User Manual

Program: Class\_Generator.cpp

This program prompts you for data in regard to a class, then creates a skeleton code based on the input. The program takes your class name, name, date, and purpose for the header. The class name is also used repeatedly throughout the program to get information behind the scenes. You are then asked to input any variable that you would like, as well as their types. If you’re entering these in, you must make sure they are valid types and that your class is one word or you will end up with syntax errors in your class.

To run the program double click on the file Class\_Generator.exe, or open Class\_Generator.cpp with Microsoft Visual Studio 2010 or above and hit Control F5. A console window will appear asking you to enter the name of the class, your name, the date, and the purpose of the header. These other data fields should accept any input, but be wise in your choices, as long names will lead to a bulky class. Hit enter after typing your input for each field. The program will then ask you if you want to include operators or private variables, it will repeat until you choose “n” for no. Repeat this process until finished with the prompts. Press any key to terminate the program when you are done.

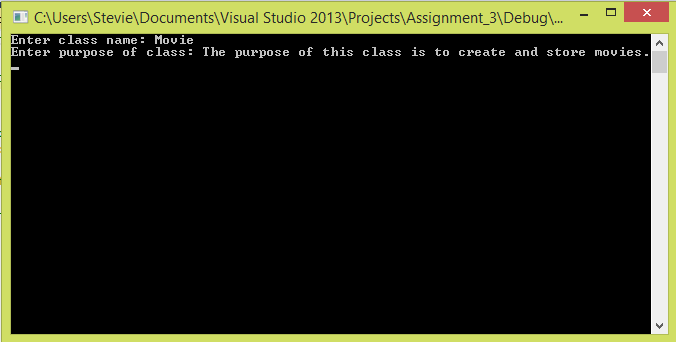
System Manual

Program: Class\_Generator.cpp

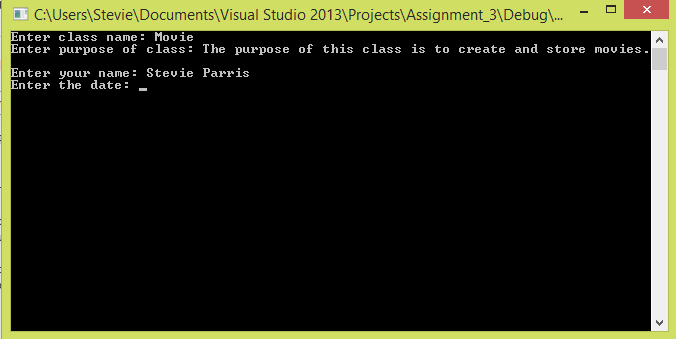
This program focuses on the construction of a class skeleton. It uses the little information gained from the user as well as a basic class layout in order to create this. The user enters the names of private variables, which are turned into getters and setters as methods. The toString() class is given only a skeleton, and not true functionality, as it cannot be automated without extensive information and discernment. The output method utilizes the toString() method in order to accurately create it without further need. The constructor is set to allow to be created blank, with any private variables given by the user, and a copy constructor. It includes a deconstructor for this class as well. The code is simultaneously writing information to the filename + “.h” and filename + “.cpp” files where filename is the user inputted filename. At the end, a basic skeleton of a test application is created under the name filename + ”Tester.cpp”. These are the basic functions implimented

The user is expected not to use special characters that C++ will not recognize. They are also expected to enter a non-negative integer for the year. The year is declare as an integer so all decimal places will be truncated. The program malfunctions if you try to use anything but a number using 0-9 for the year and skips past all other commands…

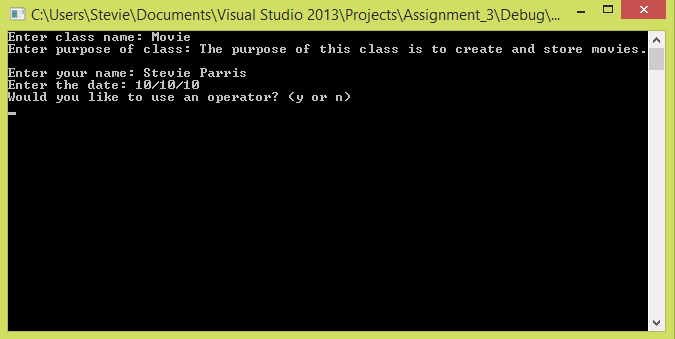
Test Log



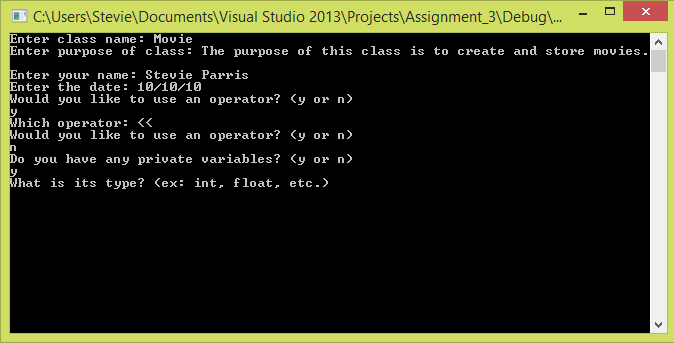
Example of entering class information, name and purpose.



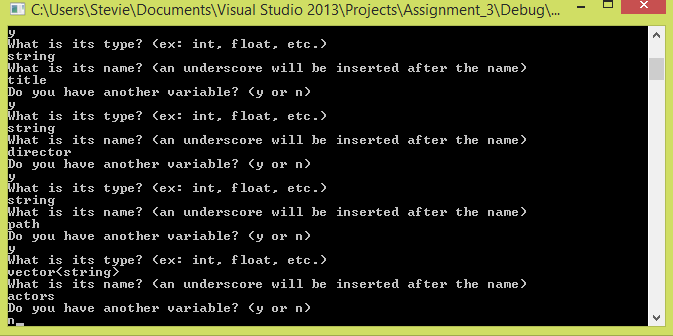
The program then asks for personal information, including your name and the date for the header.



The program asks whether you would like to add an operator, and if not answer “n” for no, proceeds. The program is only designed to handle the “<<” operator, however.



The program asks if you would like to add variables, and then asks their type and name.



The program repeats this with variable until the user has no further variables. The user is expect to know the valid variable types, and that variable names must be a single word.