# PROGRAMAÇÃO A

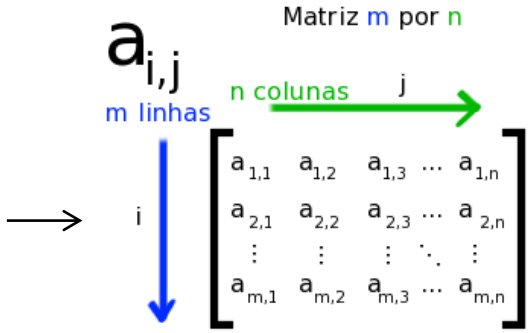
Matrizes

# DEFINIÇÃO

Em linguagem C, uma **matriz** é um vetor cujos itens são também vetores. Uma matriz pode ter qualquer número de dimensões, mas as matrizes **bidimensionais** são as mais usadas.

Uma **matriz** é uma **variável composta homogênea multidimensional.** Ela é formada por uma sequencia de variáveis, todas do mesmo tipo, com o mesmo identificador (mesmo nome), e alocadas sequencialmente na memória. Uma vez que as variáveis têm o mesmo nome, o que as distingue são índices que referenciam sua localização dentro da estrutura. **Uma variável do tipo matriz precisa de um índice para cada uma de suas dimensões.**

Organização de uma matriz
Representação matemática



# INTRODUÇÃO

- Uma matriz representa um conjunto bidimensional de valores.
- Similar a variáveis simples e vetores, matrizes devem ser declaradas para que o espaço de memória apropriado seja reservado.
- Como a matriz representa um conjunto bidimensional, devemos especificar as duas dimensões na declaração: o número de linhas e o número de colunas.

**Representação da matriz na memória do computador:**

Matriz A (2 linhas e 3 colunas)

| | | |
|---|---|---|
| 1 | 6 | 2 |
| 4 | 3 | 8 |

Matriz A armazenada na memória

| 1 | 6 | 2 | 4 | 3 | 8 | → Elementos da matriz |
|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | → Endereços da memória (linear e sequencial) |

# INTRODUÇÃO

Uma **matriz** pode ser definida como um conjunto de variáveis do mesmo tipo e identificadas pelo mesmo nome. Essas variáveis são diferenciadas por meio da especificação de suas posições dentro dessa estrutura.

Os índices usados na linguagem C, para identificar as posições de uma matriz, começam sempre em 0 (zero) e vão até o tamanho da dimensão menos uma unidade.

Os índices de uma matriz em C devem ser sempre representados por um dos tipos inteiros disponíveis na linguagem.

**Exemplos:**

```
char m[3][2];      // declaração de uma matriz de nome m com 3 linhas e 2 colunas que
                      pode armazenar até 6 caracteres, sendo 2 por linha.

int v[4][4];       // declaração de uma matriz de nome v com 4 linhas e 4 colunas que
                      pode armazenar até 16 números inteiros, sendo 4 por linha.

float p[2][5];     // declaração de uma matriz de nome p com 2 linhas e 5 colunas que
                      pode armazenar até 10 números reais, sendo 5 por linha.
```

# DECLARAÇÃO

A declaração (criação) de uma matriz no linguagem C é realizada especificando o tipo de dados da mesma, um nome e o seu tamanho (número de linhas e colunas).

**tipo_dos_dados nome_variável[dimensão1][dimensão2];**

Onde:

**tipo_dos_dados:** é o tipo dos dados que serão armazenados na matriz (int, float, char, etc.).

**nome_variável:** é o nome dado à variável do tipo matriz.

**[dimensão1]:** representa o tamanho da 1ª dimensão da matriz (número de linhas).

**[dimensão2]:** representa o tamanho da 2ª dimensão da matriz (número de colunas).

Exemplos:

```
int mNumeros[2][2];
float mValores[3][4];
char mLetras[6][8];
```

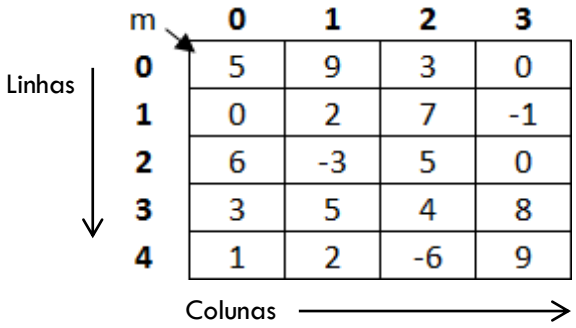
# REPRESENTAÇÃO GRÁFICA DE UMA MATRIZ

A declaração **int m[5][4];** cria uma matriz com **20 posições**, cada uma delas capaz de armazenar um número inteiro. Em cada linha da matriz é possível armazenar 4 valores (coluna).

Se for desejado iniciar essa matriz com valores específicos, podemos declarar:

```
int m[5][4] = {{5, 9, 3, 0},
               {0, 2, 7, -1},
               {6, -3, 5, 0},
               {3, 5, 4, 8},
               {1, 2, -6, 9}};
```

Com essa declaração, criamos uma variável **m** conforme ilustrado:




Se m[0][0] = 5 então:

m[0][1] = ?
m[2][2] = ?
m[4][3] = ?
m[1][2] = ?
m[3][3] = ?

# EXERCÍCIOS

Sendo a matriz M igual a:

| M | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 5 | 7 | 3 | 1 |
| 1 | 4 | 9 | 2 | -10 |
| 2 | 1 | -1 | 5 | 0 |

exemplo_slide_7.c

e as variáveis **L = 0** e **C = 1**, escreva o valor correspondente à solicitação:

a) M[L][C]

b) M[L][C-1]

c) M[L+C][C-L]

d) M[L*2][C*3]

e) M[3-C][L+1]

f) M[M[2][L]][3]

g) M[C][L]

h) M[M[2][3]][0]

i) M[M[L][3]][M[L][2]]

# DECLARAÇÃO, ATRIBUIÇÃO E EXIBIÇÃO DE VALORES

exemplo_slide_8.c

```c
/*
    Esse programa mostra as diversas formas de declaração
    e utilização de matrizes em linguagem C.
*/

#include <stdio.h>  // funções scanf e printf

int main()
{
    // declaração de matrizes
    int Mi[2][3] = {{1, 2, 3}, {4, 5, 6}};      // matriz de números inteiros com tamanho definido
    float Mf[2][2] = {{1.5, 2.8}, {3.0, 0.5}};  // matriz de números reais com tamanho definido
    char Mc[3][2];                              // matriz de caracteres com tamanho definido

    // atribuição de alguns valores para as matrizes
    Mc[0][0] = 'D'; // Elemento da 1ª linha e 1ª coluna da matriz
    Mc[1][1] = 'A'; // Elemento da 2ª linha e 2ª coluna da matriz
    Mc[2][1] = 'N'; // Elemento da 3ª linha e 2ª coluna da matriz

    // exibição de alguns elementos das matrizes
    printf("Mi[0][0] = %i\n", Mi[0][0]);
    printf("Mi[1][2] = %i\n", Mi[1][2]);
    printf("Mf[1][1] = %.2f\n", Mf[1][1]);
    printf("Mf[1][0] = %.2f\n", Mf[1][0]);
    printf("Mc[0][0] = %c\n", Mc[0][0]);
    printf("Mc[0][1] = %c\n", Mc[0][1]);
    printf("Mc[5][5] = %c\n", Mc[5][5]);    // ERRADO! CUIDADO!

    return 0;
}
```

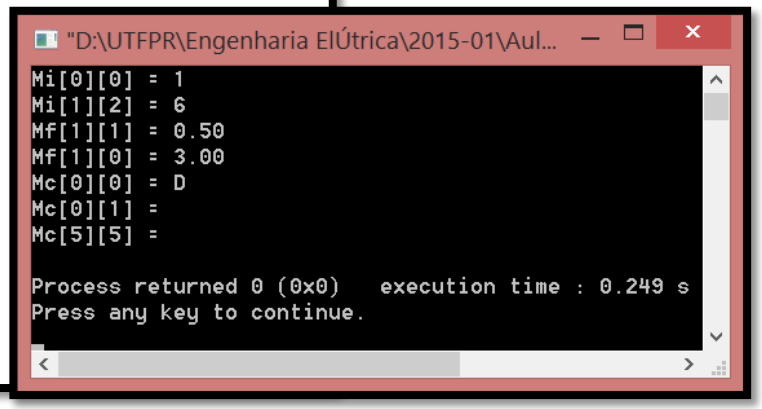


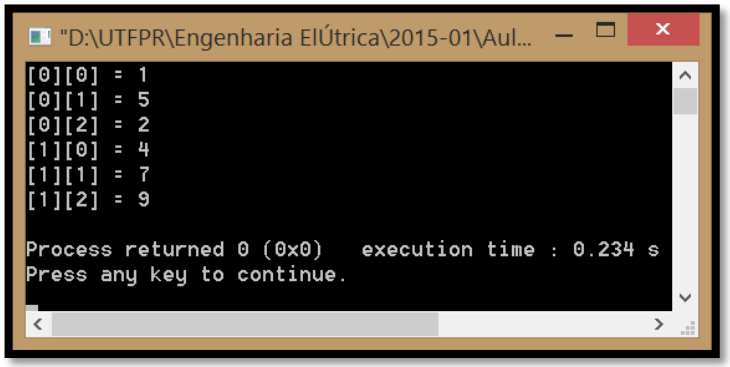

Figura 1 - Exemplo de declaração de matrizes e atribuição de valores em linguagem C

* As matrizes criadas nas linhas 11 e 12 foram inicializadas com valores fixos

# DECLARAÇÃO, ATRIBUIÇÃO E EXIBIÇÃO DE VALORES

Este primeiro exemplo mostra como declarar **uma matriz do tipo inteiro** de tamanho **2 x 3**, isto é, com **duas linhas e três colunas**. A matriz é então preenchida inicialmente com **valores fixos** e depois a **estrutura de repetição FOR** é utilizada para mostrar os elementos da matriz. Este exemplo exibe um elemento da matriz e pula uma linha para exibir o elemento subsequente.

```c
#include <stdio.h>  // printf, scanf

// constantes que controlam o tamanho da matriz
#define L 2
#define C 3

int main()
{
    // declaração de uma matriz 2x3 do tipo inteiro
    int M[L][C] = {{1, 5, 2}, {4, 7, 9}};

    // variáveis do tipo contador (i=linha, j=coluna)
    int i, j;

    // exibe os dados da matriz
    for(i=0; i<L; i++)  // para cada valor de i
    {
        for(j=0; j<C; j++) // para cada valor de j
        {
            // exibe o elemento da posição M[i,j]
            printf("[%d][%d] = %d\n", i, j, M[i][j]);
        }
    }

    return 0;
}
```

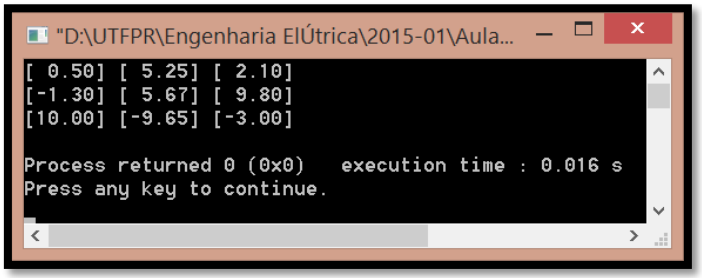



exemplo_slide_9.c

# DECLARAÇÃO, ATRIBUIÇÃO E EXIBIÇÃO DE VALORES

Este segundo exemplo mostra como **declarar uma matriz do tipo float (números reais) de tamanho 3 x 3,** isto é, com **três linhas e três colunas (chama-se de matriz quadrada).** A matriz é **preenchida inicialmente com valores fixos** e depois a **estrutura de repetição FOR** é utilizada para **mostrar os elementos da matriz.** Este exemplo exibe os elementos da matriz de cada linha e pula uma linha para exibir os elementos da linha subsequente.

```c
#include <stdio.h>  // printf, scanf

// constantes que controlam o tamanho da matriz
#define L 3
#define C 3

int main()
{
    // declaração de uma matriz 3x3 do tipo float
    float M[L][C] = {{0.5, 5.25, 2.1}, {-1.3, 5.67, 9.8}, {10, -9.65, -3}};

    // variáveis do tipo contador (i=linha, j=coluna)
    int i, j;

    // exibe os dados da matriz
    for(i=0; i<L; i++)  // para cada valor de i
    {
        for(j=0; j<C; j++)  // para cada valor de j
        {
            // exibe o elemento da posição M[i,j]
            printf("[%5.2f] ", M[i][j]);
        }

        // a cada linha da matriz pula uma linha na tela
        printf("\n");
    }

    return 0;
}
```

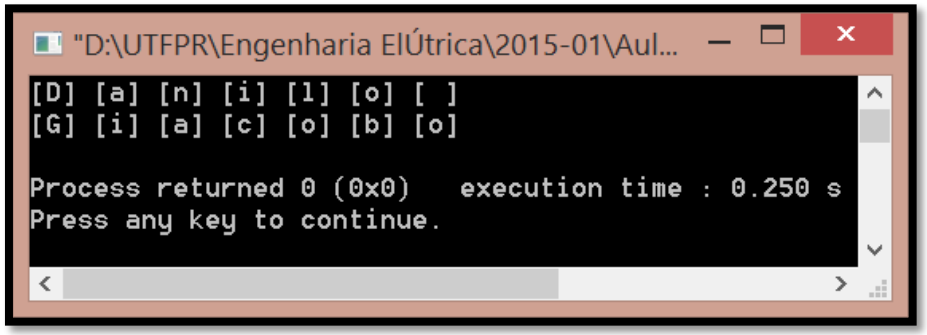



exemplo_slide_10.c

# DECLARAÇÃO, ATRIBUIÇÃO E EXIBIÇÃO DE VALORES

Este terceiro exemplo mostra como **declarar uma matriz do tipo char (caracteres) de tamanho 2 x 7**, isto é, com **duas linhas e sete colunas**. A matriz é **preenchida inicialmente com valores fixos** e depois a **estrutura de repetição FOR** é utilizada para **mostrar os elementos da matriz**. Este exemplo exibe os elementos da matriz de cada linha e pula uma linha para exibir os elementos da linha subsequente. Note que para cada linha da matriz um texto foi atribuído e que para as posições que não foram ocupadas na matriz um espaço em branco foi armazenado.

```c
#include <stdio.h>  // printf, scanf

// constantes que controlam o tamanho da matriz
#define L 2
#define C 7

int main()
{
    // declaração de uma matriz 2x7 do tipo char
    char M[L][C] = {"Danilo", "Giacobo"};

    // variáveis do tipo contador (i=linha, j=coluna)
    int i, j;

    // exibe os dados da matriz
    for(i=0; i<L; i++)  // para cada valor de i
    {
        for(j=0; j<C; j++)  // para cada valor de j
        {
            // exibe o elemento da posição M[i,j]
            printf("[%c] ", M[i][j]);
        }

        // a cada linha da matriz pula uma linha na tela
        printf("\n");
    }

    return 0;
}
```




exemplo_slide_11.c

# ESTUDO DE CASO - MATRIZ TRANSPOSTA

A **matriz transposta** é uma matriz onde ocorre a troca de linhas por colunas.

Uma **matriz simétrica** é toda a matriz que é igual à sua transposta.

A **matriz identidade** é simétrica. Portanto, a matriz transposta da matriz identidade é a própria matriz identidade.

Exemplos:

**Matriz 1x2**

$$\begin{bmatrix}1 & 2\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}1\\2\end{bmatrix}.$$

**Matriz 2x2**

$$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}1 & 3\\2 & 4\end{bmatrix}.$$

A

$$\begin{bmatrix}1 & 2\\3 & 4\\5 & 6\end{bmatrix}$$

# ESTUDO DE CASO - MATRIZ TRANSPOSTA

O objetivo é criar um programa em linguagem C para ler os elementos de uma matriz qualquer (por meio do teclado), apresentar os dados da mesma na tela para a pessoa ver eles organizados em uma matriz e depois exibir a matriz transposta da mesma.

Os passos a seguir compõem o algoritmo para solucionar este problema:

1. Pedir para a pessoa informar o tamanho da matriz (linhas e colunas);
2. Declarar uma matriz do tipo inteiro a partir dos dados fornecidos;
3. Ler os dados da matriz e armazenar os mesmos em suas respectivas posições;
4. Apresentar os elementos da matriz original;
5. Percorrer a matriz do início ao fim trocando o elemento que está na linha da matriz pela coluna da mesma;
6. Para simplificar o processamento basta trocar os índices da matriz quando estiver percorrendo a mesma e colocar o elemento na matriz transposta invertendo os índices;
7. Apresentar os elementos da matriz transposta.

exemplo_matriz_transposta.c

# ESTUDO DE CASO - MATRIZ IDENTIDADE

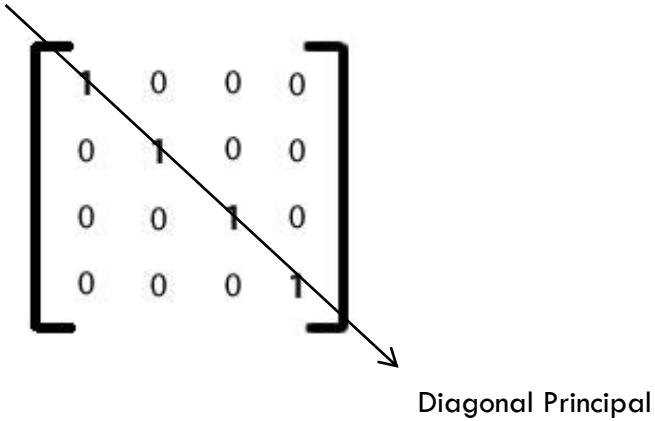
A **matriz identidade** é uma **matriz quadrada** em que todos os elementos da diagonal principal são iguais a 1 (um) e os demais são iguais a 0 (zero). Ela é chamada de matriz identidade pois multiplicá-la por outra matriz não altera a matriz original.

Exemplo:

**Matriz quadrada 4x4**

**- 4 linhas e**

**- 4 colunas**



# ESTUDO DE CASO - MATRIZ IDENTIDADE

O objetivo é criar um programa em linguagem C para ler os elementos de uma matriz quadrada qualquer (por meio do teclado), apresentar os dados da mesma na tela para a pessoa ver eles organizados em uma matriz e depois exibir uma mensagem informando se a matriz é classificada como identidade ou não.

Os passos a seguir compõem o algoritmo para solucionar este problema:

1. Pedir para a pessoa informar o tamanho da matriz (linhas e colunas);
2. Declarar uma matriz do tipo inteiro a partir dos dados fornecidos;
3. Ler os dados da matriz e armazenar os mesmos em suas respectivas posições;
4. Apresentar os elementos da matriz;
5. Percorrer a matriz do início ao fim verificando se a diagonal principal possui apenas valores iguais a 1 e se os demais elementos são 0;
6. Para simplificar o cálculo basta contar quantos números 1 (um) a matriz possui na diagonal principal e contar também os elementos iguais a 0 (zero) que não estão nesta diagonal;
7. Se a quantidade de números 1 (um) for igual ao tamanho de qualquer uma das dimensões da matriz e o número de valores iguais a 0 (zero) for igual ao total de elementos da matriz menos a quantidade calculada anteriormente, então ela é uma matriz identidade;
8. Sabe-se que o elemento pertence a diagonal da matriz quando a linha for igual à coluna onde este foi armazenado.

exemplo_matriz_identidade.c

# EXERCÍCIOS

1. Construir um programa para mostrar os dados de uma matriz de ordem 3 x 3, seguindo a orientação **aij = 3i + 2j**. Apresentar o resultado na tela

2. Construir um programa para mostrar os dados de uma matriz de ordem 4 x 4, de modo que **aij é i + j, se i = j e i – j, se i ≠ j**. Apresentar o resultado na tela.

3. Construir um programa para calcular o determinante de uma matriz de ordem 2 x 2. A pessoa irá informar os dados da matriz.

4. Construir um programa para calcular o determinante de uma matriz de ordem 3 x 3. A pessoa irá informar os dados da matriz.

exercicio_matriz_1.c

exercicio_matriz_2.c

exemplo_matriz_determinante_2x2.c

exemplo_matriz_determinante_3x3.c

# REFERÊNCIAS BIBLIOGRÁFICAS


- ASCENCIO, A. F. G.; CAMPOS, E. A. V. D. **Fundamentos da Programação de Computadores: Algoritmos, Pascal, C/C++ (Padrão ANSI) e Java**. 3. ed. São Paulo: Pearson Education do Brasil, 2012. 569 p.
- FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de Programação: A construção de algoritmos e estruturas de dados.** 3. ed. São Paulo: Prentice Hall, 2005. 218p.
- PEREIRA, S. D. L. **Algoritmos e Lógica de Programação em C: Uma abordagem didática**. 1. ed. São Paulo: Érica, 2010. 190 p.