ELE078 - Programação Orientada a Objetos

Atividade Prática 01

Parte 01: Manipulando Programas Simples em C++: ¶

Dica: usar as classes *fstream* , *vector* e *string*.

1.1. Escreva um programa que abra um arquivo e conte o número de espaços em branco do arquivo.

In [ ]:

```
#include <string>
#include <fstream>
#include <iostream>
using namespace std;

...
```

1.2. Dado um vetor de números inteiros positivos aleatórios entrados via teclado (número negativo indica fim da entrada dos dados), faça um programa utilizando a classe template vector para comprimir o vetor suprimindo as repetições de números vizinhos através da contagem do número de repetições de cada um da seguinte forma:

- Vetor de entrada: 1 1 1 4 1 1 4 4 25 67 67 67 67 2 2
- Vetor de saída: 3 1 1 4 2 1 2 4 1 25 4 67 2 2

In [ ]:

```
#include <iostream>
#include <vector>
#include <cstdio>
using namespace std;

...
```

Parte 02: *Structs* and Alocação Dinâmica de Memória:

O código a seguir tem como objetivo implementar um tipo abstrato de dados (TAD) Matriz referente a um *array* bidimensional de elementos do tipo *double*. Os principais atributos da Matriz são:

- número de linhas: nLinhas (int)
- número de colunas nColunas (int)
- array 2D que armazena os elementos: m (double pointer)

Com base nesse código inicial fornecido, implemente as demais funções cuja definição encontram-se nas células a seguir:

In [ ]:

```
#include <iostream>
using namespace std;

typedef struct{
    double** m;
    int nLinhas;
    int nColunas;
}Matriz;
```

2.1. função para inicializar a matriz *X* com todos os elementos iguais a *elem* (double). A função deve fazer alocação dinâmica de memória com base nos argumentos *ls* e *cs*, que representam o número de linhas e o número de colunas, respectivamente.

In [ ]:

```
void inicializaMatriz(Matriz &X, int ls, int cs, const double elem){
...
}
```

2.2. função para liberar a memória da Matrix *X*.

In [ ]:

```
void apagaMatriz(Matriz &X){
...
}
```

2.3. função que retorna uma matriz transposta de *X*.

In [ ]:

```
Matriz transposta(Matriz &X){
...
}
```

2.4. função para multiplicar a matriz *X* por um valor constante *k* (tipo double). Retorna uma nova matriz.

In [ ]:

```
Matriz multiplica_por_cte(Matriz &X, double k){
...
}
```

2.5. função para imprimir os elementos da matriz *X*.

In [ ]:

```
void imprimeMatriz(Matriz &X){
...
}
```

2.6. Programa Teste:

As funções implementadas devem ser testadas com o programa principal a seguir.

In [ ]:

```
int main()
{
    Matriz A, B, T, R;

    cout << "A:: " << endl;
    inicializaMatriz(A,2,3,7.0);
    imprimeMatriz(A);

    cout << "Transposta de A:: " << endl;
    T = transposta(A);
    imprimeMatriz(T);
    apagaMatriz(T);

    cout << endl << "B:: " << endl;
    inicializaMatriz(B,4,4,5.0);
    imprimeMatriz(B);

    cout << endl << "R = k*B " << endl;
    R = multiplica_por_cte(B,5);
    imprimeMatriz(R);

    apagaMatriz(B);
    apagaMatriz(R);

    return 0;
}
```