

matrix.h

```
1: #include <iostream>
2: #include <stdexcept>
3: #include <vector>
4: using namespace std;
5:
6: class matrix {
7: private:
8:     int _nr;           /* number of rows */
9:     int _nc;           /* number of columns */
10:    vector<double> data; /* underlying data storage */
11:
12: public:
13:    matrix() : _nr(0), _nc(0), data() {};
14:
15:    matrix(int numRows, int numColumns, double value=0)
16:        : _nr(numRows), _nc(numColumns), data(numRows*numColumns, value) {}
17:
18:    int numRows() const {
19:        return _nr;
20:    }
21:
22:    int numColumns() const {
23:        return _nc;
24:    }
25:
26:    matrix size() const {
27:        matrix result(1,2);
28:        result(0,0) = numRows();
29:        result(0,1) = numColumns();
30:        return result;
31:    }
32:
33:    bool operator==(const matrix &other) const {
34:        return (_nr == other._nr && _nc == other._nc && data == other.data);
35:    }
36:
37:    bool operator!=(const matrix &other) const {
38:        return !(*this == other);
39:    }
40:
41:    // provides read-only access to a matrix entry
42:    double operator()(int r, int c) const {
43:        if (r < 0 || r >= _nr || c < 0 || c >= _nc)
44:            throw out_of_range("Invalid indices for matrix");
45:
46:        return data[r + c * _nr]; // column-major
47:    }
48:
49:    // provides write-access to a matrix entry
50:    double& operator()(int r, int c) {
51:        if (r < 0 || r >= _nr || c < 0 || c >= _nc)
52:            throw out_of_range("Invalid indices for matrix");
53:
54:        return data[r + c * _nr]; // column-major
55:    }
```

matrix.h

```
56:  //-----
57:  // addition
58:  //-----
59:  matrix operator+(const matrix& other) const { // produce sum of two matrices
60:      if (_nr != other._nr && _nc != other._nc)
61:          throw invalid_argument("Matrix dimensions must agree.");
62:
63:      matrix result = matrix(*this);
64:      for (int r=0; r < _nr; r++)
65:          for (int c=0; c < _nc; c++)
66:              result(r,c) += other(r,c);
67:      return result;
68:  }
69:
70:  matrix operator+(double scalar) const { // add scalar to all elements
71:      matrix result = matrix(*this);
72:      for (int r=0; r < _nr; r++)
73:          for (int c=0; c < _nc; c++)
74:              result(r,c) += scalar;
75:      return result;
76:  }
77:
78:
79:  //-----
80:  // multiplication
81:  //-----
82:  matrix operator*(double scalar) const { // multiply each element by scalar
83:      matrix result = matrix(*this);
84:      for (int r=0; r < _nr; r++)
85:          for (int c=0; c < _nc; c++)
86:              result(r,c) *= scalar;
87:      return result;
88:  }
89:
90:  matrix operator*(const matrix& other) const { // matrix multiplicaiton
91:      if (_nc != other._nr)
92:          throw invalid_argument("Inner matrix dimensions must agree.");
93:
94:      // *** rest of implementation missing ***
95:
96:
97:
98:  }
99: };
100:
101: //-----
102: // define additional support for reading/writing matrices
103: // (implementations are given in matrixio.cpp)
104: //-----
105: ostream& operator<<(ostream& out, const matrix& m);
106: istream& operator>>(istream& in, matrix& m);
```