# **Matriz em C**

Disciplina: PROGRAMAÇÃO II

Prof. Jean Eduardo Glazar Curso de Sistemas de Informação Campus Colatina-ES



### Definição

Matriz é um conjunto de espaços de memória, de mesmo tipo, referenciados por um mesmo nome e que necessita de mais de um índice para que seus elementos sejam endereçados. Graficamente:

	0	1	2	3	4	5
0						
1						
2						
3						
4						



#### Matriz - exemplo

 A seguinte matriz pode representar os preços das passagens de uma empresa de transportes – comercial, executivo e leito (eixo x) das cidades (eixo y) até Colatina:

	Comercial	Executivo	Leito
São Paulo	43.40	60,99	97,50
Vitória	17,80	33,56	46,40
Rio de Janeiro	24,60	37,15	64,55



#### Declaração de Matriz bi-dimensional

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

#### **Exemplos:**

```
int matriz[10][20];
float passagens[3][3];
```

**Vetor** é uma matriz de uma dimensão.





### Manipulação de Matriz

A manipulação de matrizes é bastante semelhante a manipulação de vetores. Assim como em vetores, utilizamos índices para acessar cada elemento. Um índice para cada dimensão. Uma matriz bi-dimensional, por exemplo, pode ser acessada por dois índices, conforme programa a seguir.



#### Manipulação de Matriz - exemplo

```
20
int main() {
  int matriz[10][20];
  matriz [0][0] = 10;
  matriz [1][2] = 20;
  printf("O valor da posição [0][0] da matriz é %d", matriz[0][0]);
  printf("O valor da posição [1][2] da matriz é %d", matriz[1][2]);
```



3

#### **Percorrer Matriz**

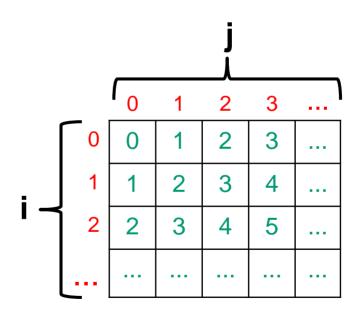
- Como temos várias dimensões (índices), devemos utilizar estruturas de repetição aninhadas (de preferência o for) para percorrer cada dimensão da matriz.
- Por exemplo, em uma matriz bi-dimensional, utilizaremos os índices i e j, onde i corresponde à linha e j corresponde à coluna.

matriz[i][j]



#### **Percorrer Matriz - exemplo**

```
int main() {
  int matriz[10][20];
  int i, j;
  for (i=0; i<10; i++) {
     for (j=0; j<20; j++) {
         matriz[i][j] = i + j;
```





### Matriz – passagem de parâmetro

```
void imprimir ( int mat[ ][20] ) {
  int i, j;
  for (i=0; i<10; i++) {
     for (j=0; j<20; j++) {
         printf("%3d", mat[i][j]);
     printf("\n");
```

Pode ou não colocar o tamanho no primeiro colchetes da matriz.





#### **Vetor de Strings**

Vetores de strings são matrizes bi-dimensionais. Imagine uma string, ela é um vetor. Se fizermos um vetor de strings estaremos fazendo um vetor onde cada posição é outro vetor, ou seja, uma matriz bidimensional de char. Podemos ver a forma geral de uma matriz de strings como sendo:

	0	1	2	3	4	5
0	7	0	ã	0	\0	
1	M	a	r	-	a	\0
2	Ξ	u	9	0	\0	
3	Z	é	\0			
4	M	a	n	œ	\0	



### Declaração de Vetor de Strings

char <nome\_matriz> [<qtde>] [<comprimento>];

#### Onde:

- <qtde> → é a quantidade de strings que se deseja armazenar.
- <comprimento> → é o comprimento máximo de cada string.



### Declaração de Vetor de Strings

Como acessar uma string individual?



Fácil. É só usar apenas o primeiro índice. Assim:

matriz [indice]



#### **Vetor de Strings - exemplo**

```
int main() {
  char strings [5][100];
                                              Lê 5 strings, armazena
                                              no vetor e exibe na
  int i;
                                              tela.
  for (i=0; i<5; i++) {
     printf ("\n\nDigite uma string: ");
     fgets ( strings[i], 100, stdin );
  printf ("\n\n\nAs strings que voce digitou foram:\n\n");
  for (i=0; i<5; i++) {
     printf ("%s\n", strings[i] );
```



### Inicialização de Matriz

Podemos inicializar matriz assim como inicializamos vetor. Exemplos:

```
float vect [6] = { 1.3, 4.5, 2.7, 4.1, 0.0, 100.1 };
int matrx [3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
char str1 [10] = { 'J', 'o', 'a', 'o', '\0' };
char str2 [10] = "Joao";
char str_vect [3][10] = { "Joao", "Maria", "Jose" };
```



#### **Matriz multi-dimensional**

 Para cada dimensão, coloca-se entre colchete, o tamanho dessa dimensão.

```
<tipo> <nome_matriz> [<tam1>] [<tam2>]... [<tamN>];
```

#### **Exemplo com 3 dimensões:**

int cuboMagico [3] [3];

#### Exemplo com 4 dimensões (cubo mágico com string):

char cuboMagicoStr [3] [3] [50];



## Alocação Dinâmica

 Podemos visualizar matrizes bidimensionais como vetores de vetores, dessa forma uma matriz pode ser alocada dinamicamente como se segue no exemplo:

```
int **matriz; // Vetor de vetor → Ponteiro para ponteiro
int i, quant_linhas, quant_colunas;
... // Lendo quant_linhas e quant_colunas;
matriz = (int **) malloc (quant_linhas * sizeof(int *) );
for (i = 0; i < quant_linhas; i++) {
    matriz[i] = (int *) malloc (quant_colunas * sizeof(int) );
}</pre>
```



