main(void)
{
    int A = 35, B = 10, C = 5, D = 3;
    float X = 0;

    if( A > 0 )
    {
        if( A < 10 )
            X = 10 / A;
    }
    else if( B >= 10 && C < 20)
        X = B + C / 5;
    else if( D != 5 )
        X = 4;
    else
        X = C + !D;

    printf("%f", X);

    return 0;
}
```

# Estrutura de repetição

8

## □ Problema:

- Realizar a leitura de duas notas.
- Calcular a média e exibir se o aluno foi aprovado, reprovado, ou está em recuperação.
- Realizar a operação para uma turma de 50 alunos.
  - Executar o programa 50 vezes?
  - Copiar e colar a estrutura de decisão 50 vezes?



```
int main(void)
{
    float nota1, nota2, media;

    printf("Informe a nota da P1: ");
    scanf("%f", &nota1);
    printf("Informe a nota da P2: ");
    scanf("%f", &nota2);
    media = (nota1 + nota2) / 2;
    if( media >= 6 ) printf("Aluno 1 Aprovado.\n");
    else if( media >= 5 && media < 7 ) printf("Aluno 1 Em recuperação.\n");
    else printf("Aluno 1 Reprovado.\n");

    printf("Informe a nota da P1: ");
    scanf("%f", &nota1);
    printf("Informe a nota da P2: ");
    scanf("%f", &nota2);
    media = (nota1 + nota2) / 2;
    if( media >= 6 ) printf("Aluno 2 Aprovado.\n");
    else if( media >= 5 && media < 7 ) printf("Aluno 2 Em recuperação.\n");
    else printf("Aluno 2 Reprovado.\n");

    // ...

    printf("Informe a nota da P1: ");
    scanf("%f", &nota1);
    printf("Informe a nota da P2: ");
    scanf("%f", &nota2);
    media = (nota1 + nota2) / 2;
    if( media >= 6 ) printf("Aluno 50 Aprovado.\n");
    else if( media >= 5 && media < 7 ) printf("Aluno 50 Em recuperação.\n");
    else printf("Aluno 50 Reprovado.\n");

    return 0;
}
```

# Estrutura de repetição

10

- **Repete** um conjunto de instruções enquanto um teste lógico for verdadeiro.
- Testes lógicos são realizados entre valores (variáveis e constantes) através de *operadores relacionais* e *operadores lógicos*.
  - ▣ O resultado de testes lógicos é representado por apenas dois valores:
    - Verdadeiro: 1;
    - Falso: 0;

# Estrutura de repetição

11

- **Repete** um conjunto de instruções enquanto um teste lógico for verdadeiro.
- Laços de repetição ou *Loops*.
- A repetição pode:
  - ▣ Ter o número de repetições conhecido.
  - ▣ Ser indeterminada e depender de uma leitura ou do resultado de um cálculo interno.

# Estrutura de repetição

12

- **Repete** um conjunto de instruções enquanto um teste lógico for verdadeiro.
  
- Principais estruturas de repetição:
  - ▣ while (enquanto);
  - ▣ do ... while (faça ... enquanto);
  - ▣ for (para);

# Estrutura de repetição: while

13

- **while()**
- **Sintaxe:**

```
while ( CONDIÇÃO )  
{  
    INSTRUÇÕES;  
}
```

- Executa as *instruções* **enquanto** a condição for verdadeira.

# Estrutura de repetição: while

14

## □ Sintaxe:

```
while ( CONDIÇÃO )  
{  
    INSTRUÇÕES;  
}
```

## □ **condição**: expressão lógica que verifica se o critério final foi atingido.

▣ Ex:  $i \leq 100$ ;

▣ Ex:  $j > 0$ ;

▣ Verifica **antes** de executar as instruções pela primeira vez.

# Estrutura de repetição: while

15

## □ Sintaxe:

```
while ( CONDIÇÃO )  
{  
    INSTRUÇÕES;  
}
```

- **instruções**: contém os comandos que são executados enquanto a condição for verdadeira.

- Incluir a alteração da condição para interromper o loop.

- Ex: incremento de variável  $i++$ ;
- Ex: leitura de um valor com `scanf()`;

# Estrutura de repetição: while

16

## □ Exemplo:

```
int main(void)
{
    int num;

    num = 0;

    while( num != 100 )
    {
        printf("Digite um numero: ");
        scanf("%d", &num);
    }

    printf("Ate que enfim digitou 100.\n");

    return 0;
}
```



# Estrutura de repetição: while

17

## □ Exemplo:

```
int main(void)
{
    int cont = 0;

    while( cont < 50 )
    {
        printf("Contador = %d \n", cont);
        cont++;
    }

    printf("Contador final = %d \n", cont);

    return 0;
}
```

# Estrutura de repetição: while

18

## Exemplo:

```
int main(v  
{  
    int co  
    while(  
    {  
        pr  
        co  
    }  
    printf  
    return  
}
```

```
C:\Users\Muriel\Desktop\exemplos.exe  
Contador = 39  
Contador = 40  
Contador = 41  
Contador = 42  
Contador = 43  
Contador = 44  
Contador = 45  
Contador = 46  
Contador = 47  
Contador = 48  
Contador = 49  
Contador final = 50
```

# Estrutura de repetição: while

19

## □ Exemplo:

```
int main(void)
{
    int cont = 100;

    while( cont >= 0 )
    {
        printf("Contador = %d \n", cont);
        cont--;
    }

    printf("Contador final = %d \n", cont);

    return 0;
}
```

# Estrutura de repetição: do while

20

- **do while()**

- **Sintaxe:**

```
do
{
    INSTRUÇÕES;
}while ( CONDIÇÃO );
```

- Executa as *instruções* **enquanto** a condição for verdadeira.

# Estrutura de repetição: do while

21

## □ Sintaxe:

**do**

{

INSTRUÇÕES;

} **while** ( CONDIÇÃO );

- **condição**: expressão lógica que verifica se o critério final foi atingido.

- Ex:  $i \leq 100$ ;

- Ex:  $j > 0$ ;

- Verifica **depois** de executar as instruções pela primeira vez.

# Estrutura de repetição: do while

22

## □ Sintaxe:

```
do  
{  
    INSTRUÇÕES;  
} while ( CONDIÇÃO );
```

- **instruções**: contém os comandos que são executados enquanto a condição for verdadeira.

- Incluir a alteração da condição para interromper o loop.

- Ex: incremento de variável  $i++$ ;
- Ex: leitura de um valor com `scanf()`;

# Estrutura de repetição: do while

23

## □ Exemplo:

```
int main(void)
{
    float num;

    do
    {
        printf("Digite um numero positivo: ");
        scanf("%f", &num);
    } while( num <= 0 );

    printf("Valor positivo = %f \n", num);

    return 0;
}
```

# Estrutura de repetição: do while

24

## □ Exemplo:

```
int main(void)
```

```
{
```

```
    C:\Users\Muriel\Desktop\exemplos.exe
```

```
Digite um numero positivo: -8
```

```
Digite um numero positivo: 0
```

```
Digite um numero positivo: -36
```

```
Digite um numero positivo: -1
```

```
Digite um numero positivo: 50
```

```
Valor positivo = 50.000000
```

```
);
```

```
    return 0;
```

```
}
```



# Estrutura de repetição: do while

25

## □ Exemplo:

```
int main(void)
{
    char letra;

    do
    {
        printf("Digite uma letra: ");
        scanf(" %c", &letra); //espaço antes de %c
    } while( letra != 'A' && letra != 'a' );

    printf("Letra 'A' ou letra 'a' = %c \n", letra);

    return 0;
}
```

# Estrutura de repetição: do while

26

## □ Exemplo:

```
int main(void)
{
    int cont = 0;

    do
    {
        printf("contador = %d\n", cont);
        cont++;
    }while( cont < 35 );

    printf("Contador final = %d\n", cont);

    return 0;
}
```

# Estrutura de repetição: do while

27

## □ Exemplo:

```
int main()
{
    int contador = 27
    do
    {
        contador = 28
        contador = 29
        contador = 30
        contador = 31
    } while (contador < 32)
    printf("Contador = %d\n", contador);
    contador = 33
    printf("Contador = %d\n", contador);
    contador = 34
    printf("Contador final = %d\n", contador);
    return 0;
}
```

# Estrutura de repetição: do while

28

## □ Exemplo:

```
int main(void)
{
    int cont = 0;

    do
    {
        cont++;
        printf("contador = %d\n", cont);
    }while( cont < 35 );

    printf("Contador final = %d\n", cont);

    return 0;
}
```

# Estrutura de repetição: do while

29

## □ Exemplo:

```
int main()
{
    int contador = 28
    do
    {
        contador = 30
        contador = 31
        contador = 32
    } while (contador < 33)
    printf("Contador = %d\n", contador);
    contador = 34
    contador = 35
    return 0;
}
```

C:\Users\Muriel\Desktop\exemplos.exe

contador = 28

contador = 29

contador = 30

contador = 31

contador = 32

contador = 33

contador = 34

contador = 35

Contador final = 35

cont) ;

cont) ;

# Estrutura de repetição

30

- **Regra geral de while e do while:**
  - ▣ Sempre **inicializar** as variáveis da condição **antes** da estrutura de repetição.
  - ▣ Sempre incluir a **alteração** das variáveis da condição **dentro das instruções**.
    - De preferência colocar como a **última linha** do bloco de repetição.

# Estrutura de repetição: for

31

- **for()**

- **Sintaxe:**

```
for ( INICIALIZAÇÃO ; CONDIÇÃO ; INCREMENTO/DECREMENTO )  
{  
    INSTRUÇÕES;  
}
```

- Executa as *instruções* **começando na inicialização e enquanto** a *condição* for verdadeira.

# Estrutura de repetição: for

32

## □ Sintaxe:

```
for ( INICIALIZAÇÃO ; CONDIÇÃO ; INCREMENTO/DECREMENTO )  
{  
    INSTRUÇÕES;  
}
```

□ **inicialização**: expressão que determina o início.

▣ Ex:  $i = 0$ ;

▣ Ex:  $j = 20$ ;



# Estrutura de repetição: for

33

## □ Sintaxe:

```
for ( INICIALIZAÇÃO ; CONDIÇÃO ; INCREMENTO/DECREMENTO )  
{  
    INSTRUÇÕES;  
}
```

- **condição**: expressão lógica que verifica se o critério final foi atingido.

- Ex:  $i \leq 100$ ;

- Ex:  $j > 0$ ;

- Verifica **antes** de executar as instruções pela primeira vez. **Após** a inicialização.

# Estrutura de repetição: for

34

## □ Sintaxe:

```
for ( INICIALIZAÇÃO ; CONDIÇÃO ; INCREMENTO/DECREMENTO )  
{  
    INSTRUÇÕES;  
}
```

- **incremento/decremento**: expressão para determinar a quantidade de iterações realizadas/faltantes.

Determina o passo de cada iteração.

- Ex:  $i++$ ;

- Ex:  $j--$ ;

- Sempre incrementará após as instruções.

# Estrutura de repetição: for

35

## □ Sintaxe:

```
for ( INICIALIZAÇÃO ; CONDIÇÃO ; INCREMENTO/DECREMENTO )  
{  
    INSTRUÇÕES;  
}
```

- ***instruções***: contém os comandos que são executados enquanto a condição for verdadeira.

# Estrutura de repetição: for

36

## □ Exemplo:

```
int main(void)
{
    int cont;

    for(cont = 0; cont < 50; cont++)
    {
        printf("Contador = %d\n", cont);
    }

    printf("Contador final = %d\n", cont);

    return 0;
}
```

# Estrutura de repetição: for

37

## □ Exemplo:

```
int main (
{
    int Contador = 42
    for (Contador = 43; Contador <= 50; Contador++)
    {
        printf("Contador = %d\n", Contador);
    }
    printf("Contador final = 50\n");
    return 0;
}
```

# Estrutura de repetição: for

38

## □ Exemplo:

```
int main(void)
{
    int i, soma;

    soma = 0;
    for(i = 0; i < 20; i++)
    {
        soma += i;
    }

    printf("Somatorio = %d\n", soma);

    return 0;
}
```

# Estrutura de repetição: for

39

## □ Exemplo:

```
int main(void)
{
    int i, inicio, fim;

    printf("Informe o inicio: ");
    scanf("%d", &inicio);
    printf("Informe o final: ");
    scanf("%d", &fim);

    for(i = inicio; i <= fim; i++)
    {
        printf("i = %d\n", i);
    }

    printf("i final = %d\n", i);

    return 0;
}
```

# Estrutura de repetição: for

40

## □ Exemplo:

```
int main(void)
```

```
{ C:\Users\Murie\Desktop\exemplos.exe
```

```
Informe o inicio: 5
```

```
Informe o final: 10
```

```
i = 5
```

```
i = 6
```

```
i = 7
```

```
i = 8
```

```
i = 9
```

```
i = 10
```

```
i final = 11
```

```
}
```



# Estrutura de repetição: for

41

## □ Exemplo:

```
int main(void)
{
    int i, fatorial, num;

    printf("Informe o valor para calcular: ");
    scanf("%d", &num);

    fatorial = 1;
    for(i = num; i > 0; i--)
    {
        fatorial *= i;
    }

    printf("Fatorial = %d\n", fatorial);

    return 0;
}
```

# Estrutura de repetição: for

42

## □ Exemplo:

```
int main(void)
{
    int i, fatorial, num;

    printf("Informe o valor para calcular: ");
    scanf("%d", &num);

    fatorial = 1;
    for(i = num; i > 0; i--)
    {
        fatorial *= i;
    }

    printf("Fatorial = %d\n", fatorial);

    return 0;
}
```

SE O USUÁRIO INSERIR  
UM VALOR NEGATIVO?

# Estrutura de repetição

43

## □ Exemplo:

```
int main(void)
{
    int i, fatorial, num;

    do{
        printf("Informe o valor para calcular: ");
        scanf("%d", &num);
    }while(num < 0);

    fatorial = 1;
    for(i = num; i > 0; i--)
    {
        fatorial *= i;
    }

    printf("Fatorial = %d\n", fatorial);

    return 0;
}
```

# Estrutura de repetição: Problemas

44

- ❑ Entrar em loop infinito (não para de executar):
  - ❑ Variável de controle não inicializada.
  - ❑ Teste inconsistente: condição nunca é alcançada.
  - ❑ Incremento/decremento inconsistente: valor não é compatível com a expressão lógica da condição.
  - ❑ Instruções inconsistentes: altera a variável da condição de parada.

# Estrutura de repetição

45

- Indispensável:
  - ▣ Identificar instruções que se repete;
  - ▣ Identificar condição de parada;
  - ▣ Definir forma de alcançar a condição de parada;
  
- Em loops com quantidade fixa:
  - ▣ Inicializar variáveis de controle;
  - ▣ Incrementar/Decrementar variáveis até alcançar a condição de parada.

# Estrutura de repetição

46

## ❑ Problema:

- ❑ Realizar a leitura de duas notas.
- ❑ Calcular a média e exibir se o aluno foi aprovado, reprovado, ou está em recuperação.
- ❑ Realizar a operação para uma turma de 50 alunos.
  - Executar o programa 50 vezes?
  - Copiar e colar a estrutura de decisão 50 vezes?
  - USAR ESTRUTURA DE REPETIÇÃO!



```
int main(void)
{
    int i;
    float nota1, nota2, media;

    for(i = 0; i < 50; i++)
    {
        printf("Informe a nota da P1: ");
        scanf("%f", &nota1);
        printf("Informe a nota da P2: ");
        scanf("%f", &nota2);

        media = (nota1 + nota2) / 2;

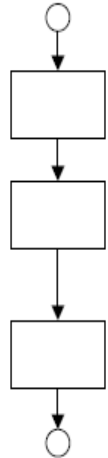
        if( media >= 6 )
            printf("Aluno %d Aprovado.\n", i + 1);
        else if( media >= 5 && media < 7 )
            printf("Aluno %d Em recuperação.\n", i + 1);
        else
            printf("Aluno %d Reprovado.\n", i + 1);
    }

    return 0;
}
```

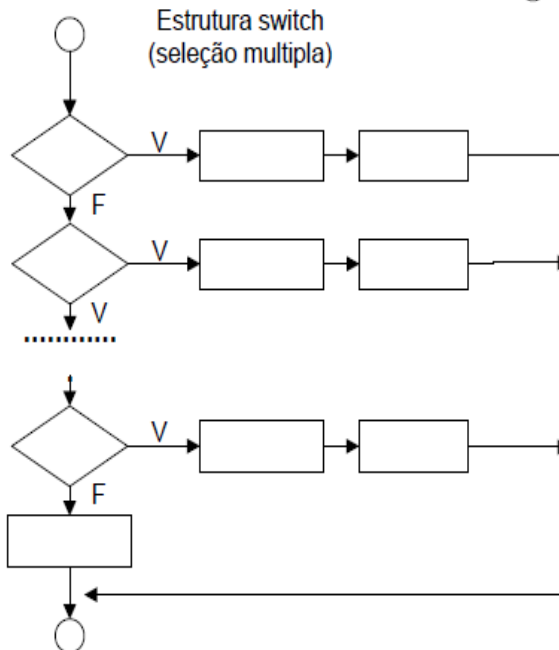
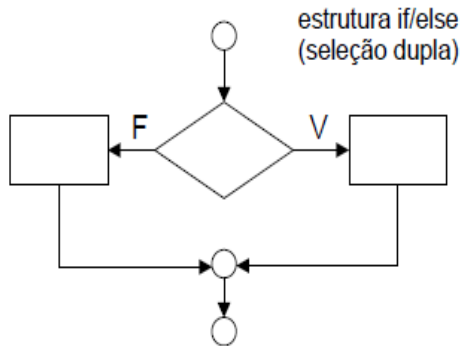
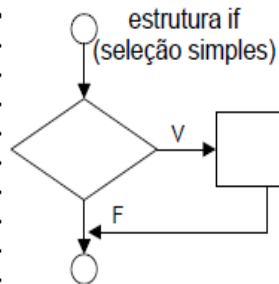
# Estruturas

48

## Sequencial

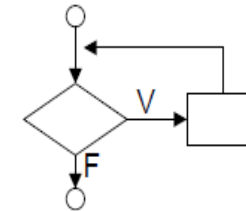


## Decisão

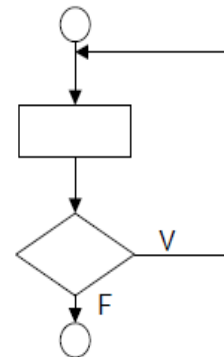


## Repetição

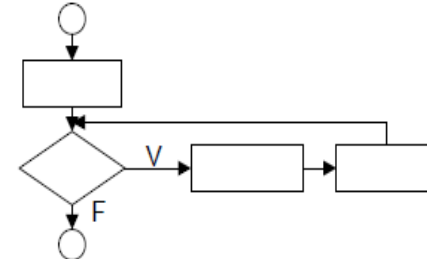
### Estrutura while



### Estrutura do/while



### Estrutura for

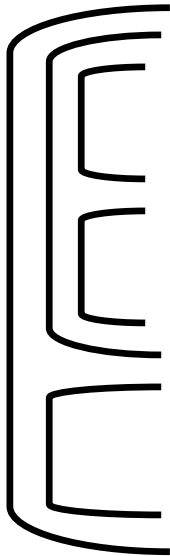




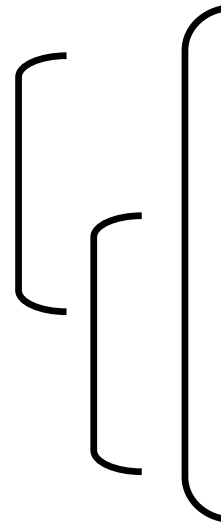
# Aninhamento

49

- Quando estruturas de controle estão inseridas uma dentro da outra;
- A estrutura mais interna deve estar inteiramente contida na estrutura imediatamente mais externa.



Válido



Inválido

# Aninhamento

50

```
#include <stdio.h>
int main(void)
{
    int num, qtd = 0, senha = 1234, senha_root = 4321;
    do{
        printf("Digite a senha: ");
        scanf("%d", &num);

        qtd ++;
        if(qtd == 5)
        {
            printf("Numero maximo de tentativas.\n");
            break;
        }

        if(num == senha_root)
        {
            printf("Acesso root.\n");
            break;
        }

    }while(num!=senha && num!=1000);

    if(num == 1000 || num == senha) printf("Acesso usuario.\n");

    return 0;
}
```

# Aninhamento

51

**break FORÇA  
A PARADA DA  
ESTRUTURA  
DE  
REPETIÇÃO.**

```
#include <stdio.h>
int main(void)
{
    int num, qtd = 0, senha = 1234, senha_root = 4321;
    do{
        printf("Digite a senha: ");
        scanf("%d", &num);

        qtd ++;
        if(qtd == 5)
        {
            printf("Numero maximo de tentativas.\n");
            break;
        }

        if(num == senha_root)
        {
            printf("Acesso root.\n");
            break;
        }

    }while(num!=senha && num!=1000);

    if(num == 1000 || num == senha) printf("Acesso usuario.\n");

    return 0;
}
```

# Aninhamento

52

```
int main(void)
{
    int i, j, tam;

    printf("Informe o tamanho do lado do quadrado: ");
    scanf("%d", &tam);

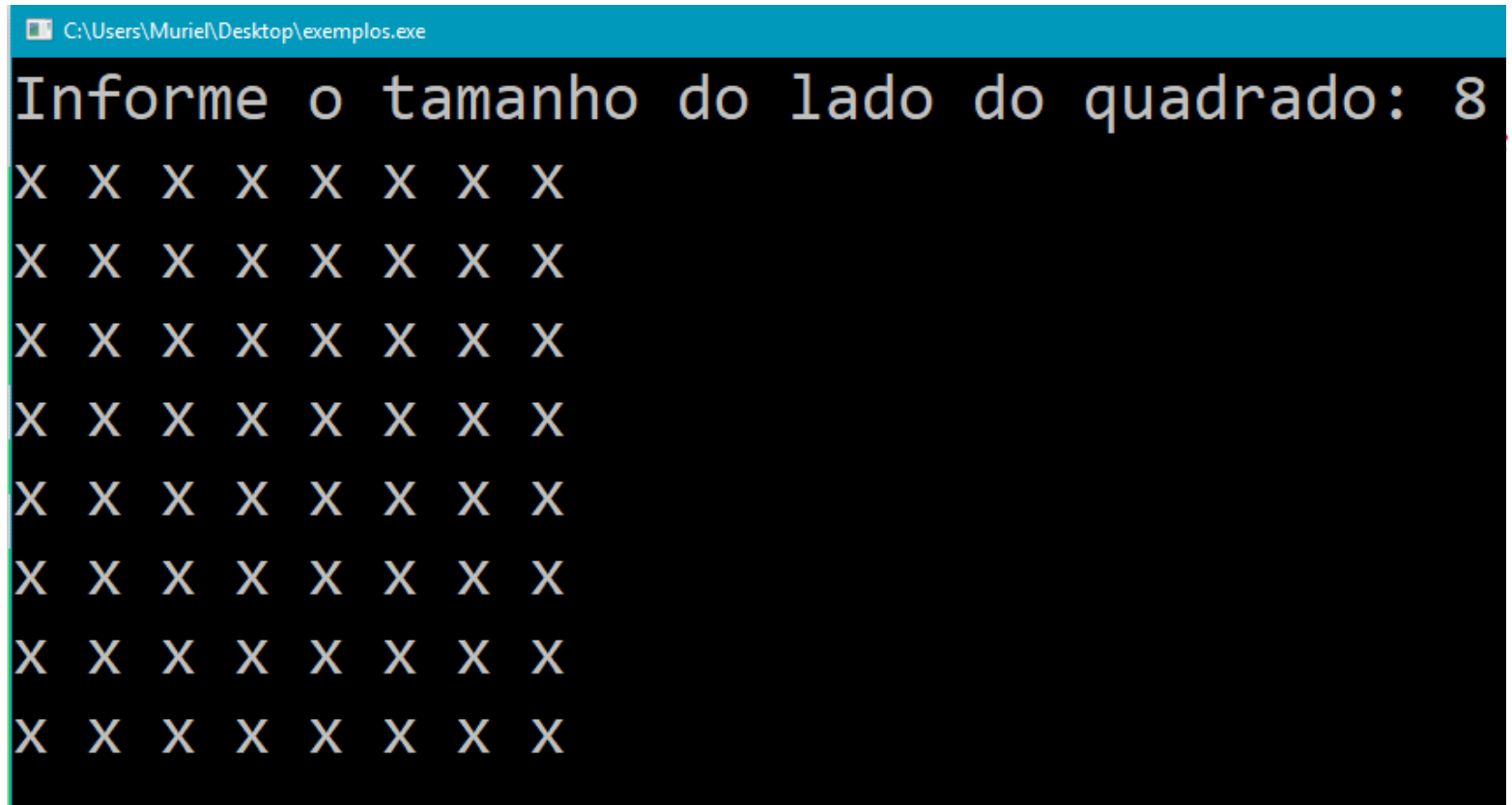
    for(i = 0; i < tam; i++)
    {
        for(j = 0; j < tam; j++)
        {
            printf("x ");
        }
        printf("\n");
    }

    return 0;
}
```

# Aninhamento

53

```
int main(void)
{
```



The screenshot shows a Windows command prompt window with the title bar "C:\Users\MurIEL\Desktop\exemplos.exe". The prompt displays the text "Informe o tamanho do lado do quadrado: 8". Below this, an 8x8 grid of 'X' characters is printed, forming a square. The grid consists of 8 rows and 8 columns of 'X' characters.

```
}
```

# Questionário de revisão

54

- 1 Como pode ser criada uma estrutura de repetição em C? Comente sobre cada tipo.
  
- 2 Qual a diferença básica entre for e do while?

# Estrutura de repetição: Exercícios

55

- 1) Imprimir os números de 100 a 0 (nessa ordem) utilizando uma estrutura while.

# Estrutura de repetição: Exercícios

56

- 2) Ler números informados pelo usuário. Parar a leitura quando for informado o número zero, que não deve ser considerado.
  - ▣ Faça a média dos valores negativos informados.
  - ▣ Conte quantos dos valores positivos são ímpares.



# Estrutura de repetição: Exercícios

57

- 3) Uma árvore A tem 1,50 metros e cresce 2 centímetros por ano, enquanto uma árvore B tem 1,10 metros e cresce 3 centímetros por ano. Construa um algoritmo que calcule e imprima quantos anos serão necessários para que B seja maior que A.