MC-102 — Aula 18
Matrizes e Vetores Multidimensionais

Eduardo C. Xavier

Instituto de Computação – Unicamp

9 de Maio de 2017

Roteiro

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 uma matriz 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;
```

Vetores multi-dimensionais e funções

- Ao passar um **vetor simples** como parâmetro, não é necessário fornecer o seu tamanho na declaração da função.
- Quando o **vetor é multi-dimensional** a possibilidade de não informar o tamanho na declaração se restringe à primeira dimensão apenas.

```
void mostra_matriz(int mat[][10], int n_linhas) {
  ...
}
```

Vetores multi-dimensionais e funções

- Pode-se criar uma função deixando de indicar a primeira dimensão:

```
void mostra_matriz(int mat[][10], int n_linhas) {
  ...
}
```

- Ou pode-se criar uma função indicando todas as dimensões:

```
void mostra_matriz(int mat[5][10], int n_linhas) {
  ...
}
```

- Mas não pode-se deixar de indicar outras dimensões (exceto a primeira):

```
void mostra_matriz(int mat[5][], int n_linhas) {
  //ESTE NÃO FUNCIONA
  ...
}
```

Vetores multi-dimensionais em funções

```
void mostra_matriz(int mat[][10], int n_linhas) {
  int i, j;

  for (i = 0; i < n_linhas; i++) {
    for (j = 0; j < 10; j++)
      printf("%2d ", mat[i][j]);
    printf("\n");
  }
}

int main() {
  int mat[][10] = { { 0,  1,  2,  3,  4,  5,  6,  7,  8,  9},
                    {10, 11, 12, 13, 14, 15, 16, 17, 18, 19},
                    {20, 21, 22, 23, 24, 25, 26, 27, 28, 29},
                    {30, 31, 32, 33, 34, 35, 36, 37, 38, 39},
                    {40, 41, 42, 43, 44, 45, 46, 47, 48, 49},
                    {50, 51, 52, 53, 54, 55, 56, 57, 58, 59},
                    {60, 61, 62, 63, 64, 65, 66, 67, 68, 69},
                    {70, 71, 72, 73, 74, 75, 76, 77, 78, 79}};

  mostra_matriz(mat, 8);
  return 0;
}
```

Vetores multi-dimensionais em funções

Lembre-se que vetores (multi-dimensionais ou não) são alterados quando passados como parâmetro em uma função

```
void    teste (int mat[2][2]) {
  int i, j;

  for (i = 0; i < 2; i++) {
    for (j = 0; j < 2; j++){
          mat[i][j] = -1;
    }
  }
}

int main() {
  int mat[2][2] = { { 0, 1},
                    { 2, 3} };

  teste(mat);
  //Neste ponto mat tem que valores em suas posições???

  return 0;
}
```

Exemplo

Criar aplicações com operações básicas sobre matrizes quadradas:

- Soma de 2 matrizes com dimensões $l \times c$.
- Subtração de 2 matrizes com dimensões $l \times c$.
- Cálculo da transposta de uma matriz de dimensão $l \times c$.
- Multiplicação de 2 matrizes com dimensões $l \times c$ e $l_2 \times c_2$ onde $c = l_2$.

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_SIZE 100

void readMat(double mat[MAX_SIZE][MAX_SIZE], int l, int c){
  int i, j;

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

- **MAX_SIZE** é uma constante inteira definida com valor 100 (tam. max. matriz).
- Note porém que o tamanho efetivo da matriz é o número de linhas **l** ≤ 100 e colunas **c**≤ 100 passado como parâmetros para a função.

Exemplo: Lendo e Imprimindo uma Matriz

- Agora o código da função que faz a impressão de uma matriz:

```
void printMat(double mat[MAX_SIZE][MAX_SIZE], int l, int c){
  int i, j;

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

- 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

- Com as funções anteriores podemos criar uma função **main** como abaixo:

```
#include <stdio.h>

#define MAX_SIZE 100

void readMat(double mat[MAX_SIZE][MAX_SIZE], int l, int c);
void printMat(double mat[MAX_SIZE][MAX_SIZE], int l, int c);


int main(){
  double m1[MAX_SIZE][MAX_SIZE], m2[MAX_SIZE][MAX_SIZE], m3[MAX_SIZE][MAX_SIZE];
  int l1, c1, l2, c2, l3, c3;

  printf("Dimensoes da matriz 1: ");
  scanf("%d %d", &l1, &c1);
  printf("Dimensoes da matriz 2: ");
  scanf("%d %d", &l2, &c2);

  readMat(m1, l1, c1);
  readMat(m2, l2, c2);

  printMat(m1, l1, c1);
  printMat(m2, l2, c2);

}
```

Exemplo: Soma de Matrizes

- Vamos implementar a funcionalidade de soma de matrizes.
- A função recebe como parâmetro a matriz resposta mat3.

```
int soma(double mat1[][MAX_SIZE], int l1, int c1, double mat2[][MAX_SIZE],
                                                  int l2, int c2, double mat3[][MAX_SIZE]){
  int i, j;

  if(c1 != c2 || l1 != l2)
    return 0;
  ....
  ....

  return 1;
}
```

- A função devolve 1 caso a soma foi feita (matrizes de dimensões compatíveis) ou 0 caso contrário (matrizes de dimensões incompatíveis).

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 soma(double mat1[][MAX_SIZE], int l1, int c1, double mat2[][MAX_SIZE],
                                   int l2, int c2, double mat3[][MAX_SIZE]){
  int i, j;

  if(c1 != c2 || l1 != l2)
    return 0;

  for(i=0; i<l1; i++){
    for(j=0; j<c1; j++){
      mat3[i][j] = mat1[i][j] + mat2[i][j];
    }
  }

  return 1;
}
```

Exemplo: Soma de Matrizes

- Com as funções anteriores podemos alterar o **main** para:

```
int main(){
  double m1[MAX_SIZE][MAX_SIZE], m2[MAX_SIZE][MAX_SIZE], m3[MAX_SIZE][MAX_SIZE];
  int l1, c1, l2, c2, l3, c3;

  printf("Dimensoes da matriz 1: ");
  scanf("%d %d", &l1, &c1);
  printf("Dimensoes da matriz 2: ");
  scanf("%d %d", &l2, &c2);
  readMat(m1, l1, c1);
  readMat(m2, l2, c2);
  printMat(m1, l1, c1);
  printMat(m2, l2, c2);

  if(soma(m1, l1, c1, m2, l2, c2, m3)){
    l3 = l1; c3 = c1;
    printf("Resultado da soma:\n");
    printMat(m3, l3, c3);
  }

}
```

Exemplo: Multiplicação de Matrizes

- Vamos implementar a funcionalidade de multiplicação de matrizes.
- Vamos multiplicar duas matrizes M_1 e M_2 (de dimensões $l_1 \times c_1$ e $l_2 \times c_2$ com $c_1 = l_2$).
- O resultado será uma terceira matriz M_3 (de dimensões $l_1 \times c_2$).
- 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:

$$M_3[i,j] = \sum_{k=0}^{c_1-1} M_1[i,k] \cdot M_2[k,j]$$

Exemplo: Multiplicação de Matrizes

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

$$\text{mat3}[i,j] = \sum_{k=0}^{c_1-1} \text{mat1}[i,k] \cdot \text{mat2}[k,j]$$

```
...
for(i=0; i < l1; i++){
  for(j=0; j < c2; j++){
    mat3[i][j] = 0;
    for(k=0; k < c_1; k++){
      mat3[i][j] = mat3[i][j] + (mat1[i][k] * mat2[k][j]);
    }
  }
}
...
```

Exemplo: Multiplicação de Matrizes

- Abaixo temos a função que devolve 1 caso a multiplicação possa ser feita e 0 caso contrário.

```
int mult(double mat1[][MAX_SIZE], int l1, int c1, double mat2[][MAX_SIZE],
                                  int l2, int c2, double mat3[][MAX_SIZE]){
  int i, j, k;

  if(c1 != l2)
    return 0;

  for(i=0; i < l1; i++){
    for(j=0; j < c2; j++){
      mat3[i][j] = 0;
      for(k=0; k < c1; k++)
        mat3[i][j] = mat3[i][j] + (mat1[i][k] * mat2[k][j]);
    }
  }

  return 1;
}
```

Exemplo: Multiplicação de Matrizes

- Podemos usar a função de multiplicação na **main**:

```
int main(){
  double m1[MAX_SIZE][MAX_SIZE], m2[MAX_SIZE][MAX_SIZE], m3[MAX_SIZE][MAX_SIZE];
  int l1, c1, l2, c2, l3, c3;

  printf("Dimensoes da matriz 1: ");
  scanf("%d %d", &l1, &c1);
  printf("Dimensoes da matriz 2: ");
  scanf("%d %d", &l2, &c2);
  readMat(m1, l1, c1);
  readMat(m2, l2, c2);
  printMat(m1, l1, c1);
  printMat(m2, l2, c2);

  if(soma(m1, l1, c1, m2, l2, c2, m3)){
    l3 = l1; c3 = c1;
    printf("Resultado da soma:\n");
    printMat(m3, l3, c3);
  }

  if(mult(m1, l1, c1, m2, l2, c2, m3)){
    l3 = l1; c3 = c2;
    printf("Resultado da multiplicacao:\n");
    printMat(m3, l3, c3);
  }
}
```

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} }
   };

   .
   .
   .
   .
}
```