Studio Session on C++ IO Streams

These studio exercises are intended to introduce basic features of C++ IO streams, and to give you experience using them for input and output of user defined data types (i.e., structs) within the Visual C++ environment.

As before, students who are more familiar with the material are encouraged to help those for whom it is less familiar. Asking questions of your professor and teaching assistant (as well as of each other) during the studio sessions is highly encouraged as well.

Please record your answers you work through the following exercises. After you have finished, please submit your answers to the required exercises and to any of the enrichment exercises you completed to Blackboard or our OJ system (if applicable).

The enrichment exercises are optional but are a good way to dig into the material a little deeper, especially if you breeze through the required ones.

PART I: REQUIRED EXERCISES

1. Open Visual Studio and create a new Visual C++ Win32 Empty Project for this studio. Create a new header file and source file in your project and copy over the declarations and definitions for the struct from the previous studio exercises (the Classes studio) into those files, respectively. In your main function, declare an object of the struct type, and make sure your program can compile and run correctly. Then, try to insert the object directly into **`cout`** (as in **`cout << obj << endl;`**) and as the answer to this exercise explain what happens when you try to build your program.

2. Declare and define an operator to insert objects of your struct type into an **`ostream`**, and in your program's main function again try to insert the object directly into **`cout`**. Make sure that your program can compile and run, and as the answer to this exercise please show your implementation of the insertion operator and also show your program's output (which should print out the values contained in the object this time around).

3. Declare and define an operator to extract objects of your struct type from an **`istream`**, and in your program's main function repeatedly (1) prompt the user to input values for the object (or hit Ctrl-C to quit), (2) extract values from **`cin`** into the object, and (3) insert the object into **`cout`**. Make sure that your program can compile and run, and as the answer to this exercise please show your implementation of the extraction operator and also show your program's output with several repetitions of input and output.

4. Repeat the previous exercise, but instead of taking input from **`cin`**, take it from an input file stream – pass the name of the file into the program as a command line argument, open the file, test whether or not it opened correctly, and if it did open correctly repeatedly extract values from the file until you reach the end of the file. Build your program, and in the directory where the executable program file was created,

create a text file containing a number of different input values. Use that file to test your program, and as the answer to this exercise please show your code, the contents of the input file, and the output your program produced.

5. Repeat the previous exercise, but instead of inserting the object into **`cout`**, insert it into an output file stream – pass the names of the input and output files into the program as command line arguments, open the files (for reading and writing respectively), test whether or not they opened correctly, and if they did open correctly repeatedly extract values from the input file stream into the object and then insert the object into the output file stream, until you reach the end of the input file. Build and run your program using the input file from the previous exercise, and as the answer to this exercise please show your code, the contents of the input file, and of the output file that was produced.

PART II: ENRICHMENT EXERCISES (optional, do the ones that interest you).

6. Modify your code from the previous exercise so instead of extracting data from, and inserting data into files it instead uses string streams to get the data out of and back into strings. Try reading from and writing to different strings, and as the answer to this exercise please show your code and the output your program produced.

7. Try using different stream manipulators in the extraction and insertion operators for your struct to "pretty print" the data (and still be able to read it back in correctly) no matter what streams are involved. For example, you might want to put some kind of delimiter (square brackets, angle brackets, etc.) around the object's values, and possibly separate them with a different delimiter (spaces, commas, etc.). Make sure that whatever format your insertion operator uses can be read successfully by your extraction operator. Build and run your program, and test it with different kinds of streams, and as the answer to this exercise explain briefly how you modified your extraction and insertion operators, show your code, and show the output of a representative run of your program.