

# MC102 – Algoritmos e Programação de Computadores

Instituto de Computação

UNICAMP

Segundo Semestre de 2013

# Roteiro

- 1 Matrizes
- 2 Exemplos com matrizes
- 3 Inicialização de matrizes e vetores
- 4 Representação de matrizes por linearização de índices
- 5 Exercícios

# Matrizes

Suponha que queremos ler as notas de 4 provas para cada aluno e então calcular a média do aluno e a média da turma. O tamanho máximo da turma é de 50 alunos.

## Solução:

Criar 4 vetores de tamanho 50 cada. Cada vetor representa as notas dos alunos de uma prova.

```
float prova1[50], prova2[50], prova3[50], prova4[50];
```

# Matrizes

- Agora suponha que o número de provas possa ser igual a 100. Torna-se inconveniente criar 100 vetores diferentes, um para cada prova.
- Para resolver esse problema, podemos utilizar matrizes. Uma matriz é um vetor (ou seja, um conjunto de variáveis de mesmo tipo) que possui duas ou mais dimensões.

# Declarando uma matriz

```
<tipo> nome_da_matriz [<linhas>][<colunas>;
```

- Uma matriz possui *linhas*  $\times$  *colunas* variáveis do tipo `<tipo>`.
- As linhas são numeradas de 0 a *linhas*  $- 1$ .
- As colunas são numeradas de 0 a *colunas*  $- 1$ .

## Exemplo de declaração de matriz

```
int matriz[5][4];
```

	0	1	2	3
0				
1				
2				
3				
4				

## Acesso aos elementos de uma matriz

- O acesso a um elemento da matriz pode ser feito da seguinte forma:

```
nome_da_matriz [<linha>][<coluna>]
```

Ex: `matriz[1][3]`: refere-se ao elemento na 2ª linha e na 4ª coluna da matriz.

- Lembre-se que, assim como vetores, a primeira posição em uma determinada dimensão começa no índice 0.
- O compilador não verifica se você utilizou valores válidos para a linha e para a coluna.

# Declarando uma matriz de múltiplas dimensões

```
<tipo> nome_da_matriz [< dim1 >] [< dim2 >] ... [< dimN >];
```

- Essa matriz possui  $dim_1 \times dim_2 \times \dots \times dim_N$  variáveis do tipo `<tipo>`.
- Cada dimensão é numerada de 0 a  $dim_i - 1$ .

## Declarando uma matriz de múltiplas dimensões

- Você pode criar, por exemplo, uma matriz para armazenar a quantidade de chuva em um dado dia, mês e ano:

```
double chuva[31][12][3000];
```

```
chuva[23][3][1979] = 6.0;
```

- O código acima indica que no dia 24/04/1980 choveu 6mm.

## Exemplos com matrizes

Lendo uma matriz  $5 \times 4$  da entrada padrão:

```
/* Leitura */
for (i = 0; i < 5; i++)
    for (j = 0; j < 4; j++) {
        printf("Matriz[%d] [%d]: ", i, j);
        scanf("%d", &matriz[i][j]);
    }
```

## Exemplos com matrizes

Escrevendo uma matriz  $5 \times 4$  na saída padrão:

```
/* Escrita */  
for (i = 0; i < 5; i++) {  
    for (j = 0; j < 4; j++)  
        printf ("%d ", matriz[i][j]);  
    printf ("\n");  
}
```

# Exemplos com matrizes

- Ler duas matrizes  $3 \times 3$  e calcular a soma das duas.

# Exemplos com matrizes

```
#include <stdio.h>

int main() {
    double mat1[3][3], mat2[3][3], mat3[3][3];
    int i, j;

    printf("*** Dados da Matriz 1 ***\n");
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++) {
            printf("Entre com dado da linha %d e coluna %d: ", i, j);
            scanf("%lf", &mat1[i][j]);
        }

    printf("*** Dados da Matriz 2 ***\n");
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++) {
            printf("Entre com dado da linha %d e coluna %d: ", i, j);
            scanf("%lf", &mat2[i][j]);
        }
    ...
}
```

# Exemplos com matrizes

```
...  
  
for (i = 0; i < 3; i++)  
    for ( j = 0; j < 3; j++) {  
        mat3[i][j] = mat1[i][j] + mat2[i][j];  
    }  
  
printf("*** Dados da Matriz 3 ***\n");  
for (i = 0; i < 3; i++) {  
    for (j = 0; j < 3; j++)  
        printf("%f, ", mat3[i][j]);  
    printf("\n");  
}  
  
return 0;  
}
```

# Inicialização de matrizes

- Em algumas situações, ao criarmos uma matriz, pode ser útil atribuir valores já na sua criação.
- No caso de vetores, a inicialização é simples: basta atribuir uma lista de valores constantes de mesmo tipo separados por vírgulas e entre chaves.

## Exemplo

```
int vet[5] = {10, 20, 30, 40, 50};
```

- No caso de strings, pode-se atribuir diretamente uma constante.

## Exemplo

```
char st1[100] = "sim, isto eh possivel";
```

# Inicialização de matrizes

- No caso de matrizes, use-se chaves para delimitar as linhas:

## Exemplo

```
int vet[2][4] = { {10, 20, 30, 40}, {50, 60, 70, 80} };
```

- No caso tridimensional, cada primeiro índice é uma matriz inteira:

## Exemplo

```
int v3[2][3][4] = {  
    { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} },  
    { {0, 0, 0, 0}, {5, 6, 7, 8}, {0, 0, 0, 0} },  
};
```

# Inicialização de matrizes

```
#include <stdio.h>

int main() {
    int i, j, k;
    int v1[5] = {1, 2, 3, 4, 5};
    int v2[2][3] = { {1, 2, 3}, {4, 5, 6}};
    int v3[2][3][4] = {
        { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} },
        { {0, 0, 0, 0}, {5, 6, 7, 8}, {0, 0, 0, 0} }
    };
    char st1[100] = "exemplo de texto";
    ...
}
```

# Inicialização de matrizes

```
#include <stdio.h>

int main() {
    ...
    printf("Vetor v1:\n");
    for (i = 0; i < 5; i++)
        printf("%d, ", v1[i]);

    printf("Matriz bidimensional v2:\n");
    for (i = 0; i < 2; i++) {
        for (j = 0; j < 3; j++)
            printf("%d, ", v2[i][j]);
        printf("\n");
    }
    ...
}
```

# Inicialização de matrizes

```
#include <stdio.h>

int main() {
    ...
    printf("Matriz tridimensional v3:\n");
    for (i = 0; i < 2; i++) {
        for (j = 0; j < 3; j++) {
            for (k = 0; k < 4; k++) {
                printf("%d, ", v3[i][j][k]);
            }
            printf("\n");
        }
        printf("\n");
    }

    printf("String st1:\n");
    printf("%s", st1);
    return 0;
}
```

# Linearização de índices

- Podemos usar sempre vetores simples para representar matrizes (na prática o compilador faz isto por você).
- Ao declarar uma matriz como `int mat[3][4]`, sabemos que serão alocados 12 posições de memória associadas com a variável `mat`.
- Poderíamos simplesmente criar `int mat[12]`. Entretanto, perdemos a simplicidade de uso dos índices em forma de matriz.
  - ▶ Você não mais poderá escrever `mat[1][3]`, por exemplo.

# Linearização de índices

- A *linearização de índices* é justamente a representação de matrizes usando-se um vetor simples.
- Entretanto, devemos ter um padrão para acessar as posições deste vetor como se sua organização fosse na forma de matriz.

# Linearização de índices

- Considere o exemplo:  
`int mat[12]; /* ao inves de int mat[3][4] */`
- Fazemos a divisão por linhas como segue:
  - ▶ Primeira linha: `mat [0]` até `mat [3]`
  - ▶ Segunda linha: `mat [4]` até `mat [7]`
  - ▶ Terceira linha: `mat [8]` até `mat [11]`
- Para acessar uma posição `[i] [j]`, usamos:
  - ▶ `mat [i*4 + j];`  
tal que  $0 \leq i \leq 2$  e  $0 \leq j \leq 3$ .

# Linearização de índices

- De forma geral, seja matriz  $\text{mat}[n*m]$ , representando  $\text{mat}[n][m]$ .
- Para ter acesso à posição correspondente a  $[i][j]$  usamos:
  - ▶  $\text{mat}[i*m + j]$ ;  
tal que  $0 \leq i \leq n - 1$  e  $0 \leq j \leq m - 1$ .
- Note que  $i$  salta blocos de tamanho  $m$  e  $j$  indexa a posição dentro de um bloco.

# Linearização de índices

- Podemos estender para mais dimensões. Seja matriz  $\text{mat}[n*m*q]$ , representando  $\text{mat}[n][m][q]$ .
  - ▶ As posições de 0 até  $(m * q) - 1$  são da primeira matriz.
  - ▶ As posições de  $(m * q)$  até  $(2 * m * q) - 1$  são da segunda matriz.
  - ▶ Assim por diante...
- De forma geral, seja matriz  $\text{mat}[n*m*q]$ , representando  $\text{mat}[n][m][q]$ .
- Para ter acesso à posição correspondente a  $[i][j][k]$ , usamos:
  - ▶  $\text{mat}[i*m*q + j*q + k]$ ;

# Linearização de índices

```
#include <stdio.h>

int main() {
    int mat[40]; /* representando mat[5][8] */
    int i,j;

    for (i = 0; i < 5; i++)
        for (j = 0; j < 8; j++)
            mat[i*8 + j] = i*j;

    for (i = 0; i < 5; i++) {
        for (j = 0; j < 8; j++)
            printf("%d, ",mat[i*8 + j]);
        printf("\n");
    }
    return 0;
}
```

# Exercícios

Escreva um programa que leia todas as posições de uma matriz  $10 \times 10$ . Em seguida, mostra o índice da linha e o índice da coluna e o valor das posições não nulas. No final, exibe o número de posições não nulas.

## Exercícios

Escreva um programa que lê todos os elementos de uma matriz  $n \times m$  e mostra a matriz e a sua transposta na tela.

Matriz

$$\begin{bmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \\ 51 & 52 & 53 & 54 \end{bmatrix}$$

Transposta

$$\begin{bmatrix} 11 & 21 & 31 & 41 & 51 \\ 12 & 22 & 32 & 42 & 52 \\ 13 & 23 & 33 & 43 & 53 \\ 14 & 24 & 34 & 44 & 54 \end{bmatrix}$$

## Exercícios

Escreva um programa que lê 2 matrizes  $5 \times 5$ , mostre-as na tela e mostre a soma entre as duas matrizes em seguida.

$$\begin{array}{c} \text{A} \\ \left[ \begin{array}{ccccc} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{array} \right] \end{array} + \begin{array}{c} \text{B} \\ \left[ \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{array} \right] \end{array} = \begin{array}{c} \text{C} \\ \left[ \begin{array}{ccccc} 0 & 1 & 0 & 4 & 3 \\ 1 & 2 & 1 & 3 & 4 \\ 0 & 1 & 0 & 2 & 3 \\ 2 & 3 & 2 & 4 & 5 \\ 3 & 4 & 3 & 5 & 6 \end{array} \right] \end{array}$$

## Exercícios

Escreva um programa que lê 2 matrizes  $5 \times 5$ , mostre-as na tela e então calcule o produto entre as duas matrizes, mostrando-o em seguida.

$$\begin{array}{c} \text{A} \\ \left[ \begin{array}{ccccc} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{array} \right] \end{array} \times \begin{array}{c} \text{B} \\ \left[ \begin{array}{ccccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{array} \right] \end{array} = \begin{array}{c} \text{C} \\ \left[ \begin{array}{ccccc} 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \end{array} \right] \end{array}$$

# Exercícios

Escreva um programa lê uma matriz e depois verifica se esta é uma matriz triangular inferior.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 3 \end{bmatrix}$$