

Matrizes e Vetores Multidimensionais

Algoritmos e Programação de Computadores – ABI/LFI/TAI

Prof. Daniel Saad Nogueira Nunes



**INSTITUTO
FEDERAL**

Brasília

Campus
Taguatinga

Sumário

Introdução

Matrizes

Exemplos

Sumário

Introdução

Matrizes

Exemplos

Introdução

- ▶ Imagine o seguinte cenário: queremos armazenar as notas de 10 avaliações de 30 estudantes de uma determinada turma.
- ▶ Uma opção é criar 30 vetores de tamanho 10, sendo um vetor para cada aluno.

```
1  double notas_1[10];  
2  double notas_2[10];  
3  double notas_3[10];  
4  double notas_4[10];  
5  double notas_5[10];  
6  double notas_6[10];  
7  // ...  
8  double notas_30[10];
```

Introdução

- ▶ As matrizes fornecem um meio de fazer a mesma coisa através de um único identificador.
- ▶ `double notas[30][10]` .
- ▶ Também é possível criar vetores de 3 ou mais dimensões em C.

Sumário

Introdução

Matrizes

Exemplos

Sumário

Matrizes

Sintaxe

Acesso

Vetores n -dimensionais

Vetor de strings

Vetores n -dimensionais e funções

Inicialização

Sintaxe

- ▶ Uma matriz deve ser declarada com a seguinte sintaxe:

```
tipo nome_matriz[nro_linhas][nro_colunas] .
```

- ▶ A declaração é similar a de vetores, mas é necessário especificar duas dimensões: o número de linhas e o número de colunas.
- ▶ Exemplo: `double notas_alunos[30][100];`
- ▶ Se a matriz tiver n linhas e m colunas, as linhas são indexadas de 0 a $n - 1$ e as colunas de 0 a $m - 1$.

Exemplo

- ▶ A declaração de uma matriz de inteiros, de tamanho 4×4 , `int matriz[4][4]`, nos traria a seguinte configuração na memória.

	0	1	2	3
0				
1				
2				
3				

Sumário

Matrizes

Sintaxe

Acesso

Vetores n -dimensionais

Vetor de strings

Vetores n -dimensionais e funções

Inicialização

Acesso

- ▶ Para acessar qualquer dado de uma matriz de tamanho $n \times m$, utilizamos os `[]`, como nos vetores.
- ▶ `matriz[i][j]`
- ▶ Em que i é o índice da linha e j o índice da linha, com $0 \leq i < n$ e $0 \leq j < m$.

Acesso

```
1  int matriz[100][100]
2  // ...
3  int aux = matriz[10][12];
```

- ▶ Neste exemplo, acessamos o elemento que se localiza na décima primeira linha e na décima terceira coluna.
- ▶ Lembre-se que os índices começam de 0!

Acesso

- ▶ Assim como em vetores, o C não verifica se os índices utilizados para referenciar a matriz estão dentro dos limites da mesma.
- ▶ Caso não estejam, não há como prever o que pode acontecer. É possível inclusive que o programa aborte ou manipule os dados de outras variáveis.
- ▶ **Erro de lógica!**
- ▶ O programador tem total responsabilidade sobre a manipulação correta dos índices.

Sumário

Matrizes

Sintaxe

Acesso

Vetores n -dimensionais

Vetor de strings

Vetores n -dimensionais e funções

Inicialização

Vetores n -dimensionais

- ▶ Em C é possível declarar e utilizar estruturas com mais de 2 dimensões.
- ▶ Vetores n -dimensionais.
- ▶ Por exemplo, suponha que queiramos armazenar a quantidade de chuva que caiu em cada dia, de cada mês, nos anos de 0 a 2021.
- ▶ `double qtd_chuva[2022][12][31];`
- ▶ Para acessar a quantidade de chuva no dia **10 de junho de 1960**, fazemos:
`double qtd = qtd_chuva[1960][5][9];`

Sumário

Matrizes

Sintaxe

Acesso

Vetores n -dimensionais

Vetor de strings

Vetores n -dimensionais e funções

Inicialização

Vetor de strings

- ▶ Em C uma string é um vetor de caracteres.
- ▶ Um vetor de strings, é uma *matriz de caracteres!*.
- ▶ Cada linha da matriz é uma string.
- ▶ Se quiséssemos armazenar 30 nomes com até 30 caracteres sem espaços, poderíamos declarar.
- ▶

```
char nomes[30][31];
```

Vetor de strings

```
1  int i;  
2  char nomes[30][31];  
3  for(i=0;i<30;i++){  
4      printf("Digite o %d nome: ",i+1);  
5      scanf("%s",nomes[i]);  
6  }
```

Sumário

Matrizes

Sintaxe

Acesso

Vetores n -dimensionais

Vetor de strings

Vetores n -dimensionais e funções

Inicialização

Vetores n -dimensionais e funções

- ▶ Assim como em vetores, matrizes e vetores n -dimensionais não são copiados quando passados para funções.
- ▶ **Qualquer alteração na matriz em uma função, refletirá na matriz original.**

Vetores n -dimensionais e funções

- ▶ Nos parâmetros de uma função, podemos omitir apenas a **primeira dimensão!**.

```
1 void imprime_matriz(int matriz[][10], int n_linhas) {  
2     //...  
3 }
```

- ▶ Uma omissão de qualquer dimensão que não seja a primeira acarretará em um erro de compilação.

Exemplo

```
3 void mostra_matriz(int mat[][10], int n_linhas) {  
4     int i, j;  
5  
6     for (i = 0; i < n_linhas; i++) {  
7         for (j = 0; j < 10; j++) {  
8             printf("%2d ", mat[i][j]);  
9         }  
10        printf("\n");  
11    }  
12 }
```

Exemplo

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

Exemplo

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


Sumário

Matrizes

Sintaxe

Acesso

Vetores n -dimensionais

Vetor de strings

Vetores n -dimensionais e funções

Inicialização

Inicialização

- ▶ Assim como em vetores, é possível inicializar matrizes e vetores n -dimensionais em sua declaração.

```
1  int matriz[3][3] = {  
2      {1, 2, 3},  
3      {4, 5, 6},  
4      {7, 8, 9}  
5  };
```

Inicialização

- ▶ Em um vetor com 3 dimensões, teríamos:

```
1  int v3d[2][3][4] = {  
2      {  
3          {1, 2, 3, 4},  
4          {5, 6, 7, 8},  
5          {9, 10, 11, 12}  
6      },  
7      {  
8          {0, 0, 0, 0},  
9          {5, 6, 7, 8},  
10         {0, 0, 0, 0}  
11     }  
12 };
```

Sumário

Introdução

Matrizes

Exemplos

Exemplos

- ▶ Para fixar os conceitos, construiremos funções que forneçam as seguintes operações entre matrizes:
 - ▶ Leitura;
 - ▶ Impressão;
 - ▶ Soma matricial;
 - ▶ Subtração matricial;
 - ▶ Cálculo da matriz transposta;
 - ▶ Produto matricial.
- ▶ O tamanho máximo das matrizes será 100×100 .

Exemplos

- ▶ Como o tamanho das matrizes será no máximo 100×100 , podemos definir através de uma macro.

```
#define MAX 100
```

- ▶ Mesmo que o tamanho máximo 100×100 , o espaço útil, efetivamente utilizado pelo programa, pode ser menor.
- ▶ Dependerá do que o usuário digitar na leitura das estruturas.
- ▶ Sempre que o pré-processador encontrar a palavra `MAX` ele substituirá por `100`

Definição de tamanho

```
1  #include <stdio.h>
2
3  #define MAX 100
```

Sumário

Exemplos

Leitura

Escrita

Adição matricial

Subtração matricial

Transposição

Produto

Leitura

```
5 void le_matriz(double matriz[MAX][MAX], int n, int m) {  
6     int i, j;  
7     for (i = 0; i < n; i++) {  
8         for (j = 0; j < m; j++) {  
9             scanf("%lf", &matriz[i][j]);  
10        }  
11    }  
12 }
```

Sumário

Exemplos

Leitura

Escrita

Adição matricial

Subtração matricial

Transposição

Produto

Escrita

```
14 void imprime_matriz(double matriz[MAX][MAX], int n, int m) {  
15     int i, j;  
16     for (i = 0; i < n; i++) {  
17         for (j = 0; j < m; j++) {  
18             printf("%.2f ", matriz[i][j]);  
19         }  
20         printf("\n");  
21     }  
22     printf("\n");  
23 }
```

Sumário

Exemplos

Leitura

Escrita

Adição matricial

Subtração matricial

Transposição

Produto

Adição

```
25 void soma_matriz(double matriz_a[MAX] [MAX], double matriz_b[MAX] [MAX],  
26                 double matriz_res[MAX] [MAX], int n, int m) {  
27     int i, j;  
28     for (i = 0; i < n; i++) {  
29         for (j = 0; j < m; j++) {  
30             matriz_res[i][j] = matriz_a[i][j] + matriz_b[i][j];  
31         }  
32     }  
33 }
```

Adição

```
int main(void) {  
    double matriz_a[MAX][MAX], matriz_b[MAX][MAX], matriz_c[MAX][MAX];  
    int n1, m1, n2, m2;  
    scanf("%d %d %d %d", &n1, &m1, &n2, &m2);  
    if (n1 != n2 || m1 != m2) {  
        printf(  
            "Impossível realizar soma. Dimensões das matrizes incompatíveis\n");  
        return 0;  
    }  
    le_matriz(matriz_a, n1, m1);  
    le_matriz(matriz_b, n2, m2);  
    imprime_matriz(matriz_a, n1, m1);  
    imprime_matriz(matriz_b, n1, m1);  
    soma_matriz(matriz_a, matriz_b, matriz_c, n1, m1);  
    imprime_matriz(matriz_c, n1, m1);  
    return 0;  
}
```

Sumário

Exemplos

Leitura

Escrita

Adição matricial

Subtração matricial

Transposição

Produto

Subtração

```
35 void subtrai_matriz(double matriz_a[MAX][MAX], double matriz_b[MAX][MAX],
36                     double matriz_res[MAX][MAX], int n, int m) {
37     int i, j;
38     for (i = 0; i < n; i++) {
39         for (j = 0; j < m; j++) {
40             matriz_res[i][j] = matriz_a[i][j] - matriz_b[i][j];
41         }
42     }
43 }
```


Subtração

```
int main(void) {
    double matriz_a[MAX][MAX], matriz_b[MAX][MAX], matriz_c[MAX][MAX];
    int n1, m1, n2, m2;
    scanf("%d %d %d %d", &n1, &m1, &n2, &m2);
    if (n1 != n2 || m1 != m2) {
        printf("Impossível realizar subtração. Dimensões das matrizes "
               "incompatíveis\n");
        return 0;
    }
    le_matriz(matriz_a, n1, m1);
    le_matriz(matriz_b, n2, m2);
    imprime_matriz(matriz_a, n1, m1);
    imprime_matriz(matriz_b, n1, m1);
    subtrai_matriz(matriz_a, matriz_b, matriz_c, n1, m1);
    imprime_matriz(matriz_c, n1, m1);
    return 0;
}
```

Sumário

Exemplos

Leitura

Escrita

Adição matricial

Subtração matricial

Transposição

Produto

Transposição

- ▶ A transposição de uma matriz $M_{n \times m}$ nos dá uma matriz $M'_{m \times n}$, em que $M'[i][j] = M[j][i]$ para $0 \leq i < m$ e $0 \leq j < n$.

Transposição

```
45 void transpoe_matriz(double matriz_a[MAX][MAX], double matriz_res[MAX][MAX],  
46                     int n, int m) {  
47     int i, j;  
48     for (i = 0; i < m; i++) {  
49         for (j = 0; j < n; j++) {  
50             matriz_res[i][j] = matriz_a[j][i];  
51         }  
52     }  
53 }
```

Transposição

```
int main(void) {  
    double matriz_a[MAX][MAX], matriz_b[MAX][MAX];  
    int n, m;  
    scanf("%d %d", &n, &m);  
    le_matriz(matriz_a, n, m);  
    imprime_matriz(matriz_a, n, m);  
    transpoe_matriz(matriz_a, matriz_b, n, m);  
    imprime_matriz(matriz_b, m, n);  
    return 0;  
}
```

Sumário

Exemplos

Leitura

Escrita

Adição matricial

Subtração matricial

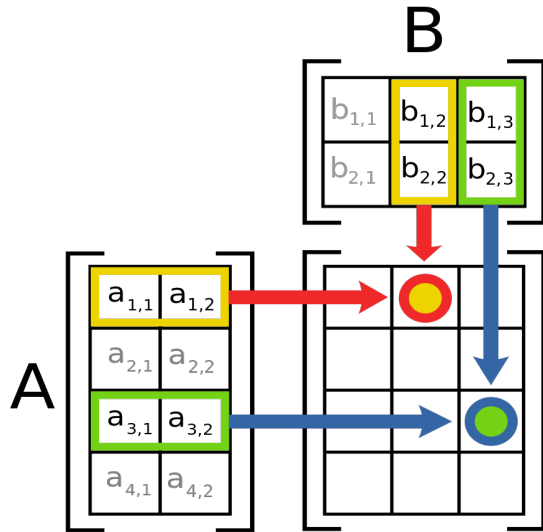
Transposição

Produto

Produto

- ▶ O produto matricial de duas matrizes $A_{n_1 \times m_1}$ por $B_{n_2 \times m_2}$ nos dá uma matriz $C_{n_1 \times m_2}$.
- ▶ Só está definido se $m_1 = n_2$.
- ▶ Cada célula $C[i][j]$ é definida como um produto escalar de $A[i][k]$ e $B[k][j]$, com $0 \leq k < m_1$.

Produto



Produto

```
55 void multiplica_matriz(double matriz_a[MAX][MAX], double matriz_b[MAX][MAX],
56                       double matriz_res[MAX][MAX], int n1, int m1, int n2,
57                       int m2) {
58     for (int i = 0; i < n1; i++) {
59         for (int j = 0; j < m2; j++) {
60             matriz_res[i][j] = 0;
61             for (int k = 0; k < m1; k++) {
62                 matriz_res[i][j] += matriz_a[i][k] * matriz_b[k][j];
63             }
64         }
65     }
66 }
```

Produto

```
int main(void) {  
    double matriz_a[MAX][MAX], matriz_b[MAX][MAX], matriz_c[MAX][MAX];  
    int n1, m1, n2, m2;  
    scanf("%d %d %d %d", &n1, &m1, &n2, &m2);  
    if (m1 != n2) {  
        printf("Impossível realizar multiplicação. Dimensões das matrizes "  
               "incompatíveis\n");  
        return 0;  
    }  
    le_matriz(matriz_a, n1, m1);  
    le_matriz(matriz_b, n2, m2);  
    imprime_matriz(matriz_a, n1, m1);  
    imprime_matriz(matriz_b, n2, m2);  
    multiplica_matriz(matriz_a, matriz_b, matriz_c, n1, m1, n2, m2);  
    imprime_matriz(matriz_c, n1, m2);  
    return 0;  
}
```