

Introdução a Programação – 2023.1

MATRIZES

Prof^a. Giorgia Mattos –
giorgiamattos@gmail.com

Essa definição aloca
isso na memória

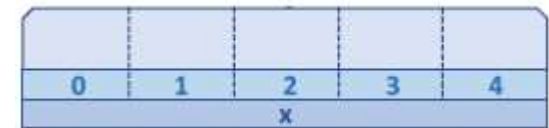
`int x;`



Um **espaço** associado a
um **identificador**,
capaz de armazenar um
único valor inteiro

Essa definição aloca
isso na memória

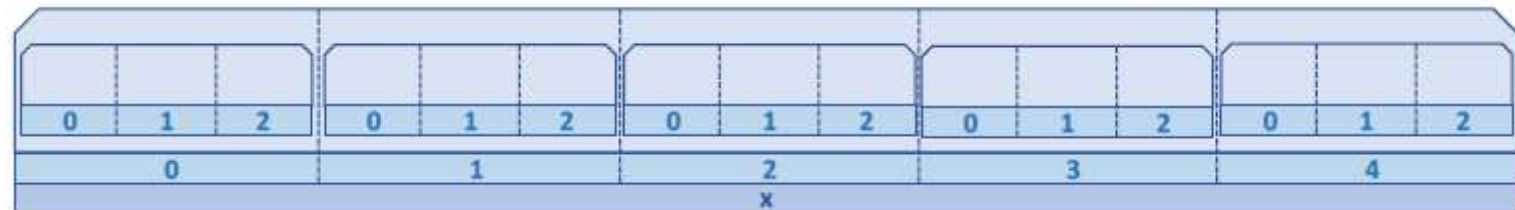
`int x[5];`



Cinco espaços contínuos
associados a **um**
identificador, capaz de
armazenar **cinco valores**
inteiros

Essa definição aloca
isso na memória

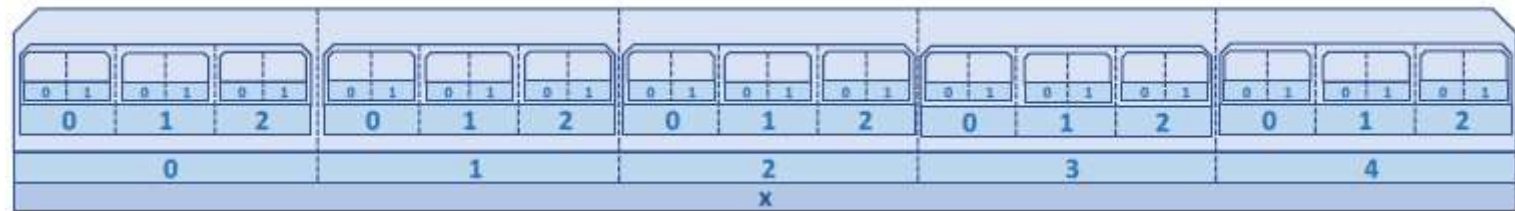
`int x[5][3];`



Uma região contígua associada a um identificador, dividida em cinco espaços onde cada subespaço é capaz de armazenar três valores inteiros

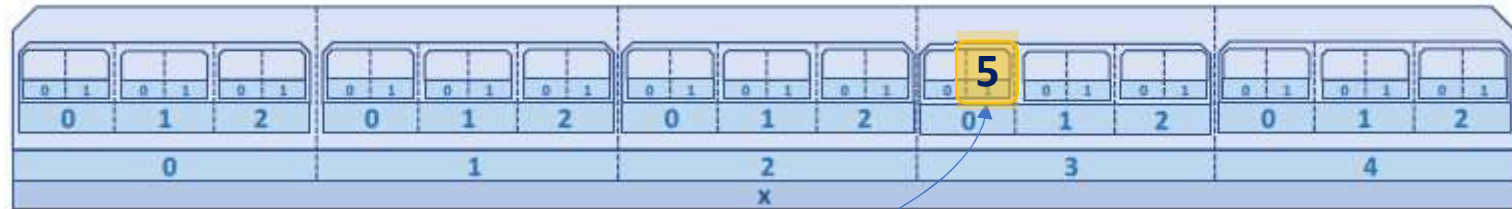
Essa definição aloca
isso na memória

`int x[5][3][2];`



Uma região contígua associada a um identificador, dividida em cinco espaços onde cada subespaço é dividido em outros três espaços onde cada subespaço armazena dois valores inteiros

```
int x[5][3][2];
```



Como acessar um elemento?

```
x [3][0][1] = 5;
```

1. Usar o **identificador** da variável

2. Indicar o **elemento** da **primeira dimensão**

3. Indicar o **elemento** da **segunda dimensão**

4. Indicar o **elemento** da **terceira dimensão**

5. Por exemplo, atribuir **5** ao elemento

Outros exemplos:

```
char arrayCaracteres[3][4][5];
```

```
arrayCaracteres[1][0][2] = 'A';
```

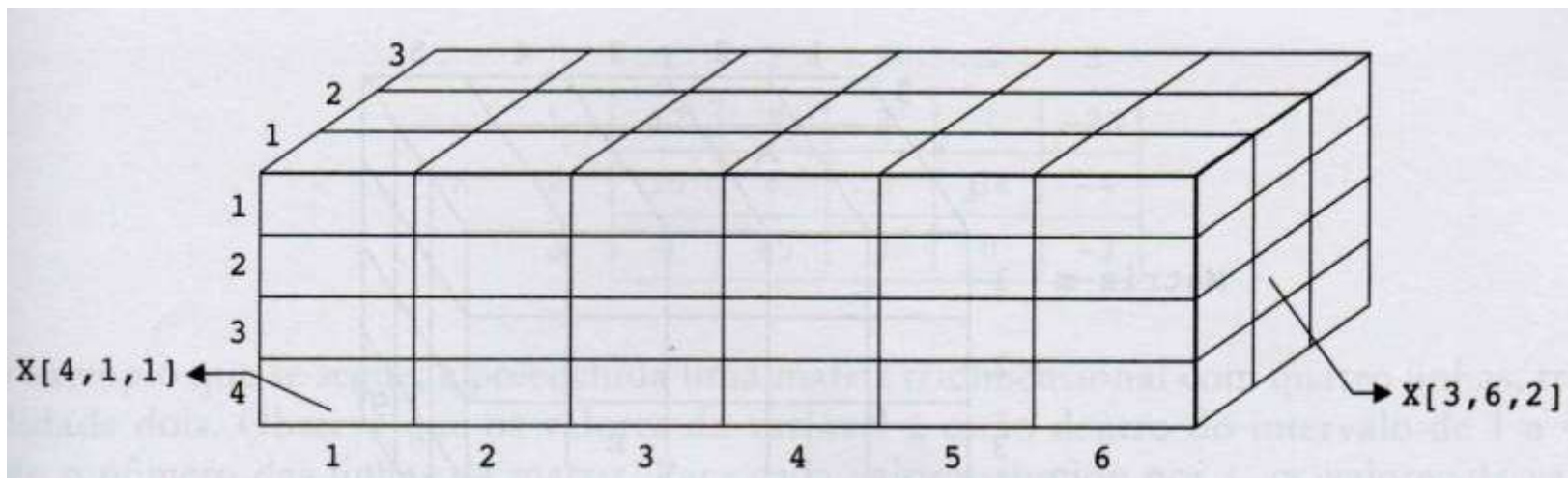
```
int arrayBi[5][3] = { {1, 2, 3},  
                     {4},  
                     {5, 6, 7} };
```

arrays de muitas dimensões são iniciados como os outros *arrays*

array bidimensionais são conhecidos como **matrizes**. Esta tem 5 linhas e 3 colunas

Array tridimensional

Array **X** tridimensional, onde o tamanho da 1ª dimensão é 4; o tamanho da 2ª dimensão é 6; e o tamanho da 3ª dimensão é 3.



Matrizes

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

M	1	2	3	4	5
1					
2					
3					

Diagram illustrating a 3x5 matrix **M**. The rows are indexed 1 to 3, and the columns are indexed 1 to 5. Arrows point to specific elements: $M[1,1]$ (top-left) and $M[3,4]$ (bottom-right).

Matriz **M** bidimensional, onde o tamanho da 1ª dimensão (linha) é 3 e o tamanho da 2ª dimensão (coluna) é 5.

Matrizes

- A linguagem C permite a declaração de matrizes unidimensionais (conhecidas como vetores), bidimensionais e multidimensionais
- O padrão ANSI prevê 12 dimensões, entretanto o limite de dimensões fica por conta da quantidade de recursos computacionais disponíveis
- As matrizes mais utilizadas possuem duas dimensões. Para cada dimensão deve ser utilizado um índice (linha e coluna)
- Os índices utilizados em C, para identificar as posições de uma matriz, começam sempre em zero e vão até o tamanho da dimensão menos um
- Os índices devem sempre ser representados pelo tipo inteiro

Matrizes

- Declaração

`tipo nomeMatriz [linhas] [colunas];`

Onde:

- `tipo`: é o tipo dos dados que serão armazenados na matriz;
 - `nomeMatriz`: é o nome dado à variável;
 - `[linhas]`: representa o tamanho da 1ª dimensão;
 - `[colunas]`: representa o tamanho da 2ª dimensão;
-
- O tamanho das dimensões de uma matriz deve ser feito por um valor inteiro fixo

Matrizes

1. float matrizX[2][6];

- Variável **matrizX** contendo 2 linhas (de 0 a 1) com 6 colunas cada (de 0 a 5), capaz de armazenar números reais

matrizX	0	1	2	3	4	5
0						
1						

2. int matrizY[4][3];

- Variável **matrizY** contendo 4 linhas (de 0 a 3) com 3 colunas cada (de 0 a 2), capaz de armazenar números inteiros

matrizY	0	1	2
0			
1			
2			
3			

Matrizes

- **Atribuir valores a uma matriz:** Significa armazenar dados em seus elementos, identificados de forma única por meio dos seus índices

matrizX[1][4] = 5.3; → Atribui o valor 5.3 à posição identificada pelos índices 1 e 4

matrizX	0	1	2	3	4	5
0						
1					5.3	

matrizY[3][2] = 4; → Atribui o valor 4 à posição identificada pelos índices 3 e 2

matrizY	0	1	2
0			
1			
2			
3			4

Matrizes

- **Preencher uma matriz:** Significa percorrer todos os seus elementos, atribuindo-lhes um valor. Esse valor pode ser recebido via teclado, pelo usuário, ou ser gerado automaticamente pelo programa

```
for (i=0; i<7; i++)  
    for (j=0; j<3; j++)  
        scanf("%d", &M[i][j]);
```

Todos os elementos de M são percorridos, atribuindo-lhes valores digitados pelo usuário.

Como M possui 7 linhas e 3 colunas, o exemplo apresenta duas estruturas for para garantir que a variável i assuma todos os valores para linha (0 a 6) e a variável j assuma todos os valores possíveis para coluna (0 a 2). Assim, para cada execução das estruturas de repetição, uma posição diferente da matriz é preenchida por um valor digitado pelo usuário.

Matrizes

- **Mostrar os elementos de uma matriz:** Significa percorrer todos os elementos de uma matriz acessando seu conteúdo

```
for (i=0; i<10; i++)  
    for (j=0; j<6; j++)  
        printf("%f", X[i][j]);
```

Todos os elementos da matriz X são percorridos, mostrando os seus valores através do *printf*.

Como X possui 10 linhas e 6 colunas, o exemplo apresenta duas estruturas for para garantir que a variável *i* assuma todos os valores para linha (0 a 9) e a variável *j* assuma todos os valores possíveis para coluna (0 a 5). Assim, para cada execução das estruturas de repetição, uma posição diferente da matriz é acessada e seu conteúdo mostrado por meio do comando *printf*.

Matrizes

- **Percorrer uma matriz:** Significa passar por todos os seus elementos (suas posições)
 - Uma das formas mais simples de percorrer uma matriz pode ser por meio do uso de uma estrutura de repetição para cada dimensão
 - A disposição de tais estruturas de repetição define a forma como a matriz será percorrida

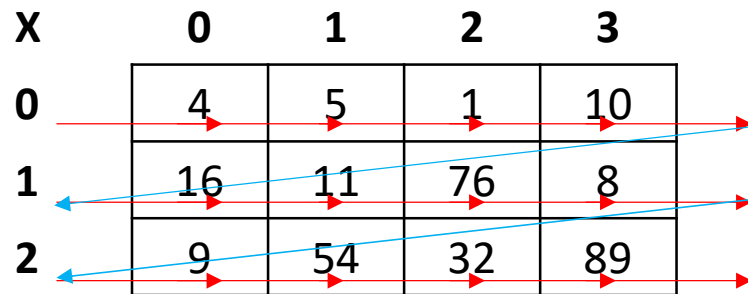
```
for (i=0; i<3; i++) {  
    printf ("Elementos da linha %d: ", i);  
    for (j=0; j<4; j++)  
        printf ("%d\\n", X[i][j]);  
}
```

X	0	1	2	3
0	4	5	1	10
1	16	11	76	8
2	9	54	32	89

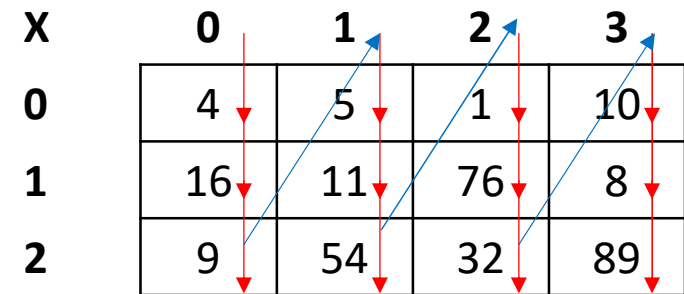
Matrizes

- Outra visão da forma utilizada para percorrer a matriz
 - A direção das setas indica a mudança no valor das variáveis i e j e o caminho utilizado para percorrer a matriz

X	0	1	2	3
0	4	5	1	10
1	16	11	76	8
2	9	54	32	89



X	0	1	2	3
0	4	5	1	10
1	16	11	76	8
2	9	54	32	89



Matrizes

- As matrizes podem ser inicializadas no momento da sua declaração

```
int M [3][3] = {{4, -1, 5},{ 10, 3, 6},{ 21, -5, 1}};
```

```
int M[][3] = {4, -1, 5, 10, 3, 6, 21, -5, 1};
```

```
int M [][] = {4, -1, 5, 10, 3, 6, 21, -5, 1 }; /* ERRADO!!*/
```

Esta função preenche uma matriz com os dados fornecidos pelo usuário.

```
void PreencheMatriz(int m[][2], int linhas,int colunas){
    int i,j;

    for (i = 0; i < linhas; i++) {
        for (j = 0; j < colunas; j++) {
            printf ("M[%d/[%d]", i, j);
            scanf ("%d", &m[i][ j]);
        }
    }
}
```

Esta função mostra os elementos de um array bidimensional no formato de uma matriz

```
void MostraMatriz (int m[][2], int linhas, int colunas) {  
    int i, j;  
    /* Escreve o índice de cada coluna */  
    for (i = 0; i < colunas; i++) {  
        printf("\t%d", i);  
    }  
    for (i = 0; i < linhas; i++) {  
        /* Escreve o índice de cada linha */  
        printf("\n%d", i);  
        /* Escreve o valor de cada elemento */  
        for (j = 0; j < colunas; j++) {  
            printf("\t%d", m[i][j]);  
        }  
    }  
}
```

```
int main () {  
    int matriz[3][2];  
  
    PreencheMatriz (matriz, 3, 2);  
    MostraMatriz (matriz, 3, 2);  
  
    return 0;  
}
```

	0	1
0	1	2
1	3	4
2	5	6

```
int SomaMatriz (int m[][2], int linhas, int colunas) {  
    int i, j, soma = 0;  
  
    for (i = 0; i < linhas; i++) {  
        for (j = 0; j < colunas; j++) {  
            soma = soma + m[i][j];  
        }  
    }  
    return soma;  
}
```

```
int main () {  
    int matriz[3][2] = {{1,2},{3,4},{5,6}};  
  
    MostraMatriz (matriz, 3, 2);  
    printf("Soma dos elementos = %d", SomaMatriz(matriz, 3, 2));  
    return 0;  
}
```

```
int ContaNegativos (int m[][2], int linhas, int colunas) {  
    int i, j, negativos = 0;  
  
    for (i = 0; i < linhas; i++) {  
        for (j = 0; j < colunas; j++) {  
            if (m[i][j] < 0) {  
                negativos = negativos + 1;  
            }  
        }  
    }  
    return negativos;  
}
```

```
int main () {  
    int matriz[3][2] = {{1,-2},{3,4},{-5,-6}};  
  
    MostraMatriz (matriz, 3, 2);  
    printf("Total de negativos = %d", ContaNegativos(matriz, 3, 2));  
    return 0;  
}
```

Matrizes - Exercícios práticos

1. Seja A uma matriz 4x5. Determine o maior elemento de A e a sua posição.
2. Seja A uma matriz 3x3. Fazer um programa que:
 - a) Calcule e mostre a soma dos elementos da diagonal principal.
 - b) Armazene os elementos da diagonal principal de A em um vetor D; Mostre o vetor D.
3. Idem ao exercício anterior para diagonal secundária.
4. Seja A uma matriz 3x3. Determine a matriz T transposta de A. (obs.: $T[l][c] = A[c][l]$). Mostre a matriz T.
5. Dada uma matriz A, determine a linha de A que possui a maior soma de seus elementos.

Matrizes - Exercícios práticos

6. Faça um programa que preencha uma matriz $M(3 \times 3)$, calcule e mostre a matriz R , resultante da multiplicação dos elementos de M pelo seu menor elemento.
7. Escreva um programa que preencha uma matriz $M(6 \times 4)$ com números inteiros, calcule e mostre quantos elementos dessa matriz são maiores que 30 e, em seguida, monte uma segunda matriz com os elementos diferentes de 30. No lugar do número 30, da segunda matriz, coloque o número zero.
8. Faça um programa que preencha uma matriz $M(8 \times 8)$ com números inteiros e some cada uma das linhas, armazenando o resultado das somas em um vetor. A seguir, o programa deverá multiplicar cada elemento da matriz pela soma da linha correspondente e mostrar a matriz resultante.
9. Escreva um programa que preencha uma matriz $M(10 \times 10)$ com números inteiros, execute as trocas especificadas a seguir e mostre a matriz resultante: a linha 2 com a linha 8; a coluna 4 com a coluna 10.

Matrizes - Exercícios práticos

10. Seja a declaração: `int N[LINHA][COLUNA]`; Escrever um programa capaz de:
- a) ler os elementos da matriz.
 - b) identificar o número de elementos iguais a zero em cada uma das linhas.
 - c) identificar o número de elementos iguais a zero em cada uma das colunas.
 - d) identificar o número de elementos pares em determinada linha (a linha será fornecida pelo usuário).
 - e) identificar o número de elementos pares em determinada coluna (a coluna será fornecida pelo usuário).
 - f) calcular a média aritmética dos elementos de cada uma das linhas, armazenando esses valores em um vetor.
 - g) identificar a linha que tem a maior média de seus elementos.
 - h) mostrar todos os resultados.