Fundamentos de Programação

Aula 14 - Estruturas de Dados Básicas: Matriz

Profª. Elisa de Cássia Silva Rodrigues

Estrutura de Dados

- Uma estrutura de dados representa uma forma eficiente de armazenar e organizar dados de modo que possam ser usados de forma eficiente.
- Uma estrutura de dados armazena dados do mesmo tipo.
- A utilização de estrutura de dados é importante pois a organização eficiente dos dados diminui o custo de um algoritmo com relação ao tempo de execução e consumo de memória.
- Estruturas de dados básicas:
  - Vetores.
  - Matrizes.
  - Registros.

Definição de Matriz

- Matriz é um conjunto de variáveis de mesmo tipo alocados sequencialmente na memória.
- Esse conjunto de variáveis possui o mesmo identificador (nome).
- Também conhecido como variável composta homogênea multidimensional.
- As variáveis se distinguem pelo índice que referencia sua localização no vetor.
- Um variável do tipo matriz possui um índice para cada umas das suas dimensões.

Declaração de Matriz (Bidimensional)

Sintaxe (Pseudocódigo)

DECLARE `nome`[`dimensao1`, `dimensao2`] `tipo`

onde:

`nome` é o nome da variável do tipo matriz.

`dimensao1` é a quantidade de elementos da 1ª dimensão da matriz.

- A 1ª dimensão também é chamada de linha.

`dimensao2` é a quantidade de elementos da 2ª dimensão da matriz.

- A 2ª dimensão também é chamada de coluna.

`tipo` é o tipo básico dos dados que serão armazenados na matriz.

Declaração de Matriz (Bidimensional)

Exemplo (Pseudocódigo)

DECLARE `M[3, 5] inteiro`

// matriz chamada `M` que possui 3 linhas e 5 colunas (posições armazenadas sequencialmente na memória).

M

	0	1	2	3	4
0	10	9	13	7	1
1	20	8	11	4	6
2	30	5	18	2	12

M[0,0]←10 M[0,1]←9 M[0,2]←13
M[1,0]←20 M[1,1]←8 M[1,2]←11
M[2,0]←30 M[2,1]←5 M[2,2]←18

M[0,3]←7 M[0,4]←1
M[1,3]←4 M[1,4]←6
M[2,3]←2 M[2,4]←12

Declaração de Matriz (Bidimensional)

Sintaxe (Linguagem C)

```
tipo nome[dimensao1][dimensao2];
```

onde:

`nome` é o nome da variável do tipo matriz.

`dimensao1` é a quantidade de elementos da 1ª dimensão da matriz.

- A 1ª dimensão também é chamada de linha.

`dimensao2` é a quantidade de elementos da 2ª dimensão da matriz.

- A 2ª dimensão também é chamada de coluna.

`tipo` é o tipo básico dos dados que serão armazenados na matriz.

Declaração de Matriz (Bidimensional)

Exemplo (Linguagem C)

```
int X[3][5];
// matriz chamada M que possui 3 linhas e 5 colunas (posições armazenadas sequencialmente na memória).
```

M

	0	1	2	3	4
0	10	9	13	7	1
1	20	8	11	4	6
2	30	5	18	2	12

M[0,0] = 10 M[0,1] = 9 M[0,2] = 13
M[1,0] = 20 M[1,1] = 8 M[1,2] = 11
M[2,0] = 30 M[2,1] = 5 M[2,2] = 18

M[0,3] = 7 M[0,4] = 1
M[1,3] = 4 M[1,4] = 6
M[2,3] = 2 M[2,4] = 12

Preenchimento da Matriz

- Preencher uma matriz significa atribuir valores a todas as posições.
- Para isso usa-se dois laços de repetição aninhados (`PARA`, por exemplo).
- As variáveis contadoras `i` e `j` são os índices da matriz que se referem as linhas e as colunas, respectivamente.
- O elemento `X[i][j]` da matriz se encontra na coluna `j` da linha `i`.

Preenchimento da Matriz

Preenchendo uma matriz M com 3 linhas e 5 colunas (Pseudocódigo)

```
PARA i ← 0 ATÉ 2 FAÇA {
     PARA i ← 0 ATÉ 4 FAÇA {
          ESCREVA "Digite o número da posição (",i,",",j,"):"
          LEIA X[i,j]
     }
}
```

Preenchendo uma matriz M com 3 linhas e 5 colunas (Linguagem C)

```
for(i = 0; i <= 2; i++) {
     for(i = 0; i <= 4; i++) {
          printf("Digite o número da posição (%d,%d):", i, j);
          scanf("%d", &X[i][j]);
     }
}
```

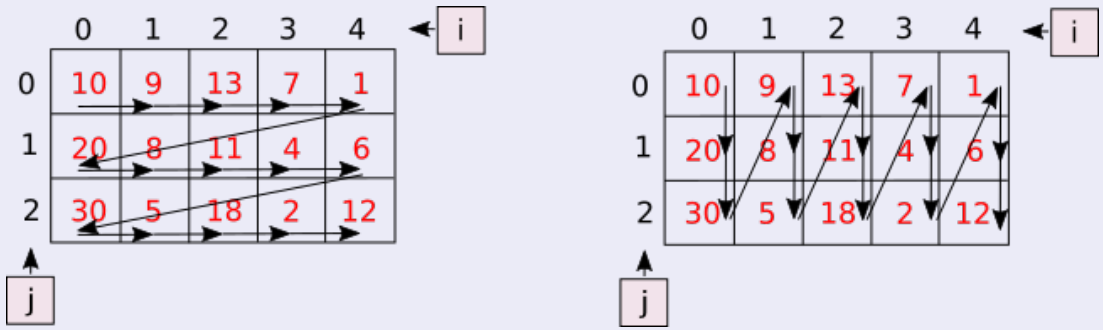
Preenchimento da Matriz

- O 1º laço de repetição deve garantir que a variável `i` assuma todas as posições entre 0 e 2 (posições válidas para as linhas da matriz com 3 linhas).
- O 2º laço de repetição deve garantir que a variável `j` assuma todas as posições entre 0 e 4 (posições válidas para as colunas da matriz com 5 colunas).
- A cada iteração do 1º laço uma linha da matriz é preenchida.
- A cada iteração do 2º laço uma coluna da linha `i` da matriz será preenchida.

Preenchimento da Matriz

- A direção das setas indica a mudança de valor das variáveis `i` e `j`.
- Caminho utilizado pelo algoritmo para percorrer a matriz.
  - Se o 1º laço de repetição (`i`) se referir as linhas e o 2º laço (`j`) se referir as colunas, então o algoritmo percorre todos os elementos de uma linha antes de ir para a próxima (Figura à esquerda).
  - Se o 1º laço de repetição (`j`) se referir as colunas e o 2º laço (`i`) se referir as linhas, então o algoritmo percorre todos os elementos de uma coluna antes de ir para a próxima (Figura à direita).

Direção de Preenchimento da Matriz



Impressão na Tela dos Elementos da Matriz

- Segue a mesma ideia do preeenchimento da matriz usando dois laços de repetição aninhados.
- As variáveis contadoras `i` e `j` são os índices da matriz que se referem as linhas e as colunas, respectivamente.
- O elemento `X[i][j]` da matriz se encontra na coluna `j` da linha `i`.

Impressão na Tela dos Elementos da Matriz

Imprimindo uma matriz M com 3 linhas e 5 colunas (Pseudocódigo)

```
PARA i ← 0 ATÉ 2 FAÇA {
     PARA i ← 0 ATÉ 4 FAÇA {
          ESCREVA "Digite o número da posição (",i,",",j,"):"
          ESCREVA X[i,j]
     }
}
```

Imprimindo uma matriz M com 3 linhas e 5 colunas (Linguagem C)

```
for(i = 0; i <= 2; i++) {
     for(i = 0; i <= 4; i++) {
          printf("Digite o número da posição (%d,%d):", i, j);
          printf("%d", X[i][j]);
     }
}
```

Exemplo

Faça um programa que preencha uma matriz de 2 linhas e 3 colunas com números inteiros positivos, encontre e mostre o maior valor contido na matriz e sua localização (linha e coluna).

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

int main()
{
    int M[2][3]; // declaração da matriz com 2 linhas e 3 colunas
    int maior = 0;

    // preenche o vetor com 2*3=6 números inteiros positivos
    for(int i=0; i<2; i++) // laço referente as linhas da matriz
    {
        for(int j=0; j<3; j++) // laço referente as colunas da matriz
        {
            printf("Digite X[%d][%d]: ", i, j);
            scanf("%d", &M[i][j]);

            // verifica se o elemento digitado é maior que o maior armazenado anteriormente
            if(maior < M[i][j])
            {
                maior = M[i][j]; // copia o elemento para a variável maior
            }
        }
    }

    printf("Maior elemento da matriz: %d", maior);

    return 0;
}
```

Dimensões de uma Matriz

- Uma matriz é uma variável composta homogênea multidimensional, ou seja, possui uma ou mais dimensões.
- Logo, vetor é uma matriz unidimensional (apenas 1 dimensão).
- A matriz do exemplo anterior, é uma matriz bidimensional (duas dimensões).
- Para para criar uma matriz com 3 dimensões ou mais, basta adicionar mais pares de colchetes na declaração até completar o número de dimensões desejadas.

Declaração de Matriz (Multidimensional)

Sintaxe (Pseudocódigo)

```
DECLARE nome[dimensao1, dimensao2, dimensao3,
..., dimensaoN] tipo
```

onde:

- `nome` é o nome da variável do tipo matriz.
- `dimensao1` é a quantidade de elementos da 1ª dimensão da matriz.
- `dimensao2` é a quantidade de elementos da 2ª dimensão da matriz.
- `dimensao3` é a quantidade de elementos da 3ª dimensão da matriz.
- `dimensaoN` é a quantidade de elementos da Nª dimensão da matriz.
- `tipo` é o tipo básico dos dados que serão armazenados na matriz.

Declaração de Matriz (Multidimensional)

Sintaxe (Linguagem C)

```
tipo nome[dimensao1][dimensao2] ...[dimensaoN]
```

onde:

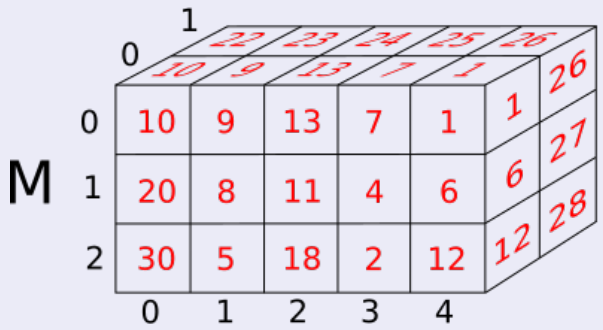
- `nome` é o nome da variável do tipo matriz.
- `dimensao1` é a quantidade de elementos da 1ª dimensão da matriz.
- `dimensao2` é a quantidade de elementos da 2ª dimensão da matriz.
- `dimensao3` é a quantidade de elementos da 3ª dimensão da matriz.
- `dimensaoN` é a quantidade de elementos da Nª dimensão da matriz.
- `tipo` é o tipo básico dos dados que serão armazenados na matriz.

Declaração de Matriz (Tridimensional)

Exemplo (Pseudocódigo)

```
DECLARE M[3, 5, 2] inteiro
```

// matriz chamada `M` que possui 3 linhas, 5 colunas e profundidade 2 (posições armazenadas sequencialmente na memória).



M[0,0,0]←10 M[0,1,0]←9 M[0,2,0]←13
M[1,0,0]←20 M[1,1,0]←8 M[1,2,0]←11
M[2,0,0]←30 M[2,1,0]←5 M[2,2,0]←18
M[0,3,0]←7 M[0,4,0]←1
M[1,3,0]←4 M[1,4,0]←6
M[2,3,0]←2 M[2,4,0]←12

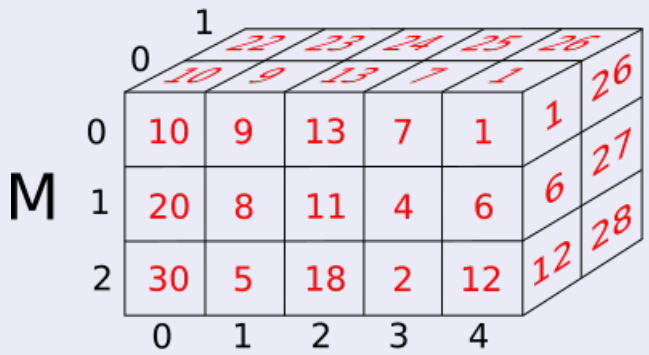
M[0,0,1]←22 M[0,1,1]←23 M[0,2,1]←24
M[1,0,1]←0 M[1,1,1]←0 M[1,2,1]←0
M[2,0,1]←0 M[2,1,1]←0 M[2,2,1]←0
M[0,3,1]←25 M[0,4,1]←26
M[1,3,1]←0 M[1,4,1]←27
M[2,3,1]←0 M[2,4,1]←28

Declaração de Matriz (Tridimensional)

Exemplo (Linguagem C)

```
int X[3][5][2];
```

// matriz chamada `M` que possui 3 linhas, 5 colunas e profundidade 2 (posições armazenadas sequencialmente na memória).



M[0,0,0] = 10 M[0,1,0] = 9 M[0,2,0] = 13
M[1,0,0] = 20 M[1,1,0] = 8 M[1,2,0] = 11
M[2,0,0] = 30 M[2,1,0] = 5 M[2,2,0] = 18
M[0,3,0] = 7 M[0,4,0] = 1
M[1,3,0] = 4 M[1,4,0] = 6
M[2,3,0] = 2 M[2,4,0] = 12

M[0,0,1] = 22 M[0,1,1] = 23 M[0,2,1] = 24
M[1,0,1] = 0 M[1,1,1] = 0 M[1,2,1] = 0
M[2,0,1] = 0 M[2,1,1] = 0 M[2,2,1] = 0
M[0,3,1] = 25 M[0,4,1] = 26
M[1,3,1] = 0 M[1,4,1] = 27
M[2,3,1] = 0 M[2,4,1] = 28

Dicas de Estudo

Sugestões de leitura:

- Capítulo 7 (Matriz) do livro texto (ASCÊNCIO, 2012).
- Capítulo 6 (Vetores e Matrizes) do livro (BACKES, 2013).

Sugestões de exercícios:

- Exercícios do capítulo 7 do livro texto (ASCÊNCIO, 2012).
  - ★ Exercícios Resolvidos: 2, 3, 10, 14, 18 e 24.
- Exercícios do capítulo 6 do livro (BACKES, 2013).
  - ★ Exercícios: 3, 4, 5 e 9.
- Lista de Exercícios 7 (Estruturas de Dados Básicas: Matriz).

Bibliografia

1. ASCÊNCIO, A. F. G.; CAMPOS, E. A. V. ***Fundamentos da Programação de Computadores***. 2012.
2. BACKES, A. ***Linguagem C: completa e descomplicada***. 2013.
3. PAES, R. B. ***Introdução à Programação com a Linguagem C: Aprenda a resolver problemas com uma abordagem prática***. 2016.

Obrigada pela atenção!