MC-102 — Aula 13
Matrizes e Vetores Multidimensionais

Alexandre M. Ferreira

IC – Unicamp

19/04/2017

Roteiro

1. Matrizes e Vetores Multidimensionais
   - Declaração de Matrizes
   - Acessando dados de uma Matriz
   - Declarando Vetores Multidimensionais
2. Exemplo com Matrizes
3. Exercícios
4. Informações Extras: Inicialização de Matrizes

Matrizes e Vetores Multidimensionais

- Matrizes e Vetores Multidimensionais são generalizações de vetores simples vistos anteriormente.
- Suponha por exemplo que devemos armazenar as notas de cada aluno em cada laboratório de MC102.
- Podemos alocar 15 vetores (um para cada lab.) de tamanho 50 (tamanho da turma), onde cada vetor representa as notas de um laboratório específico.
- Matrizes e Vetores Multidimensionais permitem fazer a mesma coisa mas com todas as informações sendo acessadas por um nome em comum (ao invés de 15 nomes distintos).

Declaração de Matrizes

- A criação de uma matriz é feita com a seguinte sintaxe:

tipo `nome_da_matriz`[**linhas**][**colunas**];

onde **tipo** é o tipo de dados que a matriz armazenará, **linhas** (respectivamente **colunas**) é um inteiro que especifica o número de linhas (respectivamente colunas) que a matriz terá.

- A matriz criada terá (`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				

Acessando dados de uma Matriz

- Em qualquer lugar onde você usaria uma variável no seu programa, você pode usar um elemento específico de uma matriz da seguinte forma:

`nome_da_matriz` [**ind_linha**][**ind_coluna**]

onde **ind_linha** (respectivamente **ind_coluna**) é um índice inteiro especificando a linha (respectivamente coluna) a ser acessada.

- No exemplo abaixo é atribuído para **aux** o valor armazenado na variável da 1^a linha e 11^a coluna da matriz:

```
int matriz[100][200];
int aux;
...
aux = matriz [0][10];
```

Acessando dados de uma 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!
- Assim como vetores unidimensionais, comportamentos anômalos do programa podem ocorrer em caso de acesso à posições inválidas de uma matriz.

Declarando Vetores Multidimensionais

- Para se declarar um vetor com 3 ou mais dimensões usamos a seguinte sintaxe:

tipo `nome_vetor`$[d_1][d_2]\ldots[d_n]$;

onde d_i, para $i = 1, \ldots, n$, é um inteiro que especifica o tamanho do vetor na dimensão correspondente.

- O vetor criado possuirá $d_1 \times d_2 \times \cdots \times d_n$ variáveis do tipo **tipo**.
- Cada dimensão i é numerada de 0 a $d_i - 1$.

Declarando Vetores Multidimensionais

- Você pode criar, por exemplo, um vetor multidimensional para armazenar a quantidade de chuva em um dado dia, mês e ano, para cada um dos últimos 3000 anos:

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

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

Exemplo

Criar aplicações com operações básicas sobre matrizes quadradas:

- Soma de 2 matrizes com dimensões $n \times n$.
- Subtração de 2 matrizes com dimensões $n \times n$.
- Cálculo da transposta de uma matriz de dimensão $n \times n$.
- Multiplicação de 2 matrizes com dimensões $n \times n$.

Exemplo: Lendo e Imprimindo uma Matriz

- Primeiramente vamos implementar o código para se fazer a leitura e a impressão de uma matriz:

```
#include <stdio.h>
#define MAX 10

int main(){
  double mat1[MAX][MAX];
  int i, j, n;

  printf("Dimensão das matrizes (max. 10): ");
  scanf("%d", &n);

  printf("Lendo dados da matriz 1, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat1[i][j]);
    }
  }
  ...
}
```

- **MAX** é uma constante inteira definida previamente com valor 10 no nosso exemplo.
- Note porém que o tamanho efetivo da matriz é lido na variável **n**.

Exemplo: Lendo e Imprimindo uma Matriz

- Agora o código da impressão de uma matriz:

```
int main(){
  double mat1[MAX][MAX];
  int i, j, n;

  printf("Dimensão das matrizes (max. 10): ");
  scanf("%d", &n);
  printf("Lendo dados da matriz 1, linha por linha\n");
  ...

  printf("Imprimindo dados da matriz 1, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      printf("%.2lf \t", mat1[i][j]);
    }
    printf("\n"); //Após a impressão de uma linha da matriz pula linha
  }

}
```

- Para imprimir linha por linha, fixado uma linha i, imprimimos todas colunas j desta linha e ao final do laço em j, pulamos uma linha, para impressão de uma próxima linha.

Exemplo: Lendo e Imprimindo uma Matriz

- Código completo para ler e imprimir uma matriz:

```
#include <stdio.h>
#define MAX 10

int main(){
  double mat1[MAX][MAX];
  int i, j, n;

  printf("Dimensão das matrizes (max. 10): ");
  scanf("%d", &n);
  printf("Lendo dados da matriz, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat1[i][j]);
    }
  }

  printf("Imprimindo dados da matriz, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      printf("%.2lf \t", mat1[i][j]);
    }
    printf("\n");
  }
}
```

Exemplo: Soma de Matrizes

- Vamos implementar a funcionalidade de soma de matrizes quadradas.
- Primeiramente lemos as duas matrizes:

```
int main(){
  double mat1[MAX][MAX], mat2[MAX][MAX], mat3[MAX][MAX];
  int i, j, n;

  printf("Dimensão das matrizes: ");
  scanf("%d", &n);

  printf("Lendo dados da matriz 1, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat1[i][j]);
    }
  }

  printf("Lendo dados da matriz 2, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat2[i][j]);
    }
  }
  ...
}
```

Exemplo: Soma de Matrizes

- Agora para cada posição (i, j) fazemos

$$\text{mat3}[i][j] = \text{mat1}[i][j] + \text{mat2}[i][j]$$

tal que o resultado da soma das matrizes estará em **mat3**.

```
int main(){
  double mat1[MAX][MAX], mat2[MAX][MAX], mat3[MAX][MAX];
  int i, j, n;
  ...
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      mat3[i][j] = mat1[i][j] + mat2[i][j];
    }
  }

  printf("Imprimindo dados da matriz 3, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      printf("%.2lf \t", mat3[i][j]);
    }
    printf("\n");
  }
}
```

Exemplo: Multiplicação de Matrizes

- Vamos implementar a funcionalidade de multiplicação de matrizes quadradas.
- Vamos multiplicar duas matrizes M_1 e M_2 (de dimensão $n \times n$).
- O resultado será uma terceira matriz M_3.
- Lembre-se que uma posição (i, j) de M_3 terá o produto interno do vetor linha i de M_1 com o vetor coluna j de M_2:

$$M_3[i,j] = \sum_{k=0}^{n-1} M_1[i,k] \cdot M_2[k,j]$$

Exemplo: Multiplicação de Matrizes

- O código da multiplicação está abaixo: para cada posição (i, j) de **mat3** devemos computar

$$\text{mat3}[i, j] = \sum_{k=0}^{MAX-1} \text{mat1}[i, k] \cdot \text{mat2}[k, j]$$

```
...
for(i=0; i<n; i++){
  for(j=0; j<n; j++){
    mat3[i][j] = 0;
    for(k=0; k<n; k++){
       mat3[i][j] = mat3[i][j] + (mat1[i][k] * mat2[k][j]);
    }
  }
}
...
```

Exemplo: Multiplicação de Matrizes

```
int main(){
  double mat1[MAX][MAX], mat2[MAX][MAX], mat3[MAX][MAX];
  int i, j, k, n;
  printf("Dimensão das matrizes: ");
  scanf("%d", &n);
  printf("Lendo dados da matriz 1, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat1[i][j]);
    }
  }
  printf("Lendo dados da matriz 2, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat2[i][j]);
    }
  }

  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      mat3[i][j] = 0;
      for(k=0; k<n; k++){
         mat3[i][j] = mat3[i][j] + (mat1[i][k] * mat2[k][j]);
      }
    }
  }
  printf("Imprimindo dados da matriz 3, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      printf("%.2lf \t", mat3[i][j]);
    }
    printf("\n");
  }
}
```

Exercícios

- Faça um programa para realizar operações com matrizes que tenha as seguintes funcionalidades:
  - Um menu para escolher a operação a ser realizada:
    1. Leitura de uma matriz_1.
    2. Leitura de uma matriz_2.
    3. Impressão da matriz_1 e matriz_2.
    4. Cálculo da soma de matriz_1 com matriz_2, e impressão do resultado.
    5. Cálculo da multiplicação de matriz_1 com matriz_2, e impressão do resultado.
    6. Cálculo da subtração de matriz_1 com matriz_2, e impressão do resultado.
    7. Impressão da transposta de matriz_1 e matriz_2.

Exercícios

Escreva um programa que leia todas as posições de uma matriz 10×10.
O programa deve então exibir o número de posições não nulas na matriz.

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 leia uma matriz do teclado e então imprime os elementos com menor e maior frequência de ocorrência na matriz.

Informações Extras: Inicialização de Matrizes

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

Exemplo

```
int vet[2][5] = { {10, 20, 30, 40, 50} , {60, 70, 80, 90, 100 } } ;
```

- No caso tridimensional, cada índice da primeira dimensão se refere a 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} },
};
```

Informações Extras: Inicialização de Matrizes

```
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} }
  };

  .
  .
  .
  .
}
```