



## **Aula 19: Matrizes**

### **Introdução a Programação**

---

**Túlio Toffolo & Puca Huachi**  
<http://www.toffolo.com.br>

## Aula anterior

- Vetores
- Vetores e funções
- Aritmética de ponteiros

# Aula de hoje

- 1 Motivação
- 2 Definição de matrizes
- 3 Matrizes e funções
- 4 Exemplos
- 5 Exercícios
- 6 Próxima aula

# Aula de hoje

- 1 Motivação
- 2 Definição de matrizes
- 3 Matrizes e funções
- 4 Exemplos
- 5 Exercícios
- 6 Próxima aula

# Motivação

Fazer um programa para ler as notas de 4 provas para 50 alunos de uma turma e calcular a média do aluno e média da turma.

Solução: criar 4 vetores de 50 posições, sendo um para cada nota:

```
1  double nota1[50], nota2[50], nota3[50], nota4[50];
```

# Motivação

	Aluno		Nota1		Nota2		Nota3		Nota4		Média
0	Pedro	0	5.6	0	6.0	0	7.3	0	5.6	0	6.1
1	Ana	1	10	1	4.0	1	5.0	1	7.3	1	6.6
2	Luiz	2	4.5	2	2.0	2	5.5	2	1.0	2	3.3
...	...	...	...	...	...	...	...	...	...	...	...
48	Matheus	48	7.2	48	6.6	48	8.1	48	8.8	48	7.7
49	Andre	49	6.0	49	9.0	49	7.3	49	4.5	49	6.6

# Motivação

E se tivermos que armazenar 100 notas?

Aluno		Nota1		Nota2		Nota3		Nota99		Nota100		Média	
0	Pedro	0	5.6	0	6.0	0	7.3	0	7.3	0	5.6	0	6.1
1	Ana	1	10	1	4.0	1	5.0	1	5.0	1	7.3	1	6.6
2	Luiz	2	4.5	2	2.0	2	5.5	2	5.5	2	1.0	2	3.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
48	Matheus	48	7.2	48	6.6	48	8.1	48	8.1	48	8.8	48	7.7
49	Andre	49	6.0	49	9.0	49	7.3	49	7.3	49	4.5	49	6.6

Criaremos 100 vetores com 100 nomes diferentes?

# Motivação

Uma solução mais eficaz para resolver o problema é o uso de **matrizes**:

Aluno		0	1	2	...	98	99	Média		
0	Pedro	0	5.6	6.0	7.3	...	7.3	5.6	0	6.1
1	Ana	1	10	4.0	5.0	...	5.0	7.3	1	6.6
2	Luiz	2	4.5	2.0	5.5	...	5.5	1.0	2	3.3
...	...	...	...	...	...	...	...	...	...	...
48	Matheus	48	7.2	6.6	8.1	...	8.1	8.8	48	7.7
49	Andre	49	6.0	9.0	7.3	...	7.3	4.5	49	6.6



# Aula de hoje

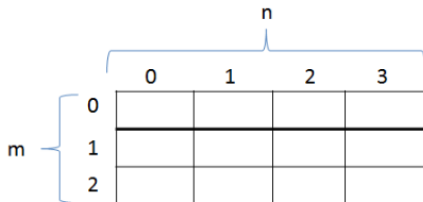
- 1 Motivação
- 2 Definição de matrizes**
- 3 Matrizes e funções
- 4 Exemplos
- 5 Exercícios
- 6 Próxima aula

## Matrizes: variáveis compostas homogêneas

- As variáveis compostas homogêneas correspondem a um conjunto de elementos de mesmo tipo e que compartilham um mesmo nome;
- Cada um dos elementos é unicamente identificado por um número inteiro (índice) que especifica a sua localização dentro da estrutura;
- Estas variáveis podem ser **unidimensionais (vetores)** ou **multidimensionais (matrizes)**;

## Matriz bi-dimensional

Por exemplo, uma matriz bi-dimensional pode ser vista como uma tabela de  $m$  linhas e  $n$  colunas.



The diagram illustrates a 3x4 matrix. A horizontal bracket above the table is labeled 'n', indicating the number of columns. A vertical bracket to the left of the table is labeled 'm', indicating the number of rows. The columns are indexed 0, 1, 2, and 3 from left to right. The rows are indexed 0, 1, and 2 from top to bottom. The table itself is a 3x4 grid of empty cells.

	0	1	2	3
0				
1				
2				

## Declaração de matrizes

`<tipo> <identificador> [<linhas>] [<colunas>];`

- `<tipo>`: tipo dos dados que serão armazenados no vetor (int, char, float, etc);
- `<identificador>`: nome dado à variável;
- `<linhas>`: número de elementos da primeira dimensão;
- `<colunas>`: número de elementos da segunda dimensão;
- As linhas e colunas são numeradas de 0 até *tamanho* - 1.

# Declaração de matrizes

Exemplo:

```
1 //matriz com 100 linhas e 50 colunas  
2 double notas[100][50];
```

# Acessando os elementos

- Forma de ter acesso ao elemento de uma matriz:  
`<variável>[<indice_linha>][<indice_coluna>]`
- Exemplos:

```
1 //imprimir o elemento da linha 3 e coluna 10 da matriz notas
2 printf("%lf", notas[3][10]);
3
4 //multiplica a posição (i, j) da matriz mat por 5;
5 mat[i][j] = mat[i][j] * 5;
```

## Observação

- C/C++ não verifica o limite das dimensões das variáveis compostas;
- Se uma instrução for feita com índices além do limite, é possível que não ocorra um erro de execução do programa e outros valores sejam sobrepostos na memória;
- É responsabilidade do programador providenciar a verificação dos limites das dimensões das variáveis compostas;

## Exemplo

Faça um programa que leia e imprima uma matriz  $4 \times 3$  (4 linhas e 3 colunas).



# Declaração da matriz

Matriz (M x N)

```
1  #define M 4
2  #define N 3
3
4  int matriz[M][N]; // note que M e N são constantes
```

# Leitura dos dados da matriz

Matriz (M x N)

```
1 // capturando dados
2 for (int i = 0; i < M; i++) { //para as linhas
3     for (int j = 0; j < N; j++) { //para as colunas
4         scanf("%d", &matriz[i][j]);
5     }
6 }
```

# Impressão da matriz

Matriz (M x N)

```
1 // imprimindo o conteúdo da matriz
2 for (int i = 0; i < M; i++) { //para as linhas
3     for (int j = 0; j < N; j++) { //para as colunas
4         printf("%d ", matriz[i][j]);
5     }
6     printf("\n"); //salta uma linha
7 }
```

```

1  #define M 4
2  #define N 3
3
4  int main()
5  {
6      int matriz[M][N];
7
8      // capturando dados
9      for (int i = 0; i < M; i++) {           // para as linhas
10         for (int j = 0; j < N; j++) {       // para as colunas
11             printf("matriz[%d][%d] = ", i, j);
12             scanf("%d", &matriz[i][j]);
13         }
14     }
15
16     // imprimindo o conteudo da matriz
17     for (int i = 0; i < M; i++) {           // para as linhas
18         for (int j = 0; j < N; j++) {       // para as colunas
19             printf("%4d ", matriz[i][j]);
20         }
21         printf("\n");
22     }
23     return 0;
24 }

```

# Inicialização de Matrizes I

Inicializando cada elemento da matriz ( $m \times n$ ) com o valor 0.

```
1  int matriz[M][N];  
2  
3  for (int i = 0; i < M; i++) //para as linhas  
4      for (int j = 0; j < N; j++) //para as colunas  
5          matriz[i][j] = 0;
```

## Inicialização de Matrizes II

Inicializando na declaração. Processo semelhante à inicialização de vetores.

```
1  int matriz[3][4] = { {10, 20, 30, 40},  
2                        {50, 60, 70, 80},  
3                        {90, 11, 22, 33} };
```

Mas podemos fazer também:

```
1  int matriz[3][4] = { 10, 20, 30, 40,  
2                        50, 60, 70, 80,  
3                        90, 11, 22, 33 };
```

Ou ainda:

```
1  int matriz[3][4] = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 11, 22, 33 };
```

## Exemplo

Dada uma matriz ( $4 \times 5$ ), calcular a soma de todos os elementos da matriz. Calcular também o somatório dos elementos de cada linha da matriz, armazenando o somatório em um vetor.

### SOMALINHA

**MAT**

1	2	3	4	5
0	-1	0	-3	1
2	-2	-2	2	0
0	0	6	0	0

15
-3
0
6

## Exemplo

```
1  int main()
2  {
3      // declaração das variáveis
4      float mat[4][5], somaLinha[4], total;
5
6      // total se inicia com zero
7      total = 0;
8
9      for (int i = 0; i < 4; i++) {
10         // a soma da cada linha é inicializada com zero
11         somaLinha[i] = 0;
12
13         // somando os valores da linha em somaLinha[i]
14         for (j = 0; j < 5; j++)
15             somaLinha[i] += mat[i][j];
16
17         // somando o total de cada linha
18         total += somaLinha[i];
19     }
20 }
```



# Aula de hoje

- 1 Motivação
- 2 Definição de matrizes
- 3 Matrizes e funções**
- 4 Exemplos
- 5 Exercícios
- 6 Próxima aula

## Passando matrizes por parâmetro

Em C/C++ você precisa indicar o tamanho de todas as dimensões de uma matriz passada por parâmetro, exceto a dimensão mais à esquerda.

Exemplo com **todas** as dimensões da matriz:

```
1 void imprimirMatriz(int matriz[3][3], int n, int m)
2 {
3     for (int i = 0; i < n; ++i) {
4         for (int j = 0; j < m; ++j)
5             printf("%d ", matriz[i][j]);
6         printf("\n");
7     }
8 }
```

## Passando matrizes por parâmetro

Em C/C++ você precisa indicar o tamanho de todas as dimensões de uma matriz passada por parâmetro, exceto a dimensão mais à esquerda.

Exemplo **sem a dimensão mais a esquerda**:

```
1 void imprimirMatriz2(int matriz[][3], int n, int m)
2 {
3     for (int i = 0; i < n; ++i) {
4         for (int j = 0; j < m; ++j)
5             printf("%d ", matriz[i][j]);
6         printf("\n");
7     }
8 }
```

## Passando matrizes por parâmetro

Mas... porquê todas as dimensões menos a mais à esquerda??

- Por conta da forma como matrizes são representadas na memória!
- As linhas são colocadas sequencialmente em um “**vetorzão**”.
- Exemplo: seja a matriz  $3 \times 3$  a seguir

```
1  matriz[3][3] = { { 10, 20, 30 },  
2                    { 40, 50, 60 },  
3                    { 70, 80, 90 } };
```

- Ela será representada na memória como um vetor de tamanho 9:

```
1  i          --->    0  0  0  1  1  1  2  2  2  
2  j          --->    0  1  2  0  1  2  0  1  2  
3  M[i][j]    ---> { 10, 20, 30, 40, 50, 60, 70, 80, 90 }
```

## Passando matrizes por parâmetro

Logo, quando acessamos o campo  $[i][j]$  de uma matriz  $4 \times 5$ :

- C/C++ acessa o campo  $5 \times i + j$  do “vetorzão”  
(em que 5 é o número de colunas).
- Para tal, o compilador deve saber quantas colunas há em cada linha
  - Ou seria impossível multiplicar por **5** neste exemplo.

Se você não quiser definir as dimensões da sua matriz em tempo de compilação, há algumas alternativas...

- que aprenderemos em breve...  
(quando falarmos sobre **alocação dinâmica**)

## Passando matrizes por parâmetro

Agora faz mais sentido inicializar uma matriz sem separação?

```
1 // inicializando uma matriz sem separação entre linhas/colunas
2 int matriz[3][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
```

# Aula de hoje

- 1 Motivação
- 2 Definição de matrizes
- 3 Matrizes e funções
- 4 Exemplos**
- 5 Exercícios
- 6 Próxima aula

## Exemplo 1

Faça um programa em C++ que calcule a soma de duas matrizes:

$$C_{m \times n} = A_{m \times n} + B_{m \times n}$$

onde

$$c_{i,j} = a_{i,j} + b_{i,j} \quad \forall i \in \{1 \dots m\} \text{ e } j \in \{1 \dots n\}$$



```

1  #define M 4
2  #define N 3
3
4  int main()
5  {
6      int a[M][N] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12} };
7      int b[M][N] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12} };
8
9      int c[M][N]; // c não precisa ser inicializada neste momento...
10
11     // calculando o valor de cada célula da matriz c
12     for (int i = 0; i < M; i++)
13         for (int j = 0; j < N; j++)
14             c[i][j] = a[i][j] + b[i][j];
15
16     // imprimindo matriz c
17     for (int i = 0; i < M; i++) {
18         for (int j = 0; j < N; j++)
19             printf("%d ", c[i][j]);
20         printf("\n");
21     }
22
23     return 0;
24 }

```

## Exemplo 2

Escrever um programa que leia uma matriz, seus elementos e sua dimensão ( $m, n \leq 100$ ). Em seguida, o programa deve:

- 1 solicitar ao usuário o índice de uma linha ( $\ell$ ) e um valor constante ( $c$ );
- 2 multiplicar todos elementos da linha  $\ell$  por  $c$ ;
- 3 imprimir a matriz resultante;

## Exemplo 2

Lendo a matriz:

```
1  int matriz[100][100];
2
3  // lendo as dimensões da matriz
4  printf("Digite as dimensões m e n da matriz: ");
5  scanf("%d %d", &m, &n);
6
7  // lendo os elementos da matriz
8  for (int i = 0; i < m; i++) {
9      for (int j = 0; j < n; j++) {
10         printf("Digite o valor de matriz[%d][%d]: ", i, j);
11         scanf("%d", &matriz[i][j]);
12     }
13 }
```

Solicitando o índice da linha  $\ell$  e o valor de  $c$  para, em seguida, multiplicar os elementos da linha  $\ell$  por  $c$  e imprimir a matriz resultante:

```
1  int linha, constante;
2  printf("Digite o índice da linha a alterar: ");
3  scanf("%d", &linha);
4  printf("Digite o valor da constante: ");
5  scanf("%d", &constante);
6
7  // multiplicando valores da linha *linha* por *constante*
8  for (int j = 0; j < n; j++)
9      matriz[linha][j] = matriz[linha][j] * constante;
10
11 // imprimindo a matriz resultante
12 for (int i = 0; i < m; i++) {
13     for (int j = 0; j < n; j++)
14         printf("%d ", matriz[i][j]);
15     printf("\n");
16 }
```

## Exemplo 3

Escreva um programa que declare uma matriz ( $5 \times 5$ ) e inicialize cada posição com o valor 0. Em seguida, o usuário deve digitar o índice da linha e da coluna e o valor da posição.

- A leitura será feita enquanto os índices forem não negativos.
- Após a leitura o programa deve imprimir a matriz na tela.

```
1  int matriz[5][5];
2  int linha, coluna, valor;
3
4  for (int i = 0; i < 5; i++)
5      for (int j = 0; j < 5; j++)
6          matriz[i][j] = 0;
7
8  do {
9      printf("Usuário, digite linha, coluna e valor: ");
10     scanf("%d %d %d", &linha, &coluna, &valor);
11
12     if (linha >= 0 && coluna >= 0)
13         matriz[linha][coluna] = valor;
14
15 } while (linha >= 0 && coluna >= 0);
16
17 for (int i = 0; i < 5; i++) {
18     for (int j = 0; j < 5; j++)
19         printf("%d ", matriz[i][j]);
20     printf("\n");
21 }
```

# Aula de hoje

- 1 Motivação
- 2 Definição de matrizes
- 3 Matrizes e funções
- 4 Exemplos
- 5 Exercícios**
- 6 Próxima aula

## Exercícios

### Exercício 1

Escreva um programa que declare e preencha uma matriz ( $10 \times 10$ ) com valores fornecidos pelo usuário.

- 1 O programa deve imprimir o maior valor da matriz e em qual posição (linha e coluna) este valor está.
- 2 Em seguida, o programa deve imprimir os elementos da **diagonal principal** e da **diagonal secundária**.



# Aula de hoje

- 1 Motivação
- 2 Definição de matrizes
- 3 Matrizes e funções
- 4 Exemplos
- 5 Exercícios
- 6 Próxima aula**

## Próxima Aula

- Aula prática: Cadeia de caracteres (vetor/array de caracteres)
- Aula prática: **strings**, **vetores** e **matrizes**



Perguntas?