Effective and Modern C++ Programming

Lab 5 – Class templates

Exercise 1. Class template Matrix

Implement a class template Matrix<T,N,M> that has three template parameters:

- T - type of matrix elements,
- N - number of rows,
- M - number of columns.

Coefficients should be stored in static array.
Implement

- constructors: default (set coordinates to 0), copy, from initialization list, without initialization,
- access operator to the coefficients by a(i,j). Indices start from 1.
- addition and multiplication operators. Operations on matrices with incompatible dimensions should be detected during compilation.

Class should be implemented in file Matix.h.

Question: Is it possible to implement move semantics?

Exercise 2. Partial specialization Matrix<T,0,0>

Implement a partial specialization of template Matrix for both sizes equal to 0. It should store matrix coefficients in a dynamical array. The number of rows and columns are provided in the constructor. (As a base you can use implementation from Lab3).

Make interfaces of primary template and specialization the same or at least very similar (you need to extend also primary template).

Operators + and * should throw an exception if sizes do not match. Exception should be of user defined type `MatrixException` that is derived from runtime_error.

Question: Is it possible to implement move semantics here?

Exercise 3. Mixed operators

Add explicit conversions between a static Matrix and a dynamic Matrix and vice versa.
Implement operations between static and dynamic matrices:

- sum of static and dynamic matrix should be a static matrix,
- product of static and dynamic matrix should be a dynamic matrix.

Exercise 4. Transpose algorithm

Implement one function template that will transpose matrix. It should work both for static and dynamic matrices (if needed correct their interfaces). For static matrix it should return static matrix.

```
Matrix<double, 4, 2> m({{1.2, 1},{21, 2},{34, 2},{2, 32}});
auto mt = transpose(m);
printMatrix(mt);
/*  1.2   21    34    2
      1    2     2   32 */
```