

MINIAULA DE ALGORITMOS MATRIZES

Prof. Ivanilton Polato

Departamento Acadêmico de Computação (DACOM-CM)

ipolato@utfpr.edu.br

Matrizes: o que são?

- Variáveis compostas, homogêneas e multidimensionais
 - *Trabalham com DUAS ou mais dimensões!*
- Agregam um conjunto de variáveis:
 - *de mesmo tipo*
 - *com um mesmo nome*
 - *com índices (posições) diferentes*
 - *alocadas na memória (vetor de vetores)*
- Cada índice da matriz é acessado como uma variável comum

Matrizes: o que são?

- Exemplo: uma matriz de números inteiros com
2 linhas e 3 colunas

```
int m[2][3];    // Declaração da matriz
m[0][0] = 1;    // matriz L0,C0 recebe o valor 1
m[0][1] = 3;    // matriz L0,C1 recebe o valor 3
m[0][2] = 5;    // matriz L0,C2 recebe o valor 5
m[1][0] = 1;    // matriz L1,C0 recebe o valor 2
m[1][1] = 3;    // matriz L1,C1 recebe o valor 4
m[1][2] = 5;    // matriz L1,C2 recebe o valor 5
```

Matrizes: ilustrando o exemplo anterior

- Matriz m com 2 linhas e 3 colunas

	0	1	2
0	1	3	5
1	2	4	6

- Referenciar os elementos na forma:

$m[\text{ÍNDICE DA LINHA}][\text{ÍNDICE DA COLUNA}]$:

$m[0][0]$, $m[0][1]$, $m[0][2]$, ...

Matrizes: índices

- Assim como nos vetores, permitem o acesso direto a cada posição do vetor
- São representados por valores inteiros e sequenciais, e sempre vem no mínimo em pares (linha e coluna)!



Sempre começam no índice ZERO!

- Linhas e colunas manipuladas individualmente!

Matrizes: declaração

- Similar às variáveis simples, apenas adicionando os “[]”, um para cada dimensão!
- Podem ser de qualquer tipo:

```
int m[2][3]; // matriz de inteiros com 2 linhas  
e 3 colunas
```

```
float m2[5][7]; // matriz de float com 5 linhas  
e 7 colunas
```

```
char velha[3][3]; //matriz de char com 3 linhas  
e 3 colunas. Poderia ser utilizada para  
controlar o jogo da velha ('X' ou 'O')
```

Matrizes: manipulação

- Nas declarações, a matriz pode ser inicializada em sua totalidade:

```
linhas:      0      1
int m[2][3] = { {1,3,5}, {2,4,6} };
colunas:    0  1  2      0  1  2
```

- No caso acima, a matriz será declarado e inicializada com os números nas respectivas posições. Devemos ter uma **quantidade de conjuntos igual ao número de linhas**. Cada conjunto deve ter a **quantidade** de elementos igual ao **número de colunas da matriz**.

```
int mat[128][256] = {0};
```

- Nesse caso, todas as posições da matriz serão inicializadas com 0. Esse mecanismo só funciona para o número zero.

Matrizes: manipulação

- As posições da matriz recebem valores como variáveis simples:
 - *Atribuição direta:* `m[0][2] = 5;`
 - *Comando de entrada:* `scanf("%d", &m[0][2]);`



ATENÇÃO: *as posições da matriz devem ser manipuladas INDIVIDUALMENTE, uma por vez, mesmo que em uma estrutura de repetição!*

Matrizes: manipulação + repetição

```
01. #include <stdio.h>
02. int main() {
03.     int i, j;
04.     int m[2][3];
05. → for(i=0; i<2; i++) {
06.         for(j=0; j<3; j++) {
07.             printf("Número: ");
08.             scanf("%d", &m[i][j]);
09.         }
10.     }
11.     return 0;
12. }
```

i	j	m[i][j]
0	0	Número: 1
0	1	Número: 3
0	2	Número: 5
1	0	Número: 2
1	1	Número: 4
1	2	Número: 6

	0	1	2
0	1	3	5
1	2	4	6

Matrizes: inversão dos FOR!

```
01. #include <stdio.h>
02. int main() {
03.     int i, j;
04.     int m[2][3];
06. → for(j=0; j<3; j++) {
05.         for(i=0; i<2; i++) {
07.             printf("Número: ");
08.             scanf("%d", &m[i][j]);
09.         }
10.     }
11.     return 0;
12. }
```

i	j	m[i][j]
0	0	Número: 1
1	0	Número: 3
0	1	Número: 5
1	1	Número: 2
0	2	Número: 4
1	2	Número: 6

	0	1	2
0	1	5	4
1	3	2	6

Matrizes: manipulação + repetição

```
01. #include <stdio.h>
02. int main() {
03.     int i, j, m[2][3]={{1,3,5},{2,4,6}};
04.     for(i=0; i<2; i++){
05.         for(j=0; j<3; j++){
06.             printf("M[%d][%d]: %d",
07.                 i, j, m[i][j]);
08.         }
09.     return 0;
10. }
```

i	j	Tela
0	0	M[0][0]: 1
0	1	M[0][1]: 3
0	2	M[0][2]: 5
1	0	M[1][0]: 2
1	1	M[1][1]: 4
1	2	M[1][2]: 6

	0	1	2
0	1	3	5
1	2	4	6

Matrizes: impressão em formato

```
01. #include <stdio.h>
02. int main() {
03.     int i, j;
04.     int m[2][3] = {{1, 3, 5}, {2, 4, 6}};
05.     for(i=0; i<2; i++) {
06.         for(j=0; j<3; j++) {
07.             printf("%d  ", m[i][j]);
08.         }
09.         printf("\n");
10.     }
11.     return 0;
12. }
```

Saída em Tela:

```
1  3  5
2  4  6
```

	0	1	2
0	1	3	5
1	2	4	6

Matrizes: dicas

- Índices sempre começam em ZERO!
 - *Lembre-se do limite das variáveis contadora na repetição!*
 - *Associação facilitada: $i \rightarrow$ linhas e $j \rightarrow$ colunas*
 - *Jamais inverta os índices i e j na instrução da matriz!*
- Manipulação individual das posições!
 - *Todas as posições manipuladas uma por vez!*
 - *Não se esqueça do $\&$ no scanf! (`scanf("%d", &m[i][j]);`)*
- Use como uma variável comum:
 - `if (m[i][j] > 0) { ... }`
 - `switch(m[i][j]) { ... }`
 - `soma = soma + m[i][j];`