DSA Homework 1

Roger Jang

Due date: 20180326 23:59:59

Matrix class

Problem definition

The goal of this homework is to let you get familiar with C++ programming, especially in the use of classes and their operations. In particular, you need to implement a Matrix clss that supports basic matrix operations. The followings are the required member functions & variables.

- Basic:
  - Constructor
  - Copy constructor
  - Assignment constructor
  - Destructor
- Public member variables
  - int row;
  - int col;
  - int **array;
- Basic operator:
  - Operator =
  - Operator []
- Unary operatior:
  - Operator –
  - Operator +
- Binary operator:
  - Operator +
  - Operator –
  - Operator *
  - Operator /
- Other:
  - read(filename)
  - write(filename)
  - print()

Here are detailed descriptions about the above functions.

- read(filename): Read a matrix from file
  Input file format:
  - The first line of the input contains two integers, m (no. of row of the matrix) and n (no. of column of the matrix), with $1 \leq$ n,m ≤ 100000.
  - Each of the following m lines contains n integers separated by a single space, a1, a2, ..., an, representing each row of the matrix.
- write(filename): Write the matrix to the given file name, with the same format mentioned previously in read(filename).
- print(): Print the matrix to stdout, with each element separate by a single space and each row separate with a newline.
- Operator /: For matrices A and B, A/B is defined as $A * B^{-1}$. You can safely assume B^{-1} always exists.

Requirements & suggestions

- You can search Google using terms like "matrix class c++ example" to obtain similiar code for this homework, and work from there to have a jump start. (But remember you cannot change matrix.h, so you function prototypes must conform to those in the header file.)
- More info:
  - Copy constructors
  - Assignment operators
- To keep high precision, use the data type "double" to store each element. All intermediate variables should be declared as "double" too.
- In the package4student directory (download), you may find the following files:
  - readme.txt: read-me file
  - matrix.h: declare your own class
  - matrix.cpp: your member function (This is the only file you need to submit.)
  - test.cpp: sample test case
  - output: sample output files
  - outputStd: standard output files for the given test.cpp
  - makefile: the makefile to compile your code to create the default executable (a.out) for grading
  - runTest.sh: A script for compiling test.cpp, run it under several cases, and compare the outputs in "output" to the standard ones in "outputStd".
- For matrix multiplication, you can safely assume the matrices to be multiplied are of compatible dimensions.
- For matrix division, you can safely assume the inverse of the matrix always exists.
- You should be able to create the executable (a.out) by typing "g++ matrix.cpp test.cpp" under unix/linux.
- You should be able to execute your program by typing "./a.out [cmd]" under unix/linux, where [cmd] is the command to execute (see test.cpp for details).
- You only need to submit matrix.cpp. To test your program, TA will use another test.cpp to compile with your matrix.cpp. In other words, TA will embed test cases in the new test.cpp for test, compile it, run it, and obtain the output files to verify their correctness. An example of TA's test is in runTest.sh, which can be executed by typing "bash runTest.sh" under unix/linux, and all the output files will be stored under the folder "output".

Last updated on 04/13/2019 16:54:31.