ELE078 - PROGRAMAÇÃO ORIENTADA A OBJETOS

Trabalho Prático

valor: 25 Pontos

I – As seguintes regras devem ser observadas na confecção do trabalho:

1) **O trabalho é individual**. É permitido discutir os problemas e estratégias de solução com outros alunos, mas quando se tratar de escrever ou implementar computacionalmente as soluções, isto deve ser feito individualmente. Utilizar o trabalho dos outros, como se fosse seu, é plágio. Desonestidade acadêmica será punida com severidade.

2) **Forma de entrega:** O trabalho deve ser entregue em formato digital por meio do *Moodle*. Utilizar a opção *"Link para Envio do TP".* Anexe um único arquivo .zip (ou .rar) contendo todos os arquivos do trabalho (códigos-fonte, executáveis, documentação, etc.). O nome do arquivo .zip (ou .rar) a ser enviado deve ser o nome completo do aluno. Exemplo:

***Cristiano_Leite_Castro.zip*.**

3) O link para envio do TP no Moodle terá uma tolerância de 24 horas para recebimentos dos trabalhos. Após esse período, os trabalhos não serão recebidos.

II – Tarefas:

Crie uma classe *Matrix* que permita que sejam feitas operações matemáticas similares às do *Matlab.* A classe deve ser implementada de forma a permitir que os usuários manipulem seus índices (linhas e colunas) começando de 1 (um) e não de 0 (zero). Os elementos devem ser armazenados com estruturas criadas dinamicamente na memória através do uso de ponteiros duplos (ou simples).

A classe *Matrix* deve ser definida como uma classe *template* de forma a armazenar elementos de tipos quaisquer (int , long , double , etc.).

As checagens de erro (consistencia) feitas pelos métodos da classe devem considerar o lançamento de exceções através da cláusula *throw*. O programa cliente main() que testa a classe deve tratar as possíveis exceções lançadas, usando as cláusulas *try* e *catch(...)*

Use os recursos aprendidos até o momento para tornar o seu código eficiente:

- sobrecarga de construtores e funções membro;
- argumentos *default* em construtores e funções membro, funções membro *inline*, passagem de argumentos por referência, etc.
- argumentos e funções membro *constantes* (*const*) sempre que possível.
- sobrecarga de operadores, etc.

Código exemplo para a interface da Classe *Matrix*: (é só uma sugetão...)

```
// matrix.h
...
template <class TValor>
class Matrix {
      private:
            TValor** m;    // m é um array 2D (pointer of pointer)
            int rows;  // numero de linhas
            int cols;  //  numero de colunas
      public:
            // Construtores
            Matrix();                     //  construtor default
            Matrix(int rows, int cols, const TValor &value = 0.0)
            Matrix(ifstream &myFile)
            Matrix(const Matrix& that)     // construtor copia
            ~Matrix();                    // destrutor

            // basic getters
            int getRows() const;
            int getCols() const;
            TValor get(int row, int col) const;
            ...
            // exemplo de checagem em que uma Exceção é lançada
            TValor& Matrix::operator()(int row, int col) {
                  if (rows <= row || cols <= col)
                        throw std::invalid_argument("Index out of bounds.")
                  return m[row][col];
            }
};
```

Programa Teste: Complemente o programa teste a seguir para que todas as operações implementadas na classe possam ser testadas.

```
int main()
{
    ifstream in("myMatrix.txt");
    Matrix Y;
    Matrix X(3,1), A(3,3), C(3,3);
    Matrix Z(3,2,7.0);
    Matrix W(in);

    // operadores a serem implementados em sua classe:

    A(2,1) = 10;                    // altera o valor de uma posição de A
    C = A + A;                      // Soma
    C -= A;                         // Subtração
    A = C - A;                      // Subtração
    A += A;                         // Soma
    A = ~C;                         // A é igual a transposta de C
    X *= 2;                         // multiplicação por uma constante
    C = A*X;                        // multiplicação de matrizes
    C *= X;                         // multiplicação de matrizes
    if (A == C)                     // verifica a igualdade entre A e C
    if(X != Y)                      // verifica a desigualdade entre A e C
    cout << Z << endl;              // impressão de matrizes
    cin >> W                        // leitura de dados para dentro da matriz Y

    return 0;
}
```