

Linguagens de Programação I

Tema # 8

Matrizes Multidimensionais

Susana M Iglesias

MATRIZ - INTRODUÇÃO

- Os vetores ou matrizes em C podem ter vários subscritos,
- Uma matriz com mais de um subscrito é chamada multidimensional,
- Matrizes de dois subscritos ou bidimensionais são usualmente utilizados para representar tabelas de valores do mesmo tipo,
- Uma matriz bidimensional é um vetor onde cada elemento é também um vetor.

MATRIZ - INTRODUÇÃO

- Exemplos:
 - Notas de uma turma em uma disciplina considerando várias avaliações,
 - Resultados de um campeonato de futebol (equipes, rodadas, pontos)
 - Estoque de uma loja de roupa (tamanho, cores). Como considerar vários produtos?

REPRESENTAÇÃO DE UMA MATRIZ BIDIMENSIONAL

Matriz de 4 x 3

| | Coluna 0 | Coluna 1 | Coluna 2 |
|---------|----------|----------|----------|
| Linha 0 | a[0][0] | a[0][1] | a[0][2] |
| Linha 1 | a[1][0] | a[1][1] | a[1][2] |
| Linha 2 | a[2][0] | a[2][1] | a[2][2] |
| Linha 3 | a[3][0] | a[3][1] | a[3][2] |

nome da matriz

subscrito da linha

subscrito da coluna

EXEMPLO DE UMA MATRIZ BIDIMENSIONAL

Notas de LP1

| | Avaliação 0 | Avaliação 1 | Avaliação 2 |
|---------|-------------|-------------|-------------|
| Aluno 0 | 5 . 3 | 8 . 3 | 7 . 1 |
| Aluno 1 | 4 . 2 | 6 . 8 | 7 . 3 |
| Aluno 2 | 6 . 0 | 5 . 4 | 1 . 0 |
| Aluno 3 | 9 . 7 | 10 . 0 | 9 . 6 |
| Aluno 4 | 2 . 1 | 5 . 8 | 7 . 9 |

ARMAZENAMENTO NA MEMÓRIA DA MATRIZ BIDIMENSIONAL

| | Memória | Endereço |
|----------------------|---------|----------|
| <code>a[0][0]</code> | 0 | 0 |
| <code>a[0][1]</code> | 1 | 32 |
| <code>a[0][2]</code> | 2 | 64 |
| <code>a[1][0]</code> | 1 | 96 |
| <code>a[1][1]</code> | 2 | 128 |
| <code>a[1][2]</code> | 3 | 160 |
| <code>a[2][0]</code> | 2 | 192 |
| <code>a[2][1]</code> | 3 | 224 |
| <code>a[2][2]</code> | 4 | 256 |

- Os elementos de uma matriz são armazenados em uma sequência contínua na memória.
- Mecanismo de busca de um elemento da matriz
- Representação por linha
`a[r1][r2]`
Acessando a **`a[i1][i2]`**
 $\text{base}(a) + (i1 * r2 + i2) * \text{esize}$

DECLARANDO MATRIZES BIDIMENSIONAIS

- Sintaxe:

```
tipo nome_matriz[num_linhas][num_colunas];
```

- O numero de linhas e colunas indicado na declaração devem ser constantes inteiras,
- O subscrito da linha: 0 .. num_linhas -1
- O subscrito da colunas: 0 .. num_colunas -1

- Exemplos:

```
float n[5][3];  
int a[4][4];  
char ch[10][5];
```

INICIALIZANDO MATRIZES BIDIMENSIONAIS

- Ao igual que os vetores as matrizes podem ser inicializadas na declaração utilizando uma lista de inicializadores,
- As regras de inicialização descritas para os vetores também são válidas para as matrizes
- Exemplos:

```
int n[3][3] = {{0,1,2},{1,2,3},{2,3,4}};
```

```
float a[4][4] = {{1,2,3,4},{5,6},{6}};
```


EXEMPLO 1

```
#define lin 5
#define col 3

int main()
{
    float notas[lin][col] = {{5.3, 8.3, 7.1},{4.3, 6.8, 7.3},
                              {6.0, 5.4, 1.0},{9.7, 10.0, 9.6},
                              {2.1, 5.8, 7.9}};

    int i, j;

    printf("    ");
    for(j=0;j<col;j++)
        printf("%6d",j);
    printf("\n");
    for(i=0;i<lin;i++){
        printf("%3d",i);
        for(j=0;j<col;j++)
            printf("%6.1f", notas[i][j]);
        printf("\n");
    }
}
```

| | 0 | 1 | 2 |
|---|-----|------|-----|
| 0 | 5.3 | 8.3 | 7.1 |
| 1 | 4.3 | 6.8 | 7.3 |
| 2 | 6.0 | 5.4 | 1.0 |
| 3 | 9.7 | 10.0 | 9.6 |
| 4 | 2.1 | 5.8 | 7.9 |

Press any key to continue . . .

EXEMPLO 2

Modifique o exercício anterior para criar um vetor com a nota final de cada um dos alunos, e outro vetor com a média da turma em cada avaliação.

```
#include <stdio.h>
#include <stdlib.h>

#define lin 5
#define col 3

int main()
{
    float notas[lin][col] = {{5.3, 8.3, 7.1},{4.3, 6.8, 7.3},
                              {6.0, 5.4, 1.0},{9.7, 10.0, 9.6},
                              {2.1, 5.8, 7.9}};

    float alunos[lin]={0}, aval[col]={0};
    int i, j;

    printf("    ");
    for(j=0;j<col;j++)
        printf("%6d",j);
    printf("\n");
```

EXEMPLO 2

```
for(i=0;i<lin;i++){
    printf("%3d",i);
    for(j=0;j<col;j++){
        printf("%6.1f", notas[i][j]);
        alunos[i] += notas[i][j];
        aval[j] += notas[i][j];
    }
    printf("\n");
}

printf("\nNotas alunos:\n");
for(i=0;i<lin;i++)
    printf("%d %6.1f\n", i, alunos[i]/col);

printf("\nMedia das avaliacoes:\n");
for(j=0;j<col;j++)
    printf("%d %6.1f\n", j, aval[j]/lin);

system("PAUSE");
return 0;
}
```

EXEMPLO 2

| | 0 | 1 | 2 |
|---|-----|------|-----|
| 0 | 5.3 | 8.3 | 7.1 |
| 1 | 4.3 | 6.8 | 7.3 |
| 2 | 6.0 | 5.4 | 1.0 |
| 3 | 9.7 | 10.0 | 9.6 |
| 4 | 2.1 | 5.8 | 7.9 |

Notas alunos:

| | |
|---|-----|
| 0 | 6.9 |
| 1 | 6.1 |
| 2 | 4.1 |
| 3 | 9.8 |
| 4 | 5.3 |

Media das avaliacoes:

| | |
|---|-----|
| 0 | 5.5 |
| 1 | 7.3 |
| 2 | 6.6 |

Press any key to continue . . .

GERENCIAMENTO DE MEMÓRIA EM MATRIZES

- Ao trabalharmos com matrizes a memória é reservada em tempo de compilação,
- Se a quantidade de elementos da matriz é desconhecida devemos alocar uma quantidade de memória "suficiente",
- A quantidade de memória necessária para a execução do programa deve ser verificada.

GERENCIAMENTO DE MEMÓRIA EM MATRIZES

```
#define NL 3
#define NC 3

int main(){
    int m[NL][NC];
    int nl, nc;

    printf("Digite o numero de linhas e colunas:");
    scanf("%d %d", &nl, &nc);

    if (nl>NL || nc>NC){
        printf("Erro!!! Estouro de memoria.\n");
        return -1;
    }

    /*
        Processamento dos elementos da matriz
    */

    return 0;
}
```

- Qual o resultado de `m[12][15]`?

MATRIZES COMO ARGUMENTOS DE FUNÇÕES

- Matrizes bidimensionais podem ser utilizadas como parâmetros em funções,
- Ao igual que em matrizes unidimensionais, matrizes bidimensionais são pasadas por referência,
- Apenas o endereço base da matriz é enviado a função,
- No prototipo da função e na declaração do parâmetro devemos indicar o numero de colunas da matriz.

MATRIZES COMO ARGUMENTOS DE FUNÇÕES

- **Exemplo:** *Crie uma função que receba e imprima os valores de uma matriz.*


```
#include <stdio.h>
#include <stdlib.h>
#define NL 3
#define NC 3

void prn_mat(int[][NC], int, int);

int main(){
    int m[NL][NC] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

    prn_mat(m, NL, NC);

    return 0;
}

void prn_mat(int mat[][NC], int l, int c){
    int i, j;
    for(i=0;i<l;i++){
        for(j=0;j<c;j++){
            printf("%4d", mat[i][j]);
        }
        printf("\n");
    }
}
```

OPERAÇÕES COM MATRIZES

- Adição e Subtração:

$$\overline{\overline{C}}_{I \times J} = \overline{\overline{A}}_{I \times J} \pm \overline{\overline{B}}_{I \times J}$$

$$c[i][j] = a[i][j] \pm b[i][j]$$

- a operação só está definida se as matrizes A e B tiverem o mesmo número de linhas e colunas,
- a matriz C terá igual número de colunas que A e B

OPERAÇÕES COM MATRIZES

- Multiplicação de matriz por vetor:

$$\vec{v}_I = \overline{\overline{A}}_{I \times J} \vec{d}_J$$

$$v[i] = a[i][0]d[0] + a[i][1]d[1] + \dots + a[i][N-1]d[N-1]$$

$$v[i] = \sum_{j=0}^{N-1} a[i][j]d[j]$$

- a operação só está definida se o numero de colunas de A for igual ao numero de elementos de d ,
- a quantidade de elementos de v será o número de linhas de A .

OPERAÇÕES COM MATRIZES

- Multiplicação de matriz por matriz:

$$\overline{\overline{C}}_{I \times M} = \overline{\overline{A}}_{I \times J} \times \overline{\overline{B}}_{J \times M}$$

$$c[i][j] = a[i][0]b[0][j] + a[i][1]b[1][j] + \dots + a[i][N-1]b[N-1][j]$$

$$c[i][j] = \sum_{k=0}^{N-1} a[i][k]b[k][j]$$

- a operação só esta definida se o numero de colunas de A for igual ao numero de linhas de B .

MATRIZES TRIDIMENSIONAIS

- Como declarar uma matriz tridimensional?
- Como acessar um elemento de uma matriz tridimensional?
- Quantos laços são precisos para percorrer uma matriz tridimensional?
- Em C não existem limitações para o número de dimensões que uma matriz pode ter.