EE208 Laboratory Session 4

Bryan Hennelly

20 February 2018

1 Lab objectives

In this lab you will continue on from last week by solving (slightly) more complex problems in C++. We can no longer continue to use the online compiler and this week we make the jump to an integrated development envoirnment (IDE). You will be using Microsoft Visual Studio Express, which is free for you to download. While the IDE is more complex to use than the online compiler it comes with many significant advantages: firstly, we can access (read from and write to) files on our own computer drives; secondly we can create projects and debug the code. A version of the IDE is available for you on the lab computers; you will need the following details to log in:

User: your university email address
Password: your university password

A tutorial is available here that teaches you how to set up your project and to debug. You should take some time to read in detail and pay particular attention to the notes on debugging your code using breakpoints and "stepping into" and "stepping over" your code. In this lab you will be required to write programs that make use of some features of C++ that we covered since the last lab as well as some that we wilcover in the next lecture, namely and structures, pointers, and dynamic memory allocaioon. Don't forget to make full use of the online C++ tutorial if you are struggling with any of the concepts. You have one week to submit all of your solutions to this weeks lab on moodle; the deadline is 5pm Monday 26th Feb This lab is worth approximately 2

Learning Outcomes
By the end of this lab sheet you should be able to:

1. Write simple C++ programs that make use of (i) pointers and dynamic memory allocation, and (ii) structures.

2 Questions:

For each of the problems given below write a C++ program that provides a solution. Each box provides a filename to use (or in certain cases multiple filenames). Please ensure that you use those filenames. If you wish yu can submit your code to moodle.

Remember:

- **Comment your code.**
- **Use proper indentation for function and control structure.**

Exercise 2.1: This first exercise is designed to be a simple introduction to using simple structs for the first time; before beginning you are expected to read through the slides for the next lecture on structures, which have been posted on moodle for you.

A complex number is made up of two parts: the real part and the imaginary part, both of which are real numbers. Write a program, using struct called `Complex`, that contains two elements of type `float`. Now write four short functions to (i) return the modulus (or absolute value) of a complex number that is input to the function, i.e. a double should be returned by the function; (ii) return the complex conjugate of a complex number that is input to the function, i.e. a single `Complex` type object should be returned; (iiii) return the sum of two complex numbers that are input to the function, once again a single `Complex` type object should be returned. (iv) return the product of two complex numbers that are input to the function, once again a single `Complex` type object should be returned. Demonstrate your code in action by creating a number of different objects of type `Complex` and using all of the functions.

You should save the source in a file called `exercise_4_1.cpp`.

Exercise 2.2: Write a program, using struct's, that reads in the name and age of a list of people, and again, prints them out in reverse. The program should store the above details using an array of structs. Initially the program should ask the user for the number of people that will be input.

You should save the source in a file called `exercise_4_2.cpp`.

Exercise 2.3: This one is a little more fun but it's tricky. The first thing you need to do is open the online tutorial; a pdf version has been uploaded to moodle for you called `cplusplus_tutorial.pdf` under Week 4. Read pages 138-140, which cover the subject of reading from and writing to txt files. A txt file has been provided for you on moodle, called `ee208.txt`. This files contains a list of the names of all stiudents in this class, where the first line contains the number of students in the class. You should practice opening this file and reading from it by simply copy pasting the code on the top of page 141 and using it appropriately. Now practice writing the data to a new file called `results.txt` using code similar to the code provided at the bottom of page 140. Your exercise is as follows: first create a structure called 'student' that contains a string member called 'name' that will store the student's name and a float member called 'grade' that will store the student's exam result. Next use dynamic memory allocation to create an array of structures of sufficient size (you will need to use the number on the first line to determine the size) and read each name in `ee208.txt` into each structure's string member. Generate a random exam result between 0-100 for each student and store in each structure's grade member. Finally, write a function called `sort` which takes in the array of 'student's and sorts them in descenting order of grades. You should be able to use your swap functuion in a previous lab as a guide. Finally write each name and result in decenting order into `results.txt`. Good luck!

You should save the source in a file called `exercise_4_3.cpp`.

Exercise 2.4: This question looks tricky but it's actually very straight forward and should only take you a few minutes. At the end of the lecture slides on structures, there is an example of creating a linked list that stores three peoples' names. Copy this program to a file called `exercise_4_4.cpp` and compile and run it to verify it works. Change the structure name to `student` and change the contents of the structure to contain a name and a student number. Adapt the print function appropriately. Extend the functionality of the program with the following functions routines:

```
void insert_name(student * head, string name, int
student_number);
int list_length(student * head);
bool find_student(student * head, int student_number);
void print_student_details(student *head);
```

Demonstrate the functionality by writing a program that allows the user to input an aribtrary number of students, specified by the user at run time. The program should store these students in a linked list. The program should keep inputting students until the user inputs a user with the name `end` at which point all the user details should be printed to the screen.

You should save the source in a file called `exercise_4_4.cpp`.