

# Laboratório de Programação

Profa. Ms. Valéria Pinheiro



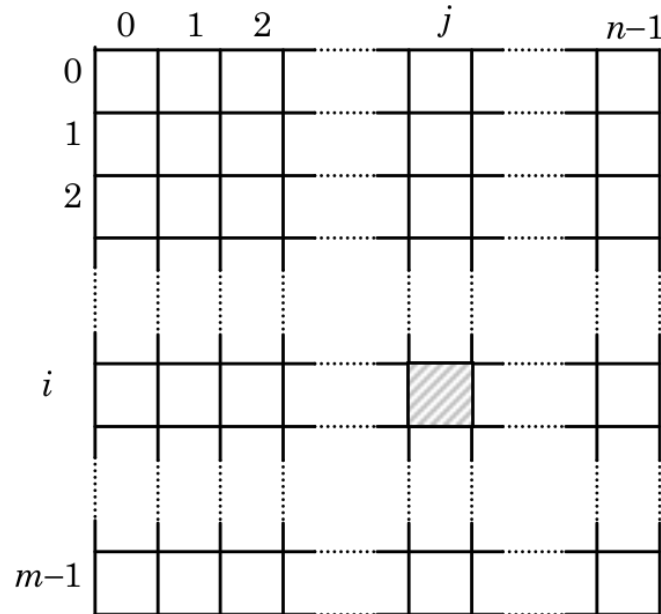
UNIVERSIDADE  
FEDERAL DO CEARÁ  
CAMPUS DE RUSSAS

# Matrices

# Matrizes

- Variável composta homogênea multidimensional

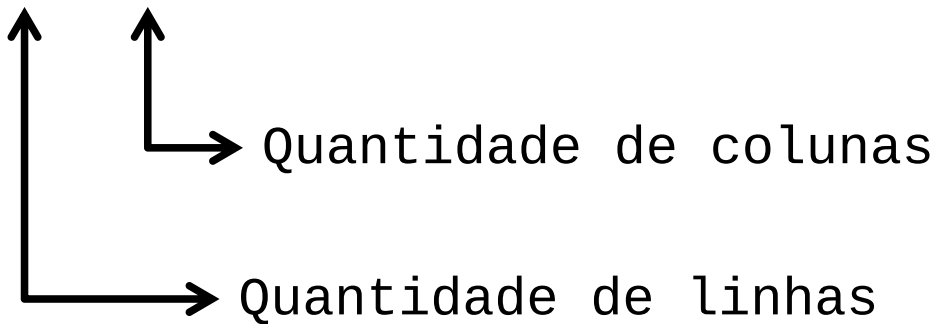
*“Coleção de variáveis de mesmo tipo, onde cada elemento é acessado por vários índices”*



# Matrizes

- Declaração / Acesso

```
int ch[4][3];
```



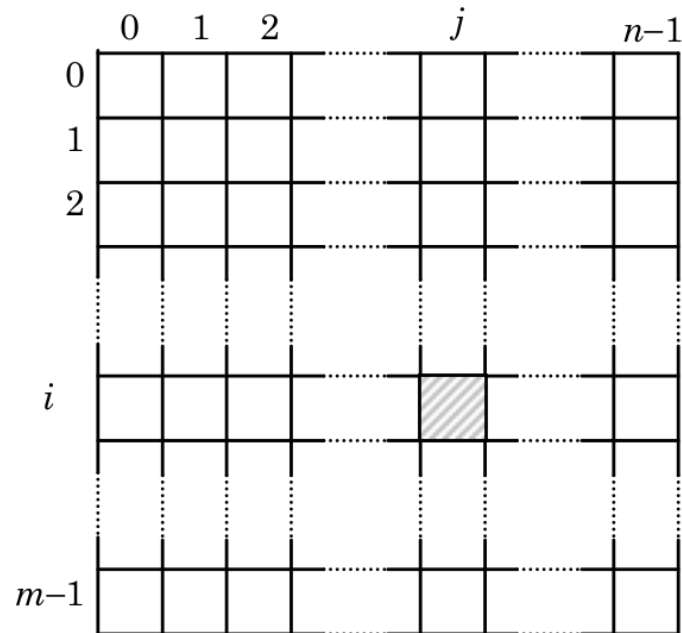
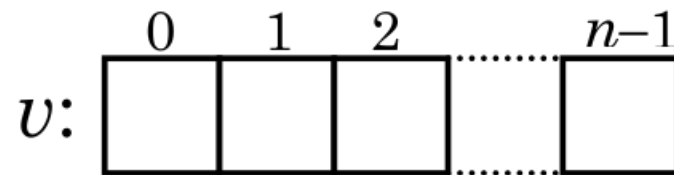
Índice direito ↔ coluna

Índice esquerdo ↔ linha

ch[0][0]	ch[0][1]	ch[0][2]
ch[1][0]	ch[1][1]	ch[1][2]
ch[2][0]	ch[2][1]	ch[2][2]
ch[3][0]	ch[3][1]	ch[3][2]

# Armazenamento de Matrizes

- Elementos sequencialmente armazenados na memória



# Armazenamento de Matrizes

- Matrizes também são armazenadas sequencialmente
  - O índice da direita varia mais rapidamente do que o da esquerda
  - Matrizes são vetores de vetores

ch[0][0]	ch[0][1]	ch[0][2]
ch[1][0]	ch[1][1]	ch[1][2]
ch[2][0]	ch[2][1]	ch[2][2]
ch[3][0]	ch[3][1]	ch[3][2]

ch[0][0]	ch[0][1]	ch[0][2]	ch[1][0]	ch[1][1]	ch[1][2]	ch[2][0]	ch[2][1]	ch[2][2]	ch[3][0]	ch[3][1]	ch[3][2]
----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------	----------

# Percorrendo uma Matriz

- Normalmente, para acessar todos os elementos de uma matriz, são usados laços aninhados

```
int i, j;
for(i = 0 ; i < 3 ; i++){
    printf("Elementos da linha %d: ", i);
    for(j = 0 ; j < 4 ; j++){
        printf("%d, ", x[i][j]);
    }
    printf("\n");
}
```

# Percorrendo uma Matriz

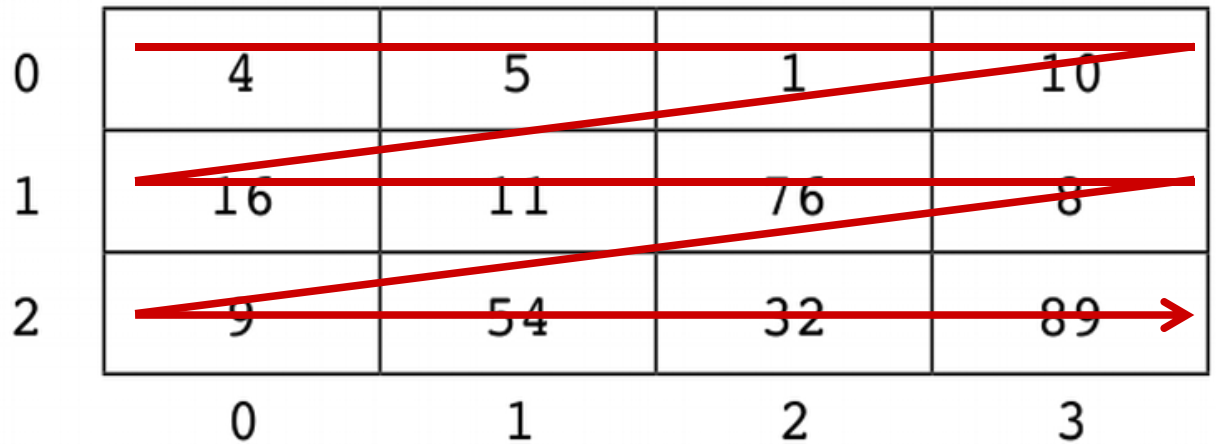
- Normalmente, para acessar todos os elementos de uma matriz, são usados laços aninhados

```
int i, j;  
for(i = 0 ; i < 3 ; i++){  
    printf("Elementos da linha %d: ",i);  
    for(j = 0 ; j < 4 ; j++){  
        printf("%d, ",x[i][j]);  
    }  
    printf("\n");  
}
```

## Saída

Elementos da linha 0: 4, 5, 1, 10,  
Elementos da linha 1: 16, 11, 76, 8,  
Elementos da linha 2: 9, 54, 32, 89,

0	4	5	1	10
1	16	11	76	8
2	9	54	32	89
	0	1	2	3



Para cada índice de linha, o laço interno varia sobre todos os índices de coluna. Isso se traduz em acessar os elementos da matriz por linhas.



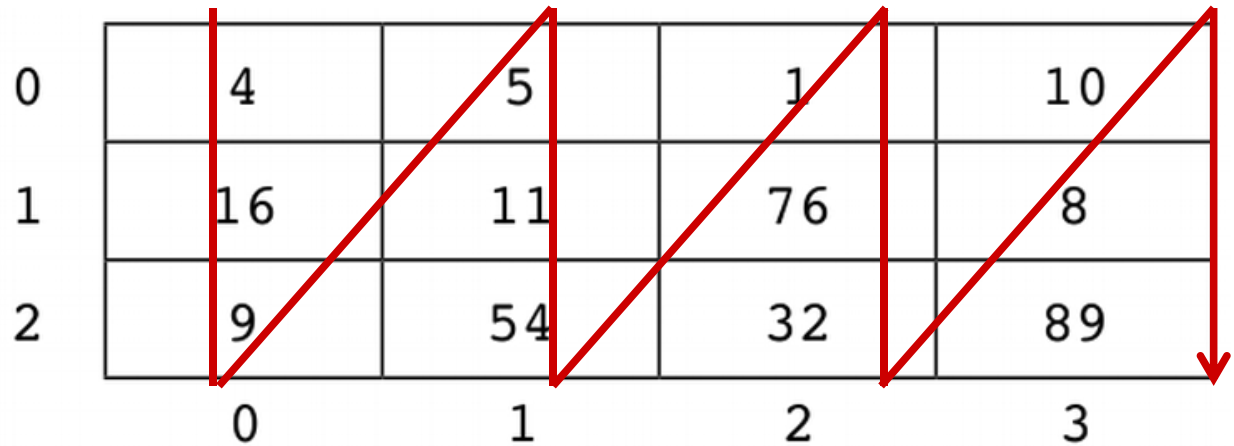
# Percorrendo uma Matriz

- Normalmente, para acessar todos os elementos de uma matriz, são usados laços aninhados

```
int i, j;  
for(j = 0 ; j < 4 ; j++){  
    printf("Elementos da coluna %d: ",j);  
    for(i = 0 ; i < 3 ; i++){  
        printf("%d, ",x[i][j]);  
    }  
    printf("\n");  
}
```

## Saída

Elementos da coluna 0: 4, 16, 9,  
Elementos da coluna 1: 5, 11, 54,  
Elementos da coluna 2: 1, 76, 32,  
Elementos da coluna 3: 10, 8, 89,



The diagram shows a 3x4 matrix with rows indexed 0 to 2 and columns indexed 0 to 3. Red arrows indicate the traversal path for each column: a vertical arrow for each column pointing downwards, and a diagonal arrow for each column pointing from the bottom-left to the top-right. This illustrates the column-major traversal order.

0	4	5	1	10
1	16	11	76	8
2	9	54	32	89
	0	1	2	3

Para cada índice de coluna, o laço interno varia sobre todos os índices de linha. Isso se traduz em acessar os elementos da matriz por colunas.

# Inicialização de Matrizes

- “Matrizes são vetores de vetores”

```
char vogais[5] = {'a', 'e', 'i', 'o', 'u'};
```

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

```
char letras[2][3] = {{ 'a', 'b', 'c' }, { 'd', 'e', 'f' }};
```

# Inicialização de Matrizes

- “Matrizes são vetores de vetores”

```
char vogais[5] = {'a', 'e', 'i', 'o', 'u'};
```

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}$$

```
char letras[2][3] = {{ 'a', 'b', 'c' }, { 'd', 'e', 'f' }};
```

# Vetor de Strings

- String → Vetor de Caracteres
- Vetor de Strings → Matriz de Caracteres
  - Strings armazenadas por linha

```
char nomes[5][30];  
int i;
```

```
for(i = 0 ; i < 5 ; i++){  
    printf("Insira o nome %d: ", i+1);  
    scanf("%s", nomes[i]);  
}
```

R	a	f	a	e	l	\0			
A	n	a	\0						
B	e	a	t	r	i	z	\0		
C	a	r	l	o	s	\0			
Z	e	\0							

# Vetor de Strings

- String → Vetor de Caracteres
- Vetor de Strings → Matriz de Caracteres
  - Strings armazenadas por linha
  - Pode-se usar a inicialização de vetores, onde cada elemento é uma string

```
char menu[][7] = {"abrir", "editar", "salvar", "sair"};  
int i;  
  
for(i = 0 ; i < 4 ; i++)  
    printf("%s\n", menu[i]);
```

	0	1	2	3	4	5	6
0	a	b	r	i	r	\0	
1	e	d	i	t	a	r	\0
2	s	a	l	v	a	r	\0
3	s	a	i	r	\0		

# Exercícios

1 - Escreva um alg que preenche, via laço de repetição, uma matriz[5][5] com o valor da linha do elemento.

Exemplo:

0 0 0 0 0

1 1 1 1 1

2 2 2 2 2

3 3 3 3 3

4 4 4 4 4

# Exercícios

2 – Escreva um alg que lê e preenche os elementos de uma matriz[3][4]. No final, exiba o conteúdo da matriz.

Exemplo:

9 4 2 1 4 3 7 8 5 4 7 6

9 4 2 1

4 3 7 8

5 4 7 6

# Exercícios

3 – Escreva um alg que lê e preenche os elementos de uma matriz[3][3]. No final, exiba o valor das somas de cada linha da matriz.

Exemplo:

1	2	2
3	2	3
4	1	1

Soma Linha 1 = 5

Soma Linha 2 = 8

Soma Linha 3 = 6



# Exercícios

4 - Escreva um alg que lê e preenche os elementos de uma matriz[4][4]. Leia um número e informe se esse número está ou não na matriz.