Pontifícia Universidade Católica de Minas Gerais

ICEI – Instituto de Ciências Exatas e Informática
DCC – Departamento de Ciência da Computação
***Campus* Lourdes**
Bacharelado em Ciência da Computação

Algoritmos e Estruturas de Dados I
Professor: Lúcio Mauro Pereira
Lista de Exercícios nº 26
21 de outubro de 2024

Arranjos bidimensionais (matrizes) e uma introdução às *strings*

Estudar:

Obra: Fundamentos da Programação de Computadores. Autora: Ana Ascêncio
Estudar o capítulo 6 – Vetor
Estudar o capítulo 7 - Matriz

Obra: C: como programar. 8ed. Autor: Deitel.
Estudar o Capítulo 6: **Arrays**

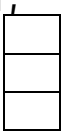
Para cada problema proposto neste caderno de exercícios:

- *Elaborar um modelo de solução. Expressá-lo através de fluxograma e/ou texto estruturado – algoritmo.*
- *Codificar a solução através da linguagem C.*
- *Fique à vontade para testar as funções criadas a partir da função principal.*

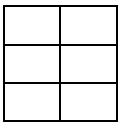
Revisitando a introdução às matrizes bidimensionais:

Um arranjo pode representar uma coleção de dados organizada em diferentes dimensões. Um vetor pode ser entendido como uma matriz de uma única dimensão. No exemplo abaixo, é declarado uma matriz, de uma única dimensão, de tamanho igual a três:

```
float A[3];
```



Considere, por exemplo, uma matriz de dimensão *(3x2)*, isto é, três linhas e duas colunas.



Para acrescentar essa segunda dimensão, em C, basta inserir um segundo par de colchetes:

```
float A[3][2];
```

Na instrução abaixo, a matriz é declarada e inicializada:

```
float A[][2]=  { 10, 11, 12, 13, 14, 15 };
```

10	11
12	13
14	15

***** *Observe ser necessário informar o número de colunas. É ele quem expressa a partir de quantos elementos o próximo deve ser interpretado como o primeiro elemento da próxima linha.*

Para o exemplo abaixo e em toda esta lista, considere o número de linhas e o número de colunas declarados nas seguintes constantes globais: MAX_LIN e MAX_COL
Por exemplo, para os cenários apresentados acima:

```
const int MAX_LIN = 3;
const int MAX_COL = 2;
```

No exemplo abaixo, a função recebe uma matriz bidimensional de reais e a escreve na tela:

```
void escreveMatriz(float Matriz[][MAX_COL])
{
    //Varrendo as linhas da matriz
    for(int i=0; i< MAX_LIN; i++){

        //Para cada linha, varrendo suas colunas
        for(int j=0; j< MAX_COL; j++){
            printf("\tM[%i][%i]=  %f", i, j, Matriz[i][j]);
        }//fim for(j)
        //Ao finalizar todas as colunas, avançar uma linha também na tela
        printf("\n");

    }//fim for(i)

}//fim escreveMatriz()
```

Questões iniciais:

Considere a função principal abaixo:

```
const int MAX_LIN = 2;
const int MAX_COL = 4;

int main() {

      float M1[MAX_LIN][MAX_COL];
      leMatriz(M1);

      float M2[MAX_LIN][MAX_COL];
      leMatriz(M2);

      printf("\n\nEscrevendo a primeira Matriz:\n");
      escreveMatriz(M1);

      printf("\n\nEscrevendo a segunda Matriz:\n");
      escreveMatriz(M2);

      if( iguais(M1, M2) ) printf("\nMatrizes iguais!");
      else                 printf("\nMatrizes diferentes!");

      return 0;
}
```

1. Implemente a função para realizar a leitura dos valores para uma matriz bidimensional.
Argumento: o endereço para uma matriz de reais
Valor gerado: nenhum

2. Implemente a função que verifica se duas matrizes bidimensionais são iguais ou não.
Argumentos: os endereços das duas matrizes a serem comparadas
Valor gerado: *true*, caso sejam as matrizes iguais, ou *false*, caso contrário

Questões:

Desenvolva as questões abaixo. Em seguida, teste-as a partir da função principal. Busque testar diferentes cenários para assegurar o "grau de corretude" de sua solução.

3. Implemente uma função que identifique o maior valor presente na linha *i* de uma matriz de reais, sendo *i* também um argumento da função.
Argumentos: o endereço da matriz de reais e um valor inteiro relativo à linha chave da pesquisa
Valor gerado: um valor real correspondente ao maior valor encontrado na linha

4. Para uma matriz quadrada de reais, construa uma função que calcule a diferença entre o maior valor presente acima de sua diagonal principal e o maior valor presente abaixo da diagonal principal.
Argumento: o endereço da matriz quadrada de reais
Valor gerado: um valor real relativo à diferença entre os maiores valores acima e abaixo da diagonal

5. Implemente uma função que receba duas matrizes de reais. A função deverá copiar a primeira matriz na segunda.
Argumentos: os endereços para duas matrizes de reais
Valor gerado: nenhum

6. Implemente uma função que receba duas matrizes de reais. A função deverá calcular a matriz transposta da primeira, armazenando-a na segunda matriz.
* Planeje, com cuidado, as dimensões de ambas as matrizes parametrizadas
* Planeje, com cuidado, os parâmetros e os valores gerados

7. Construa uma função que receba duas *strings.* A função deverá retornar se são elas absolutamente iguais ou não. Por exemplo "Ana Maria" não é igual a "Ana Maria da Silva". Por outro lado, a função não deverá ser sensível a maiúsculo/minúsculo.
Argumentos: as duas *strings* a serem comparadas
Valor gerado: *true,* caso sejam as duas *strings* iguais, ou *false,* caso contrário

8. Problema: escrever uma *string* de forma invertida. Por exemplo, a palavra ROMA deverá ser escrita como AMOR.
Argumento: a *string* a ser escrita
Valor gerado: nenhum