Exercises week 1

Klaas Isaac Bijlsma
s2394480

David Vroom
s2309939

November 18, 2017

Exercise 1

Attain some familiarity with the way functions are selected from namespaces

We used the following code,

Exercise 2

ziet ie dit?

Exercise 3

Learn to implement index operators

The Matrix class that is used here, is derived from the solutions of excercise 64. We used the following code,

matrix/matrix.h

```
#ifndef INCLUDED_MATRIX_
#define INCLUDED_MATRIX_

#include <iosfwd>
#include <initializer_list>

class Matrix
{
    size_t d_nRows = 0;
    size_t d_nCols = 0;
    double *d_data = 0;                     // in fact R x C matrix

    public:
        typedef std::initializer_list<
            std::initializer_list<double>> IniList;

        Matrix() = default;
        Matrix(size_t nRows, size_t nCols);         // 1
        Matrix(Matrix const &other);                // 2
        Matrix(Matrix &&tmp);                       // 3
        Matrix(IniList inilist);                    // 4

        ~Matrix();

        Matrix &operator=(Matrix const &rhs);
        Matrix &operator=(Matrix &&tmp);

        size_t nRows() const;
        size_t nCols() const;
        size_t size() const;            // nRows * nCols

        static Matrix identity(size_t dim);
```

```
33
34          Matrix &tr();                   // transpose (must be square)
35          Matrix transpose() const;       // any dim.
36
37          void swap(Matrix &other);
38
39              // exercise 3
40              // ==========
41          double *operator[](size_t index);
42          double *operator[](size_t index) const;
43
44              // exercise 4
45              // ==========
46          friend Matrix operator+(Matrix const &lhs, Matrix const &rhs);
47          friend Matrix operator+(Matrix &&lhs, Matrix const &rhs);
48          Matrix &operator+=(Matrix const &other) &;      // 1
49          Matrix operator+=(Matrix const &other) &&;      // 2
50
51              // exercise 5
52              // ==========
53          friend std::ostream &operator<<(
54                  std::ostream &out, Matrix const &matrix);
55
56  private:
57          double &el(size_t row, size_t col) const;
58          void transpose(double *dest) const;
59
60              // exercise 3
61              // ==========               // private backdoor
62          double *operatorIndex(size_t index) const;
63
64              // exercise 4
65              // ==========
66          void add(Matrix const &rhs);
67  };
68
69  inline size_t Matrix::nCols() const
70  {
71      return d_nCols;
72  }
73
```

```
74  inline size_t Matrix::nRows() const
75  {
76      return d_nRows;
77  }
78
79  inline size_t Matrix::size() const
80  {
81      return d_nRows * d_nCols;
82  }
83
84  inline double &Matrix::el(size_t row, size_t col) const
85  {
86      return d_data[row * d_nCols + col];
87  }
88
89      // exercise 3
90      // ==========
91  inline double *Matrix::operatorIndex(size_t index) const
92  {
93      return d_data + index * d_nCols;
94  }
95
96  inline double *Matrix::operator[](size_t index)
97  {
98      return operatorIndex(index);
99  }
100
101 inline double *Matrix::operator[](size_t index) const
102 {
103     return operatorIndex(index);
104 }
105
106     // exercise 4
107     // ==========
108 Matrix operator+(Matrix const &lhs, Matrix const &rhs);     // 1
109 Matrix operator+(Matrix &&lhs, Matrix const &rhs);          // 2
110
111     // exercise 5
112     // ==========
113 std::ostream &operator<<(std::ostream &out, Matrix const &matrix);
114
```

```
115 #endif
```

Exercise 4

Learn to implement and spot opportunities for overloaded operators

The header is shown in exercise 3, the implementations of the added functions are shown below:

matrix/add.cc

```
1  #include "matrix.ih"
2
3  void Matrix::add(Matrix const &rhs)
4  {
5          if (rhs.d_nCols != d_nCols or rhs.d_nRows != d_nRows)
6          {
7                  cerr << "Warning: Matrices have differnt size, "
8                          "so cannot be added!\n";
9                  exit(1);
10         }
11
12     for (size_t idx = size(); idx--; )
13         d_data[idx] += rhs.d_data[idx];
14 }
```

matrix/operatoradd.cc

```
1  #include "matrix.ih"
2
3  Matrix operator+(Matrix const &lhs, Matrix const &rhs)
4  {
5          Matrix tmp{ lhs };
6          tmp.add(rhs);
7          return tmp;
8  }
```

matrix/operatoradd2.cc

```
1  #include "matrix.ih"
2
```

```
3 Matrix operator+(Matrix &&lhs, Matrix const &rhs)
4 {
5         lhs.add(rhs);
6         return move(lhs);
7 }
```

matrix/operatorcompadd1.cc

```
1 #include "matrix.ih"
2
3 Matrix &Matrix::operator+=(Matrix const &other) &
4 {
5         Matrix tmp{ *this };
6         tmp.add(other);
7         swap(tmp);
8         return *this;
9 }
```

matrix/operatorcompadd2.cc

```
1 #include "matrix.ih"
2
3 Matrix Matrix::operator+=(Matrix const &other) &&
4 {
5         add(other);
6         return move(*this);
7 }
```

Exercise 5

Exercise 6

Exercise 7

Exercise 8

Exercise 9

Exercise 10