Pontifícia Universidade Católica de Minas Gerais

ICEI – Instituto de Ciências Exatas e Informática
DCC – Departamento de Ciência da Computação
Campus Belo Horizonte – Unidade Coração Eucarístico
Bacharelado em Ciência da Computação

Algoritmos e Estruturas de Dados I
Professor: Lúcio Mauro Pereira
Lista de Exercícios nº 22
3 de maio de 2023

Introdução aos Arranjos Bidimensionais (continuação)

Estudar

Obra: Fundamentos da Programação de Computadores. Autora: Ana Ascêncio
Disponível na biblioteca da PUC Minas de forma física e *e-book*.
Capítulo 7: Matriz

Obra: C: como programar. 8ed. Autor: Deitel.
Disponível na biblioteca da PUC Minas de forma física e *e-book*.
Capítulo 6: Arrays

Introdução

Analise atentamente o código abaixo a partir de sua experiência nas aulas anteriores:

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

// Constantes globais para declarar a dimensão da matriz
const int NUM_LIN = 3;
const int NUM_COL = 2;

// Função para ler os valores de uma matriz
void leMatriz(float M[][NUM_COL])
{
    for(int i=0; i<NUM_LIN; i++){
       for(int j=0; j<NUM_COL; j++){
           printf("M[%i][%i]: ", i+1, j+1) ;
           scanf("%f", &M[i][j]);
       }
    }
}

// Função para escrever os valores de uma matriz
void escreveMatriz(float M[NUM_LIN][NUM_COL])
{
    for(int i=0; i<NUM_LIN; i++){
       for(int j=0; j<NUM_COL; j++){
           printf("\nM[%i][%i]= %f", i+1, j+1, M[i][j]) ;
       }
    }
}
```

```
// versão ruim (cara)
bool iguais(float A[][NUM_COL], float B[][NUM_COL])
{
    bool saoIguais= true;

    int i=0;
    for(int i=0; i<NUM_LIN; i++)
    {
        for(int j=0; j<NUM_COL; j++)
        {
            if(A[i][j] != B[i][j]) saoIguais= false;
        }
    }
    return saoIguais;
}

int main() {
  float M1[NUM_LIN][ NUM_COL];
  leMatriz(M1);

  float M2[NUM_LIN][ NUM_COL];
  leMatriz(M2);

  printf("\n\nEscreve primeira matriz\n");
  escreveMatriz(M1);

  printf("\n\nEscreve segunda matriz\n");
  escreveMatriz(M2);

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

  return 0;
}
```

Questões:

1) Durante a aula foi discutido o custo de tempo do algoritmo apresentado para verificar a igualdade entre duas matrizes. Proponha sua versão para uma função eficiente, utilizando alguma estratégia que reduza seu custo de tempo.
* Note que os demais elementos da matriz continuam sendo visitados mesmo após encontrar um elemento diferente. Isto é, o custo é levado, necessariamente, ao pior caso.

2) Construa uma função que verifique o número de ocorrências da chave de pesquisa em uma matriz de reais.
Argumentos da função: uma matriz de reais e o valor da chave de pesquisa
Retorno: um valor inteiro relativo ao número de ocorrência da chave na pesquisa

3) Construa uma função que implemente um algoritmo de pesquisa em uma matriz de reais.
Argumentos: Uma matriz de reais e o valor chave da pesquisa.
Retorno: Verdadeiro, se a chave existir na matriz, ou falso, caso contrário.
* Busque uma estratégia eficiente.

4) Construa uma função que calcule a média dos valores que compõe a diagonal principal de uma matriz quadrada.
Argumento: Uma matriz quadrada de reais, de dimensão NUM_LIN x NUM_LIN.
Retorno: Um valor real relativo ao resultado do cálculo da média da diagonal principal.