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

```
  //-----------------------------------------------
  // addition
  //-----------------------------------------------
  matrix operator+(const matrix& other) const {  // produce sum of two matrices
    if (_nr != other._nr && _nc != other._nc)
      throw invalid_argument("Matrix dimensions must agree.");

    matrix result(_nr, _nc);
    for (int r=0; r < _nr; r++)
      for (int c=0; c < _nc; c++)
        result(r,c) = (*this)(r,c) + other(r,c);
    return result;
  }

  matrix operator+(double scalar) const {   // add scalar to all elements
    matrix result(_nr, _nc);
    for (int r=0; r < _nr; r++)
      for (int c=0; c < _nc; c++)
        result(r,c) = (*this)(r,c) + scalar;
    return result;
  }


  //-----------------------------------------------
  // multiplication
  //-----------------------------------------------
  matrix operator*(double scalar) const {   // multiply each element by scalar
    matrix result = matrix(*this);
    for (int r=0; r < _nr; r++)
      for (int c=0; c < _nc; c++)
        result(r,c) *= scalar;
    return result;
  }

  matrix operator*(const matrix& other) const {  // matrix multiplicaiton
    if (_nc != other._nr)
      throw invalid_argument("Inner matrix dimensions must agree.");

    //   *** rest of implementation missing ***
  }
};

//-----------------------------------------------------
// define additional support for reading/writing matrices
// (implementations are given in matrixio.cpp)
//-----------------------------------------------------
ostream& operator<<(ostream& out, const matrix& m);
istream& operator>>(istream& in, matrix& m);
```