Técnicas de Programação A

Luiz Fernando Carvalho

luizfcarvalhoo@gmail.com

Matrizes

- Os vetores vistos até agora eram usados para guardar variáveis escalares;
- vamos explorar agora outra possibilidade: **usar um vetor para guardar um conjunto de vetores;**
- Por exemplo, se temos 3 vetores de 5 inteiros, podemos criar um vetor que contém esses 3 vetores, e podemos acessar os inteiros usando dois índices: primeiro o índice que identifica **cada um dos três vetores**, depois o índice que identifica **cada inteiro dentro de cada vetor**;
- Podemos interpretar isso como uma matriz: o primeiro índice **indica a linha** em que um elemento está, e o segundo **indica a posição (coluna)** desse elemento dentro da linha correspondente.

Matrizes

- Em suma, cada linha de uma matriz é um vetor de ***n*** números, e a matriz é um vetor de **m** vetores-linha, formando assim uma matriz **m × n** (**m** linhas, **n** colunas);

$a_{0,0}$	$a_{0,1}$	...	$a_{0,n-1}$
$a_{1,0}$	$a_{1,1}$	...	$a_{1,n-1}$
⋮			
$a_{m-1,0}$	$a_{m-1,1}$	...	$a_{m-1,n-1}$

- Muitas vezes são chamadas de vetores multidimensionais.

Matrizes: Declaração

- Para declarar uma variável do tipo matriz, usa-se uma sintaxe similar à declaração de vetores:

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

- Não há uma obrigação computacional que indique que o índice de linha deva ser o primeiro a ser informado, seguido pelo o de coluna.
  - É só uma convenção matemática.

```
int valores[3][2] = {{2, 3}, {5, 7}, {9, 11}};  // correto
int valores[][2] = {{2, 3}, {5, 7}, {9, 11}};   // correto
Int valores[][] = {{2, 3}, {5, 7}, {9, 11}};    // inválido
```

Matrizes: Declaração

- Aplicam-se as mesmas observações apontadas para os vetores
  - Os **números de linhas e colunas** devem ser ***constantes;***
  - Os **índices** dos elementos são numerados **a partir do zero**;

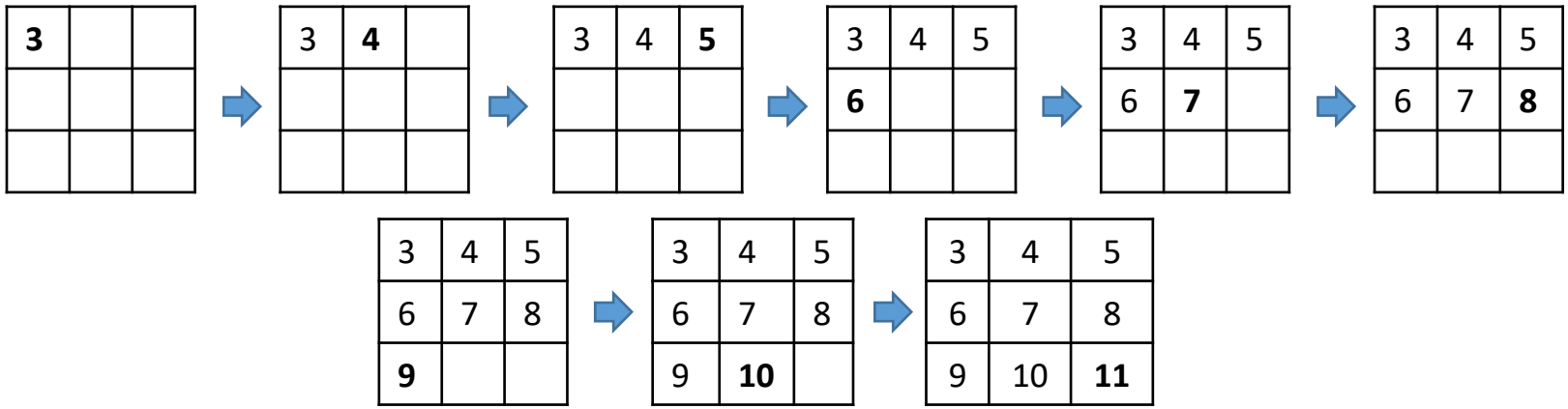
```
int matriz[3][3], i, j;
// i é o índice de linhas e j de colunas

for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        printf("Digite o valor da linha %d e coluna %d: ", i, j);
        scanf("%d", &matriz[i][j]);
    }
}
```

Matrizes: Declaração

```
int matriz[3][3], i, j;
// i é o índice de linhas e j de colunas

for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
    {
        printf("Digite o valor da linha %d e coluna %d: ", i, j);
        scanf("%d", &matriz[i][j]);
    }
}
```



Matrizes: Alteração

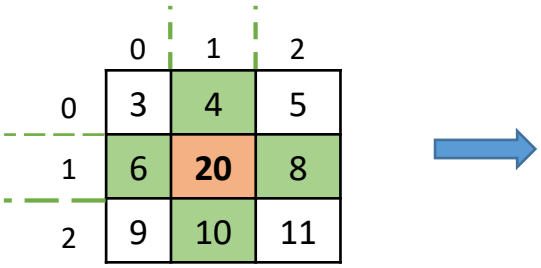
- Deve-se informar a posição do elemento que sofrerá a alteração na matriz
  - Informar linha e coluna;

```
int m1[3][3];
```

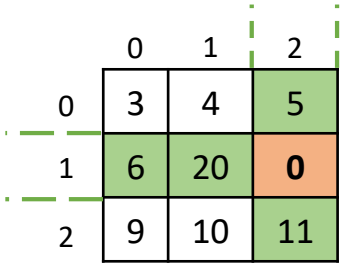
m1 =

3	4	5
6	7	8
9	10	11

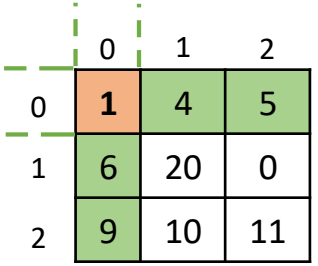
```
m1[1][1] = 20;
```




```
m1[1][2] = 0;
```




```
m1[0][0] = 1;
```



Matrizes: Impressão

- Um bom algoritmo para imprimir a matriz é:
  1. Definir uma variável para contar a quantidade de linhas impressas;
  2. Atribuir zero à essa variável;
  3. Imprimir todas colunas da linha corrente;
  4. Incrementar o contador de linha;
  5. Imprimir o comando "\n" para começar a impressão na próxima linha;
  6. Voltar ao passo 3.

```
int matriz[3][3], i, j;
// i é o índice de linhas e j de colunas

for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
        printf(" %d ", matriz[i][j]);
    printf("\n");
}
```

Exercícios

1. Crie uma matriz identidade com dimensões 5 x 5;
2. Faça um algoritmo que leia uma matriz 3 por 3 (3x3) e retorna a soma dos elementos da sua diagonal principal e da sua diagonal secundária;

- $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

Soma Principal = 15

Soma Secundária = 15

3. Leia uma matriz quadrada de inteiros com dimensão de 3x3 e verifique se ela é simétrica em relação à diagonal principal. Exemplos para teste.

- $B = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 7 & 4 \\ 0 & 4 & 5 \end{bmatrix}$ $C = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 7 & 6 & 9 \end{bmatrix}$

Exercícios

1. Construa um programa que leia uma matriz de tamanho 5 x 5 e escreva a localização (linha, coluna) do maior valor encontrado na matriz.
2. Na teoria de Sistemas define-se elemento minimax de uma matriz, o menor elemento da linha em que se encontra o maior elemento da matriz. Escrever um algoritmo que lê uma matriz 5 por 5 (5x5) e determine o elemento minimax desta matriz, escrevendo-o e a posição na matriz em que ele se encontra.
3. Construa um programa que leia uma matriz 2 x 7. O programa deverá fazer uma busca de um valor N na matriz e, como resultado, escrever a localização (linha, coluna) do elemento. Caso o valor de **N** não constar na matriz lida, o programa deverá mostrar uma mensagem de "*elemento não encontrado*".

Exercícios

1. Crie um programa que calcule o determinante de qualquer matriz 3 x 3 fornecida pelo usuário.
2. Construa um programa que entre com duas matrizes e com suas respectivas dimensões. Em seguida, verifique se é possível fazer a multiplicação entre as matrizes. Caso seja possível, calcule e exiba em tela o produto entre elas.
3. Desenvolva um programa que leia uma matriz 6 x 6 e escreva quantos valores maiores que **N** ela possui. Obs.: O valor de **N** será fornecido pelo usuário.

Exercícios Extras

1. Escreva um algoritmo que lê uma matriz M(5, 5) e a imprima para que o usuário possa conferi-la. Calcula e mostre as seguintes somas:

   a) da linha 4 de M

   b) da coluna 2 de M

   c) da diagonal principal

   d) da diagonal secundária

   e) de todos os elementos da matriz M.

Exercícios Extras

2. Escrever um algoritmo que lê uma matriz M(5, 5) e a escreva. Verifique, a seguir, quais os elementos de M que estão repetidos e quantas vezes cada um está repetido. Escrever cada elemento repetido com uma mensagem dizendo que o elemento aparece X vezes em M.

3. Receba uma matriz M(5, 5) do usuário e então troque os elementos da primeira linha, com os elementos da terceira linha.