CSCE 221 Cover Page
Programming Assignment #1
Due by February 4 midnight to eCampus

First Name: Asa Last Name: Hayes UIN: 525003952

User Name AsaHayes E-mail address: asahayes@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more: Aggie Honor System Office

Type of sources	
People	Josiah Egner
Web pages (provide URL)	Clarification on use of templated functions.
Other Sources	Several programming assignments from CSCE-121 with Dr. Moore

I certify that I have listed all the sources that I used to develop the solutions/codes to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name: Asa R. Hayes Date: 7 February, 2018

Report

February 7, 2018

Part I

Program Description and Purpose of Assignment

You want to implement a data structure that allows to tabulate data coming from many different models such as communication or social networks. The entries in a table may express a relation between two groups of people, e.g. the number 1 could denote friends and 0 otherwise. This type of data structure could be also used to represent a location of an object in a two-dimensional space using its coordinates, e.g. a pair (i, j) to refer to a particular element of the table. These tables are called two-dimensional arrays or matrices. The Background section of this assignment provides some basic information about matrices and their operations.

The purpose of this individual programming assignment is to learn about an elementary design, implementation, and testing of a simple C++ class called My_matrix. The class implementation allows you to understand and overview the basic C++ concepts like pointers, memory allocation, deallocation, dynamic arrays, constructors, copy or move constructors and assignments, destructors, operator overloading, reading from and writing to a file.

Part II

Description of Data Structures

In the first phase of the assignment, implement in C++ a class My_matrix that can hold data of integer type (int). The two parameters representing a matrix dimensions are usually not known in advance so it is necessary to allocate the arrays dynamically. In the second phase, you will need to write a generic version of the class My_matrix that can handle different types of numerical data.

The My_matrix class takes either a set of two dimensions (for rows and columns) or another My_matrix object as input. From this, the object creates an int** (or T** in part 2), which is an array containing pointers to the sub-arrays that make up the matrix part of the My_matrix object. Once the matrix is created, it can be filled with data from a file, data from another My_matrix object, or manually set per element. Input and display are facilitated by the overloading of the input and output operators. Operators * and + are also overloaded, to allow for some commonly used matrix operations for compatible matrices.

Part III

Instructions for Compilation

No additional effort is needed for compilation, the makefile is configured for each part to compile entirely with "make all" and run with "./main". The only caution would be to not remove any of the test input or output files, as the testing part of main{} requires them.

Part IV

Logical Exceptions; Bug Descriptions

Apart from the code blocks that lead to errors intentionally, there do not seem to be any places for logical errors or bugs besides the plain invalid input errors.

Part V

C++ Object Oriented/Generic Programming Features

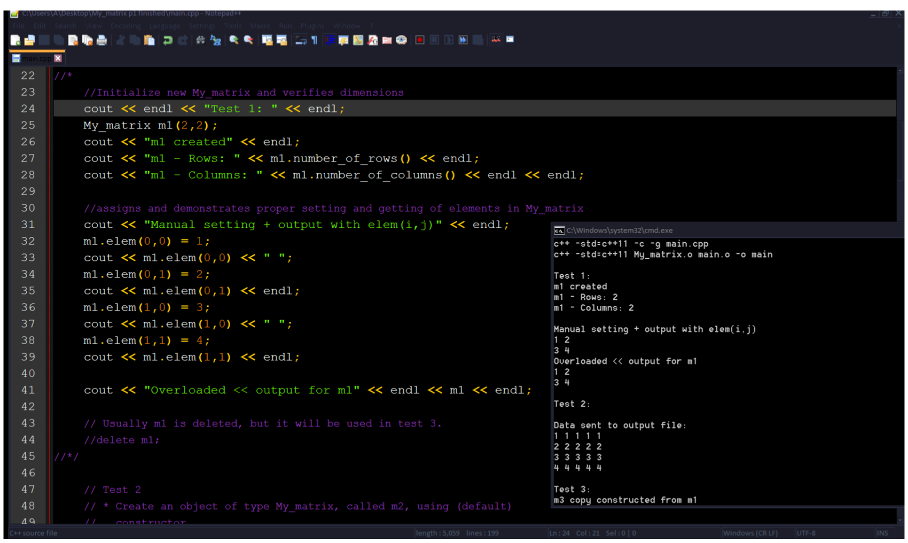
The most important features of object-oriented programming that are shown in this project are reusability and modularity. As My_matrix has a large amount of usages, having to manually create a structure each and every time one was needed would ruin readability and maintainability. Being able to simply create new My_matrix objects with one command not only creates better looking code, but reduces debug time needed, due to less variables having to be dragged around. The modularity of the My_matrix class is also important, as most of the functions contained within have no analogue within standard libraries and need to be tailored to the function. For example, the multiplication operator is much differently applied to a matrix than just 2 numbers as it would usually be interpreted, but allowing a specialized multiplier, we conserve the idea of the operator while still having it carry out the proper function, instead of creating an entirely new operator. As for general programming, the use of generic type T allows for much more versatility of what My_matrix is able to hold, allowing all types of numbers instead of just the default integers.

Part VI
Evidences of Testing

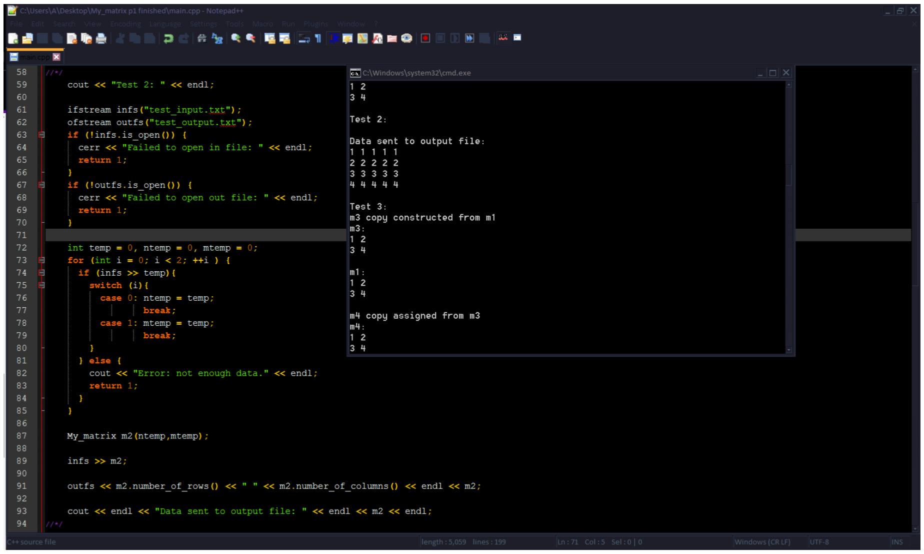
Note: The tests in the screenshots were run on Windows 7 with MinGW, but results were verified and identical when run on the TAMU UNIX server.

1 Evidence: Part 1

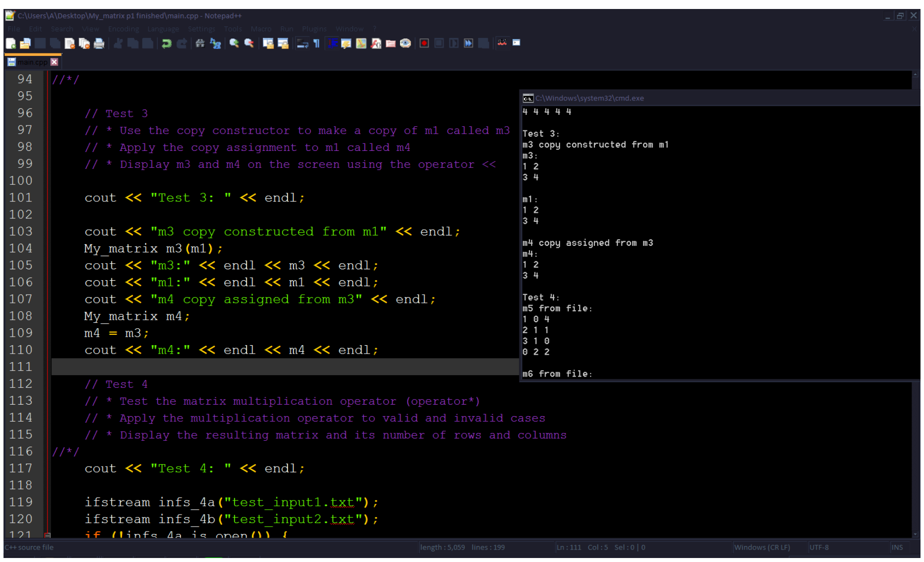
1.1 Test 1: Initialization of My_matrix objects, setting of matrix elements, demonstration of output operator



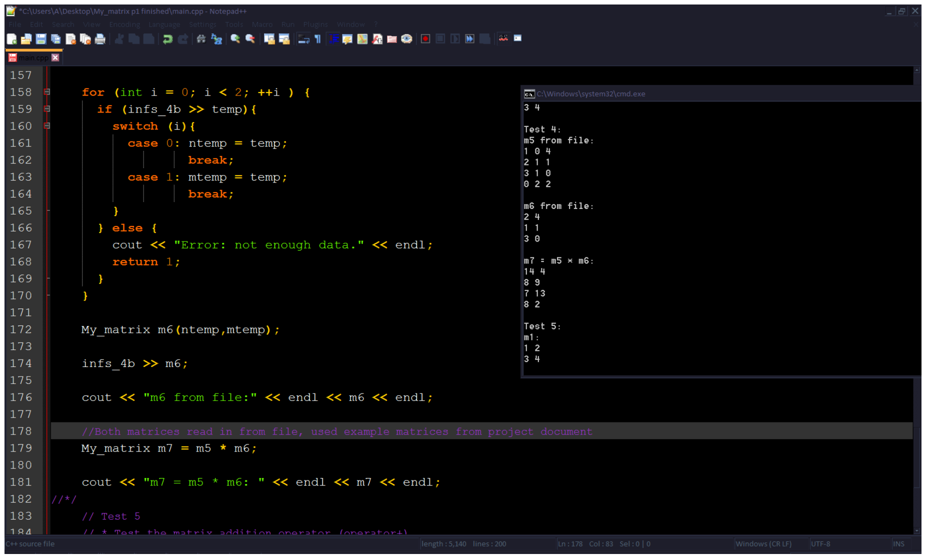
1.2 Test 2: Reading from input file into new My_matrix, displaying, reading to output file



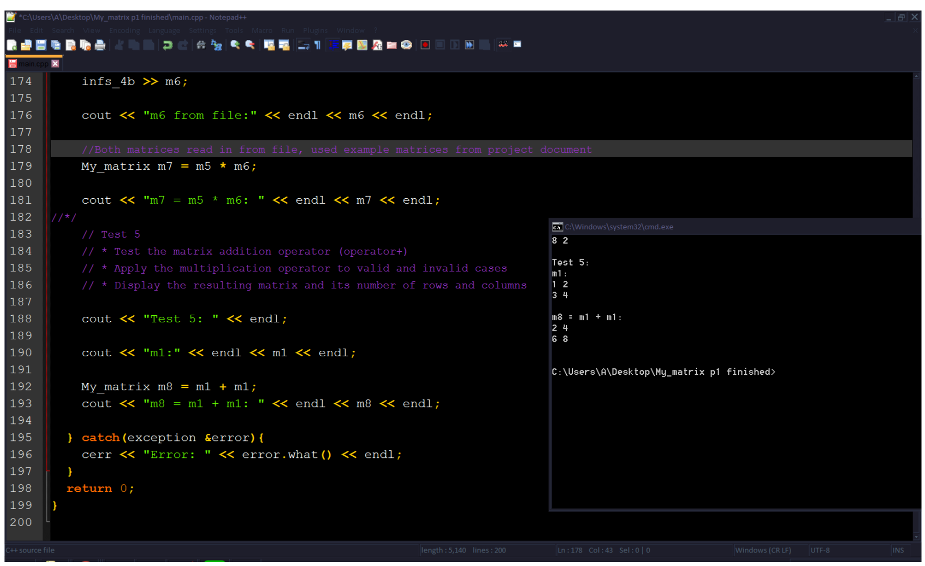
1.3 Test 3: Copy Constructor and Copy Assignment Operators



1.4 Test 4: Multiplication Operator



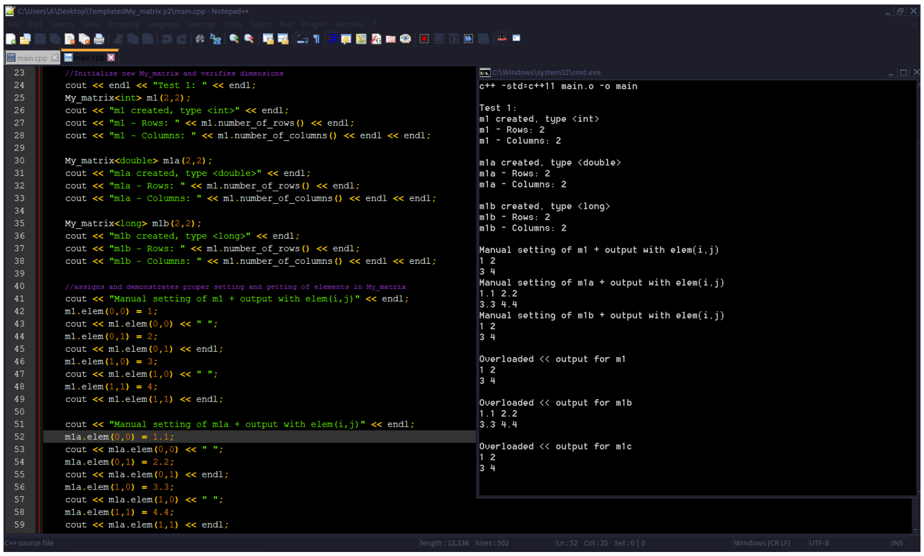
1.5 Test 5: Addition Operator



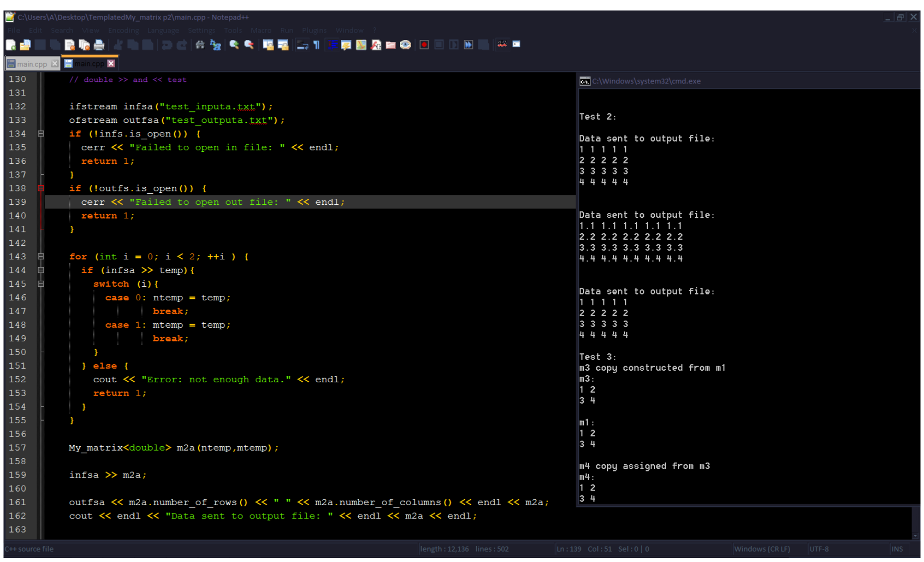
2 Evidence: Part 2

Screenshots will be focusing on the My_matrix<double> type, as the My_matrix<long> type is visibly identical to My_matrix<int> which was demonstrated in part 1.

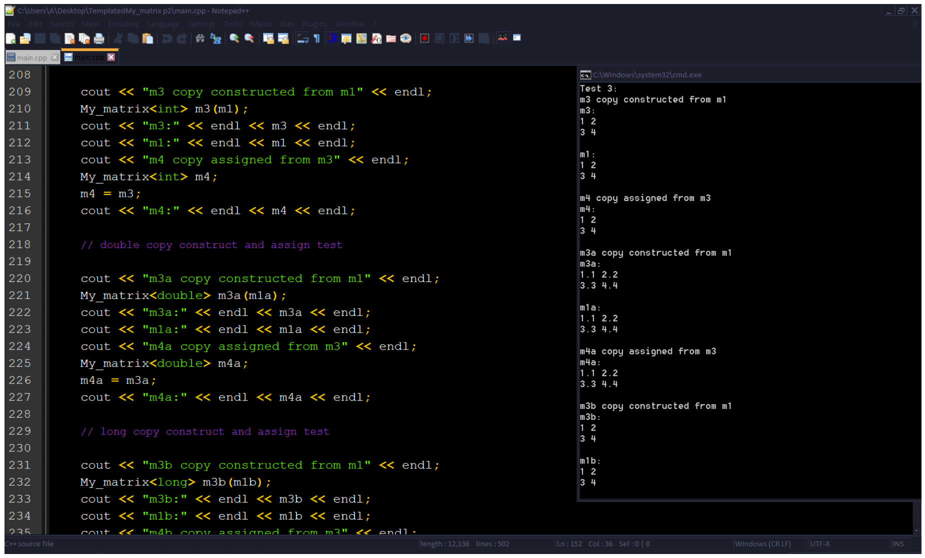
2.1 Test 1: Initialization of My_matrix<double> object, setting of matrix elements, demonstration of output operator



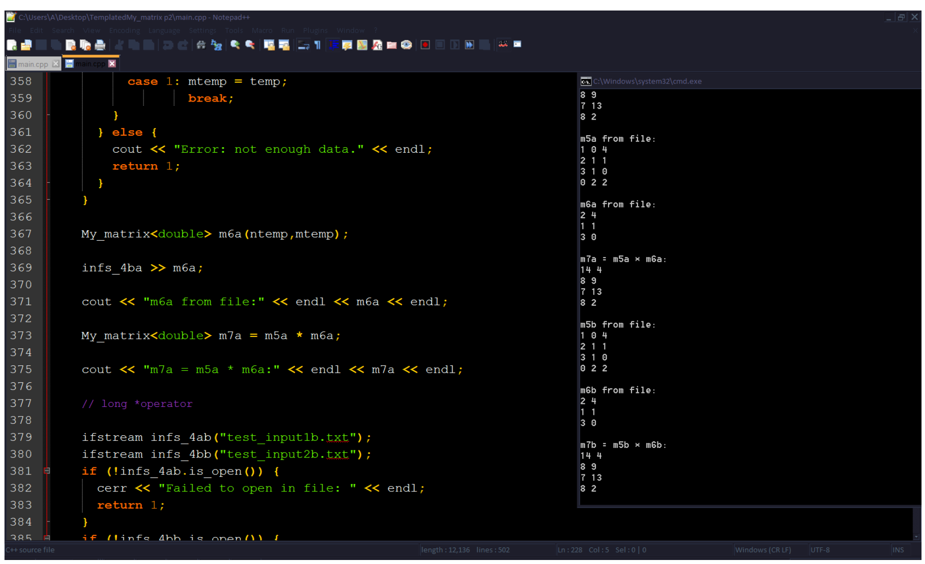
2.2 Test 2: Reading from input file into new My_matrix<double>, displaying, reading to output file

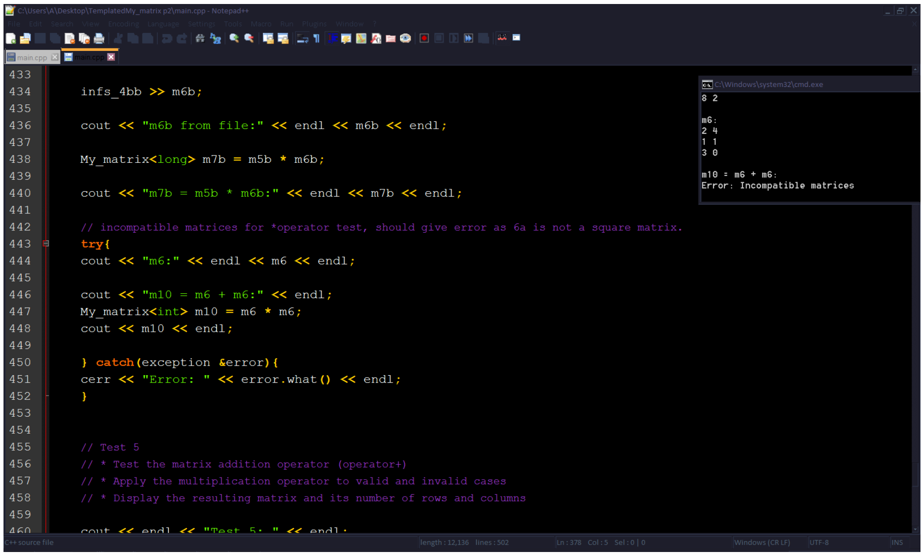


2.3 Test 3: Copy Constructor and Copy Assignment Operators



2.4 Test 4: Multiplication Operator, with test for incompatible matrices





2.5 Test 5: Addition Operator, with test for incompatible matrices