

# Matrizes

DCC 119 – Algoritmos



# ***Matrizes: vetores multidimensionais***

- Assim como os vetores, as matrizes são estruturas de dados **homogêneas**. Podem ser construídas dos diversos tipos básicos primitivos (real, inteiro, caractere).
- Principal diferença em relação aos vetores (unidimensionais): possui uma ou mais dimensões adicionais.
- Maioria dos casos: utiliza-se matrizes bidimensionais.

- São utilizadas quando os dados homogêneos necessitam de uma estruturação com mais de uma dimensão.
- Exemplos:
  - Programar um jogo de xadrez
    - o tabuleiro é naturalmente bidimensional
  - Imagens
  - Caracteres de um livro
    - três dimensões:
      - 2 para os caracteres de uma página;
      - e uma terceira para indicar as páginas.



# Matrizes: declaração



- A sintaxe para declaração de uma variável deste tipo é semelhante a declaração dos vetores.
- Considera-se porém a quantidade de elementos da outra dimensão:

**<tipo\_primitivo> <identificador>[numLinhas][numColunas];**

- Exemplo:

```
int MAT[3][4];  
// matriz 3 linhas e 4 colunas do tipo inteiro  
// esta matriz tem 12 elementos
```

*OBS: Os índices variam de 0 a 2 para as linhas e de 0 a 3 para as colunas neste caso.*

# Matrizes: declaração



- Caso geral:

**<tipo\_primitivo>** **<identificador>** **[dim1][dim2]...[dim n];**

matriz n-dimensional

- Exemplo:

```
float mat[3][6][5];  
// matriz real 3x6x5  
// esta matriz tem 90 elementos
```

*OBS: Os índices variam de 0 a 2 para a 1ª dimensão,  
0 a 5 para a 2ª dimensão e  
0 a 4 para a 3ª dimensão.*

# Matrizes: declaração

Representação:

	0	1	2	3
0	0,0	0,1	0,2	0,3
1	1,0	1,1	1,2	1,3
2	2,0	2,1	2,2	2,3

**mat[1][2]**

# Matrizes: atribuição

- A atribuição a um valor na matriz é feito explicitando a posição em que o valor será ser armazenado.
- Exemplo:

```
float num[2][3];  
num[0][0] = 3.6;  
num[0][1] = 2.7;  
num[0][2] = 1.5;  
num[1][0] = 5.0;  
num[1][1] = 4.1;  
num[1][2] = 2.3;
```



3.6	2.7	1.5
5.0	4.1	2.3

# Matrizes: manipulação

- Os elementos das matrizes são manipulados individualmente por meio de índices (iniciando de zero) entre colchetes.

Exemplo:

	0	1	2
0	3	8	5
1	9	2	1

*Acesso aos elementos da matriz acima:*

```
a = mat[0][0];  
b = mat[1][2];
```

*A instrução ao lado atribui um valor ao elemento da linha zero e coluna um da matriz **MAT**:*

```
i = 0;  
j = 1;  
mat[0][1] = 15;  
// ou  
mat[i][j] = 15;
```



# Matrizes: exemplo

- O programa a seguir, inicializa com zeros os elementos de uma matriz inteira **n** de 5 linhas e 4 colunas e imprime.

```
#include <stdio.h>
int main()
{
    int n[5][4], i, j;
    for (i=0; i< 5; i++)
        for (j=0; j< 4; j++)
            n[i][j] = 0;

    printf("%s\n", "Matriz");
    for (i=0; i < 5; i++)
    {
        printf("\nLinha %2d\n", i);
        for (j=0; j < 4; j++)
            printf("%d ", n[i][j]);
    }
    return 0;
}
```

```
Matriz
Linha 0
0 0 0 0
Linha 1
0 0 0 0
Linha 2
0 0 0 0
Linha 3
0 0 0 0
Linha 4
0 0 0 0
```

# Matrizes: exemplo



- O programa abaixo inicializa os elementos de uma matriz **m** com os valores iguais a soma dos índices de cada elemento e imprime cada valor.

```
#include <stdio.h>
int main()
{
    int m[3][2], i, j;
    for (i=0; i < 3; i++)
        for (j=0; j < 2; j++)
        {
            m[i][j] = i+j;
            printf("i=%d j=%d elemento=%d\n",
                i, j, m[i][j]);
        }
    return 0;
}
```

i= 0	j= 0	elemento=0
i= 0	j= 1	elemento=1
i= 1	j= 0	elemento=1
i= 1	j= 1	elemento=2
i= 2	j= 0	elemento=2
i= 2	j= 1	elemento=3

# Exercícios

1) Quais são os elementos do vetor referenciados pelas expressões ?

**MAT**

3.2	4.1	2.7
5.9	0.6	9.0
1.1	8.3.	6.4

a) `mat[2][0];`

b) `mat[1][1];`

c) `mat[3][1];`

2) Qual é a diferença entre os números “3” das duas instruções abaixo ?

```
int mat[6][3];
```

```
mat[6][3] = 5;
```

3) Faça um programa que crie uma matriz 4x2 de números reais, preencha-a com valores digitados pelo usuário e imprima-a, organizando os elementos em linhas e colunas.

Dica: utilize “%8.2f” no printf para imprimir cada real alinhado à direita em um espaço de 8 casas, sendo duas decimais.

4) Faça um programa que crie uma matriz quadrada de inteiros de tamanho 3, inicialize-a como uma matriz identidade e imprima-a, organizando os elementos em linhas e colunas.

Dica: utilize “%5d” no printf para imprimir cada inteiro alinhado à direita.

5) Faça um programa que crie uma matriz NxM e inicialize-a com valores digitados pelo usuário. Seu programa deve calcular e imprimir a soma dos valores de cada linha e a média dos valores de cada coluna.

# Matrizes e funções



- Matrizes serão passadas para funções da mesma forma como vetores.
- Nas matrizes, apenas a primeira dimensão pode ser omitida.
- Uma função para imprimir uma matriz teria a seguinte declaração:

```
void imprimeMatriz (float mat[][3], int lin)
```

ou

```
void imprimeMatriz (float mat[3][3])
```

# Matrizes e funções



- Neste primeiro caso, a função funciona para matrizes com 3 colunas e qualquer número de linhas.

```
void imprimeMatriz (float mat[][3], int lin)
```

- No segundo caso, a função funciona apenas para matrizes com 3 linhas e 3 colunas.

```
void imprimeMatriz (float mat[3][3])
```

- Os parâmetros com número de linhas e colunas podem ser úteis para especificar quantas linhas e colunas estão preenchidas ou estão em uso.
- Exemplo:  
Se cada disciplina pode ter até 10 avaliações e até 100 alunos, uma matriz  $100 \times 10$  pode armazenar as notas de qualquer disciplina. No entanto, se uma disciplina tem 53 alunos e 3 avaliações, nem todas as posições da matriz precisam ser usadas.

# Matrizes e funções



```
void imprime(float mat[100][10], int lin, int col) {
    int i, j;
    for (i = 0; i < lin; i++)
    {
        printf ("\n %3d) ", i); //imprime num da linha
        for (j = 0; j < col; j++)
            printf ("%6.1f", mat[i][j]);
    }
}

int main( ) {
    float dcc119[100][10], mat155[100][10];
    ...
    //imprime somente 3 notas de apenas 53 alunos
    imprime(dcc119, 53, 3);

    //imprime somente 2 notas de 95 alunos
    imprime(mat155, 95, 2);
    ...
}
```



# Exercício resolvido



**Problema:** Crie uma função que receba uma matriz  $2 \times 3$  de números reais e retorne a média dos valores da matriz. Crie uma função principal que chame a função e imprima a média.

## **Solução:**

Todos os valores da matriz serão acumulados em uma variável real. A função retornará o valor dessa variável dividido pelo número de elementos de matriz.

# Exercício - Solução



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0}, {2.0, 1.1, 4.9}};
    float media = mediaMatriz(mat);
    printf("A media da matriz foi %.2f", media);
    return 0;
}
```

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}
```

```
int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};
    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da função:

i =  
j =  
media =

	0	1	2
0	3.4	5.6	4.0
1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
```

```
{  
    int i, j;  
    float media = 0;  
    for (i = 0; i < 2; i++)  
    {  
        for (j = 0; j < 3; j++)  
        {  
            media = media + m[i][j];  
        }  
    }  
    return media / 6.0;  
}
```

```
int main()  
{  
    float mat[2][3] = {{3.4, 5.6, 4.0},  
                       {2.0, 1.1, 4.9}};  
    float media = mediaMatriz(mat);  
    printf("Media = %.2f", media);  
    return 0;  
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i =  
j =  
media =

	0	1	2
0	3.4	5.6	4.0
1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = ?  
j = ?  
media = 0

	0	1	2
0	3.4	5.6	4.0
1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 0

j = ?

media = 0

	0	1	2	
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa

```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 0

j = 0

media = 0

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 0

j = 0

media = 0 + 3.4 = 3.4

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9



# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 0

j = 1

media = 3.4

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 0

j = 1

media = 3.4 + 5.6 = 9.0

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 0

j = 2

media = 9.0

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 0

j = 2

media = 9.0 + 4.0 = 13.0

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 0  
j = 3  
media = 13.0

		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 1

j = 3

media = 13.0

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 1

j = 0

media = 13.0

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 1

j = 0

media = 13.0 + 2.0 = 15.0

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9



# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 1

j = 1

media = 15.0

				j
		0	1	2
	0	3.4	5.6	4.0
i	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 1

j = 1

media = 15.0 + 1.1 = 16.1

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 1

j = 2

media = 16.1

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 1

j = 2

media = 16.1 + 4.9 = 21

		j		
		0	1	2
i	0	3.4	5.6	4.0
	1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 1  
j = 3  
media = 21

	0	1	2
0	3.4	5.6	4.0
1	2.0	1.1	4.9

j

i

# Exercício – Teste de Mesa

```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 2  
j = 3  
media = 21

	0	1	2
0	3.4	5.6	4.0
1	2.0	1.1	4.9

i

j

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }

    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 2  
j = 3  
media = 21

	0	1	2
0	3.4	5.6	4.0
1	2.0	1.1	4.9

# Exercício – Teste de Mesa



```
float mediaMatriz(float m[2][3])
{
    int i, j;
    float media = 0;
    for (i = 0; i < 2; i++)
    {
        for (j = 0; j < 3; j++)
        {
            media = media + m[i][j];
        }
    }
    return media / 6.0;
}

int main()
{
    float mat[2][3] = {{3.4, 5.6, 4.0},
                       {2.0, 1.1, 4.9}};

    float media = mediaMatriz(mat);
    printf("Media = %.2f", media);
    return 0;
}
```

## Entrada:

matriz com 6 elementos  
2 linhas e 3 colunas

## Variáveis da Sub-Rotina:

i = 2  
j = 3  
media = 21

	0	1	2
0	3.4	5.6	4.0
1	2.0	1.1	4.9

## Saída:

Media = 3.5



6) Implemente as funções abaixo para que o programa funcione corretamente.

```
#include <stdio.h>
#define MAX 10
int main()
{
    float a[MAX][MAX], b[MAX][MAX];
    int m, n;
    printf("\nDigite o no de linhas e de colunas das matrizes: ");
    scanf("%d%d", &m, &n);
    printf("\nForneça os elementos da matriz a:\n");
    lematriz(a, m, n); // leitura da matriz a
    printf("\nForneça os elementos da matriz b:\n");
    lematriz(b, m, n); // leitura da matriz b
    copiazera(a, b, m, n); // a matriz a "recebe" b e b é zerada.
    printf("\nMatriz a:\n");
    imprimematriz(a, m, n);
    printf("\nMatriz b:\n");
    imprimematriz(b, m, n);
    return 0;
}
```

- 7) Faça um programa para exibir a soma de duas matrizes quadradas  $3 \times 3$ . Deverá ser criada uma função para ler uma matriz (será chamado duas vezes com parâmetros diferentes) e uma segunda função que irá imprimir a soma das matrizes passadas como parâmetro.
- 8) a) Faça uma função que leia do teclado uma matriz quadrada de tamanho  $N$ .
- b) Faça uma função que receba duas matrizes e inicialize a segunda matriz com a transposta da primeira (linhas e colunas invertidas).
- c) Faça um programa que crie as matrizes, chame as funções e imprima a segunda matriz.

9) Faça uma função que leia um vetor de dimensão  $M$  e outra função que leia uma matriz quadrada de dimensão  $M$ . Crie também uma função que multiplique o vetor pela matriz, atualizando-o. Faça um programa que chame as funções e imprima o resultado.

**DESAFIO:** Faça uma função que receba três matrizes de dimensões  $N \times M$ ,  $M \times P$  e  $N \times P$  e multiplique as duas primeiras matrizes, armazenando o resultado na terceira matriz.

# **Matrizes**

# **Vetores multidimensionais**

---

DCC 120



# Declaração de matrizes

`int mat3 [3] [4];`

	0	1	2	3
0	0,0	0,1	0,2	0,3
1	1,0	1,1	1,2	1,3
2	2,0	2,1	2,2	2,3

`mat[1][2]` →

- Declaração de matrizes com mais de 2 dimensões

```
float mat2 [3] [5] [2];
```

```
int mat3 [3] [5] [2] [7];
```

- Operações em matrizes

```
mat2 [2] [4] [1] = 8.6;
```

```
mat3 [0] [0] [0] [1] = 5;
```

# Inicialização de matrizes



- Pode-se fornecer valores de cada elemento de uma matriz na declaração, da mesma forma que nos vetores.

- Exemplo:

```
float num[2][3] = {{3.6, 2.7, 1.5},  
                  {5.0, 4.1, 2.3}};
```

- Ou seja, fornece-se os valores linha a linha.

```
int val[5][2] = {{3, 7}}  
// inicializou a primeira com os  
// valores 3 e 7. As demais receberao 0
```

# Inicialização de matrizes



- Observações:
- As matrizes não são inicializadas automaticamente
- Se houver menos valores do que o número de elementos da matriz, os elementos restantes são inicializados automaticamente com o valor zero.

```
int num[5][3] = {{32, 64, 27}};
```

```
//primeiro elemento é 1 e o restante zero
```

```
int n[4][4] = {{1}};
```

- A seguinte declaração causa um erro de sintaxe:

```
int n[2][5] = {{32, 64, 27, 18, 95, 14},  
               {12, 15, 43, 17, 67, 31}};
```

# Matrizes: exemplo

- O programa abaixo inicializa os elementos de uma matriz **m** com os valores iguais a soma dos índices de cada elemento e imprime cada valor.

```
#include <stdio.h>
int main()
{
    int m[3][2], i, j;
    for (i=0; i < 3; i++)
        for (j=0; j < 2; j++)
        {
            m[i][j] = i+j;
            printf("i=%d j=%d elemento=%d\n",
                i, j, m[i][j]);
        }
    return 0;
}
```

```
i= 0 j= 0 elemento=0
i= 0 j= 1 elemento=1
i= 1 j= 0 elemento=1
i= 1 j= 1 elemento=2
i= 2 j= 0 elemento=2
i= 2 j= 1 elemento=3
```



- Matrizes serão passadas para funções da mesma forma como os vetores são passados, a menos de um detalhe: **apenas a primeira dimensão pode ser omitida.**
- **Exemplo:**  
`void imprimeMatriz (float m[3][3])`  
`void imprimeMatriz (float m[][3])`  
(apenas estas duas formas são válidas)
- Havendo mais dimensões, a mesma regra deverá ser seguida:

```
void imprimeMatriz (float m[3][3][4])  
void imprimeMatriz (float m[][3][4])
```

# Exercícios



- 1) Faça uma função para calcular a multiplicação de uma matriz  $3 \times 4$  por um escalar. Faça também uma função capaz de imprimir esta matriz. Ao final, desenvolva a função principal onde será criada e lida uma matriz  $3 \times 4$ . A função principal deverá chamar os dois procedimentos criados anteriormente.
- 2) Faça um programa que leia uma matriz  $7 \times 5$  e imprima a terceira coluna.
- 3) Faça um programa que leia uma matriz quadrada de dimensão 10, uma função que encontre o maior valor desta matriz e uma função que encontre o menor valor. Imprima os valores encontrados na função principal.
- 4) Faça um programa que leia uma matriz  $6 \times 3$  e um procedimento que gere duas matrizes  $3 \times 3$ , a primeira com as 3 primeiras linhas e a outra com as restantes.

- 5) Faça um programa que leia uma matriz de caracteres 5 x 10 e uma função que conte quantas letras “a” aparecem na matriz. Você poderá fazer a leitura letra a letra ou considerar que cada linha da matriz é uma *string*.
- 6) Faça um programa que leia 3 vetores reais de dimensão 10 e um procedimento que gere uma matriz (10 x 3) onde cada coluna é dada por um destes vetores.
- 7) Crie uma matriz tridimensional onde as linhas indicam as notas de matemática, história e geografia de 10 alunos e crie uma função que verifique quantos alunos passaram, ou seja, os que tenham média aritmética  $> 60$  nas 3 disciplinas.

# Exercícios

8) Faça um programa para ler a quantidade de um total de 5 produtos que uma empresa tem em suas 7 lojas e imprimir em uma tabela:

- a) o total de cada produto em toda a empresa
- b) a loja que tem menos produtos

9) Faça uma função para manipulação de matrizes quadradas  $N \times N$  com o seguinte “Menu”:

*Escolha uma opção de cálculo para matrizes:*

- 1) *Soma*
- 2) *Diferença*
- 3) *Transposta*
- 4) *Produto*
- 5) *Sair*

*Opção:\_\_\_*

Na opção (3) o usuário só precisa fornecer uma matriz, nas opções (1), (2) e (4) o usuário deve fornecer duas matrizes. O resultado deve apenas ser impresso.

# Exercícios

## **DESAFIO:** Jogo da velha

Considerando que o jogo usa uma única matriz de tamanho 3x3, faça funções para:

- Inicializar a matriz
- Imprimir o “tabuleiro”
- Ler a jogada de um jogador e marcar a posição escolhida (o caractere ou número que representa o jogador deve ser um parâmetro). Lembre-se que o jogador deve repetir a escolha, caso a posição escolhida seja inválida!
- Verificar se a linha *i* foi preenchida por um único jogador
- Verificar se a coluna *i* foi preenchida por um único jogador
- Verificar se alguma diagonal foi preenchida por um único jogador
- Controlar o jogo: verificar qual é o próximo jogador, se deu velha ou se há vencedor.