Lab Assignment 2

CS 361 – Principles of Programming Languages I

Fall 2017

You are given a header file *matrix.h* which declares a class `Matrix`. Write a file *matrix.cpp* which implements the class.

The header declares multiple constructors, a destructor, some methods, and various operators which are described below. Do not add any additional public methods. You are, however, allowed to add private methods to avoid redundancy in your code.

The elements of the matrix are stored in dynamic allocated memory, namely in the private member `double values**` which is used as two-dimensional array. Therefore, when implementing the class, make sure that your implementation handles dynamic allocated memory appropriately. For example, make a copy or delete it if needed.

The Class `Matrix`

Constructors

`Matrix():` Creates an empty matrix of size 0×0.
`Matrix(int size):` Creates an identity matrix of size `size` $\times$ `size`.
`Matrix(int height, int width):` Creates a matrix of size `height` $\times$ `width` filled with 0s.
`Matrix(Matrix&):` Copy constructor.
`Matrix(Matrix&&):` Move constructor.

$$\texttt{Matrix}(3) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \texttt{Matrix}(3, 2) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Methods

`int getWidth()`, `int getHeight():` Returns the width or height of the matrix, respectively.
`void resize(int height, int width):` Changes the size of the matrix to `height` $\times$ `width`. Existing fields keep their value and new fields are set to 0. Note that it changes the matrix and does *not* create a copy.
`void transpose():` Changes the matrix to its transposition. Note that it changes the matrix and does *not* create a copy.

$$\begin{bmatrix} 1 & 2 \\ 4 & 3 \\ 5 & 6 \end{bmatrix}.\texttt{resize}(2, 3) = \begin{bmatrix} 1 & 2 & 0 \\ 4 & 3 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 2 \\ 4 & 3 \\ 5 & 6 \end{bmatrix}.\texttt{transpose}() = \begin{bmatrix} 1 & 4 & 5 \\ 2 & 3 & 6 \end{bmatrix}$$

Operators

`Matrix& operator= (Matrix&)`: Copy assignment.

`Matrix& operator= (Matrix&&)`: Move assignment.

`double& operator()(int row, int col)`: Returns or sets the value at the specified position in the matrix.

`Matrix operator+(Matrix&)`: Creates a new matrix which is the sum of this and another given matrix. This does not change the current matrix. If the size ob 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 ob 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. This does not change the current matrix. If the size ob 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 ob 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.

$$\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix} * \begin{bmatrix}1&2&3\\4&5&6\end{bmatrix} = \begin{bmatrix}7&12&15\\19&26&33\\29&40&51\end{bmatrix} \qquad \begin{bmatrix}1&2&3\\4&5&6\end{bmatrix} * \begin{bmatrix}1&2&3\\4&5&6\end{bmatrix} = \begin{bmatrix}9&12\\24&33\end{bmatrix}$$

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

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

$$\begin{bmatrix}1&2\\3&4\\5&6\end{bmatrix} * 0.5 = \begin{bmatrix}0.5&1\\1.5&2\\2.5&3\end{bmatrix}$$

`bool operator==(Matrix&)`: Determines if two matrices are equal. Two matrices are equal if they have the same height, the same width, and all corresponding elements are equal in both matrices.

Members

`int width`, `int height`: Store the width or height of the matrix, respectively.
`double** values`: Stores the elements of the matrix (as two-dimensional array).

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. Your implementation is expected to work with Visual Studio 2012 (the version on the lab computers).

Deadline: Sunday, November 12, 11:59 pm.