ESTRUTURA DE DADOS

AULA 02 – MATRIZES

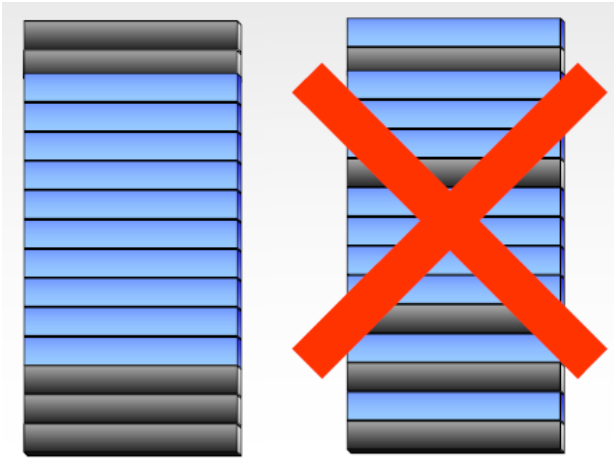
(ARRAY BIDIMENSIONAL E MULTIDIMENSIONAL)

VARIÁVEIS COMPOSTAS

✓ **São um conjunto de variáveis identificadas por um mesmo nome que correspondem a posições de memória distintas.**

- Homogêneas (vetores e matrizes)
- Heterogêneas (estruturas)

✓ **Aloca-se uma porção contígua da memória para armazenar os elementos de vetores e matrizes.**



PORÇÃO CONTÍGUA DE MEMÓRIA

Porção contígua da memória para armazenar os elementos de vetores e matrizes.

Característica *fundamental* destas estruturas de dados

Define como os dados estarão organizados na memória

Permite a manipulação por meio de índices, associados aos endereços de memória

Permite a manipulação por meio de aritmética para o deslocamento sobre endereços de memória

MATRIZES >> ARRAYS MULTIDIMENSIONAIS

- ✓ São ED homogêneas (compostas por elementos de um único tipo ou do mesmo tipo);
- ✓ Os elementos são representados por um único nome de variável;
- ✓ Podem variar quanto a sua dimensão, isto é, a quantidade de índices necessária para identificar (individualizar) cada elemento do conjunto;
- ✓ Muito utilizado para trabalhar com conjuntos de dados que são semelhantes em tipo. *Ex: o conjunto das alturas dos alunos de uma turma, ou um conjunto de seus nomes.*
- ✓ Objetivo: colocar as informações sob um mesmo conjunto, e poder referenciar cada dado individualmente por índices.

MATRIZES>> ARRAY BIDIMENSIONAL

- ✓ **Conhecidas como arranjos multidimensionais;**
- ✓ **Exemplo:**

arranjo de 2 dimensões

MATRIZES>> ARRAY MULTIDIMENSIONAL

Para cada valor da matriz:

- ✓ **haverá um par de índices diferentes para acessá-los, mesmo que os valores armazenados sejam repetidos;**
- ✓ **Cada par de índices da matriz representa um endereço de memória diferente e consecutivo aos demais elementos dessa matriz.**
- ✓ **Para cada elemento da matriz existe um endereço de memória diferente.**
- ✓ ***Exemplo: matriz 3x4***

M[0][0]=3 M[1][0]=0 M[2][0]=2
M[0][1]=8 M[1][1]=2 M[2][1]=5
M[0][2]=1 M[1][2]=4 M[2][2]=9
M[0][3]=5 M[1][3]=7 M[2][3]=3

M =

Matriz

3	8	1	5
0	2	4	7
2	5	9	3

MATRIZES>> ARRAY MULTIDIMENSIONAL

✓**Características**:

✓**matriz é um arranjo que permite acesso direto ao elemento;**

✓**Os elementos são independentes >>> para acessar qualquer elemento da matriz, basta referenciar o par de índices desejado, não é necessário acessar todos os elementos e nem os elementos anteriores ou posteriores;**

✓ **Uma variável matriz armazena vários valores simultaneamente sob o mesmo nome de variável.**

Matriz

M =

3	8	1	5
0	2	4	7
2	5	9	3

MATRIZES>> ARRAY MULTIDIMENSIONAL

✓ **As matrizes são listas ordenadas, pois possuem índices que variam de 0 a n, em cada dimensão.**

✓ **Os índices iniciam no 1° elemento da matriz e vai até o último elemento declarado na variável.**

✓ **O primeiro elemento da matriz tem índices 0 e 0 e o último tem índices *tamanho1 – 1 e* , *tamanho2 – 1*, referente ao tamanho de cada dimensão da matriz.**

✓**Sintaxe:**

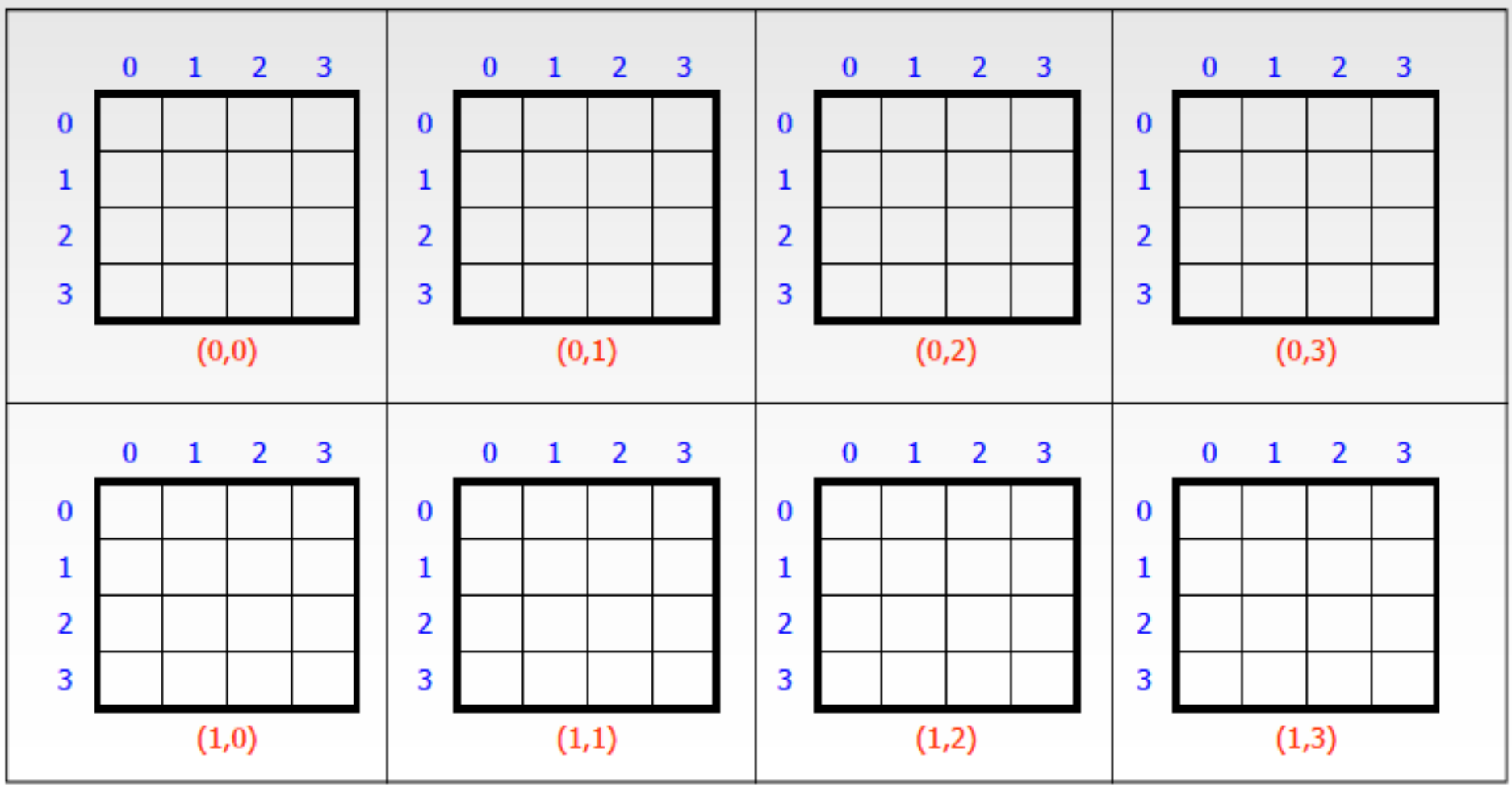
```
<tipo> <nome> [<tamanho1>][<tamanho2>]...;
```

✓**Ex.:**

```
int   matriz[5][9];
float cubo[20][12][7];
```

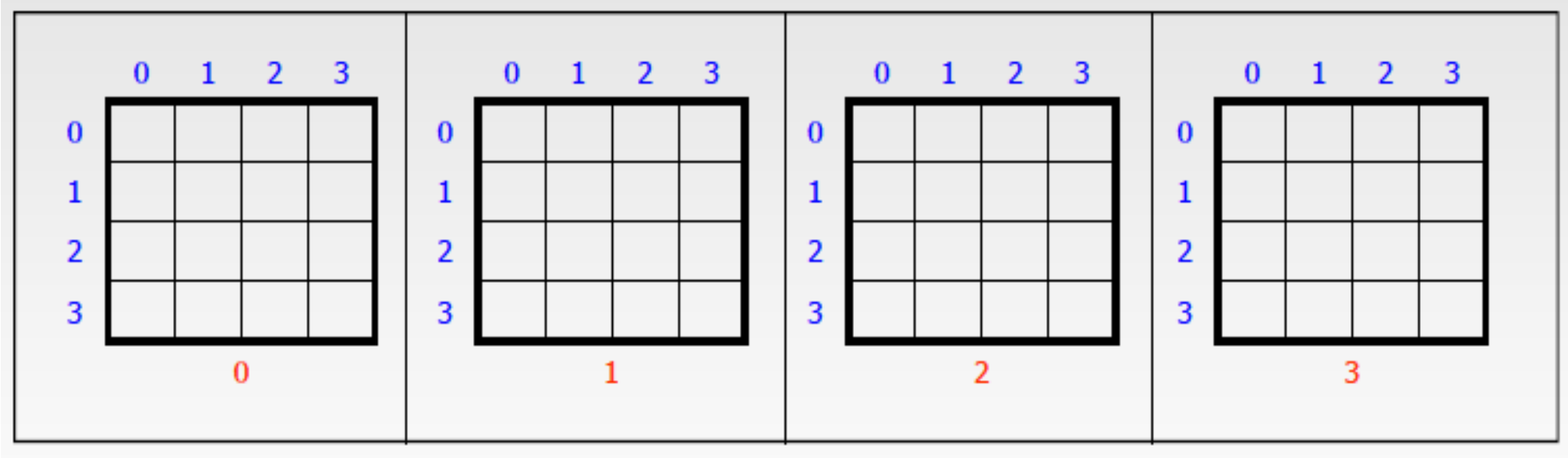
MATRIZES>> ARRAY MULTIDIMENSIONAL

- ✓ **Exemplo:** **Arranjo de 4 dimensões**



MATRIZES>> ARRAY MULTIDIMENSIONAL

✓ **Exemplo:** **Arranjo de 3 dimensões**



MATRIZES>> ARRAY MULTIDIMENSIONAL

✓**Ao fazer a declaração de uma matriz é reservado espaço de memória para os seus elementos.**

✓ **A quantidade de memória (em *bytes*) usada para armazenar uma matriz pode ser calculada como: *quantidade de memória = tamanho do tipo * qtde elementos da matriz*.**

✓ **As matrizes também podem ser inicializadas na declaração. Sintaxe da inicialização na declaração:**

tipo nome_matriz[tam][tam] = {{valores_linha0}, {valores_linhan}};

ATENÇÃO! A lista de valores é uma lista, separada por vírgulas, dos valores de cada elemento da matriz e também para separar os elementos de cada linha.

MATRIZES>> EXEMPLO

```
int tabela [2][3] = {{1,2,3}, {4,5,6}};
float tabela[3][5] = {{1.0, 2.0, 3.0, 4.0, 5.0},
                      {6.0, 7.0, 8.0, 9.0, 10.0},
                      {11.0, 12.0, 13.0, 14.0, 15.0}};
```

✓ **A forma clássica de inicializar uma matriz pode ser feita por meio de dois laços *for* aninhados, mas pode ser utilizada qualquer outra estrutura de repetiçã.**

```
for (i=0; i < N; i++){
  for (j=0; j < M; j++){
        Matriz[i][j] = 0;
}
}
```

✓ **Uma vez definido o tamanho da matriz, este tamanho não poderá ser alterado em tempo de execução, pois essa estrutura foi alocada estaticamente.**

VETORES >> ARRAYS UNIDIMENSIONAIS

QUIZ: Vetores e matrizes – Mitos e verdades

( ) Sintaticamente, não há problema apenas declarar e não inicializar uma matriz.
( ) Vetores e matrizes estão organizados na memória principal da mesma forma.
( ) Os vetores e as matrizes podem ser alocados estaticamente ou dinamicamente.
( ) Matrizes não inicializadas possuem valores desconhecidos considerados "lixo".
( ) Não é possível manipular matrizes que não foram inicializadas.

( ) Manipular valores de um vetor não inicializado pode ocasionar erros em tempo de execução ou de processamento.
( ) Uma matriz não inicializada possui valores nulos, branco ou 0 de acordo com o tipo de dado que ele foi declarado.
( ) Uma matriz não inicializada na declaração pode ter valores armazenados pelo usuário em tempo de execução.
( ) Não faço ideia.

PONTOS IMPORTANTES!

✓ **1. Definir o tamanho das matrizes e dos vetores utilizando constantes flexibiliza a manutenção do código. Além de ser uma boa prática.**

✓ **2. Pode-se fazer a inicialização de uma matriz sem tamanho definido. Ex:**

```
int m[][2] = {23, 45, 54, 55,77, 65}
```

✓ **3. A matriz é a uma estrutura de dados do tipo vetor com duas ou mais dimensões.**

✓ **4. Os itens de uma matriz tem que ser todos do mesmo tipo de dado.**

✓ **5. Na prática, as matrizes formam "tabelas" mas os valores estão sequenciados na memória.**

✓ **6. Para o preenchimento de uma matriz, deve-se percorrer todas as suas posições e atribuir-lhes um valor.**

✓ **7. Para identificar algum item de uma matriz, deve-se utilizar um par de índices.**

MATRIZES>> EXEMPLO

```
#include <stdio.h>
#include <stdlib.h>
main()
{
   int linha, coluna, valor[2][2];      //matriz de 4 elementos

   //inicializa matriz [2][2] com os valores digitados pelo usuario
   for (linha = 0; linha < 2; linha++)
   {
        for (coluna = 0; coluna < 2; coluna++)
        {
                printf("Digite um valor inteiro para a linha%d e coluna %d: ", linha, coluna);
                scanf("%d",&valor[linha][coluna]);
        }
    }
    //imprime os valores do vetor
    for (linha = 0; linha < 2; linha++)
    {
        for (coluna = 0; coluna < 2; coluna++)
        {
                printf("Linha%d e coluna %d: %d\n", linha, coluna,valor[linha][coluna]);
        }
     }
     system("pause");
}
```

PRIMEIROS PASSOS

1º:

Criar uma matriz de 5x5 para armazenar números reais;

2º :

Inicializar a matriz com valores fornecidos pelo usuário;

3º :

Exibir os valores da diagonal principal;

4º :

Calcular e exibir a quantidade de números inferiores a 10;

5º :

Calcular e exibir a soma dos valores negativos.

DESAFIOS...

1) **Faça um programa que leia duas matrizes A e B, contendo cada um 10 elementos numéricos quaisquer. Crie uma nova matriz, também de 10 elementos, onde cada elemento corresponde a soma dos elementos de A e B. Escreva na tela os elementos do novo conjunto, obtido.**
2) **Dada uma matriz de 4x5 elementos inteiros. Faça um programa que:**
   a) a inicialização de todos os elementos da matriz seja com valores informados pelo usuário;
   b) calcule e mostre na tela a soma de cada linha;
   c) calcule e mostre na tela a soma de cada coluna;
   d) calcule e mostre na tela a soma de todos os seus elementos, bem como a média desses elementos;
   Obs: utilize um vetor para armazenar o resultado da soma de cada linha e outro para a soma de cada coluna.

DESAFIOS...

3) **Faça um programa que leia uma matriz A que armazene valores inteiros quaisquer informados pelo usuário. O algoritmo deverá procurar na matriz se existe um determinado valor que está sendo buscado pelo usuário. Caso exista, exibir uma mensagem avisando que encontrou e em qual posição da matriz. Caso não, exibir uma mensagem de que não foi encontrado.**

4) **Crie uma matriz quadrada e a inicialize de forma que seja atribuído o valor 2 quando os índices forem iguais e -3 quando os índices forem diferentes. Calcule e imprima na tela apenas o somatório da diagonal principal.**

5) **Escreva um programa que tenha uma matriz de 12 elementos quaisquer informados pelo usuário e imprima quantos elementos são pares e quantos são ímpares, bem como a soma total de cada um.**