

# *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 exercise 64.  
We used the following code,

```
matrix/matrix.h
1 #ifndef INCLUDED_MATRIX_
2 #define INCLUDED_MATRIX_
3
4 #include <iosfwd>
5 #include <initializer_list>
6
7 class Matrix
8 {
9     size_t d_nRows = 0;
10    size_t d_nCols = 0;
11    double *d_data = 0;           // in fact R x C matrix
12
13    public:
14        typedef std::initializer_list<
15            std::initializer_list<double>> IniList;
16
17        Matrix() = default;
18        Matrix(size_t nRows, size_t nCols);           // 1
19        Matrix(Matrix const &other);                 // 2
20        Matrix(Matrix &&tmp);                         // 3
21        Matrix(IniList inilist);                     // 4
22
23        ~Matrix();
24
25        Matrix &operator=(Matrix const &rhs);
26        Matrix &operator=(Matrix &&tmp);
27
28        size_t nRows() const;
29        size_t nCols() const;
30        size_t size() const;                         // nRows * nCols
31
32        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
47     friend Matrix operator+(Matrix const &lhs, Matrix const &rhs);
48     friend Matrix operator+(Matrix &&lhs, Matrix const &rhs);
49     Matrix &operator+=(Matrix const &other) &;      // 1
50     Matrix operator+=(Matrix const &other) &&;      // 2
51
52 private:
53     double &el(size_t row, size_t col) const;
54     void transpose(double *dest) const;
55
56         // exercise 3
57         // =====                                // private backdoor
58     double *operatorIndex(size_t index) const;
59
60         // exercise 4
61         // =====
62     void add(Matrix const &rhs);
63 };
64
65 inline size_t Matrix::nCols() const
66 {
67     return d_nCols;
68 }
69
70 inline size_t Matrix::nRows() const
71 {
72     return d_nRows;
73 }

```

```

74
75 inline size_t Matrix::size() const
76 {
77     return d_nRows * d_nCols;
78 }
79
80 inline double &Matrix::el(size_t row, size_t col) const
81 {
82     return d_data[row * d_nCols + col];
83 }
84
85     // exercise 3
86     // =====
87 inline double *Matrix::operatorIndex(size_t index) const
88 {
89     return d_data + index * d_nCols;
90 }
91
92 inline double *Matrix::operator[](size_t index)
93 {
94     return operatorIndex(index);
95 }
96
97 inline double *Matrix::operator[](size_t index) const
98 {
99     return operatorIndex(index);
100 }
101
102     // exercise 4
103     // =====
104 Matrix operator+(Matrix const &lhs, Matrix const &rhs);           // 1
105 Matrix operator+(Matrix &&lhs, Matrix const &rhs);                 // 2
106
107 #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 |    size_t matSize = size();
12 |    for (size_t idx = 0; idx != matSize; ++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