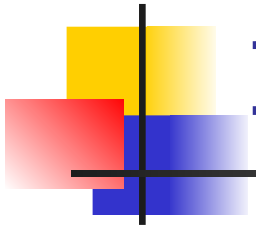
SCC0221 - Introdução à Ciência de Computação I

Prof.: Dr. Rudinei Goularte

(rudinei@icmc.usp.br)

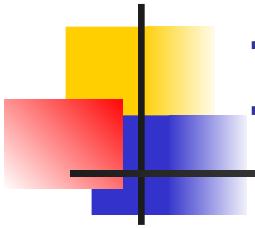
Matrizes

Instituto de Ciências Matemáticas e de Computação - ICMC
Sala 4-229



1. Introdução

- Uma matriz é uma coleção de **variáveis do mesmo tipo** referenciadas por um nome comum.
- Uma determinada variável da matriz é chamada de **elemento** da matriz.
- Os elementos de uma matriz podem ser acessados, individualmente, por meio de índices.



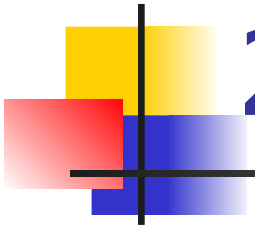
1. Introdução

- Em C, os elementos de uma matriz ocupam posições contíguas na memória.
- Em C, matrizes podem ter uma, duas ou várias dimensões.
- Em C, matrizes e ponteiros são assuntos relacionados.
  - Matrizes são arrays com mais de uma dimensão.

2. Matrizes Bidimensionais

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

- int mat [3][4];
- Conceito de linha-coluna.



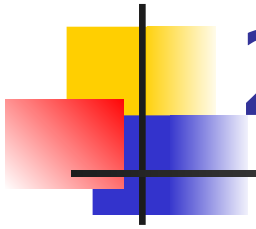
2. Matrizes Bidimensionais

- Acesso através dos índices:
  - mat [1][3] é igual a 8.
  - mat [0][2] é igual a 3.
  - mat [2] [1] é igual a 10.
- Bytes necessários:
  - Bytes = tamanho do 1º índice * tamanho do 2º índice * sizeof (tipo)

2. Matrizes Bidimensionais

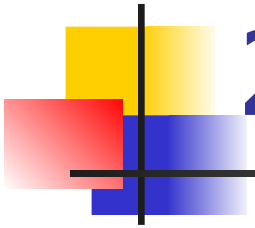
	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

- *Storage Mapping Function*
  - Como mapear/calcular uma posição da matriz na memória?
    - Arrays são arranjos lineares de memória



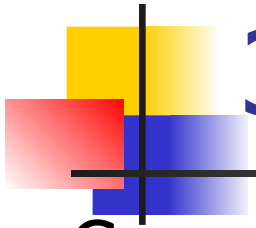
2. Matrizes Bidimensionais

- Exercício:
  - Faça um programa em C que calcule a soma de todos os elementos de uma matriz de inteiros de 3 linhas x 4 colunas.



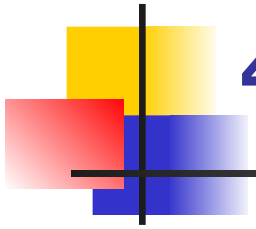
2. Matrizes Bidimensionais

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

for(i=0; i<3; i++)
   for (j=0; j<4; j++)
         soma += mat[i][j];
printf("Soma: %d", soma);
```



3. Matrizes Multidimensionais

- C permite matrizes com mais de duas dimensões.
  - tipo nome [tam 1] [tam 2] [tam 3] ... [tam n]
- O limite, se existente, é determinado pelo compilador.
- Matrizes de 3 ou mais dimensões são pouco frequentes.
  - Quantidade de memória.
  - Ex.: double mat[10][3][5][10];
    tamanho = 10 * 3 * 5 * 10 * sizeof(double) = 12000 bytes (assumindo double de 8 bytes).
- Para esse tipo de matriz é comum usar alocação dinâmica.



4. Inicialização de Matrizes

- C permite inicializar matrizes no momento da declaração:
  - int a[5] = {1, 2, 3, 4, 5};
  - char str[4] = “USP”;

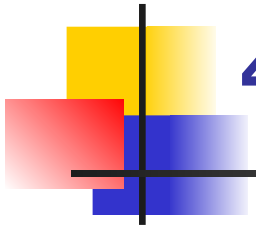
equivale a

  - char str[4] = {‘U’,’S’,’P’,’\0’};

4. Inicialização de Matrizes

- Inicialização de matrizes multidimensionais

```
int mat[5][2] = {1,3,5,8,3,1,2,2,7,4};

int mat[5][2] = {{1,3},{5,8},{3,1},{2,2},{7,4}};

int mat[5][2] = {
                    {1,3},
                    {5,8},
                    {3,1},
                    {2,2},
                    {7,4}
                };
```



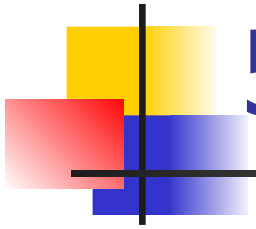
4. Inicialização de Matrizes

- Inicialização de matrizes não dimensionadas: somente a dimensão mais à esquerda não precisa ser especificada.

```
int mat[][2] = {1,3,5,8,3,1,2,2,7,4};
```

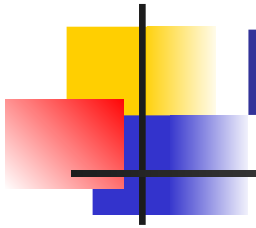
- Com 3 dimensões:
  - int mat[][2][3] = {1,3,5,8,3,1,2,2,7,4,5,6};
  - int mat[][2][3] = {1,3,5,8,3,1,2,2,7,4,5,6};
  - int mat[][2][3] = {

```
                {{1,3,5},  {8,3,1}},
                {{2,2,7},  {4,5,6}}
        };
```



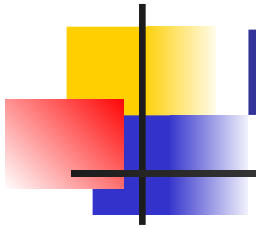
5. typedef

```
#include <stdio.h>

#define N 3
typedef int vector[N];

int main (void){
      vector vet;  /*int vet[N];*/
      vector matrix[N]; /* int matrix[N][N];*/
      vector mat[10][15];
      ...

      return(0);
}
```



Exercício

- Faça um programa em C para realizar a soma dos elementos de uma matriz MxN do tipo double que sejam maiores que um valor dado como enrtrada. Os elementos da matriz são entradas do usuário.
- Utilize funções para modulizar seu código: entrada de dados, cálculo da soma, impressão do(s) resultado(s).



Exercício

- Faça um programa em C para multiplicar duas matrizes (bidimensionais) de inteiros, de dimensões quaisquer. O cálculo deve contemplar matrizez não-quadradas.
- Obs.: Modularize seu código.


Fim.