Effective Programming Techniques - mini-project

Note: The purpose of the mini-project is to use the skills acquired during the course, including the completion of previous checklists. Therefore, within a mini-project, you can use language constructs offered by the C ++ 11 standard and higher. Restrictions that still apply include:
1. Prohibition of use unjustified exceptions.
2. Prohibition of using smart pointers, unless you write them yourself (you can use / extend the solution of homework no. 5).

Define and implement the CMatrix class that should provide at least the following functionalities:
• matrix addition and subtraction
• matrix multiplication (including vector and matrix multiplication and matrix multiplication by number)
• computing the scalar product
• matrix transposition
• matrix inversion
• loading matrices from a file (sample files for **int** and **double / float** types are on eportal).

Additionally, the following operations should be implemented:
• creating a matrix with the given dimensions (it can be done by the constructor or using member functions). Do not forget about proper handling and reporting errors, using exception is prohibited,
• assigning and reading values of the indicated matrix elements,
• creating vectors by selecting the indicated row or column of the matrix
Assigning a square matrix of values corresponding to the identity matrix.

All of the above functionalities are to be accessed via methods. Moreover, where it is possible, functionalities are to be wrapped in operators (e.g. matrix multiplication by number, matrix addition, access to individual matrix elements). If the functionality wrapped in an operator is to report an error, an exception can be thrown.

The CMatrix class is to be a template class and is to accept at least int, float, and double types. The CMatrix class is meant to be its own implementation. In particular, it cannot use any libraries to compute matrix determinants, etc.