

MC-102 — Aula 10

Matrizes

Instituto de Computação – Unicamp

Segundo Semestre de 2009



Matrizes
Exemplos
Exercícios

Roteiro

- 1 Matrizes
- 2 Exemplos
- 3 Exercícios



MC-102 — Aula 10

Matrizes
Exemplos
Exercícios

Vetores — Relembrando

Coleção de variáveis do mesmo tipo referenciada por um nome comum.

- Sequência de valores
- Todos do mesmo tipo
- Nome único para a variável
- Acesso por meio de índice
- Numeração de 0 até tamanho-1
- Tamanho fixo
- Posições contíguas na memória
- Índices fora dos limites podem causar comportamento anômalo do código

```
int v[10]
```

| |
|---------|
| v0(int) |
| v1(int) |
| v2(int) |
| v3(int) |
| v4(int) |
| v5(int) |
| v6(int) |
| v7(int) |
| v8(int) |
| v9(int) |



MC-102 — Aula 10

Matrizes
Exemplos
Exercícios

Usando um vetor

Elementos do vetor

v[0], v[1], v[2], v[3], ..., v[9]

Atribuição

v[índice] = valor;

Exemplo

```
int vetor[10];  
a = vetor[1];  
vetor[3] = 1;  
vetor[7] = 12;  
vetor[9] = vetor[7]+vetor[3];
```



MC-102 — Aula 10

Vetores

Na última aula, criamos um programa que lia as notas de uma prova para um conjunto de alunos e então calculava a média da turma. Para resolver este problema, utilizamos vetores.

```
float nota[100];
int n, i;

printf("Número de alunos: ");
scanf("%d", &n);

for (i = 0; i < n; i++) {
    printf("Nota do aluno %d: ", i+1);
    scanf("%f", &nota[i]);
}
```

Reveja o código: notas.c

Vetores

Agora queremos ler as notas de 3 provas para cada aluno e então calcular a média do aluno e a média da classe. O tamanho máximo da turma é de 50 alunos.

solução

Criar 3 vetores cada um com 50 posições. E então ler as respectivas informações.

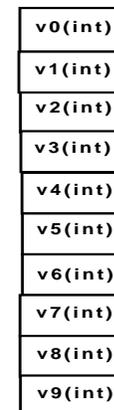
```
float nota0[50], nota1[50], nota2[50];
```

Matrizes

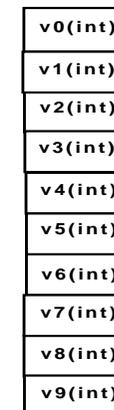
- Agora suponha que estamos trabalhando com no máximo 100 provas e 100 alunos. Seria muito cansativo criar 100 vetores e atribuir 100 nomes diferentes. **(Parece que esse problema não tem fim !!!)**.
- Para resolver esse problema podemos **utilizar matrizes**. Uma **matriz é um vetor** (ou seja, um conjunto de variáveis de mesmo tipo) **que possui** duas ou mais **dimensões**, resolvendo para sempre essa questão.

Matrizes? vetores? dimensões?

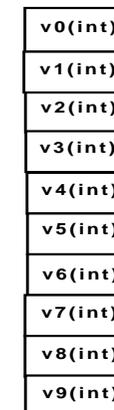
int v[10]



int v[10]



int v[10]



Matrizes = vetor + dimensões

```
int M[10][3]
```

| | | |
|-------|-------|-------|
| M 0 0 | M 0 1 | M 0 2 |
| M 1 0 | M 1 1 | M 1 2 |
| M 2 0 | M 2 1 | M 2 2 |
| M 3 0 | M 3 1 | M 3 2 |
| M 4 0 | M 4 1 | M 4 2 |
| M 5 0 | M 5 1 | M 5 2 |
| M 6 0 | M 6 1 | M 6 3 |
| M 7 0 | M 7 1 | M 7 2 |
| M 8 0 | M 8 1 | M 8 2 |
| M 9 0 | M 9 1 | M 9 2 |

Exemplo de declaração de matriz

```
<tipo> nome_da_matriz [< num_linhas >] [< num_colunas >];
```

```
int a [4] [4];
```

| | | | | |
|---|---|---|---|---|
| | 0 | 1 | 2 | 3 |
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

| | | | | |
|---|-----|-----|-----|-----|
| | 0 | 1 | 2 | 3 |
| 0 | 0,0 | 0,1 | 0,2 | 0,3 |
| 1 | 1,0 | 1,1 | 1,2 | 1,3 |
| 2 | 2,0 | 2,1 | 2,2 | 2,3 |
| 3 | 3,0 | 3,1 | 3,2 | 3,3 |

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 $dim_i - 1$

Acessando uma matriz

- Em qualquer lugar onde você escreveria uma variável no seu programa, você pode usar um elemento de sua matriz
- Acesso aos valores através de um **único nome de variável**

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

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

Ex: `matriz [1] [2]` — Refere-se a variável na 2ª linha e na 3ª coluna da matriz.

| | | | | |
|---|-----|-----|-----|-----|
| | 0 | 1 | 2 | 3 |
| 0 | 0,0 | 0,1 | 0,2 | 0,3 |
| 1 | 1,0 | 1,1 | 1,2 | 1,3 |
| 2 | 2,0 | 2,1 | 2,2 | 2,3 |
| 3 | 3,0 | 3,1 | 3,2 | 3,3 |

Acessando uma matriz

- Dimensões são fixas (assim como acontece com os vetores!)
- O compilador não verifica se você utilizou valores válidos para a linha e para a coluna.

| | 0 | 1 | 2 | 3 |
|---|-----------|-----------|-----------|-----------|
| 0 | $a_{0,0}$ | $a_{0,1}$ | $a_{0,2}$ | $a_{0,3}$ |
| 1 | $a_{1,0}$ | $a_{1,1}$ | $a_{1,2}$ | $a_{1,3}$ |
| 2 | $a_{2,0}$ | $a_{2,1}$ | $a_{2,2}$ | $a_{2,3}$ |
| 3 | $a_{3,0}$ | $a_{3,1}$ | $a_{3,2}$ | $a_{3,3}$ |

Lendo uma matriz do teclado

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

```
scanf("%d",&matriz[linha][coluna]);
```

Escrevendo uma matriz na tela

```
/*Escrita*/
for (linha = 0; linha < 4; linha++) {
  for (coluna = 0; coluna < 4; coluna++) {
    printf ("%3d ", matriz[linha][coluna]);
    printf ("\n");
  }
```

Veja o exemplo para os três últimos slides em leitura.c

Exercícios

- Escreva um programa que inicialize uma matriz 10×10 com 0s em todas as posições. Em seguida imprima a matriz na tela.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \dots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Exercícios

- Escreva um programa que leia todas as posições de uma matriz 10×10 . 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.

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \dots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{array}{l} 0 \ 0 \ -1 \\ 1 \ 2 \ 2 \\ 9 \ 9 \ 1 \\ 3 \text{ elementos não} \\ \text{nulos} \end{array}$$

Exercícios

- Escreva um programa que lê todos os elementos de uma matriz 4×4 e mostra a matriz e a sua transposta na tela.

$$\begin{array}{cc} \text{Matriz} & \text{Transposta} \\ \begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 2 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix} \end{array}$$

Exercícios

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

$$\begin{bmatrix} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 2 & 3 \\ 1 & 2 & 1 & 3 & 4 \\ 0 & 1 & 0 & 2 & 3 \\ 2 & 3 & 2 & 4 & 5 \\ 3 & 4 & 3 & 5 & 6 \end{bmatrix}$$

Exercícios

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

$$\begin{bmatrix} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{bmatrix} * \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \end{bmatrix}$$

Exercícios

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

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