CSCE 2014 – Programming Project 1

Midpoint Due Date – Feb 1, 2017 at 11:59pm
Final Due Date – Feb 8, 2017 at 11:59pm

1. Problem Statement:

The objective of this programming assignment bring together several key programming concepts (file input/output, arrays, and classes) to build an interactive program for looking up information about famous computer scientists. To initialize the application, students must read an ascii file called people.txt. Each line of the file contains the following pieces of information:

- first_name
- last_name
- birth_year
- death_year
- contribution

If you look at the people.txt file, you will notice that the first and last names are single strings, and that the birth and death years are integers. A death year of -1 is used to indicate that the person is still alive. The contribution field is a string with a variable number of words. Therefore, you will have to use getline to read to the end of a line in order to read in the contribution.

Your task is to read/store the information above into an object oriented data structure. Your program should then allow the user to search this data structure for computer scientists three ways: (1) using their first name, (2) using their last name, or (3) using a range of birth years. Your program should print out all records that match the search criteria.

2. Design:

Students are required to define two classes "Person" and "Table" to contain the information in the "people.txt" file. Each line of the input file corresponds to a Person to be inserted into the Table. Therefore, the data contained in each line should be used to set a Person object's data members. The Table class should contain an array of Person objects. Students must define these two classes and specify the constructor destructor functions, the public methods and private variables.

The Person class must contain the following methods.

- constructor - to initialize the object.
- copy constructor - to copy an object.
- destructor - to delete the object.
- set - allow the user to set the properties of a person.
- get - allow the user to get the properties of a person.
- print - print out the record.
- read - read a person's properties from a file stream.

The Table class must contain the following methods.

- constructor - to initialize the object.
- copy constructor - to copy an object.
- destructor - to delete the object.
- print - print out all of the records as a nicely a formatted table.
- read - read a list of persons from a file formatted as above.
- search_first_name - allows the user to search the table based on a person's first name. All matching records should be printed.
- search_last_name - allows the user to search the table based on a person's last name. All matching records should be printed.
- search_year - allows the user to search the table based on a range of birth years. All matching records should be printed.

The methods above should be used to implement the interactive search program. Students may want to refer to previous labs for examples of file input/output and classes.

3. Implementation:

To simplify the implementation process, you will be given the class definitions for the Person class in a file "person.h" and the class definitions for the Table class in "table.h". Your task will be to complete the implementations of these classes in two files "person.cpp" and "table.cpp". Then you should implement the user interface for your program in "main.cpp". Remember to use #include at the top of your *.cpp files to include your class definitions. Also, to compile your program you can use:

g++ -Wall -o hw7.exe person.cpp table.cpp main.cpp

Since you are starting with an existing class definition, your first task is to implement each of the methods. It might be a good idea to start with "skeleton methods" to get something to compile, and then add the desired code to each method incrementally writing comments, adding code, compiling, debugging, a little bit at a time. Once you have the methods implemented, it should be fairly simple to create a main program that calls these methods to complete your project.

Remember to use good programming style when creating your program -- good names for variables and constants, proper indenting for loops and conditionals, clear comments, etc. Also, be sure to save backup copies of your program somewhere safe. Otherwise, you may end up retyping your whole program if something goes wrong.

4. Testing:

Test your program to check that it operates correctly for all of the requirements listed above. Also check for the error handling capabilities of the code. Try your program with several input values, and save your testing output in text files for inclusion in your project report.

5. Documentation:

When you have completed your C++ program, write a short report using the project report template describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this report to be submitted electronically.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above go to Blackboard to upload your documentation (a single docx or pdf file), and all of your C++ program files. Do NOT upload an executable version of your program.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

- 10% off if less than 1 day late,
- 20% off if less than 2 days late,
- 30% off if less than 3 days late,
- no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

7. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.