Fundamentos de Programação

Aula 10 - Matrizes

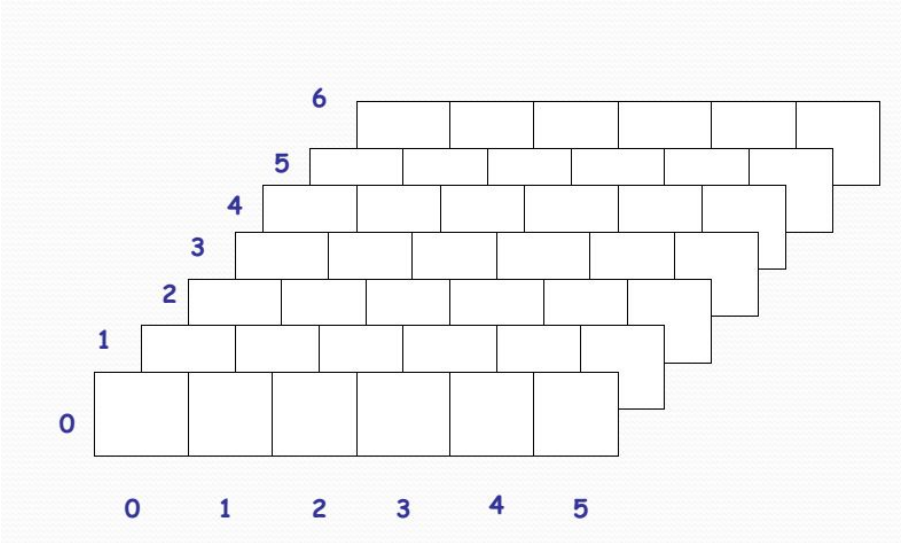
Arnaldo Barreto Vila Nova

Matrizes

- Definição
- Declaração
- Utilização
- Exercícios

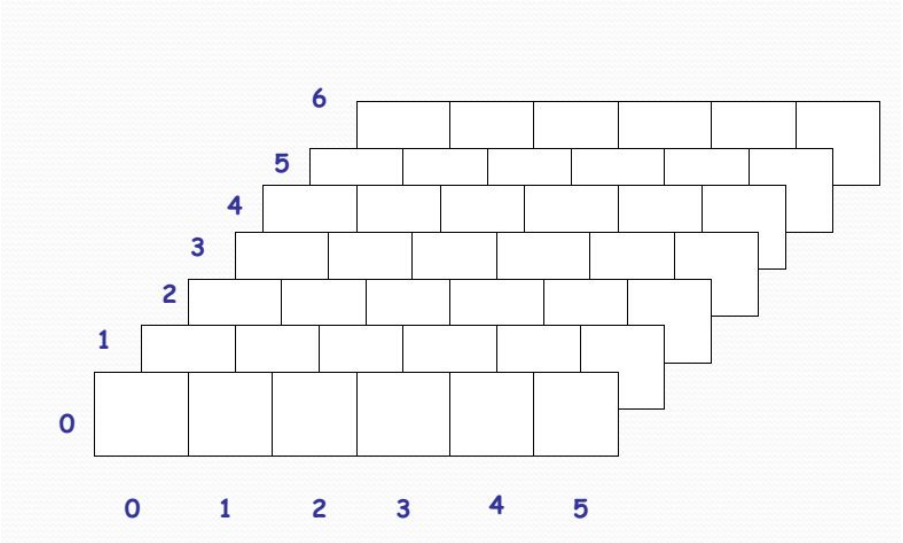
Matrizes - Definição

- Estruturas de Dados Homogêneas Multidimensionais
- Bidimensionalmente falando, é um “vetor de vetores”



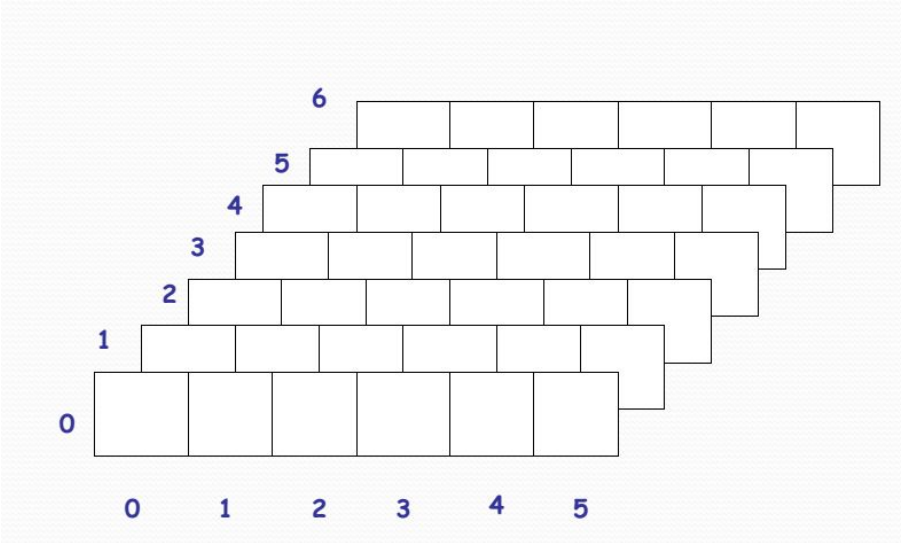
Matrizes - Definição

- Lembrando, um vetor é um conjunto de valores (mesmo tipo), acessados por índices através de um único nome



Matrizes - Definição

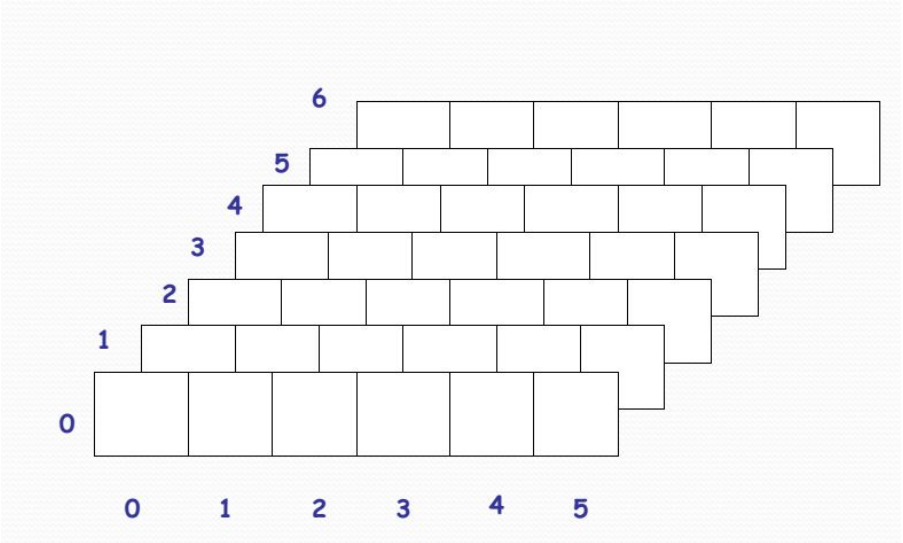
- Então, vamos precisar de um índice para cada dimensão (normalmente chamados de linhas e colunas)



Exemplo - Matriz 7x6 (7 linhas e 6 colunas)

Matrizes - Definição

- A alocação em memória é sequencial a partir do primeiro índice (linha 0, coluna 0) até o último índice

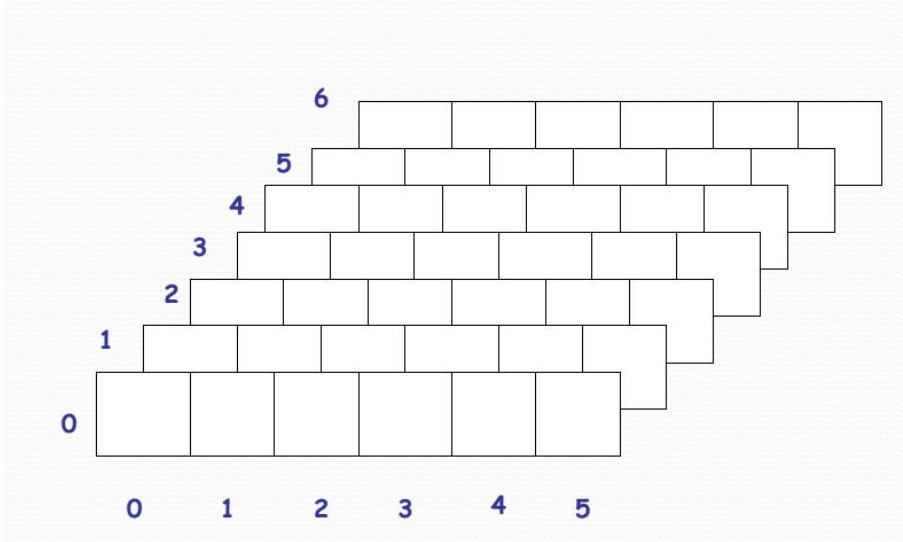


Exemplo - Matriz 7x6 (7 linhas e 6 colunas)

Matrizes - Definição

- A alocação em memória é sequencial a partir do primeiro índice

Memória



Exemplo - Matriz 7x6 (7 linhas e 6 colunas)

Matrizes - Definição

- A alocação em memória é sequencial a partir do primeiro índice

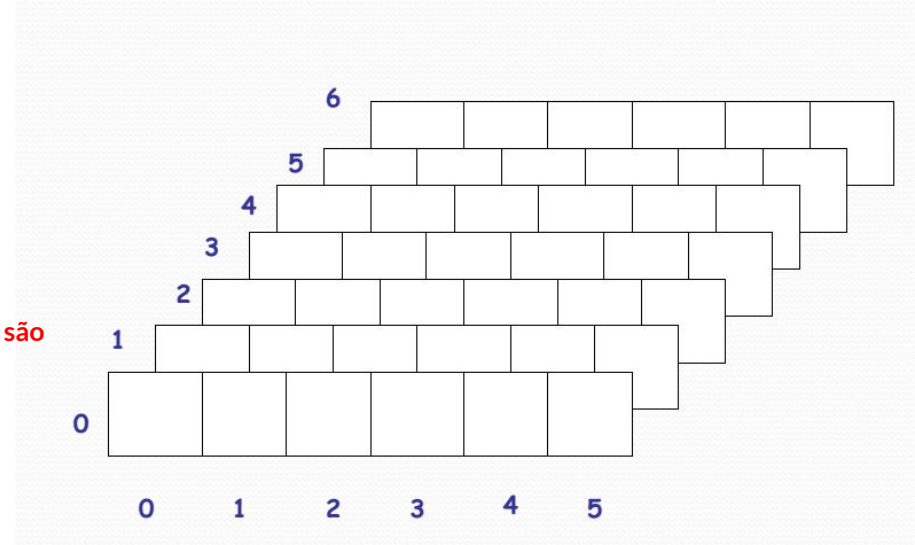
[0][5]

...

[0][0]

Memória

Os elementos da 1ª linha são alocados primeiro



Exemplo - Matriz 7x6 (7 linhas e 6 colunas)

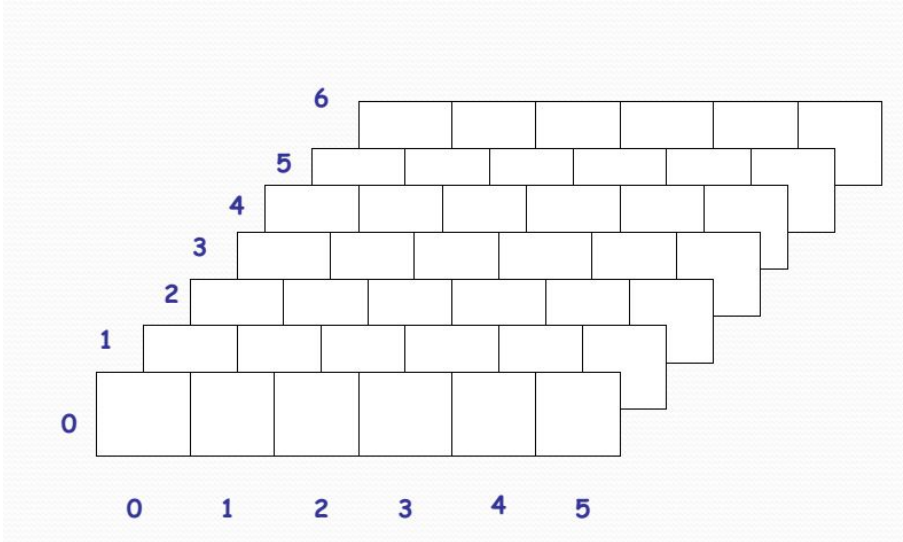
Matrizes - Definição

- A alocação em memória é sequencial a partir do primeiro índice

[1][5]
...
[1][0]
[0][5]
...
[0][0]

Memória

Depois os da 2ª linha



Exemplo - Matriz 7x6 (7 linhas e 6 colunas)

Matrizes - Definição

[6][5]

…

[6][0]

…

[1][5]

…

[1][0]

[0][5]

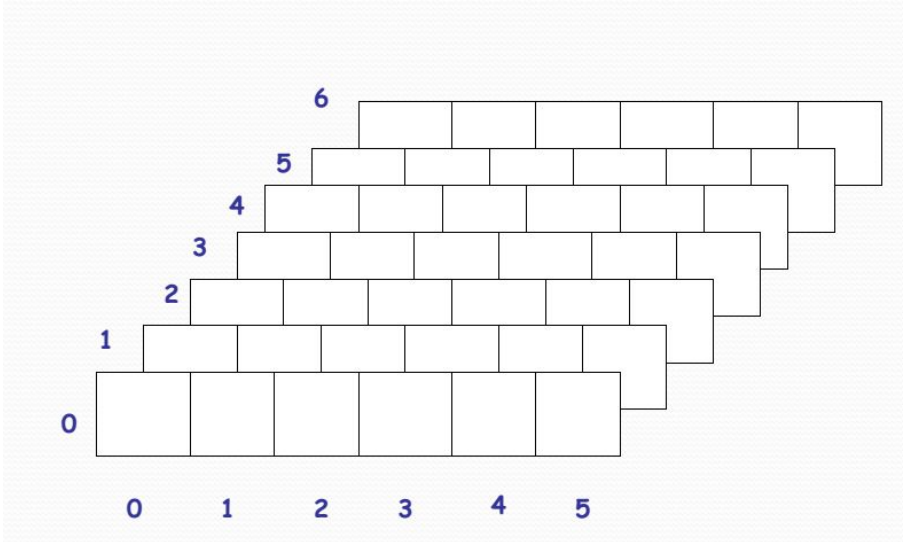
…

[0][0]

Memória

E assim vai até chegar na última

- A alocação em memória é sequencial a partir do primeiro índice



Exemplo - Matriz 7x6 (7 linhas e 6 colunas)

Matrizes - Declaração

- Array de 1 dimensão (vetor)
  - `tipo nome [tam];`
- Array de 2 dimensões (matriz)
  - `tipo nome [tam1][tam2];`
- Array de n dimensões
  - `tipo nome [tam1][tam2]...[tamN];`

Matrizes de dimensão 3 ou superior são casos muito específicos e bastante complexos. Em FUP vamos trabalhar somente com matrizes bidimensionais

Matrizes - Declaração

- Inicialização
  - `float vetor[5] = {1,2,3,4,5};`
  - `float vetor[] = {1,2,3,4,5};`

Vimos que podemos inicializar um vetor com valores específicos dessas formas

Matrizes - Declaração

- Inicialização
  - `float vetor[5] = {1,2,3,4,5};`
  - `float vetor[] = {1,2,3,4,5};`
  - `float matriz [2][3] = {1,2,3,4,5,6};`

Podemos seguir a mesma lógica para matrizes.
Neste exemplo, uma matriz 2x3, os 3 primeiros valores vão para a primeira linha e os 3 últimos para a segunda linha.

Matrizes - Declaração

- Inicialização
  - `float vetor[5] = {1,2,3,4,5};`
  - `float vetor[] = {1,2,3,4,5};`
  - `float matriz [2][3] = {1,2,3,4,5,6};`
  - `float matriz [2][3] = {{1,2,3},{4,5,6}};`

Para facilitar a visualização, podemos separar cada linha com chaves também

Matrizes - Declaração

- Inicialização
  - `float vetor[5] = {1,2,3,4,5};`
  - `float vetor[] = {1,2,3,4,5};`
  - `float matriz [2][3] = {1,2,3,4,5,6};`
  - `float matriz [2][3] = {{1,2,3},{4,5,6}};`
  - `float matriz [2][3] = {1,2,4,5,6};`

Se tiver menos valores que a matriz comporta,
os últimos elementos ficarão vazios

Matrizes - Declaração

- Inicialização
  - `float vetor[5] = {1,2,3,4,5};`
  - `float vetor[] = {1,2,3,4,5};`
  - `float matriz [2][3] = {1,2,3,4,5,6};`
  - `float matriz [2][3] = {{1,2,3},{4,5,6}};`
  - `float matriz [2][3] = {1,2,4,5,6};`
  - `float matriz [2][3] = {{1,2},{4,5,6}};`

A não ser que você identifique com chaves os valores de cada linha.
Nesse exemplo, o último elemento da primeira linha estará vazio.

Matrizes - Declaração

- Inicialização
  - `float vetor[5] = {1,2,3,4,5};`
  - `float vetor[] = {1,2,3,4,5};`
  - `float matriz [2][3] = {1,2,3,4,5,6};`
  - `float matriz [2][3] = {{1,2,3},{4,5,6}};`
  - `float matriz [2][3] = {1,2,4,5,6};`
  - `float matriz [2][3] = {{1,2},{4,5,6}};`
  - `float matriz [][3] = {1,2,3,4,5,6};`
  - `float matriz [][3] = {1,2,3,4,5,6,7,8,9};`

Ao inicializar na declaração, podemos deixar de especificar a quantidade de linhas. As linhas vão sendo preenchidas pelo número de colunas.

Matrizes - Utilização

- Usando uma determinada posição da matriz

```
vetor[3] = 5;
```

Em um vetor, podemos atribuir um valor a uma posição dessa forma.

Matrizes - Utilização

- Usando uma determinada posição da matriz

```
vetor[3] = 5;

matriz[1][2] = 4;
```

Em uma matriz, podemos podemos fazer de forma semelhante, especificando a linha e a coluna que queremos alterar. Neste exemplo, estamos atribuindo o valor 4 ao elemento da linha 1 e coluna 2 da matriz.

Matrizes - Utilização

- Usando uma determinada posição da matriz

```
vetor[3] = 5;

matriz[1][2] = 4;

int lin = 0, col = 3;

matriz[lin][col] = 8;
```

Podemos também usar variáveis para lidar com o posicionamento na matriz.

Matrizes - Utilização

- Usando uma determinada posição da matriz

```
vetor[3] = 5;

matriz[1][2] = 4;

int lin = 0, col = 3;

matriz[lin][col] = 8;

matriz[lin+1][col-1] = 5;
```

Até mesmo com operações

Matrizes - Utilização

- Usando uma determinada posição da matriz

```
vetor[3] = 5;

matriz[1][2] = 4;

int lin = 0, col = 3;

matriz[lin][col] = 8;

matriz[lin+1][col-1] = 5;

matriz[lin][col] = matriz[lin][col-1]*2;
```

Ou atribuir valores em uma posição com base no valor de outra posição

Matrizes - Utilização

- Percorrendo uma matriz

```
int matriz[2][3], i, j;
```

As operações com matrizes dependem muito de como percorremos ela, seja para preenchê-la ou para usar seus valores de alguma forma.

Na grande maioria das vezes, vamos precisar de 2 índices, uma para as linhas e outro para as colunas. Neste exemplo, vamos usar **i** para linhas e **j** para colunas.

Matrizes - Utilização

Primeiro vamos preencher a matriz com valores do usuário.

Vamos utilizar um laço de repetição para irmos da linha 0 até a última linha. É comum usarmos o **for**, mas dá para utilizar os outros laços.

- Percorrendo uma matriz

```
int matriz[2][3], i, j;
for (i=0; i<2; i++)
{

}
```

Matrizes - Utilização

- Percorrendo uma matriz

```
int matriz[2][3], i, j;
for (i=0; i<2; i++)
{
    for (j=0; j<3; j++)
    {
        scanf("%d",&matriz[i][j]);
    }
}
```

Para cada linha, vamos precisar visitar cada coluna, então precisamos de outro laço de repetição para irmos da primeira coluna à última.

Dessa forma, podemos receber um valor do usuário e armazená-lo na posição **[i][j]**.

Matrizes - Utilização

- Percorrendo uma matriz

```
int matriz[2][3], i, j;
for (i=0; i<2; i++)
{
    for (j=0; j<3; j++)
    {
        scanf("%d",&matriz[i][j]);
    }
}
for (i=0; i<2; i++)
{
    for (j=0; j<3; j++)
    {
        printf("matriz[%d][%d]=%d\n",i, j, matriz[i][j]);
    }
}
```

Sempre que precisarmos passar por toda a matriz, precisaremos dos dois laços de repetição.

Por exemplo, para imprimir os valores em uma matriz, vamos percorrer cada linha **i** e coluna **j** imprimindo a posição **[i][j]**.

Matrizes - Utilização

- Percorrendo uma matriz

```
int matriz[2][3], i, j;
for (j=0; j<3; j++)
{
   for (i=0; i<2; i++)
   {
      scanf("%d",&matriz[i][j]);
   }
}
for (i=0; i<2; i++)
{
   for (j=0; j<3; j++)
   {
      printf("matriz[%d][%d]=%d\n",i, j, matriz[i][j]);
   }
}
```

Note que se você trocar a linha pela coluna, você vai percorrer a matriz em uma ordem diferente.

Exercícios

- Faça um programa que receba do usuário os elementos de uma matriz 3x3 e a exiba na tela da seguinte forma:

  1 2 3
  4 5 6
  7 8 9

- Multiplique cada elemento da matriz anterior por 5

- Some os elementos da matriz

Exercícios

- Faça um algoritmo que pergunte para o usuário o número de linhas e o número de colunas que a matriz vai ter, preencha ela com valores sorteados de 0 a 9 e depois calcule a média dos valores da matriz.

- Receba duas matrizes 3x3 (A e B) e cria uma terceira matriz (C) com a soma dos elementos das duas primeiras (C[0][0] = A[0][0] + B[0][0]...)