# DISCIPLINA: ESTRUTURAS DE DADOS

## MATRIZES

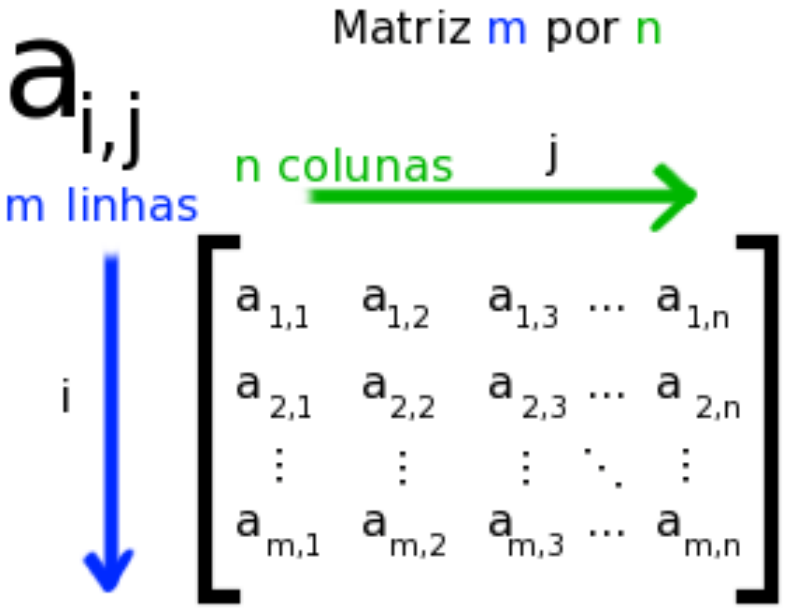

- Prof. Alexandre Ponce de Oliveira

alexandreponceo@gmail.com


Aulas 05 e 06

# Matrizes

Matemática: matriz bidimensional é uma tabela de m *x* n símbolos, representada sob a forma de um quadro com m linhas e n colunas e utilizado, entre outras coisas, para a resolução de sistemas de equações lineares e transformações lineares.



# Matrizes - Utilização

- Animações do cinema: desde o movimento dos personagens até o quadro de fundo podem ser criados por softwares que combinam pixels em formas geométricas, que são armazenadas e manipuladas.

- Os softwares codificam informações como posição, movimento, cor e textura de cada pixel. Para isso, utilizam vetores, matrizes e aproximações poligonais de superfícies para determinar a característica de cada pixel.

- Um simples quadro de um filme criado no computador tem mais de dois milhões de pixels, o que torna indispensável o uso de computadores para realizar todos os cálculos necessários.

# Matrizes - Estrutura

São caracterizadas por terem todos os elementos pertencentes ao mesmo tipo de dado.

Para declarar uma matriz utiliza a seguinte forma:

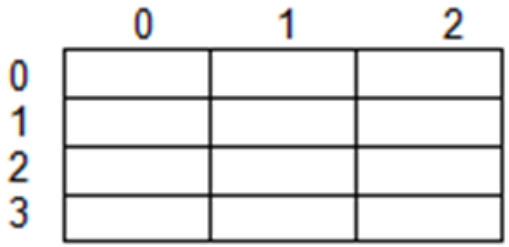
```
tipo_da_variável    nome_da_variável [qtdel] [qtdec];
```

- índice da esquerda indexa as linhas.
- índice da direita indexa as colunas.

# Matrizes

Declaração de matriz reserva um espaço na memória suficientemente grande para armazenar o número de células especificadas em tamanho.
Por exemplo, se declararmos: int matriz[4][3];

Código acima criou uma variável chamada matriz contendo quatro linha (0 a 3) com três colunas (0 a 2), capazes de armazenar números inteiros, como pode ser observado a seguir:



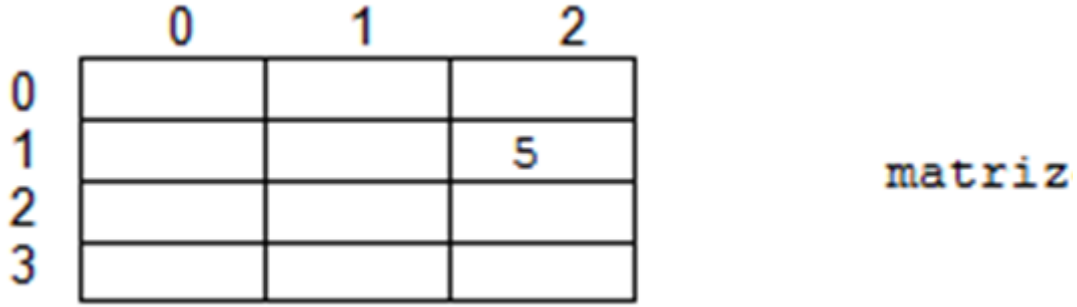

matriz

# Matrizes

**Atribuindo valores à matriz**

matriz[1][2]=5 atribui o valor 5 na posição referente:

linha 1 (2ª linha) coluna 2 (3ª coluna) da matriz.



# Matrizes

**Carregando uma matriz**

Para ler dados do teclado e atribuir a uma matriz, supondo que a mesma tenha sido declarada como int MAT[7][3], pode-se executar os comandos a seguir.

```
for (i=0;i<7;i++)
 {
    for (j=0;j<3;j++)
            scanf("%d", &MAT[i][j]);
 }
```

Matriz possui sete linhas, o **for** externo deve variar de 0 a 6 (percorrendo as sete linhas da matriz) e o **for** interno deve variar de 0 a 2 (percorrendo as três colunas da matriz).

# Matrizes

**Mostrando os elementos de uma matriz**

Para mostrar os valores armazenados dentro de uma matriz, supondo que a mesma tenha sido declarada como int X[10][6], pode-se executar os comandos a seguir.

```
for (i=0;i<10;i++)
 {
    for (j=0;j<6;j++)
            printf ("%d ",X[i][j]);
 }
```

Como a matriz possui dez linhas, o **for** externo deve variar de 0 a 9 (percorrendo as dez linhas da matriz) e o **for** interno deve variar de 0 a 5 (percorrendo as seis colunas da matriz).

# Exemplo1: Preenchimento de matriz

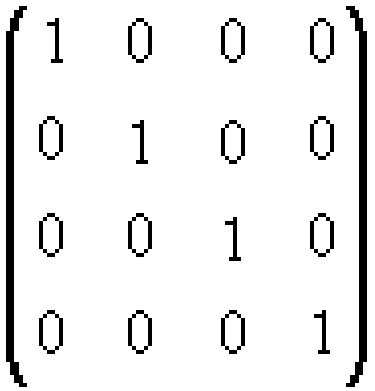
```
#include <stdio.h>
#include <conio.h>
main()
{
 int teste [4][4];
 int i,j;
```

?

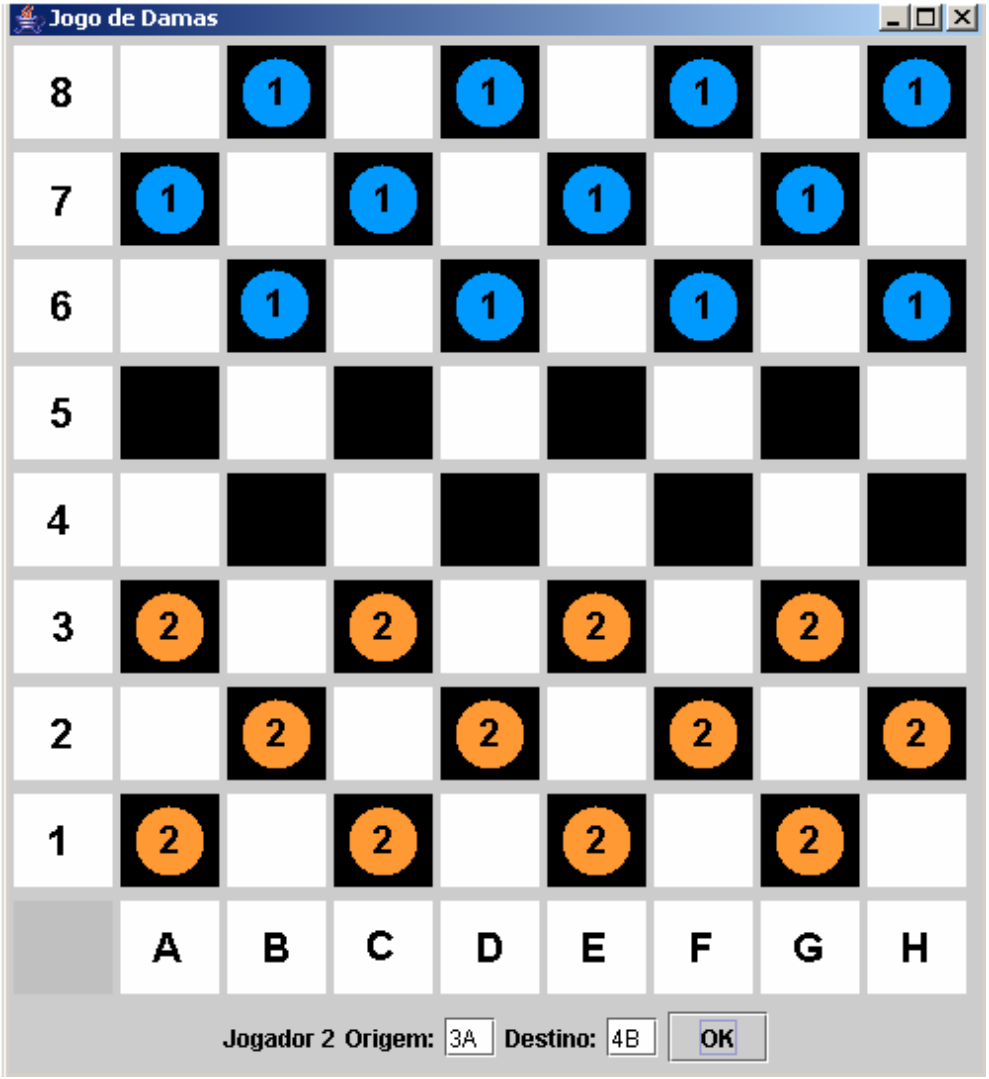
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

# Exemplo1: Preenchimento de matriz

```
#include <stdio.h>
#include <conio.h>
main()
{
 int teste [4][4];
 int i,j;
 for (i=0;i<4;i++)
    for (j=0;j<4;j++)
    {
      if (i == j)
                teste[i][j]=1;
     else
                teste[i][j]=0;
    }
 for (i=0;i<4;i++)
 {
    printf ("\n");
    for (j=0;j<4;j++)
                printf ("%d ",teste[i][j]);
 }
 getch();
}
```



# Exemplo2: Preenchimento de matriz



# Exemplo2: Preenchimento de matriz

```
#include <stdio.h>
#include <conio.h>
main()
{
 int dama [8][8];
 int i,j;
 for (i=0;i<8;i++)
                for (j=0;j<8;j++)
                {
      if (i==0 && j%2!=0)
                 dama[i][j]=1;
      else if (i==1 && j%2==0)
                 dama[i][j]=1;
      else if (i==2 && j%2!=0)
                 dama[i][j]=1;
      else if (i==5 && j%2==0)
                 dama[i][j]=2;
     else if (i==6 && j%2!=0)
                 dama[i][j]=2;
      else if (i==7 && j%2==0)
                 dama[i][j]=2;
     else
                dama[i][j]=0;
    }
 for (i=0;i<8;i++)
 {
    printf ("\n");
              for (j=0;j<8;j++)
                        printf ("%d ",dama[i][j]);
 }
 getch();
}
```

# Exercícios

1) Faça um programa em C que gere a seguinte matriz, é necessário usar laço:

```
1 1 1 1 1 1
1 2 2 2 2 1
1 2 3 3 2 1
1 2 3 3 2 1
1 2 2 2 2 1
1 1 1 1 1 1
```

2) Faça um programa em C que gere a seguinte matriz, é necessário usar laço:

```
1 3 3 3 3 2
3 1 3 3 2 3
3 3 1 2 3 3
3 3 2 1 3 3
3 2 3 3 1 3
2 3 3 3 3 1
```

3) Faça um programa em C que gere a seguinte matriz, é necessário usar laço:

```
0 0 0 1
0 0 1 0
0 1 0 0
1 0 0 0
```

# Exercícios

4) Faça algoritmo utilizando a linguagem C, que faça a soma das matrizes A e B, adicionando os elementos correspondentes: (A + B)[i,j] = A[i, j] + B[i,j]. Você pode definir o tamanho da matriz e o usuário deve alimentar as duas matrizes com valores. É necessário usar laço.

Por exemplo:

$$\begin{bmatrix} 1 & 3 & 2 \\ 1 & 0 & 0 \\ 1 & 2 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 5 \\ 7 & 5 & 0 \\ 2 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1+0 & 3+0 & 2+5 \\ 1+7 & 0+5 & 0+0 \\ 1+2 & 2+1 & 2+1 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 7 \\ 8 & 5 & 0 \\ 3 & 3 & 3 \end{bmatrix}$$

# Exercícios

5) Faça um algoritmo em linguagem C que preencha uma matriz e um vetor com tamanho definido pelo usuário (as linhas e colunas da matriz deve ser igual ao tamanho das colunas do vetor), depois some os valores da primeira linha da matriz com o valor da primeira coluna do vetor e armazene esta soma na primeira coluna de um segundo vetor. Faça isso para todas as linhas existentes. É necessário usar laço.
Exemplo: matriz [4] [4] e vetor[4]