University of Wollongong

School Information Computer Science & Software Engineering

CSCI124 **Applied Programming** **Spring 2012**

Assignment 1 **(Due: Week 4, Wednesday 15 August)** **6 marks**

Aim:

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

Instructions:

If you have difficulty understanding any part of this assignment ask your Lab Supervisor, Tutor or Lecturer to explain. Read the submission instructions at the end of this document carefully before submitting your work. Also, regularly read the Messages page in the subject website incase any suggestions are provided or changes are made to the assignment specification. (http://www.itacs.uow.edu.au/subjects/csci124)

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. Compile on linux using makefile (see below) or the compile directive:
$ g++ main.cpp ass1.cpp

Step 1 Write an appropriate makefile to compile the program on linux or UNIX. Information on how to do this can be found in the Week 1 lecture notes named "Compilation and Linking.pdf".
Your makefile should produce an executable named ass1. Test your makefile on linux or UNIX.
Now 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 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 BinaryIO.pdf and the *C++ Guide* available in the Lecture Notes folder. To implement binary file I-O follow these steps.

(a) Alter the ReadFile() function so that it does the following:

open ass1.dat
if ass1.dat is found and opened then
    read the binary data into the gRecs[] array from ass1.dat and then close the file
else
    open ass1.txt
    if ass1.txt is not found and opened then print an error message and exit() the program
    else
        read the text file into the array and then close the text file
        create and open the binary file ass1.dat
        write the contents of the gRecs[] array to the binary file and then close the file

(b) Alter the UpdateRecord() function so that it also updates the binary file thus:

    ....
    open the ass1.dat file
    seek() to the appropriate record
    writes the record to the binary file
    close the file

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

Submit:

Before submitting your ass1.cpp file check the format of your source files to ensure tabs and newlines appear correct on UNIX. Submit your **ass1.cpp** & **makefile** file using the submit facility on UNIX ie:

```
$ submit -u login -c CSCI124 -a 1 ass1.cpp makefile
```

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 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 must accompany the request for any extension. Late assignment submissions without granted extension will be marked but the points awarded will be reduced by 5 points for each day late. Assignments will not be accepted if more than three days late.