Computação Eletrônica

Vetores e Matrizes

Prof: Luciano Barbosa

(Slides adaptados do Prof. Hansenclever Bassani)

Site da disciplina: www.cin.ufpe.br/~hfb/ce

Site da turma: www.cin.ufpe.br/~luciano/cursos/ce/

Centro de Informática U·F·P·E

Universidade Federal de Pernambuco

Vetores

- Até agora: **variável** e **constante**
- Armazenar centenas de valores -> criar centenas de variáveis
  - Não é uma boa solução
- Ex: armazenar as notas dos alunos
  - Cálculo da média
  - Ordernar as notas

Exemplo

- A média aritmética de um conjunto de valores é dada pela seguinte expressão:

$$m = \frac{\sum_{i=1}^{n} x_i}{n}$$

  - O programa abaixo calcula a média aritmética entre cinco valores:

```
int main()
{
    float  nota1, nota2, nota3, nota4, nota5;
    printf("Entre com a 1a. nota:");
    scanf("%f", &nota1);
    printf("Entre com a 2a. nota:");
    scanf("%f", &nota2);
    printf("Entre com a 3a. nota:");
    scanf("%f", &nota3);
    printf("Entre com a 4a. nota:");
    scanf("%f", &nota4);
    printf("Entre com a 5a. nota:");
    scanf("%f", &nota5);
    printf("Média = %f",(nota1+nota2+nota3+nota4+nota5)/5);
}
```

Como seria este programa se precisarmos calcular a média entre 200 notas?

Solução: Variáveis Compostas

- Conjunto (coleção) de valores de um mesmo tipo de dados
- Podem ser:
  - Unidimensionais: vetores
  - Multidimensionais: matrizes

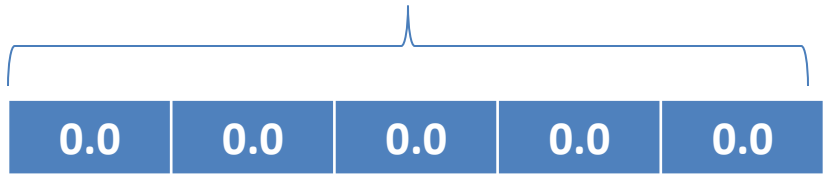
Variáveis Compostas Unidimensionais: Vetores (Arrays)

- Tipo de dado usado para representar uma coleção de variáveis de um mesmo tipo.
- Estrutura de dados homogênea e unidimensional.
- Sintaxe:

```
tipo nome_do_vetor[tamanho];
```

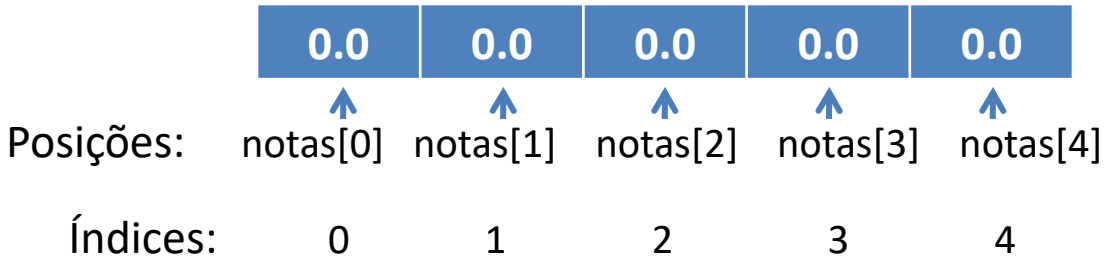
- Tamanho representa o número de elementos;
- O índice do vetor varia de 0 a (tamanho - 1);
- Ex.:

```
int main()
{
    float notas[5];
    ...
}
```



Vetores (Arrays)

- Coleção de caixas de variáveis
- Variáveis alocadas sequencialmente na memória
- Endereço inicial corresponde ao primeiro elemento (índice 0) do vetor.
- O acesso a cada posição do vetor realizado utilizando-se seu índice:



Acesso aos Elementos

- Nome do vetor seguido do índice do elemento entre colchetes
- **notas[2]** é 3º elemento do array:

- **notas[2]** é uma variável do tipo float

```
int main()
{
    float notas[5];
    ...
}
```

Exemplo

- Ler a nota de 5 alunos de uma disciplina e calcular a média.

```
int main()
{
    float nota[5];
    float soma = 0;
    int i;

    for (i = 0; i < 5; i++) {
        printf("Entre com a %da. nota: ", (i + 1));
        scanf("%f", &nota[i]);
        soma = soma + nota[i];
    }
    printf("Média da disciplina = %f", soma/5);
}
```

Declara um vetor de 5 posições.

Armazena o valor lido na i-ésima posição.

Acessa o valor da i-ésima posição.

Vetores: Valores nos Colchetes

- Significados diferentes:
  - **Na *declaração* de um vetor**: informa a quantidade N de posições que devem ser alocadas
  - **Após a declaração**: informa a posições que será acessada para leitura ou gravação de informação. Deve ser um valor entre $[0...(N\text{-}1)]$

```
int main()
{
    float nota[5];
    float soma = 0;
    int i;

    for (i = 0; i < 5; i++) {
        printf("Entre com a %da. nota: ", (i + 1));
        scanf("%f", &nota[i]);
        soma = soma + nota[i];
    }
    printf("Média da disciplina = %f", soma/5);
}
```

Declara um vetor de N=5 posições.

Grava informação na posição *i* em [0...4].

Lê informação da posição *i* em [0...4].

Vetores e seus limites

```c
int main()
{
    float notas[5];
    ...
}
```

- No exemplo anterior:
  - Vetor possui cinco posições
  - Índice da última posição é **4**
  - Acessar **notas[5]** **causa falha de memória**.
- Índice deve variar de *0 a [tamanho -1]*;
- C não avisa quando o limite de um vetor é excedido!
- Se o programa transpuser o fim do vetor durante a operação de atribuição, os valores serão armazenados em posições inválidas de memória, ou sobrescrevendo outras variáveis;

O programador tem a responsabilidade de verificar o limite do vetor!

Vetores e seus limites

- Erro comum: acesso fora dos limites
- Exemplo:

```
int main() {
    int pares[20];
    int i, somaPares = 0;
    for (i = 0; i <= 20; i++)      {
        pares[i] = 2 * i ;
        somaPares = somaPares + pares[i];
    }
    ...
}
```

Deveria ser: i < 20

A posição pares[20] está fora do vetor!

Onde está o erro?

Vetores: Inicialização

- Tipo 1:

```
int v[5] = {5,10,15,20,25};
```

- Tipo 2:

```
int v[]= {5,10,15,20,25};
```

- Compilador aloca espaço suficiente para armazenar todos os valores

Vetores só podem ser inicializados dessa forma em sua declaração!

Vetores: Inicialização

- Quando o tamanho do vetor for especificado e houver a lista de inicialização:
  - Se há menos inicializadores que o tamanho especificado, os outros serão zero;
  - Mais inicializadores que o necessário implica em um aviso de compilação (*warning*).
- Quando não inicializado: o tamanho deve ser especificado na declaração

Vetores: Declaração do Tamanho

- Valor literal, não uma variável
  - Determinado em tempo de compilação
- Exemplo:

```
int  nAlunos = 10;
printf ("entre com o número de alunos");
scanf ("%d", &nAlunos);
int  notas[nAlunos];
...
```

Erro!
nAlunos é desconhecido
em tempo de compilação

Vetores: Constantes #define pro Tamanho

- Conhecidas em tempo de compilação

```c
#include <stdio.h>
#define TOTAL_ALUNOS 10

int main()
{
    float nota[TOTAL_ALUNOS];
    float soma = 0;
    int i;

    for (i = 0; i < TOTAL_ALUNOS; i++) {
        printf("Entre com a %da. nota: ", (i + 1));
        scanf("%f", &nota[i]);
        soma = soma + nota[i];
    }
    printf("Média da disciplina = %f", soma/TOTAL_ALUNOS);
}
```

Recomendável! pois facilita a legibilidade e a manutenção do código. Se for preciso aumentar o número de alunos basta modifica-lo em um local.

Vetores como parâmetro de funções

- Pode ser passado como argumento para uma função
- Ao passar um vetor para uma função podemos modificar o conteúdo deste vetor dentro da função:
  - Passa-se na verdade o endereço do primeiro elemento do vetor na memória;
  - Os demais estão nas posições seguintes de memória;
- Podemos passar também um elemento em particular de um vetor para uma função
  - O parâmetro deve ser do mesmo tipo do vetor

Vetores como parâmetro de funções

```c
#include <stdio.h>

float media(int n, float num[])
{
    int i;
    float s = 0.0;
    for(i = 0; i < n; i++)
        s = s + num[i] ;
    return s/n ;
}

int main()
{
    float numeros[10] ;
    float med;
    int i ;
    for(i = 0; i < 10; i++)
        scanf ("%f", &numeros[i]) ;
    med = media(10, numeros) ;
    ...
}
```

Recebe um vetor float e o número de elementos como parâmetro e calcula a média

Declaração de um vetor float de 10 posições

Passa este vetor como argumento para a função "media"

Vetores como parâmetro de funções

```c
#include <stdio.h>

void incrementar(float num[], int n, float valor)
{
    int i;
    for(i = 0; i < n; i++)
        num[i] = num[i] + valor;
}
int main()
{
    float numeros[10];
    int i ;
    for(i = 0; i < 10; i++)
        scanf("%f", &numeros[i]);

    incrementar(numeros, 10, 1.5);

    for(i = 0; i < 10; i++)
        printf("%f ", numeros[i]);

    return 0;
}
```

Recebe um vetor float, o número de elementos e um valor que será utilizado para **modificar** o vetor

Declaração de um vetor float de 10 posições

Passa este vetor juntamente com os outros valores como argumento para a função "incrementar"

Imprime o vetor modificado por "incrementar"

Posição de vetor como parâmetro de funções

```c
#include <stdio.h>

int aprovado(float nota, float media)
{
    int resultado = 0;
    if ( nota >= media)
        resultado = 1;
    return resultado;
}
int main()
{
    float notas[] = {6.5,5.0,7.5,9.4,3.8};
    int i ;
    for(i = 0; i < 5; i++)
    {
        if (aprovado(notas[i],7.0) == 1)
            printf("Aluno %d: aprovado\n",i);
        else
            printf("Aluno %d: REPROVADO\n",i);
    }
    ...
}
```

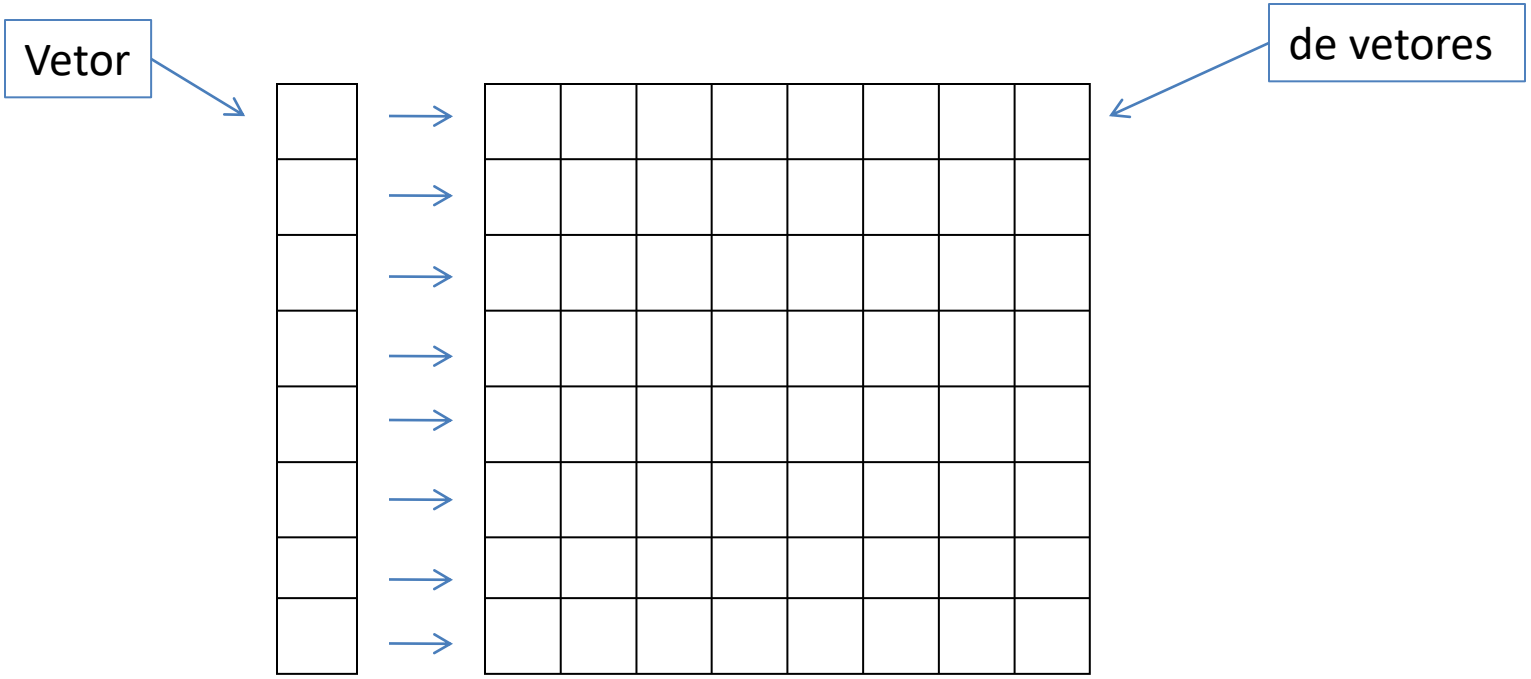
Recebe uma nota float, e a média com a qual a nota será comparada

Declaração de um vetor com as notas já preenchidas

Passa o valor na posição i do vetor como argumento para a função "aprovado"

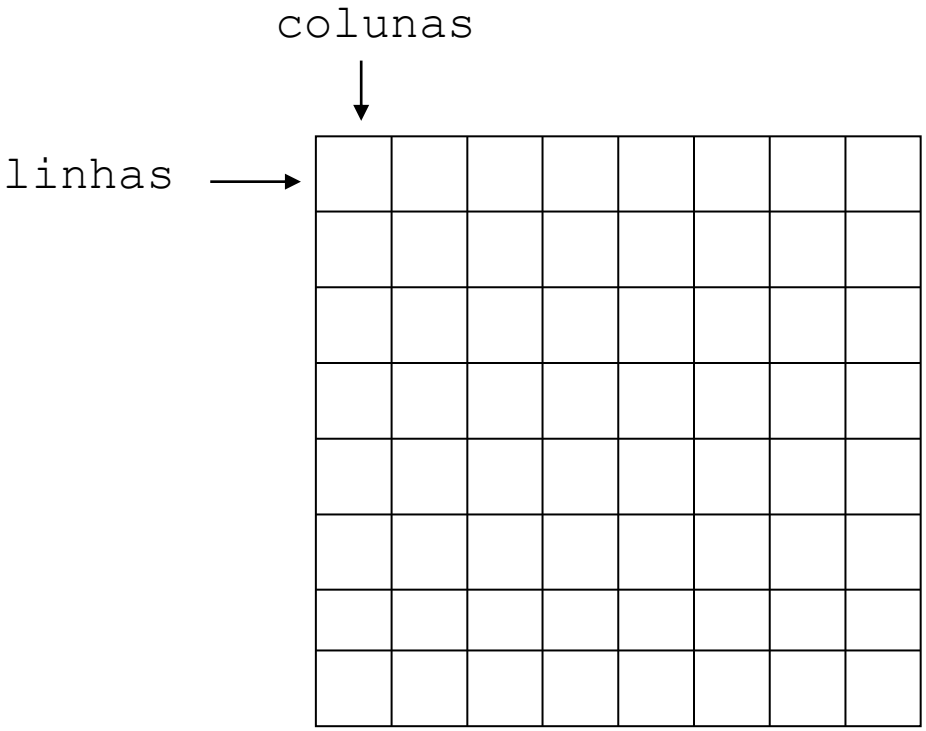
Variáveis Compostas Bidimensionais (matrizes)

- Mais de um índice para endereçamento.
- Vetor de vetores: um vetor onde cada posição contém um outro vetor:



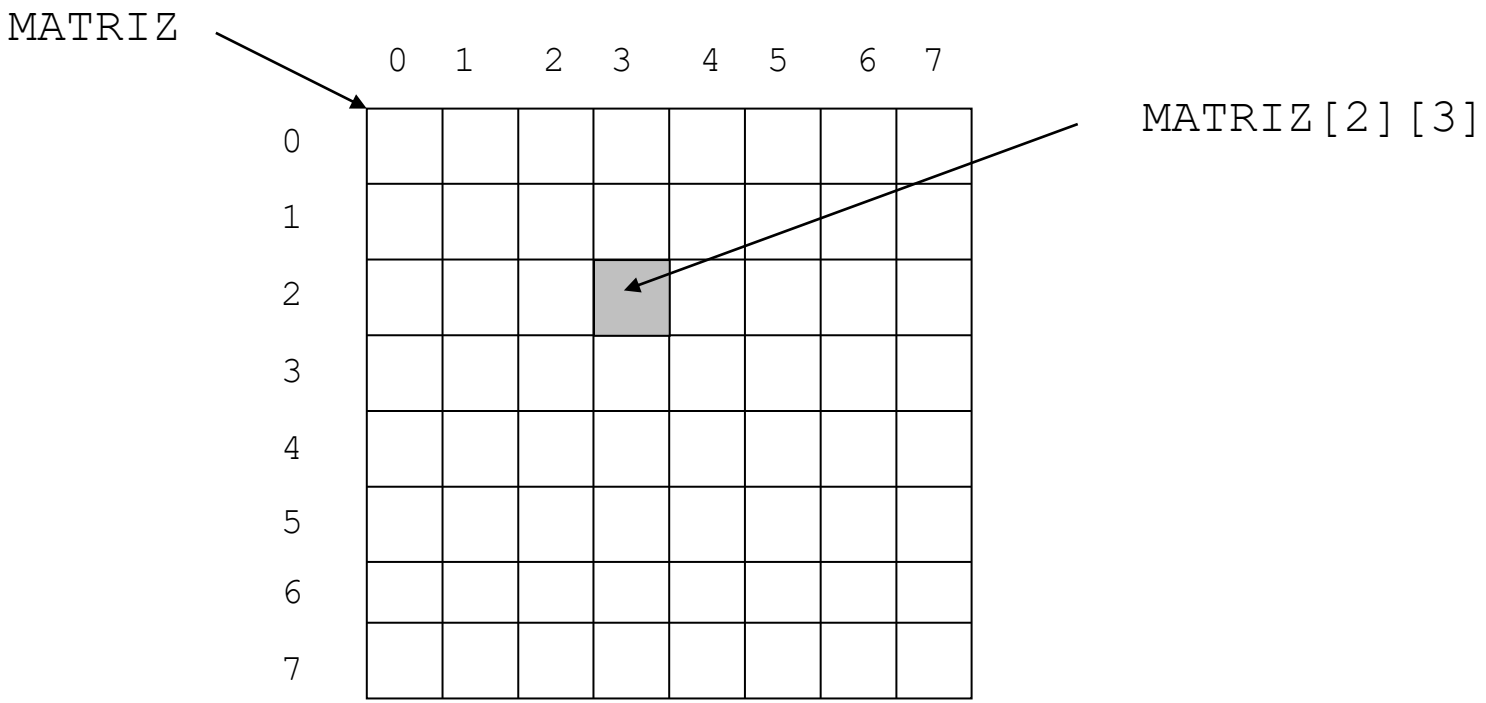
Variáveis Compostas Bidimensionais (matrizes)

- Exemplo:



Variáveis Compostas Bidimensionais (matrizes)

- Manipulação:



Variáveis Compostas Bidimensionais (matrizes)

- Matriz é uma estrutura de dados homogênea bidimensional.
- Declaração:

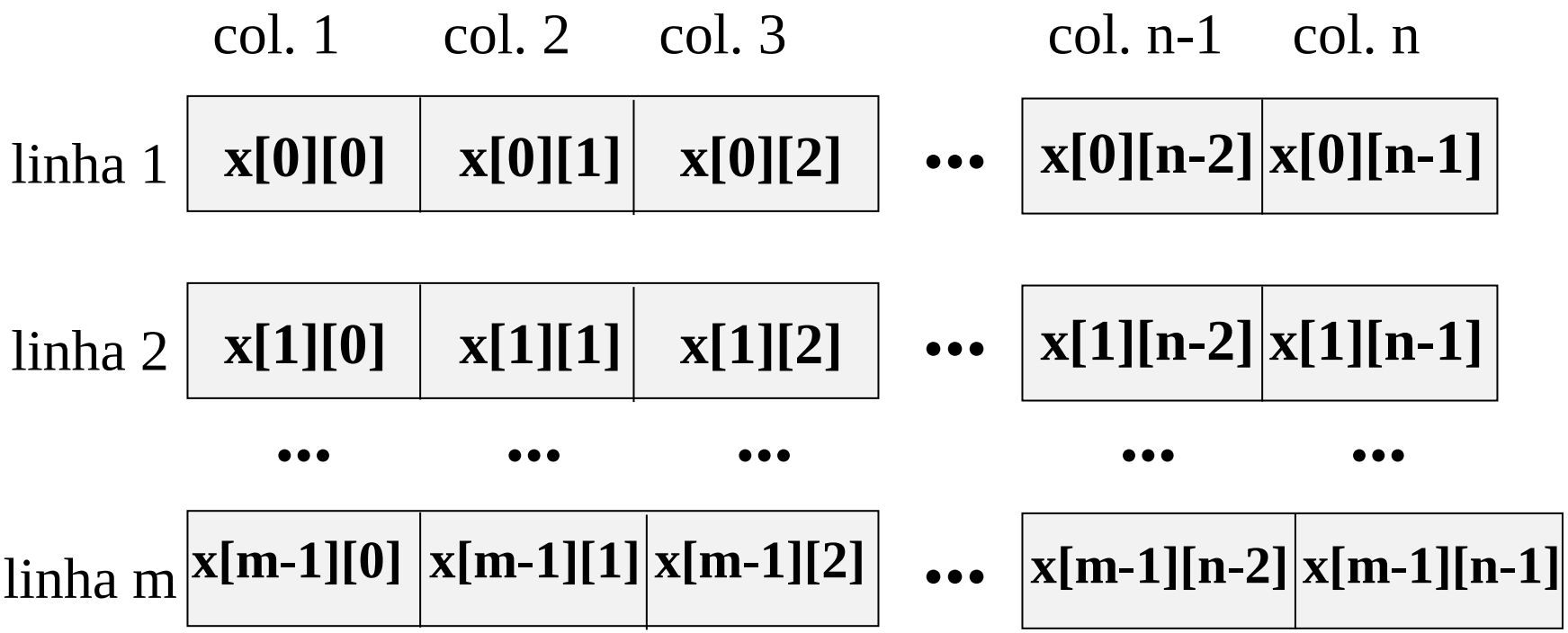
tipo nome_da_matriz[m][n];

m: representa o número de linhas da matriz

n: o número de colunas

- As duas dimensões são, respectivamente, a quantidade de linhas e colunas da matriz
- Ex: int mat[10][3];

Variáveis Compostas Bidimensionais (matrizes)



x é uma matriz bidimensional m x n.

Matrizes: Inicialização

- Pode ser feita na declaração:

```
float mat[4][3]={{5.0,10.0,15.0},{20.0,25.0,30.0},
                 {35.0,40.0,45.0},{50.0,55.0,60.0}};

float mat [][3]= {5.0, 10.0, 15.0, 20.0, 25.0, 30.0,
                  35.0,40.0,45.0,50.0,55.0,60.0};
```

- No segundo caso, deve ser informada ao menos a segunda dimensão
- Usando laços:

```
int mat[3][4];
int i,j;

//Inicialza a matriz com 1's
for (i=0; i<3; i++) {
    for (j=0; j<4; j++) {
        mat[i][j] = 1;
    }
}
```

Matrizes - Impressão

- Exemplo: imprimindo o conteúdo de uma matriz utilizando laços aninhados:

```c
#include <stdio.h>

int main()
{
    int i, j, matriz[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    for (i = 0; i < 3; i++)
    {
        for(j = 0; j < 3; j++)
        {
            printf("%d ",  matriz[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
Z:\UFPE\Graduacao\Disciplinas\ComputacaoEletronica\Exemplos\Ma
1 2 3
4 5 6
7 8 9

Process returned 0 (0x0)   execution time : 0.340 s
Press any key to continue.
```

Matrizes passadas por parâmetro

- É necessário especificar a quantidade de colunas da matriz ou todas as dimensões:

```
float media(float a[][2], int lin)
{
    int i;
    float avg, sum=0.0;
    for(i=0; i<lin; ++i)
    {
        for(j=0; i<2; ++j)
            sum+=a[i][j];
    }
    avg =(sum/(lin*2));
    return avg;
}
```

Atividade - Matrizes

1. Construa um algoritmo que efetue e apresente o resultado da soma entre duas matrizes 3 x 5. Inicialize a matriz com valores quaisquer e imprima o resultado na tela.
2. Faça um programa que multiplica uma matriz 3 x 3 de inteiros por um escalar k e imprima o resultado na tela. O usuário deve fornecer os valores da matriz e de k.
3. Leia uma matriz 20 x 20. Leia também um valor X. O programa deverá fazer uma busca desse valor na matriz e, ao final escrever a localização (linha e coluna) ou uma mensagem de “não encontrado”.
4. Dada uma matriz 5x5, elabore um algoritmo que imprima:
   - A diagonal principal
   - A diagonal secundária
   - A soma da linha 4
   - A soma da coluna 2
   - Tudo, exceto a diagonal principal
5. Refaça as questões anteriores criando uma função para cada uma delas.

Atividades adicionais - Matrizes

1. Faça um programa para multiplicar duas matrizes com tamanho até 10x10, armazenando o resultado em uma terceira matriz.
   - O programa deve solicitar ao usuário as duas dimensões das duas matrizes;
   - O programa deve verificar se as matrizes podem ser multiplicadas e apresentar uma mensagem de erro, caso não seja possível.
2. Refaça o programa anterior transformando **apenas** o código que faz a multiplicação das matrizes em uma função.
   - A função recebe como parâmetro as três matrizes e as dimensões das duas primeiras matrizes. O resultado da multiplicação das duas primeiras matrizes deve ser armazenado na terceira matriz.;
   - A função deve retornar falso se não for possível multiplicar as matrizes, e verdadeiro caso contrário.
   - A função não deve ler as matrizes, imprimir o resultado, nem a mensagem de erro. Isto deve ser feito na função principal.