

**Curso de Bacharelado em Ciência da Computação**  
**3º Semestre**

# **Estruturas de Dados I**

*Aula 02 – Revisão de Programação - Parte 01*

**Prof. Me. Dárley Domingos de Almeida**

**Universidade do Estado de Mato Grosso – UNEMAT**

**Alto Araguaia – 2024/01**

**<< Revisão >>**

# **Conceitos Básicos de Programação**

# ➤ BIBLIOTECAS

```
//Chamada de bibliotecas
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>

main()
{
    char palavra[40], r[20];
    int nl, i, j;

    //Entrada de dados
    printf("Digite uma palavra: ");
    fflush(stdin);
    scanf("%s", palavra);

    //Processamento

    nl=strlen(palavra);
    if(nl%2==0)
        j=nl/2;
    else
        j=(nl-1)/2;
    for(i=0;i<=j-1;i++)
    {
        if(palavra[i]==palavra[nl-1])
        {
            strcpy(r,"eh palindrome");
        }
    }
}
```

**#include<stdio.h>**: biblioteca que contém a definição da estrutura *FILE*, usada para todas as entradas e saídas.

**#include<stdlib.h>**: biblioteca que possui um conjunto de funções padrão que não se enquadram em outras bibliotecas.

**#include<conio.h>**: contém funções para parar a execução do software até um toque do teclado, limpar a tela e etc.

**#include<string.h>**: biblioteca que possui um conjunto de funções específicas para manipulação de strings.

```

//Chamada de bibliotecas

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>

main()
{
    char palavra[10], r[20];
    int nl,i,j;
    //Entrada de dados

    printf("Digite uma palavra: ");
    fflush(stdin);
    scanf("%s",palavra);

    //Processamento

    nl=strlen(palavra);
    if(nl%2==0)
        j=nl/2;
    else
        j=(nl-1)/2;
    for(i=0;i<=j-1;i++)
    {
        if(palavra[i]==palavra[nl-1-i])
        {
            strcpy(r,"eh palindrome");
        }
        else
    }
}

```

**main()** primeira e principal função, cuja finalidade é iniciar o bloco de dados.

**int main(int argc, char \*argv[])**



# ➤ ENTRADA & SAÍDA DE DADOS

```
//Chamada de bibliotecas

#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#include<string.h>

main()
{
    char palavra[40], r[20];
    int nl,i,j;

    //Entrada de dados

    printf("Digite uma palavra: ");
    fflush(stdin);
    scanf("%s",palavra);

    //Processamento

    nl=strlen(palavra);
    if(nl%2==0)
        j=nl/2;
    else
        j=(nl-1)/2;
    for(i=0;i<=j-1;i++)
    {
        if(palavra[i]==palavra[nl-1])
        {
            strcpy(r,"eh palindrome");
        }
    }
}
```

*printf("...")* : função de SAÍDA de dados que permite escrever na tela o que o programador desejar, no caso a frase *"Digite uma palavra:"*.

*fflush(stdin)* basicamente ‘limpa’ o buffer (memória temporária) da entrada padrão (teclado).

*scanf("%s", palavra)* : função de ENTRADA de dados responsável por armazenar o que foi digitado pelo usuário, no caso a série de caracteres (*%s*) digitada será armazenada na variável *palavra*, um a um e seqüencialmente.

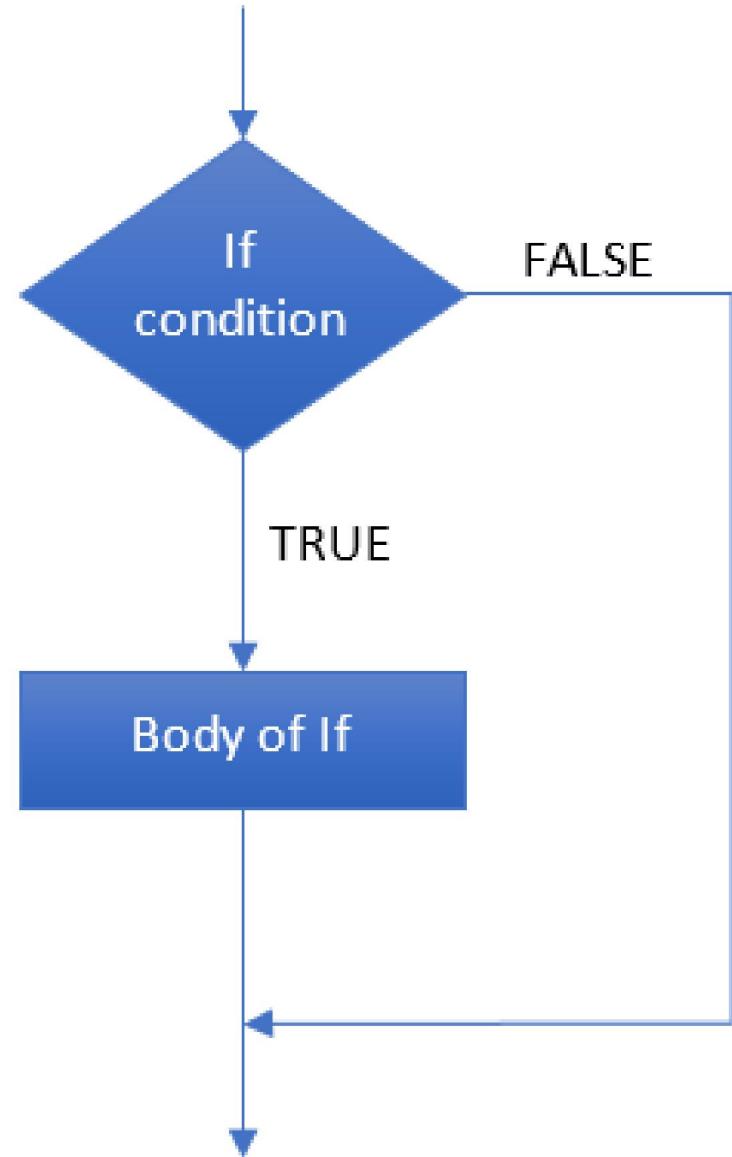
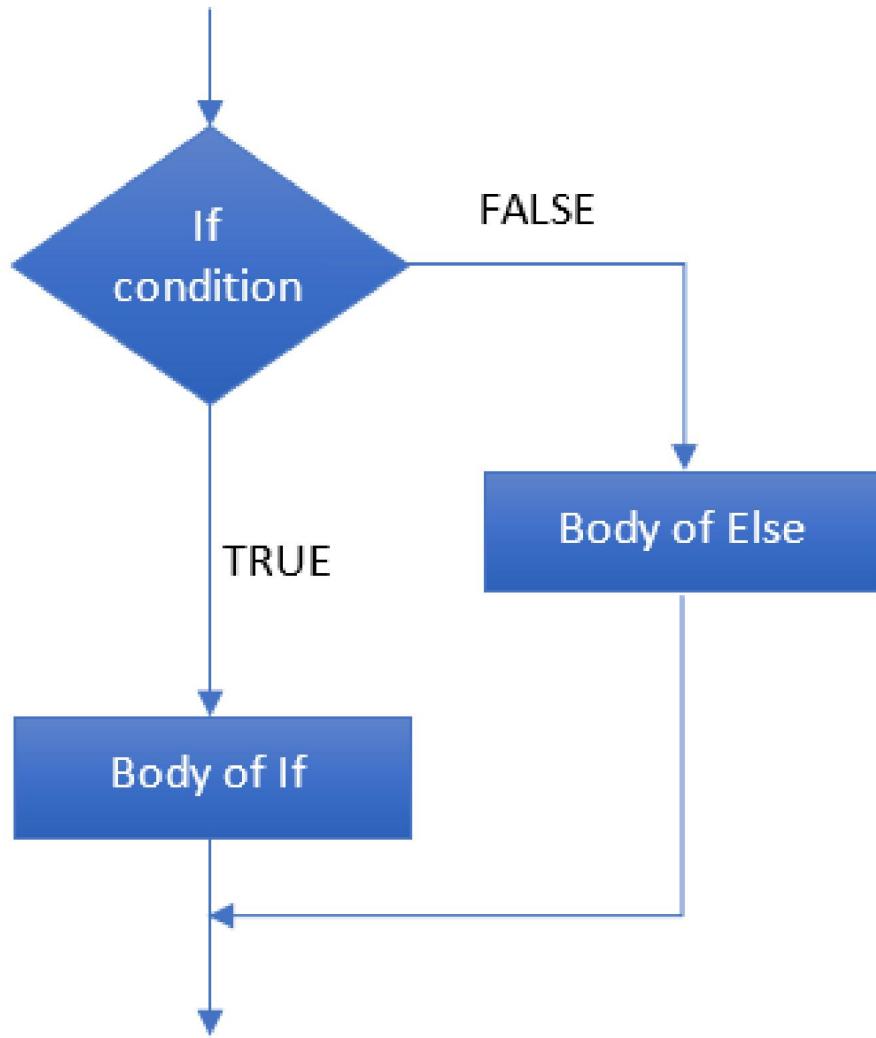
# Estruturas de Controle & Repetição

## if ... else

qualquer expressão que possa ser avaliada como *verdadeira* ou *falsa*

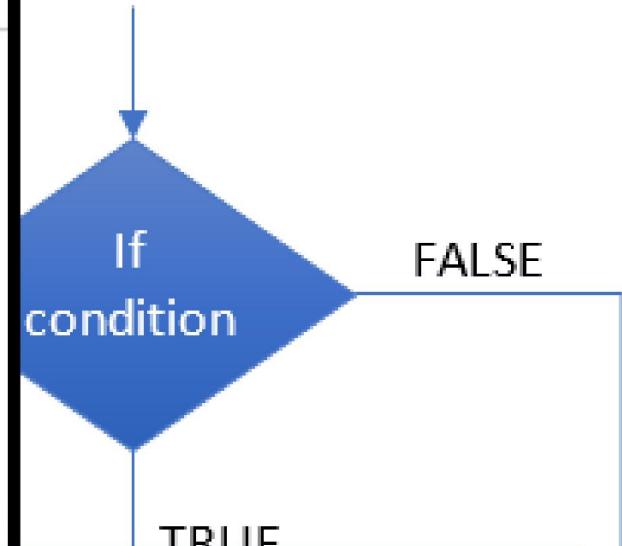
```
if ( expressão_for_verdade ) {  
    faça alguma coisa;  
}  
else {  
    faça outra coisa  
}
```

# if ... else



# if ... else

```
[*] main.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 main()
5 {
6     int x;
7     scanf("%d", &x);
8     if (x>10)
9     {
10         printf("o numero e maior que 10");
11     }
12     else
13     {
14         printf("o numero e menor ou igual a 10");
15     }
16     return 0;
17 }
```



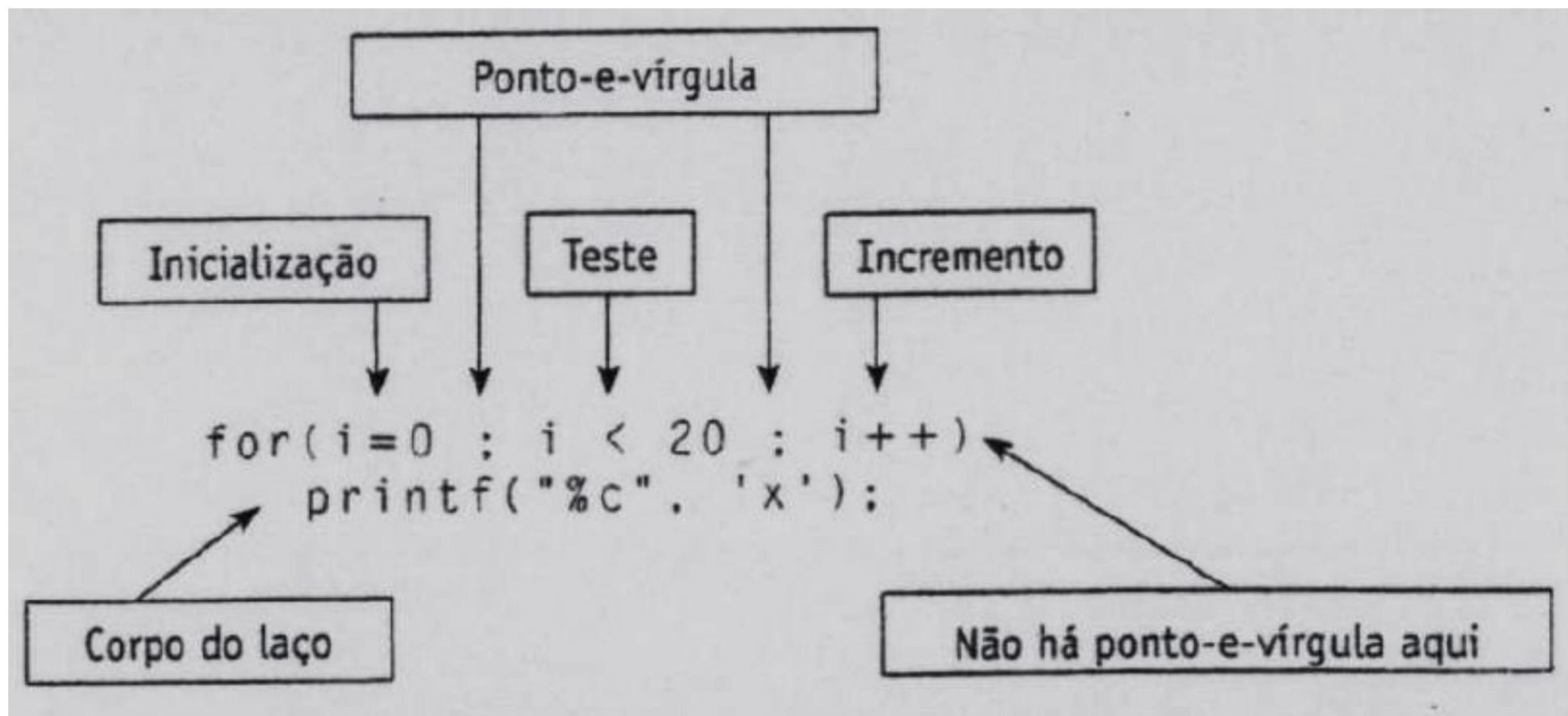
```
ternario.c
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int numero, x;
6     scanf("%d", &numero);
7
8     x = (numero > 10) ? 1 : 0;
9
10    printf("%d", x);
11
12    return 0;
13 }
```

## switch case

```
int roll = 3 ;
switch( roll )
{
    case 1 :
        printf("I am Pankaj");
        break;
    case 2 :
        printf("I am Nikhil");
        break;
    case 3 :
        printf("I am John");
        break;
    default :
        printf("No student found");
        break;
}
```

```
graph TD; A[switch] --> B["{"]; C[case 1] --> D["printf"]; E[case 2] --> F["printf"]; G[case 3] --> H["printf"]; I[default] --> J["printf"]; K[break] --> L[break]; L --> M["}"];
```

# for



# for

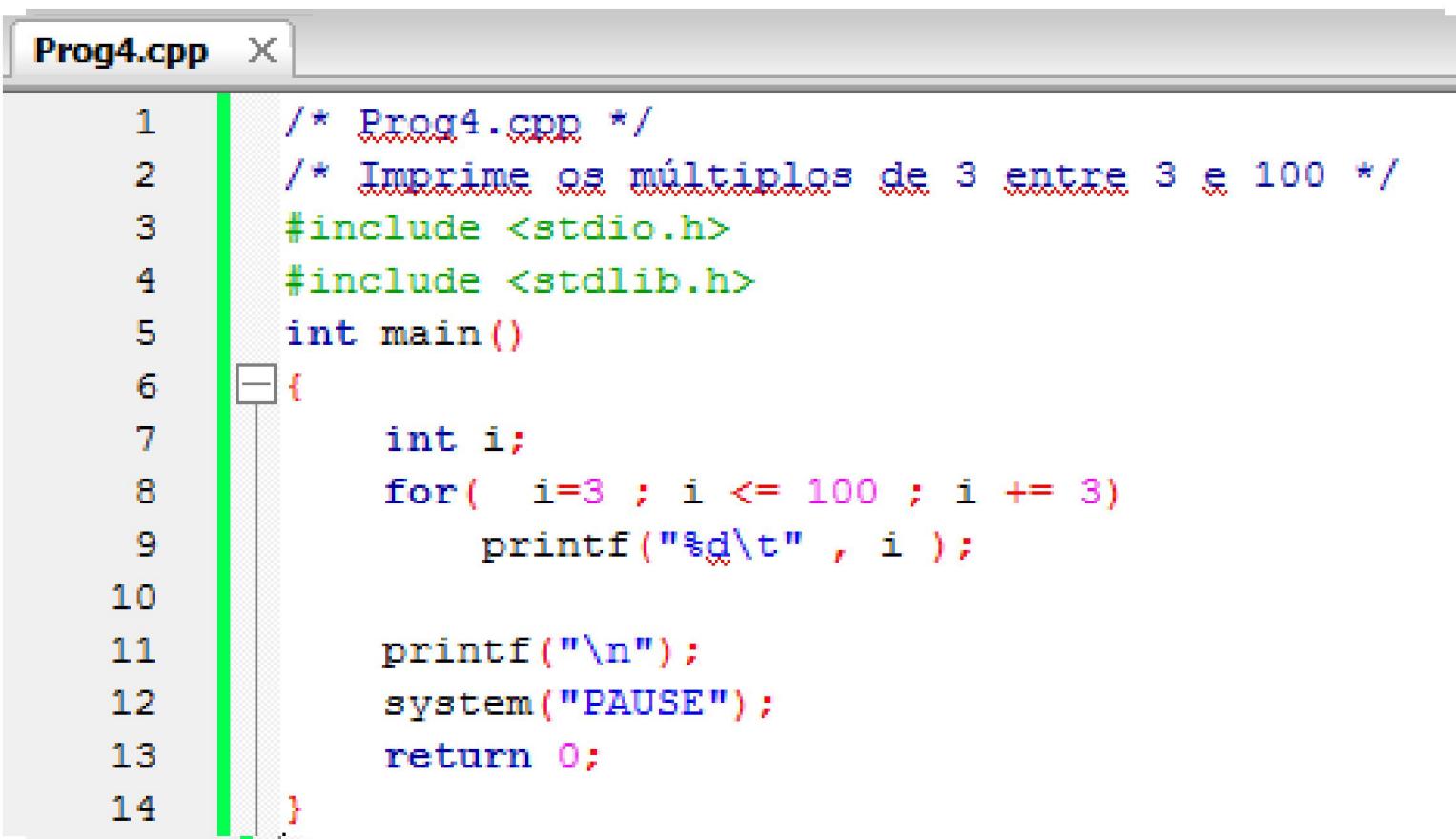
```
Prog1.cpp X

1  /* Prog1.cpp */
2  /* Mostra um uso simples do laço for */
3  #include <stdio.h>
4  #include <stdlib.h>
5  int main()
6  {
7      int i;
8      for( i=0 ; i < 20 ; i++ ) /* Imprime 20 */
9          printf("%c", '*');
10
11     printf("\n");
12     system("PAUSE");
13     return 0;
14 }
```

# for

```
Prog3.cpp X
1  /* Prog3.cpp */
2  /* Imprime a tabuada do 6 invertida */
3  #include <stdio.h>
4  #include <stdlib.h>
5  int main()
6  {
7      int i;
8      for( i=9 ; i > 0 ; i-- )
9          printf("\n%4d x 6 = %4d" , i , i*6);
10
11     printf("\n");
12     system("PAUSE");
13     return 0;
14 }
```

# for



```
1  /* Prog4.cpp */
2  /* Imprime os múltiplos de 3 entre 3 e 100 */
3  #include <stdio.h>
4  #include <stdlib.h>
5  int main()
6  {
7      int i;
8      for( i=3 ; i <= 100 ; i += 3)
9          printf("%d\t", i);
10
11     printf("\n");
12     system("PAUSE");
13     return 0;
14 }
```

# for

```
Prog5.cpp X
1  /* Prog5.cpp */
2  /* Mostra o uso do operador vírgula no laço for */
3  /* Imprime os números de 0 a 98 de 2 em 2 */
4  #include <stdio.h>
5  #include <stdlib.h>
6  int main()
7  {
8      int i,j;
9      for(i=0, j=i; (i+j) < 100 ; i++, j++)
10         printf("%d ", i +j);
11
12     printf("\n");
13     system("PAUSE");
14     return 0;
15 }
```

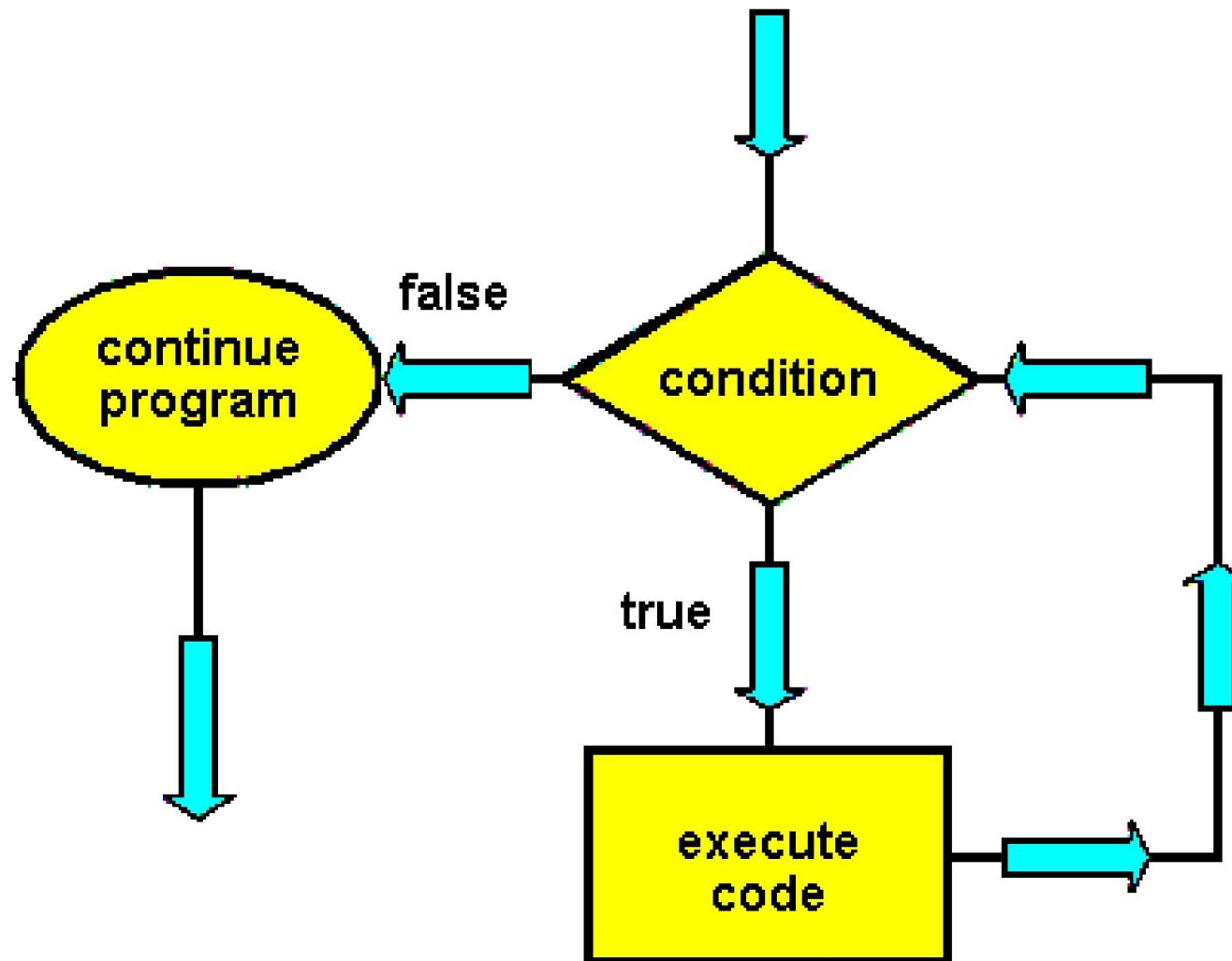
# for

```
Prog6.cpp X
1  /* Prog6.cpp */
2  /* Mostra o uso de uma variável do tipo char para controle do laço for
3   * Imprime as letras minúsculas e seus correspondentes valores
4   * em decimal na tabela ASCII */
5  #include <stdio.h>
6  #include <stdlib.h>
7  int main()
8  {
9      char ch;
10     for(ch='a'; ch <= 'z'; ch++)
11         printf("\nO valor ASCII de %c é %d", ch , ch);
12
13     printf("\n");
14     system("PAUSE");
15     return 0;
16 }
```

# for

```
Prog8.cpp X
1  /* Prog8.cpp */
2  /* Codifica a entrada digitada */
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <conio.h> /* para getch() */
6  int main()
7  {
8      unsigned char ch;
9      for( ; (ch=getch()) != 'X' ; )
10         printf("%c", ch +1);
11
12     printf("\n");
13     system("PAUSE");
14     return 0;
15 }
```

# while



# while

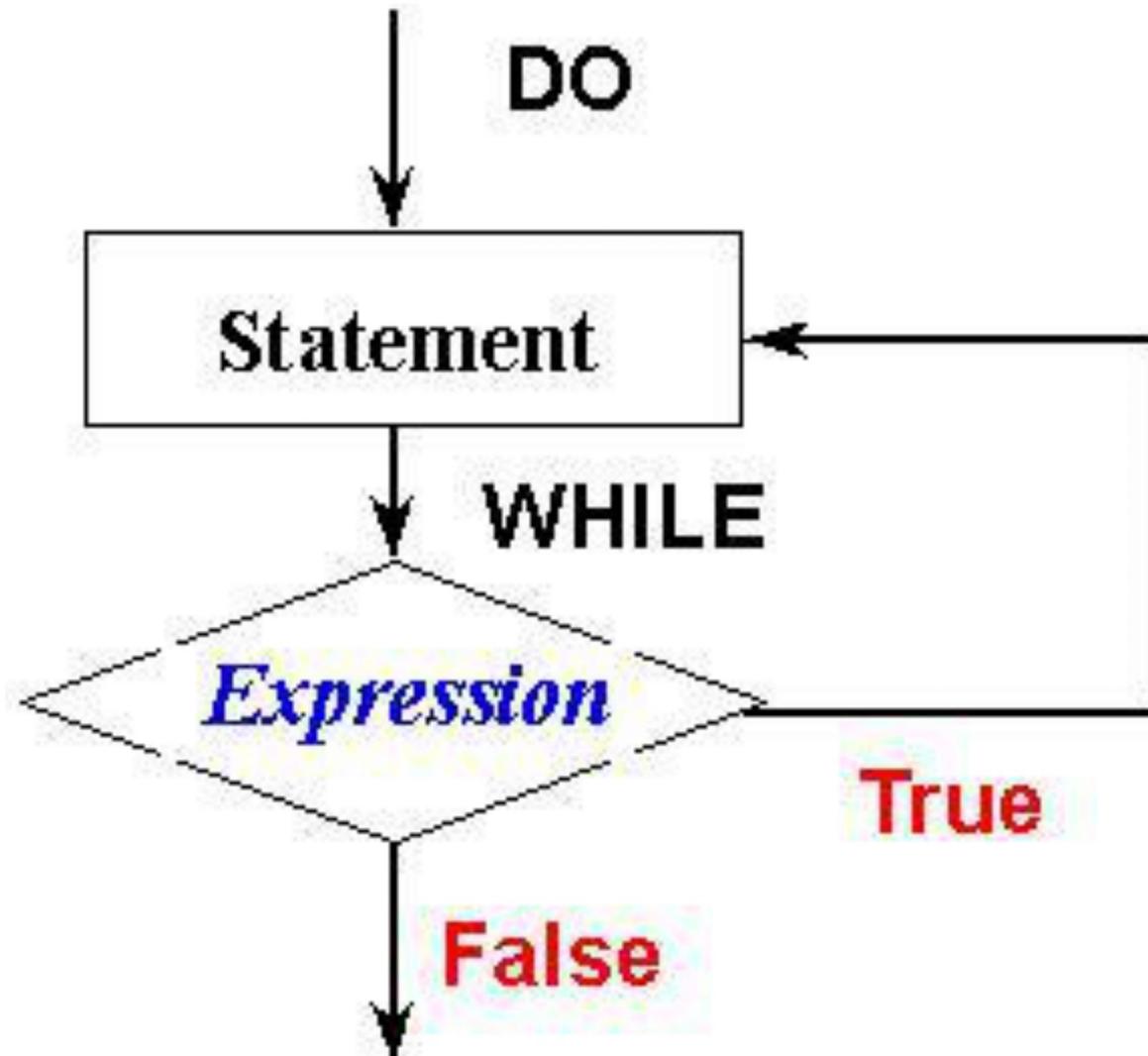
```
1 #include <iostream>
2 using namespace std;
3
4 int main()
5 {
6     int count = 0;
7
8     while (count < 2) {
9         cout << "Welcome, the count is " << count << endl;
10    }
11
12    return 0;
13 }
```

# while

```
main.cpp ×

1
2 #include <iostream>
3 using namespace std;
4
5 int main ()
6 {
7     // Local variable declaration:
8     int a = 10;
9
10    // while loop execution
11    while( a < 20 )
12    {
13        cout << "value of a: " << a << endl;
14        a++;
15    }
16
17    return 0;
18}
19
```

## do ... while



# do ... while

```
1 #include <iostream>
2
3 using namespace std;
4 int main()
5 {
6     int num, i;
7     cout << "Enter a number\t";
8     cin>> num;
9     i=0;
10    do
11    {
12        i++;
13        cout<<i<<"\n";
14    }while(num>=i);
15    return 0;
16 }
17
```

## do ... while

```
#include <stdio.h>

int main()
{
    int count = 0;
    do {
        printf("%d\n", count);
        ++count;
    }while(count < 10);
    return 0;
}
```

## while & do ... while

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

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

# Tipos de Dados

# Tipos de dados

---

- valores que uma variável pode assumir
  - representação interna desses valores
- operações que podem ser realizadas com essa variável

```
int i;           i = 10;                  i = i + 1;  
char nome[35] = "Brasil";
```

instruções incorretas (*Incompatible Types*):  
`i = nome;`  
`nome = i;`

# Classificação dos tipos de DADOS

---

- Tipos de Dados Simples
  - Tipos de Dados Inteiros: *int*, *long* e *unsigned int*
  - Tipos de Dados Reais: *float* e *double*
  - Tipos de Dados Caracteres: *char*
- Tipos de Dados Estruturados
  - cadeia de caracteres (*string*), vetores (*array*), registros (*struct*) e arquivos em disco.
- ponteiros ( alocação dinâmica de memória)
- Classes e Objetos (Orientação a Objetos)

# Tipos de Dados Estruturados

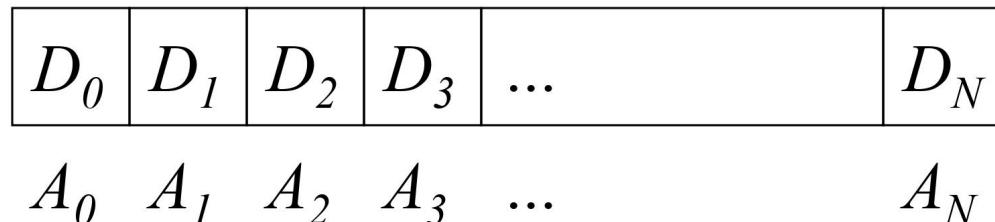
- armazenam diversos itens de uma só vez
- isto significa:
  - em uma mesma estrutura de dados, é possível ter diversas variáveis de tipos de dados simples agrupadas

# ARRAY

## (Vetores e Matrizes)

# Array (vetores, matrizes)

- ❖ Organiza dados de mesma natureza (mesmo tipo) em posições sucessivas da memória (armazenamento contíguo)
- ❖ Cada dado é identificado por um índice



- ❖ Dado um índice  $i$  é possível computar o endereço de memória correspondente

✓ Se o array é alocado a partir do endereço  $A_0$  e cada dado ocupa  $k$  posições, então o  $i$ -ésimo elemento está no endereço:

$$A_i = A_0 + i \cdot k$$

- ❖ Matrizes são construídas analogamente como vetores de vetores

# Vetor

# Vetor

- ❖ Estrutura de dados **homogênea**.
  - ✓ Manipula somente um tipo de dado
- ❖ É uma estrutura de dados linear que necessita de somente um índice para que seus elementos sejam endereçados.
- ❖ É utilizado para armazenar uma lista de valores do mesmo tipo, ou seja, o tipo vetor permite armazenar mais de um valor em uma mesma variável.
- ❖ Um dado vetor é definido como tendo um número fixo de células idênticas (seu conteúdo é dividido em posições). Cada célula armazena um e somente um dos valores de dados do vetor. Cada uma das células de um vetor possui seu próprio endereço, ou índice, através do qual pode ser referenciada. Nessa estrutura todos os elementos são do mesmo tipo, e cada um pode receber um valor diferente [ 3, 21, 154].

# Vetor

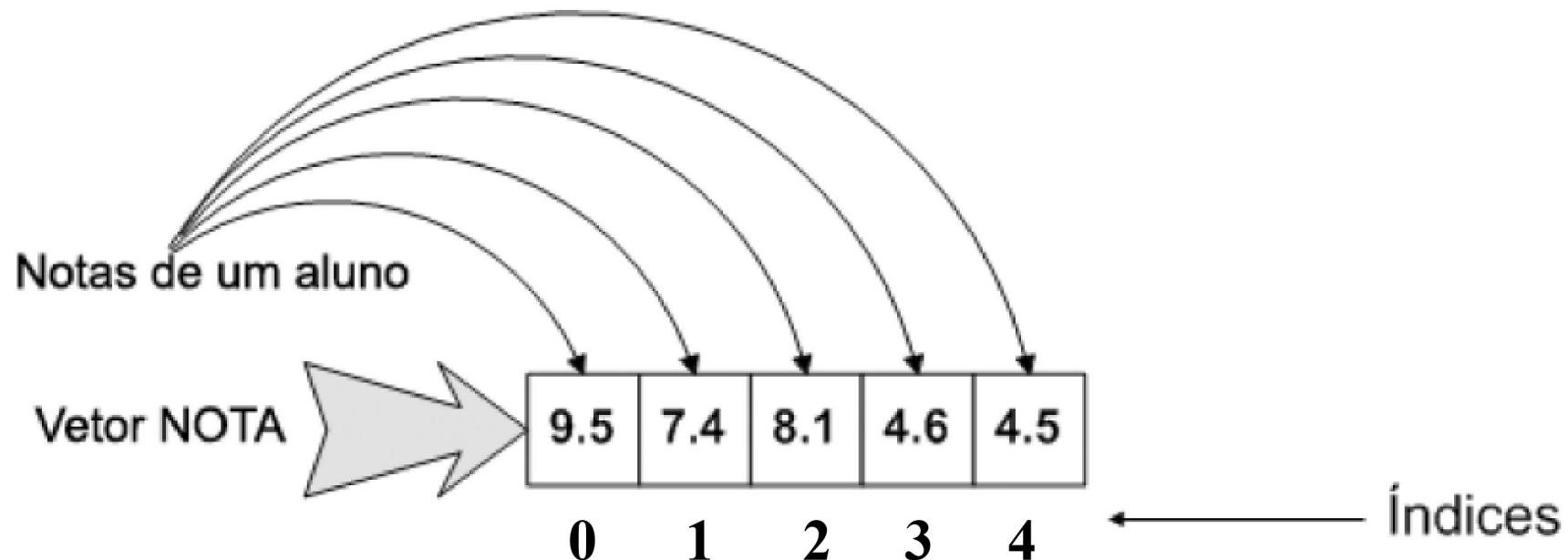


Figura 1.1: Exemplo de Vetor

A definição de um vetor em C se dá pela sintaxe:

```
tipo_do_dado nome_do_vetor[ tamanho_do_vetor ] ;
```

# Vetor

```
tipo_do_dado nome_do_vetor[ tamanho_do_vetor ] ;
```

```
int i[3];  
i[0]=21;  
i[1]=22;  
i[2]=24;
```

➤ Como declarar um vetor de inteiros de nome *i* para armazenar 3 valores?

```
char c[4];  
c[0]='a';  
c[1]='b';  
c[2]='c';  
c[3]='d';
```

➤ Como declarar um vetor de caracteres de nome *c* para armazenar 4 caracteres?

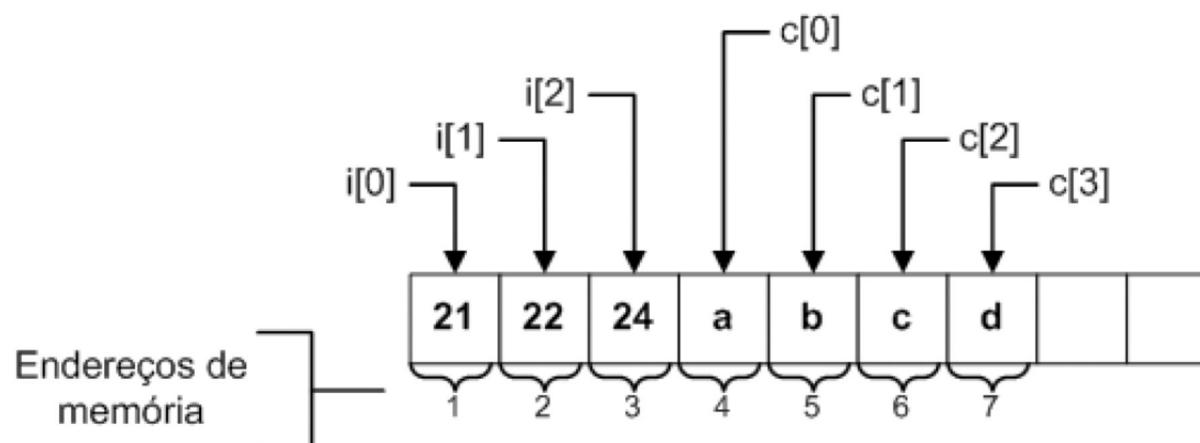


Figura 1.2: Representação de um vetor na memória

## Em Síntese:

```
#define n 5 // tamanho do vetor  
  
// declarando o vetor "v" com "n"  
// números inteiros  
int v[n];  
  
int i; // índice ou posição  
  
// processando os "n" elementos do vetor "v"  
for (i=0; i<n; i++) {  
    v[i] ---> i-ésimo elemento do vetor "v"  
}  
  
// representação interna  


|      |      |      |        |      |
|------|------|------|--------|------|
| v[0] | v[1] | v[2] | v[...] | v[n] |
|------|------|------|--------|------|


```

# Inicialização de Vetores

---

C permite a inicialização de vetores no momento da declaração, como é demonstrado nos exemplos a seguir:

```
int vetor[10] = {17, 33, 21, 67, 81, 10, 45, 29, 79, 98};
```

Isso significa que **vetor[0]** terá o valor  
e **vetor[9]** terá o valor

---

```
int matriz[2][4] = {17, 33, 21, 15,  
                    13, 81, 97, 67};
```

nos vetores bidimensionais os valores são inicializados por linha, no exemplo, **matriz[1][2]** (segunda linha e terceira coluna) terá o valor 97

# Matriz

# Matriz

- ❖ Estrutura de dados **homogênea**.
  - ✓ Manipula somente um tipo de dado
- ❖ É um arranjo bidimensional (multidimensional) de alocação estática e sequencial.
- ❖ A matriz é uma estrutura de dados que necessita de um índice para referenciar a linha e outro para referenciar a coluna para que seus elementos sejam endereçados e acessados. Da mesma forma que um vetor, uma matriz é definida com um tamanho fixo, todos os elementos são do mesmo tipo, cada célula contém somente um valor e os tamanhos dos valores são os mesmos.

A definição de uma matriz em C se dá pela sintaxe:

```
tipo_do_dado nome_da_matriz[ quantidade_linhas ] [ quantidade_colunas ]
```

Declarando um vetor bidimensional, ou matriz

**int** A[**2**][**4**];

Referenciando as posições da matriz

1a. Linha

2a. Linha

-----

$$A[0][0] = 17;$$

$$A[0][1] = 33;$$

$$A[0][2] = 21;$$

$$A[0][3] = 15;$$

$$A[1][0] = 13;$$

$$A[1][1] = 81;$$

$$A[1][2] = 97;$$

$$A[1][3] = 67;$$

0    1    2    3     $\leftarrow j$ , coluna

A[i][j]

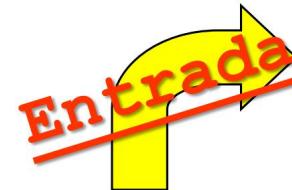
0	17	33	21	15
1	13	81	97	67

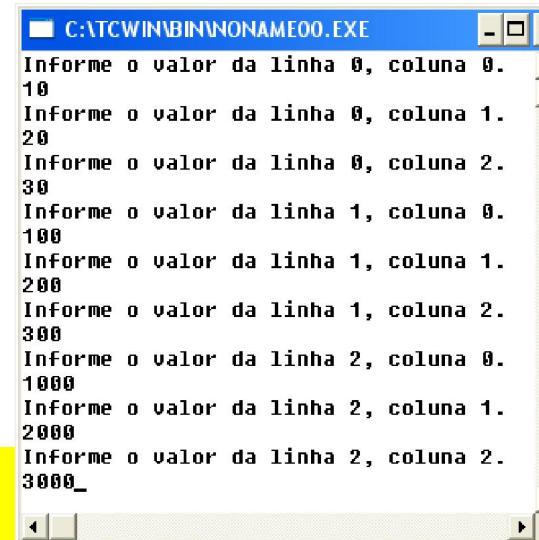
i, linha

## Lendo e escrevendo um vetor bidimensional, ou matriz

A leitura de um conjunto bidimensional é feita, como nos vetores unidimensionais, passo a passo, um componente por vez, usando a mesma sintaxe das instruções primitivas de entrada (scanf) e saída (printf) informando o nome da matriz as posições da linha e da coluna como mostra o exemplo a seguir:

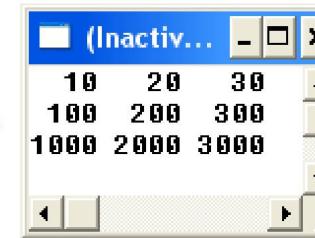
```
#include "stdio.h"
#include "stdlib.h"
#include "conio.h"
int main() {
#define n 3
    int matriz[n][n], i, j;
    for (i=0; i<n; i++)
        for (j=0; j<n; j++) {
            printf("Informe o valor da linha %d, coluna %d.\n", i, j);
            scanf("%d", &matriz[i][j]);
        }
    system("clear");
    for (i=0; i<n; i++) {
        for (j=0; j<n; j++)
            printf("%4d ", matriz[i][j]);
        printf("\n");
    }
}
```



C:\TCWIN\BIN\NONAME00.EXE

Informe o valor da linha 0, coluna 0.  
10  
Informe o valor da linha 0, coluna 1.  
20  
Informe o valor da linha 0, coluna 2.  
30  
Informe o valor da linha 1, coluna 0.  
100  
Informe o valor da linha 1, coluna 1.  
200  
Informe o valor da linha 1, coluna 2.  
300  
Informe o valor da linha 2, coluna 0.  
1000  
Informe o valor da linha 2, coluna 1.  
2000  
Informe o valor da linha 2, coluna 2.  
3000



(Inactiv...)

10	20	30
100	200	300
1000	2000	3000

# Conceitos sobre Matrizes

---

## ❖ Matriz quadrada

✓ número de linhas igual ao número de colunas

## ❖ Diagonal principal

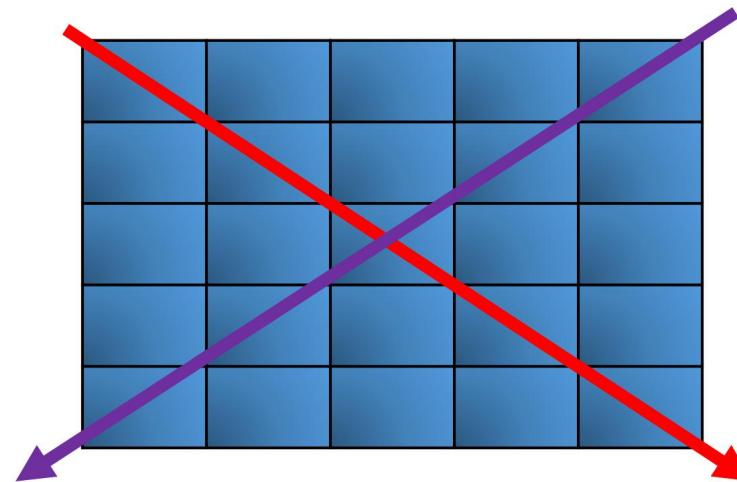
✓  $A_{i,j}$  para todo  $i == j$

## ❖ Diagonal secundária

✓  $A_{i,j}$  para todo  $(i + j) == (n - 1)$

## ❖ Matriz esparsa

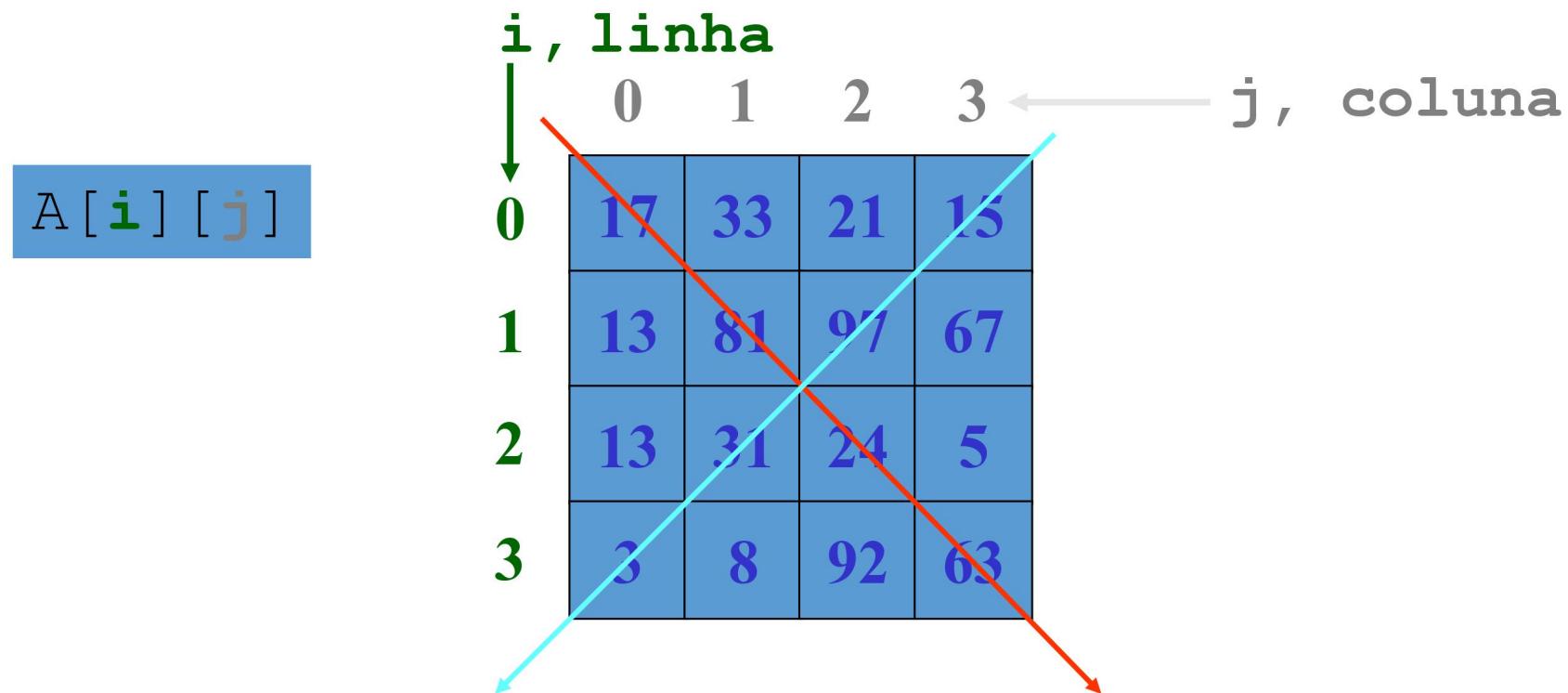
✓ a maioria dos elementos da matriz é zero



## Exemplo de Matriz Quadrada

```
#define n 4
```

```
int A[n][n] = {17, 33, 21, 15, 13, 81, 97, 67,
                13, 31, 24, 5, 3, 8, 92, 63};
```



diagonal secundária ( $(i+j) == (n-1)$ ) :

A[0][3]	15
A[1][2]	97
A[2][1]	31
A[3][0]	3

diagonal principal ( $i==j$ ) :

A[0][0]	17
A[1][1]	81
A[2][2]	24
A[3][3]	63

```

// Dada uma matriz bidimensional com 4 linhas e 4 colunas. Elaborar um programa que
// calcule e escreva o valor da soma dos elementos desta matriz.
#include "stdio.h"
#include "stdlib.h"
#include "conio.h"
#include "time.h"

int main() {
    // declaração da estrutura de dados
    #define n 4
    int A[n][n]; // matriz quadrada de n linhas x n colunas
    int i, j, sm = 0;

    // entrada- gerar nros aleatórios para armazenar no vetor bidimensional
    srand(time(NULL));
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            A[i][j] = 1 + rand() % 10;

    // processamento: somando todos os elementos da matriz
    for (i=0; i<n; i++)
        for (j=0; j<n; j++)
            sm = sm + A[i][j];

    // saída: imprime os resultados
    system("clear");
    printf("Matriz A\n");
    printf("===== \n");
    for (i=0; i<n; i++) {
        for (j=0; j<n; j++)
            printf("%2d ", A[i][j]);
        printf("\n");
    }

    printf("\nA soma dos elementos da matriz eh igual a %d", sm);
}

```

```

Matriz A
=====
 9 10  7 10
 4 10  4 10
 6  1  4  7
 4   1  2  4

```

A soma dos elementos da matriz eh igual a 93

## Operações com Matrizes - Matriz Transposta

Dada uma matriz **A**, chama-se **transposta** de **A** e indica-se por **A<sup>t</sup>** a matriz que se obtém trocando-se ordenadamente as linhas pelas colunas de **A**.

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad A^t = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix} \quad A_{j,i}^t = A_{i,j} \begin{cases} \text{para todo } 0 \leq i < n \\ \text{para todo } 0 \leq j < n \end{cases}$$

tendo a seguinte estrutura de dados:

```
#define n 3  
int i, j, a[n][n], at[n][n];
```

transposição da matriz "a":

```
for (i=0; i<n; i++) {  
  
    for (j=0; j<n; j++) {  
        at[j][i] = a[i][j];  
    }  
  
}
```

## Operações com Matrizes - Matriz Simétrica

1. Uma matriz **A** diz-se **simétrica** se  $\mathbf{A} = \mathbf{A}^t$ .
2. Uma matriz é simétrica se for quadrada e os elementos situados em posições "simétricas" relativamente à diagonal principal forem iguais.

$$\mathbf{A} = \begin{pmatrix} 1 & 3 & 5 \\ 3 & 0 & 4 \\ 5 & 4 & 7 \end{pmatrix} \quad \mathbf{A}_{i,j} == \mathbf{A}_{j,i} \begin{cases} \text{para todo } 0 \leq i < n \\ \text{para todo } 0 \leq j < i \end{cases}$$

tendo a seguinte estrutura de dados:

```
#define n 3  
int i, j, ehsimetrica, a[n][n];
```

verificando se a matriz "a" é simétrica:

```
ehsimetrica = 1; // é simétrica  
for (i=0; i<n; i++) {  
    for (j=0; j<i; j++) {  
        if (a[i][j] != a[j][i]) {  
            ehsimetrica = 0; // não é simétrica  
        }  
    }  
}
```

## Operações com Matrizes - Produto de Matrizes (1/2)

O produto **AB** da matriz **A** pela matriz **B**, apenas está definido se o número de colunas de **A** for igual ao número de linhas de **B**.

$$\mathbf{AB}_{n \times p} = \mathbf{A}_{n \times m} * \mathbf{B}_{m \times p}$$

Somatório do produto dos elementos da i-esima linha da matriz "a" com os respetivos elementos da j-esima coluna da matriz "b".

$$\mathbf{A} = \begin{pmatrix} 1 & 3 & 0 \\ 2 & 5 & 1 \\ 4 & 0 & 2 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} 0 & 4 & 5 \\ 2 & 1 & 3 \\ 6 & 1 & 2 \end{pmatrix}$$

$$\mathbf{AB} = \begin{pmatrix} (1*0) + (3*2) + (0*6) & (1*4) + (3*1) + (0*1) & (1*5) + (3*3) + (0*2) \\ (2*0) + (5*2) + (1*6) & (2*4) + (5*1) + (1*1) & (2*5) + (5*3) + (1*2) \\ (4*0) + (0*2) + (2*6) & (4*4) + (0*1) + (2*1) & (4*5) + (0*3) + (2*2) \end{pmatrix}$$

$$\mathbf{AB} = \begin{pmatrix} 6 & 7 & 14 \\ 16 & 14 & 27 \\ 12 & 18 & 24 \end{pmatrix}$$

## Operações com Matrizes - Produto de Matrizes (2/2)

tendo a seguinte estrutura de dados:

```
#define n 3
int i, j, k, sm, a[n][n], b[n][n], ab[n][n];

for (i=0; i<n; i++) {
    for (j=0; j<n; j++) {
        // somatório do produto dos elementos da i-esima linha
        // da matriz "a" com os respetivos elementos da j-esima
        // coluna da matriz "b"
        sm = 0;
        for (k=0; k<n; k++) {
            sm = sm + (a[i][k] * b[k][j]);
        }
        ab[i][j] = sm;
    }
}
```

# **Atividade 01-R**