```cpp
#ifndef METNUM_TP2_MATRIX_H
#define METNUM_TP2_MATRIX_H

#include <iostream>
#include <bitset>
#include <vector>

/*
* Matriz.
*/
class Matrix {
    friend std::ostream &operator<<(std::ostream &, const Matrix &);
    friend std::istream &operator>>(std::istream &, Matrix &);
public:
    Matrix(const Matrix &m);

    Matrix(Matrix &&m) : N(m.rows()), M(m.columns()), matrix(std::move(m.matrix)) {
        std::cerr << "Llamado al constructor por movimiento de Matriz " << this->rows() << "x" << this->columns() << std::endl;

        if (this->matrix.size() > 0) {
            std::cerr << "Dimensiones del vector de salida: " << this->matrix.size() << "x" << this->matrix[0].size() << std::endl;
        }
    }

    template <std::size_t K>
    Matrix(const Matrix &m, const std::bitset<K> &filter)
            : N((int)filter.count()), M(m.columns()), matrix((int)filter.count(), std::vector<double>(m.columns(), 0.0)) {
        if (K != (std::size_t)m.rows()) {
            throw new std::out_of_range("Filtro de bitset para Matriz con entradas insuficientes");
        }

        if (this->rows() < 0 || this->columns() < 0) {
            throw new std::invalid_argument("Dimensiones de Matriz inválidas");
        }

        std::cerr << "Filtrando matriz de " << m.rows() << "x" << m.columns() << " en " << this->rows() << "x" << this->columns() << std::endl;

        int last = 0;

        for (int i = 0; i < m.rows(); ++i) {
            if (filter.test((std::size_t) i)) {
                std::copy(m.matrix[i].begin(), m.matrix[i].begin() + m.columns(), this->matrix[last].begin());
                last++;
            }
        }

        if (this->matrix.size() > 0) {
            std::cerr << "Dimensiones del vector de salida: " << this->matrix.size() << "x" << this->matrix[0].size() << std::endl;
        }
    }

    Matrix(int N, int M);

    int inline rows() const {
        return this->N;
    }

    int inline columns() const {
        return this->M;
    }

    inline double &operator()(const int &i, const int &j) {
        #ifdef DEBUG
        if (0 > i || 0 > j || i >= this->rows() || j >= this->columns()) {
            throw new std::out_of_range("Index access out of range");
        }
        #endif

        return this->matrix[i][j];
    }

    inline const double &operator()(const int &i, const int &j) const {
```

```
73          #ifdef DEBUG
74          if (0 > i || 0 > j || i >= this->rows() || j >= this->columns()) {
75              throw new std::out_of_range("Index access out of range");
76          }
77          #endif
78
79          return this->matrix[i][j];
80      }
81
82      Matrix & operator=(const Matrix &m);
83      bool operator==(const Matrix &m) const;
84      bool operator!=(const Matrix &m) const;
85      Matrix & operator+=(const Matrix &m);
86      Matrix & operator*=(const double &c);
87  private:
88      // Matrix
89      int N;
90      int M;
91      std::vector< std::vector<double> > matrix;
92  };
93
94  std::ostream &operator<<(std::ostream &, const Matrix &);
95  std::istream &operator>>(std::istream &, Matrix &);
96  Matrix operator+(const Matrix &m, const Matrix &n);
97  Matrix operator*(const Matrix &m, const double &c);
98  Matrix operator*(const Matrix &m, const Matrix &n);
99
100 #endif //METNUM_TP2_MATRIX_H
```