

# Aula 14 – Estruturas de Dados - Matrizes

Algoritmos e Programação de Computadores

Profs: Ronaldo Castro de Oliveira – [ronaldo.co@ufu.br](mailto:ronaldo.co@ufu.br)

Anilton Joaquim da Silva – [anilton@ufu.br](mailto:anilton@ufu.br)

# Matrizes

- Uma matriz é uma variável capaz de armazenar vários dados de mesmo tipo, ou seja, é uma variável composta homogênea bidimensional.
- Declaração da Matriz  
**Tipo\_da\_matriz nome\_matriz [linhas][colunas];**
- Exemplos:
  - `int matriz1[3][3];`
  - `float matriz2[2][2];`
  - `char matriz_nomes[10][30];`

# Matrizes

- Seja o exemplo de números inteiros:

```
int matriz1 [ 3 ][ 3 ];
```

Linhas  
Colunas

Índice	0	1	2
0	7	2	5
1	4	8	1
2	9	6	3

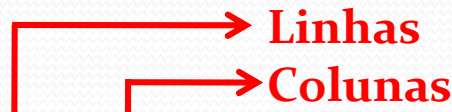
Memória	
1000	7
1001	
1002	2
1003	
1004	5
1005	
1006	4
1007	
1008	8
1009	
1010	1
1011	
1012	9
1013	
1014	6
1015	
1016	3
1017	
1018	

1 Linha  
2 Linha  
3 Linha

Obs.: os índices de uma matriz, em C++, sempre começam em 0. Logo o matriz1 pode armazenar 9 inteiros, com controle de índice começando em [0][0], [0][1], [0][2], [1][0]... até [2][2].

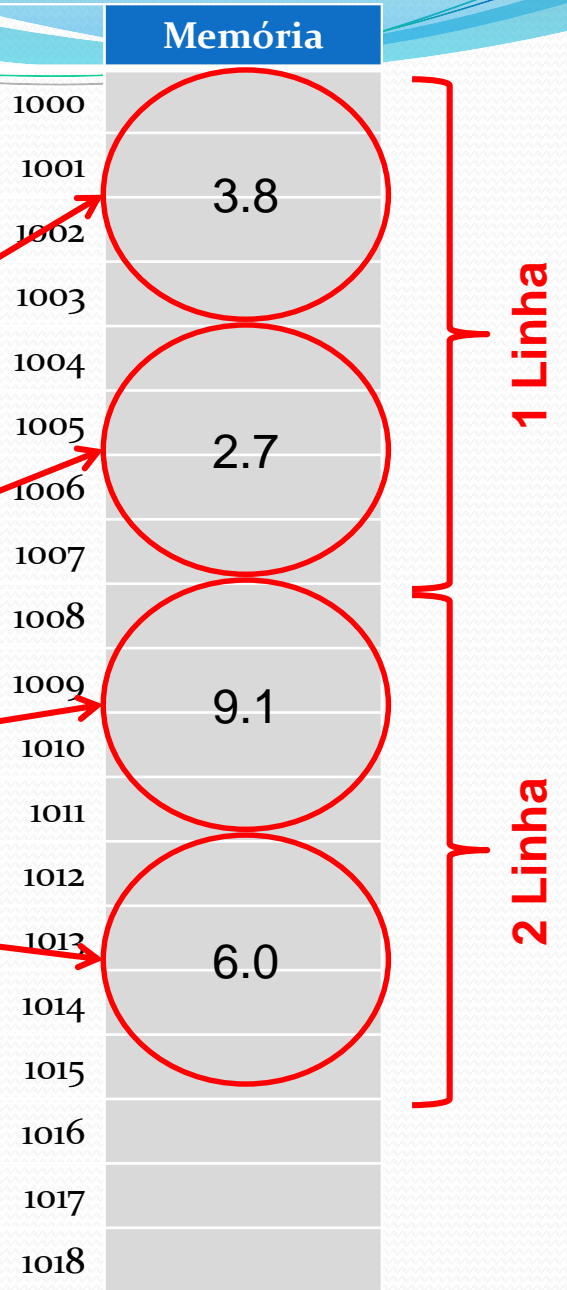
# Matrizes

- Seja o exemplo de números inteiros:



```
float matriz2 [ 2 ][ 2 ];
```

Índice	0	1
0	3.8	2.7
1	9.1	6.0

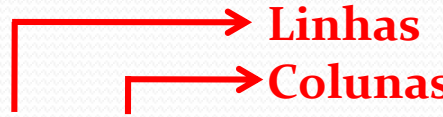


Obs.: A matriz2 armazena 4 floats, com controle de índices: [0][0], [0][1], [1][0], [1][1].

# Matrizes

- Seja o exemplo de números inteiros:

`int matriz1 [ 3 ][ 3 ];`



Índice	0	1	2
0	7	2	5
1	4	8	1
2	9	6	3

Posição:

`matriz1 [1][1] → 8`

`matriz1 [0][2] → 5`

`matriz1 [2][0] → 9`

`matriz1 [3][1] → ? → ERRO`

`matriz1 [2][3] → ? → ERRO`

# Atribuindo valores a uma matrizes

- Para atribuir diretamente um valor a uma determinada posição da matriz, pode-se usar a seguinte expressão:

**nome\_matriz[linha][coluna] = valor;**

Ex: int matriz1 [ 3 ][ 3 ];

matriz1[2][1] = 94;

matriz1[0][2] = 33;

matriz1[1][0] = 57;

Índice	0	1	2
0			33
1	57		
2		94	

matriz1[1][3] = 94; → **ERRO**

# Lendo uma matriz

- Para ler uma matriz e ter seus valores alocados em posições específicas, pode-se utilizar dois comandos for, um dentro do outro:

```
void le_matriz (int matriz[][tamanho], int L, int C)  
{  
    for (int linha = 0; linha < L; linha++)  
    {  
        for(int coluna = 0; coluna < C; coluna++)  
        {  
            cout << "Digite Matriz: [ " << linha << " ][ " << coluna << " ]: ";  
            cin >> matriz [linha][coluna];  
        }  
    }  
}
```

- OBS1: o procedimento le\_matriz recebe na variável matriz tipo inteiro uma passagem por referência ao matriz originalmente declarada. Valores atribuídos a esta variável serão atribuídos a definição original que efetuou a chamada do procedimento.
- OBS2: as variáveis L e C recebem passagens de valor, sendo definida como o tamanho da matriz em linhas e colunas.

# Imprimindo uma matriz

- Para mostra uma matriz já lida e ter seus valores apresentados na tela, também pode-se utilizar dois comandos for, um dentro do outro:

```
void mostra_matriz (int matriz [][][tamanho], int L, int C)
{
    cout << "Matriz digitada: " << endl;
    for (int linha = 0; linha < L; linha++)
    {
        for(int coluna = 0; coluna < C; coluna++)
        {
            cout << setw(10) << left << matriz[linha][coluna];

        }
        cout << endl;
    }
}
```



# Exemplo completo

```
#include <iostream>
#include <iomanip>
#define tamanho 100

using namespace std;

void le_matriz (int M[][tamanho], int L, int C)
{
    for (int linha = 0; linha < L; linha++)
    {
        for(int coluna = 0; coluna < C; coluna++)
        {
            cout << "Digite Matriz: [ " << linha << " ][ " << coluna << " ]: ";
            cin >>M[linha][coluna];
        }
    }
}

void mostra_matriz (int M[][tamanho], int L, int C)
{
    cout << "Matriz digitada: " << endl;
    for (int linha = 0; linha < L; linha++)
    {
        for(int coluna = 0; coluna < C; coluna++)
        {
            cout << setw(10) << left << M[linha][coluna];
        }
        cout << endl;
    }
}
```

```
int main()
{
    int matriz[tamanho][tamanho];
    int numeroLinhas, numeroColunas;

    cout << "Matriz - Le e mostra uma matriz" << endl;
    cout << "Digite numero de linhas: ";
    cin >> numeroLinhas;
    cout << "Digite numero de colunas: ";
    cin >> numeroColunas;

    le_matriz(matriz, numeroLinhas, numeroColunas);
    mostra_matriz(matriz, numeroLinhas, numeroColunas);

    return 0;
}
```

# Exercícios

1. Fazer um programa que leia a matriz A e a matriz B, calcula a matriz C que é a soma da Matriz A mais a matriz B.
2. Faça um programa que lê uma matriz quadrada (número de linhas é igual ao número de colunas) e retorne a soma dos elementos da matriz.
3. Faça um programa que cria e mostra na tela uma matriz  $M \times N$  de tal maneira que:
  - Se  $i=j$ , então  $\text{matriz}[i][j]= 0$ ;
  - Se  $i>j$ , então  $\text{matriz}[i][j] = i$ ;
  - Se  $i<j$ , então  $\text{matriz}[i][j] = j$ .
4. Faça um programa que lê uma matriz quadrada e retorna a soma dos elementos pertencentes à diagonal principal da matriz.

# Inicializando uma matriz com valores pré-definidos

- Uma matriz pode ter os seus valores inicializados na declaração da mesma:

```
int a[1][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}};
```

```
int b[2][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},  
               {11, 12, 13, 14, 15, 16, 17, 18, 19, 20}};
```

```
int c[3][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},  
               {11, 12, 13, 14, 15, 16, 17, 18, 19, 20},  
               {21, 22, 23, 24, 25, 26, 27, 28, 29, 30}};
```

OBS: isto também pode ser feito com vetores:

```
int vetor[6] = { 8, 7, 6, 5, 4, 3 };
```

# Exemplo completo

```
#include <iostream>
#include <iomanip>
#define tamanho 10

using namespace std;

void mostra_matriz (int M[][tamanho], int L, int C)
{ //.....mesma função
}

int main()
{
    int A[1][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}};
    int B[2][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
                   {11, 12, 13, 14, 15, 16, 17, 18, 19, 20}};
    int C[3][10] = {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
                   {11, 12, 13, 14, 15, 16, 17, 18, 19, 20},
                   {21, 22, 23, 24, 25, 26, 27, 28, 29, 30}};

    cout << endl << "Matriz A: " << endl;
    mostra_matriz (A, 1, 10);
    cout << endl << "Matriz B: " << endl;
    mostra_matriz (B, 2, 10);
    cout << endl << "Matriz C: " << endl;
    mostra_matriz (C, 3, 10);
    return 0;
}
```

# Multiplicação de Matrizes

- Sejam duas matrizes A e B, tal que A tem dimensões  $M \times N$  e B;  $N \times O$ ; então, a matriz  $A \times B$  tem dimensões  $M \times O$ , e

$$C_{i,j} = \sum_{k=1}^m A_{m,n} B_{n,o}$$

The diagram illustrates the calculation of each element in the resulting matrix  $C$  from the product of matrix  $A$  and matrix  $B$ . Matrix  $A$  is  $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$  and matrix  $B$  is  $\begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix}$ . The resulting matrix  $C$  is  $\begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$ .

Row 1, Column 1:  $1 \times 7 + 2 \times 9 + 3 \times 11 = 58$

Row 1, Column 2:  $1 \times 8 + 2 \times 10 + 3 \times 12 = 64$

Row 2, Column 1:  $4 \times 7 + 5 \times 9 + 6 \times 11 = 139$

Row 2, Column 2:  $4 \times 8 + 5 \times 10 + 6 \times 12 = 154$

# Multiplicação de Matrizes

```
#include <iostream>
#include <iomanip>
#define tamanho 100

using namespace std;

void mostra_matriz (int Mat[][tamanho], int L, int C)
{ //...mesma função }

void mult_matriz(int A[][tamanho], int B[][tamanho], int
    Mult[][tamanho], int M, int N, int O)
{ int soma = 0;
  for(int i = 0 ; i < M ; i++)
  {
    for(int j = 0 ; j < O ; j++)
    {
      for(int k = 0 ; k < N ; k++)
      {
        soma = soma + A[i][k] * B[k][j];
      }
      Mult[i][j] = soma;
      soma = 0;
    }
  }
}
```

```
int main()
{
  int A[tamanho][tamanho] = { {1, 2, 3},
                                {4, 5, 6}};

  int B[tamanho][tamanho] = { {7, 8},
                                {9, 10},
                                {11, 12}};

  int MULT[tamanho][tamanho];

  int M = 2, N = 3, O = 2;

  cout << endl <<"Matriz A: " << endl;
  mostra_matriz(A, M, N);
  cout << endl <<"Matriz B: " << endl;
  mostra_matriz(B, N, O);
  mult_matriz(A, B, MULT, M, N, O);
  cout << endl <<"Resultado da Multiplicacao: "
        << endl;
  mostra_matriz(MULT, M, O);
  return 0;
}
```

# Exercícios

1. Escreva um programa que leia uma matriz e diga se ela é a matriz identidade ou não.
2. Escreva um programa que leia uma matriz e imprima sua transposta.