Lab Assignment 6
Matrix – Part 3

CS 361 – Principles of Programming Languages I

Fall 2021

In the previous labs, we started implementing a class `Matrix`. We implemented constructors, the destructor, a resize and transpose function, and an operators to access and modify the elements of the matrix as well as to compare two matrices. This week, we will implement addition and multiplication of matrices.

Assignment

You are given a header file *matrix.h* declaring a class `Matrix` with various members which are described below. Write a file *matrix.cpp* which implements the class. You can use you implementation from Lab 5 or the file provided on Canvas as starting point. Do not add any additional public methods. You are, however, allowed to add private methods to avoid redundancy in your code.

The Class `Matrix`

Implemented in Lab 4 and Lab 5

- `Matrix()`, `Matrix(Matrix&)`, `Matrix(Matrix&&)`
- `Matrix(int size)`, `Matrix(int height, int width)`
- `~Matrix()`
- `int getWidth()`, `int getHeight()`
- `long& operator()(int row, int col)`
- `void resize(int height, int width)`, `void transpose()`
- `Matrix& operator=(Matrix&)`, `Matrix& operator=(Matrix&&)`
- `bool operator==(Matrix&)`
- `int width`, `int height`, `long** values`

Operators

`Matrix operator+(Matrix&)`: Creates a new matrix which is the sum of this and another given matrix. That does not change the current matrix. If the size of both matrices is not matching, the resulting matrix has the dimensions of the largest intersection of both given matrices.

`Matrix& operator+=(Matrix&)`: Adds a given matrix to the current. Note that it changes the matrix and does *not* create a copy. If the size of both matrices is not matching, it changes the dimensions of the current matrix to the largest intersection of both matrices.

$$\begin{bmatrix}1 & 2\\ 3 & 4\\ 5 & 6\end{bmatrix} + \begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix} = \begin{bmatrix}2 & 4\\ 7 & 9\end{bmatrix}$$

`Matrix operator*(Matrix&)`: Creates a new matrix which is the product of this and another given matrix. That does not change the current matrix. If the size of both matrices is not matching, it uses the largest sub-matrices which work (the top left element is always included).
Note that matrix multiplication is not a symmetric operation, i. e., in general for two matrices A and B, $AB \neq BA$. When multiplying two matrices, the `*`-operator of the left matrix is called.

`Matrix& operator*=(Matrix&)`: Multiplies a given matrix with the current. Note that it changes the matrix and does *not* create a copy. If the size of both matrices is not matching, it uses the largest sub-matrices which work (the top left element is always included).
Note that matrix multiplication is not a symmetric operation, i. e., in general for two matrices A and B, $AB \neq BA$. When multiplying two matrices, the `*`-operator of the left matrix is called.

$$\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix} * \begin{bmatrix}1&2&3\\4&5&6\end{bmatrix} = \begin{bmatrix}9&12&15\\19&26&33\\29&40&51\end{bmatrix} \qquad \begin{bmatrix}1&2&3\\4&5&6\end{bmatrix} * \begin{bmatrix}1&2&3\\4&5&6\end{bmatrix} = \begin{bmatrix}9&12&15\\24&33&42\end{bmatrix}$$

`Matrix operator*(long)`: Creates a new matrix which is the product of this matrix and a given number. This does not change the current matrix.

`Matrix& operator*=(long)`: Multiplies the current with a given number. Note that it changes the matrix and does *not* create a copy.

$$\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix} * 5 = \begin{bmatrix}5&10\\15&20\\25&30\end{bmatrix}$$

Submission

For your submission, upload a single zip-file to canvas. The zip-file should contain

- a file *matrix.cpp* and
- a file *matrix.h* if you added additional private methods.

This is an individual assignment. Therefore, a submission is required from each student.

Deadline: Sunday, November 7, 11:59 pm.