

```

1  #include <stdexcept>
2  #include "Matrix.h"
3
4  Matrix::Matrix(const Matrix &m) : N(m.rows()), M(m.columns()), matrix(m.matrix) {
5      std::cerr << "Copiando matriz de " << this->rows() << "x" << this->columns() << std::endl;
6  }
7
8  Matrix::Matrix(int N, int M)
9      : N(N), M(M), matrix(N, std::vector<double>(M, 0.0)) {
10     if (this->rows() < 0 || this->columns() < 0) {
11         throw new std::out_of_range("Invalid matrix dimension");
12     }
13
14     std::cerr << "Creando matriz de " << this->rows() << "x" << this->columns() << std::endl;
15
16     if (this->matrix.size() > 0) {
17         std::cerr << "Dimensiones del vector de salida: " << this->matrix.size() << "x" << this->matrix[0].size() << std::endl;
18     }
19 }
20
21 Matrix &Matrix::operator=(const Matrix &m) {
22     if (*this != m) {
23         // Poner información de la representación interna
24         this->N = m.rows();
25         this->M = m.columns();
26
27         // Crear matriz nueva
28         this->matrix = m.matrix;
29     }
30
31     return *this;
32 }
33
34 bool Matrix::operator==(const Matrix &m) const {
35     if (this->rows() != m.rows() || this->columns() != m.columns()) {
36         return false;
37     } else {
38         for (int i = 0; i < this->rows(); i++) {
39             for (int j = 0; j < this->columns(); j++) {
40                 if ((*this)(i, j) != m(i, j)) {
41                     return false;
42                 }
43             }
44         }
45
46         return true;
47     }
48 }
49
50 bool Matrix::operator!=(const Matrix &m) const {
51     return !(*this == m);
52 }
53
54 Matrix &Matrix::operator+=(const Matrix &m) {
55     if (this->rows() == m.rows() && this->columns() == m.columns()) {
56         for (int i = 0; i < m.rows(); ++i) {
57             for (int j = 0; j < m.columns(); ++j) {
58                 this->matrix[i][j] += m.matrix[i][j];
59             }
60         }
61     } else {
62         throw new std::out_of_range("Different dimensions for matrix sum");
63     }
64
65     return *this;
66 }
67
68 Matrix &Matrix::operator*=(const double &c) {
69     for (int i = 0; i < this->rows(); ++i) {
70         for (int j = 0; j < this->columns(); ++j) {
71             this->matrix[i][j] *= c;
72         }
73     }
74
75     return *this;
76 }
77

```

```

78  std::ostream &operator<<(std::ostream &os, const Matrix &m) {
79      for (int i = 0; i < m.rows(); ++i) {
80          for (int j = 0; j < m.columns(); ++j) {
81              os << (double)m(i, j) << " ";
82          }
83      }
84      os << std::endl;
85  }
86
87  os << std::endl;
88
89  return os;
90 }
91
92  std::istream &operator>>(std::istream &is, Matrix &m) {
93      for (int i = 0; i < m.rows(); ++i) {
94          for (int j = 0; j < m.columns(); ++j) {
95              is >> m(i, j);
96          }
97      }
98
99      return is;
100 }
101
102  Matrix operator+(const Matrix &m, const Matrix &n) {
103      Matrix output(m);
104      output += n;
105      return output;
106 }
107
108  Matrix operator*(const Matrix &m, const double &c) {
109      Matrix output(m);
110      output *= c;
111      return output;
112 }
113
114  Matrix operator*(const Matrix &m, const Matrix &n) {
115      if (m.columns() == n.rows()) {
116          Matrix output(m.rows(), n.columns());
117
118          for (int i = 0; i < output.columns(); ++i) {
119              for (int j = 0; j < output.columns(); ++j) {
120                  for (int k = 0; k < m.columns(); ++k) {
121                      output(i, j) += m(i, k) * n(k, j);
122                  }
123              }
124          }
125
126          return output;
127      } else {
128          throw new std::out_of_range("Matrix product between incompatible matrices.");
129      }
130 }

```