EE208 Laboratory Session 2

Bryan Hennelly

5th February 2018

1 Lab objectives

In this lab you will continue on from last week by solving (slightly) more complex problems in C++. In particular, you will be required to write programs that make use of the features of C++ that we covered since the last lab, namely functions and arrays. Don't forget to make ful 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 due date is Monday 12th Feb 5pm. This lab is worth 2.5% of your overall grade. Don't forget to comment your code for full marks. The demonstrators are here to help so don't be afraid to shout if you're having difficulties.

This week we will continue to make life really simple for you; you will do all of your work on an onlinje compiler. The reaosn for this is that most people find using Interated Development Envoirnments (IDEs) like Microsfort Visual Studio, or Eclipse, a little bit of a headache at the start, and it tends to put novice programmers off what should be fun. So, for the first few weeks you can use this really simple online compiler. By doing this you are effectively connecting to a server somewhere in the world that is running the most uptodate GCC compiler.

Once you have written your program and youre happy with it you can copy and paste your code to a simple text editor like Notepad++. A text editor is used to edit plain text files. You should then save your file with the *.cpp extension and store it somewhere on your own shared drive (Z directory). Once you save as cpp you will have syntax highlighting; syntax highlighting is a very useful feature. It means that the editor will highlight certain words or types or syntax specific to a language. For example, if you have C++ highlighting turned on, the editor might make all C++ control flow keywords appear green. This makes it much easier to follow the flow of your program. As another example, the editor might have all quoted text show up as light blue. This way, if you forget to include an opening or closing quotation mark, you will quickly realize it because of the color of the text on your screen. A text editor might also indicate mismatched parentheses or brackets by turning them red; if you have a closing brace with no corresponding opening one, the color will tell you that you made a syntax error somewhere. Once you have done all of the exercises you should upload them to moodle directly - **DO NOT ZIP**.

Learning Outcomes

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

1. Write simple C++ programs that make use of (i) functions (including pass by reference arguments), (ii) arrays, and (iii) strings.
2. Overload your functions and add default parameters.

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: Write a program that simulates the rolling of two dice. The program should call `rand` to roll the first die, and call `rand` again to roll the second die. The sum of the two values should then be calculated. Your program should roll the pair of dice a user specified number of times, keeping track of the number of times each possible *total value* occurs. If you're not sure how to call the rand function have a quick read of this. Hint: you should use an array of integers of size 11 (since there are only 11 possible outcomes i.e. the total values of 2 up to 12). When the program is run it print out, in a tabulated format, the percentage of times each pair of values occurs. You should save the source in a file called `exercise_2_1.cpp`.

Exercise 2.2: Write a function, called `swap`, that takes two integers as input and swaps their values. To demonstrate that the function works you should write a program that instructs the user to input two integers. The program should store those numbers in two separate variables. The program should then print the values stored in the two variables to the screen, swap the two numbers by calling `swap`, and then again print out the values stored in the two variables. Hint: You will need to use use pass by reference to make sure that the swapping that occurs within the swap function is 'remembered' in the main function. You should save the source in a file called `exercise_2_2.cpp`.

Exercise 2.3: You should extend the previous exercise as follows: overload your swap function with two more swap functions, which swap two float variables and two characters respectively. Demonstrate all three of your swap functions in action by calling them appropriately in the main function. You should save the source in a file called `exercise_2_3.cpp`.

Exercise 2.4: Extend the previous exercise by including a default parameter in each of your three functions. This default parameter should be of boolean type and if the input is **True**, then some text should be printed to screen giving details of which of the three swap functions was called; if it is **False** then nothing should be printed to screen. The default value should be **False**. Demonstrate the this new functonality in your main functuion. You should save the source in a file called `exercise_2_4.cpp`.

Exercise 2.5: A little harder! Write a program that allows the user to input a sequence of 5 doubles and then prints that sequence in reverse. Hint: you will have to use an array and a couple of while or for loops. You should save the source in a file called `exercise_2_5.cpp`.

Exercise 2.6: Difficult! Now write a program that prints out the first N inegetrs in the Fibnacci sequence in reverse, where N is entered by the user *at run-time*. Save the source in a file called `exercise_2_6.cpp`. You must use dynamic memory allocation to do this properly; DO NOT WRITE int N; cin >> N; int my_Array[N]; EVEN IF YOUR COMPILER ALLOWS THIS;