MC-102 — Aula 19
Matrizes

Instituto de Computação – Unicamp

Segundo Semestre de 2011

Roteiro

Vetores

- O que é um vetor?
- Coleção de variáveis do mesmo tipo referenciada por um nome comum.

Vetores

Atribuição

vetor[índice]=valor;

ou

valor=vetor[índice];

Exemplos

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

Vetores

Agora queremos ler as notas de 4 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 4 vetores cada um com 50 posições. E então ler as respectivas informações.

```
float nota0[50],nota1[50],nota2[50],nota3[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.

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 [4][4];
```

	0	1	2	3
0				
1				
2				
3				

	0	1	2	3
0	1.4	12.3	23.2	34
1	3.6	5.4	3.4	34
2	0.9	9.0	2.1	34
3	34.9	93.0	21.0	23

Declarando uma matriz de múltiplas dimensões

<tipo> nome_da_matriz [< dim_1 >] [< dim_2 >] ... [< dim_N >]

- Essa matriz possui $dim_1 \times dim_2 \times \cdots \times \dim_N$ 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, da seguinte forma:

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

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

	0	1	2	3
0				
1			aqui	
2				
3				

Vetores

- 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	a0,0	a0,1	a0,2	a0,3
1	a1,0	a1,1	a1,2	a1,3
2	a2,0	a2,1	a2,2	a2,3
3	a3,0	a3,1	a3,2	a3,3

Lendo uma matriz do teclado

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

Escrevendo uma matriz na tela

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

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

Linearização de Matrizes

- Matrizes também podem ser representadas na forma unidimensional (isto é muito comum em processamento de imagens, por exemplo).
- Por exemplo, programas de processamento de imagem usam vetores por ser mais fácil otimizar.
- Para isto é preciso armazenar a matriz em um vetor, ou seja, e linearizar a matriz.

Linearização de Matrizes

- Podemos armazenar os elementos de uma matriz da esquerda para direita e de cima para baixo iniciando em [0,0] até [NLIN-1,NCOL-1], em um vetor de NLIN x NCOL variáveis.
- Para saber o índice i do elemento do vetor correspondente à matriz m[l,c] (onde l é linhas e c é colunas), fazemos:
  i = l * NCOL + c
- O processo inverso é dado por: c = i%NCOL e l = i/NCOL.

Veja o exemplo em matrizLinear.c

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.

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \dots & & & & & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

0 0 -1
1 2 2
9 9 1
3 elementos não nulos

Exercícios

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

Matriz

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

Transposta

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

Exercícios

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

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

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.

$$\underset{A}{\begin{bmatrix} 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 2 & 3 \end{bmatrix}} * \underset{B}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \\ 14 & 14 & 14 & 14 & 14 \end{bmatrix}}$$

Exercícios

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

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