University of Wollongong

School of Computing & Information Technology

CSCI251/851 Advanced Programming Autumn 2018

Assignment 1 (Due: 11.59pm, Week 4, Friday 23 March) 10 marks

(Week-3: demo 2marks. Week-4: submission 8 marks)

Aim:

- Design a solution to a problem from a partially complete framework.
- Gain some experience writing a database application in C++.
- Using binary I-O in C++ programs.

Instructions:

To receive 2 marks for the demo you must complete Step-1 and demonstrate it to your lab supervisor in week-3. The week-4 submission instructions are provided at the end of this document. Please read the Ass1 Q/As on moodle regularly in case suggestions are provided or changes are made to the assignment specification.

You are to write a student enrollment system for a university. For each student the system is to keep the following information:

a) First Name
b) Last Name
c) Number of Subjects
d) Subjects (Codes; Status, Marks;};

In the Ass1 folder you will find an incomplete implementation of the enrollment system and a datafile called ass1.txt for testing the program. Examine the data file to get a feel for the data in this system. You may assume that the file contains no errors. All work for this assignment should be done in the ass1.cpp file. Do not modify main.cpp or ass1.h. To compile on linux/unix: g++ main.cpp ass1.cpp. To run on linux: ./a.out. The file: "How to compile with DevCpp.pdf" explains how to compile all the files on DevC++.
(Marking: Step 1 is worth 4 marks (including the demo). Steps 2 and 3 are worth 3 marks each.)

Step 1 Implement the functions: ReadFile(), FindRecord(), DisplayRecord() and PrintRecord() according to the pseudo code provided. Information and example code on reading and writing text files can be found in the C++ Guide in the Wk 1 Lecture folder. Test that your program can correctly load the array from the data file and display records on the screen. Example output is given below:

```
13 records read

Commands Available:
  d - Display Record
  u - Update Record
  q - Quit the program

Command: d
Enter student number: 4734455
Student No.      4734455
First Name       Kieren
Last Name        Legrande
Subjects:
    CSCI104  Provisional  65
    IACT123  Enrolled     67
    CSCI121  Enrolled     98
```

Step 2 Implement the UpdateRecord() and SaveFile() functions and test your program to ensure that it can amend student records and save all records to the file ass1.txt.

```
Command: u
Enter student number: 4734455
Student No.       4734455
First Name        Kieren
Last Name         Legrande
Subjects:
    CSCI104  Provisional  65
    IACT123  Enrolled     67
    CSCI121  Enrolled     98

Enter subject code: CSCI104
Select status or mark (s/m): s
Select new status
    e: enrolled
    p: provisional
    w: withdrawn
Status: e
Record 4734455 Updated.
```

Step 3 Requires implementation of binary file I-O into the program. If you are unfamiliar with binary file I-O, information on this can be found in BinaryFiles.pdf and the *C++ Guide* available in the Lecture Notes folder. To implement binary file I-O follow these steps.

(a) Add four more private functions to ass1.cpp:

```
bool ReadTextFile(char Filename[]); //reads text data from file to gRecs[] array
bool WriteTextFile(char Filename[]); //writes text data from gRecs[] to file
bool ReadBinaryFile(char Filename[]); //reads binary data from file to gRecs[] array
bool WriteBinaryFile(char Filename[]); //writes binary data from gRecs[] to file
```

To do this, copy the code in the ReadFile() and SaveFile() functions to ReadTextFile and WriteTextFile() respectively. Modify this code so that it uses the passed filename and return false if the file or path is not found when opened. ReadBinaryFile() and WriteBinaryFile() should read/write the binary contents of gRecs[] to the passed filename (see lecture notes: BinaryFiles.pdf) (Note: the binary file should store the number of records followed by the record data.)

Rewrite the ReadFile() function so that it uses the above functions and attempts to read the binary file (named in global cBinaryFileName). If this fails, it then attempts to read the text file (named in global cTextFileName) and then writes the binary file. If ReadFile() fails to read the binary file and the text file it should print an appropriate error message and exit.

(b) Add one more private function named:

```
void WriteBinaryRecord(char Filename[], int Pos);
```

This function should:

```
open the binary file (named in global cBinaryFileName)
seek() to the appropriate record at Pos
write the record to the binary file
close the file
```

Then alter the UpdateRecord() function so that it <u>also</u> updates the binary file by calling the above function.

(c) Modify the SaveFile() function so that it attempts to write the gRecs[] array to the binary file. If this fails it then attempts to write the data to the text file.

(d) Test your program to ensure that the binary data is being appropriately saved to the file.

Submit:

Before submitting check the format of your source files to ensure tabs, spaces and newlines appear correct on an editor like nedit or notepad++. Submit your files using the submit facility on UNIX ie:

```
$ submit –u login –c CSCI251 –a 1 ass1.cpp main.cpp ass1.h
```

where 'login' is your UNIX login ID.

Deductions will be made for untidy work or for failing to comply with the submission instructions. Requests for alternative demo or submission arrangements will only be considered before the due date. An extension of time for the assignment submission may be granted in certain circumstances. Any request for an extension of the submission deadline must be made to the Subject Coordinator before the submission deadline. Supporting documentation should accompany the request for any extension. Late assignment submissions without granted extension will be marked but the points awarded will be reduced by 1 mark for each day late. Assignments will not be accepted if more than four days late.