EE273 - Engineering Design for Software Development 2

Week 3, Semester 1

Learning Outcomes:

- Ability to write, compile and run simple functional C/C++ programs.
- Ability to debug and test simple non-looping functional programs.
- Ability to declare variables of specific type and recognise the role of typing within C/C++ programs.
- Ability to use logical and arithmetic operations
- Ability to write and call simple functions within a C/C++ program.
- Ability to use screen and key board input/output functions.

Preface

- The lab sheets are released one week in advance of the appropriate laboratory session.
- It is key that students prepare in advance of the actual laboratory sessions; reading through the lab sheets, mapping out draft solutions and indeed trying out sample code.
- Students are expected at arrive at each lab session fully prepared for the lab exercises; the lab exercises are not designed to be completed solely within the lab sessions but are designed to be completed by the end of each lab session if appropriate preparation takes place.
- Students are expected to maintain a logbook as part of the laboratory work – taking notes, recording code and reflective comments about their solutions and progress todate.

Part 1: Variables & Operators

1) Using an appropriate text editor, create a new program in which an `int` variable, a `float` variable and a `bool` variable are all declared. Give each of them a unique name – such that the name reflects their type and function. Add code to permit the user to enter values for each of these variables using `cin` and use `cout` to print the values on the screen.
   a) Compile the program with GCC and run it via the console.
   b) Using Visual studio, compile and run the program, ensuring that (when running the program) the output can be seen easily on the monitor.

2) Add extra code to the programme that.
   a) multiplies the `float` variable by the `bool` variable,
   b) divides the `float` variable by the `int` variable
   c) divides the `int` variable by the `float` variable.

   Compile and run the program for a range of different input values. Are the results always what you would expect? If not, why not?

Part 2: Functions

3) Taking the code developed in the previous example (part 1), add in a simple ”stub” function called `stub(),` called from within `main`, that does nothing, passes no parameters and returns no values.
   a) Compile and run the program.
      i) How do you test that the program runs correctly?
      ii) Compare the size of the executable produced by both compilers with the executables obtained previously. What would you expect – comment?
   b) Modify `stub()` so that it now takes all three variables as input (arguments) and returns true/false value associated with an internal variable “`result`” that is initialised as 0.
4) Re-write the above program so that each of the arithmetic operations is called in its own dedicated function as follows:
      i) one function that takes a `float` and a `bool` as arguments (i.e. ‘inputs’), multiples them together and returns the result as a `float`;
      ii) one function that takes a `float` and an `int` as arguments, divides the first by the second and returns the result as a `float`;
      iii) one function that takes an `integer` and a `float` as arguments, divides the first by the second and returns the result as an `integer`.
   b) After each function call from the main function, use a `cout` instruction in main to print the result to the console. Test the program for a range of input values.
5) Do the types of the variables in the function that calls another function need to be the same as those in the function that is called? Change and recompile the program to find out. Which work, and why?
6) Do the names of the variables in the function that calls another function need to be the same as those in the function that is called? Change, recompile and rerun the program to find out. Which cases work, and why?
   a) What happens if you change the first function so that it returns an `int` instead of a `float`?

Print out listings of your programs and put in your logbook. Make a note in your logbook of anything that you found difficult but learned how to do (so that you can look it up again later in case you forget it). Make a note of the answers to all key questions.

DO NOT WRITE OUT FINAL PROGRAMMES BY HAND……..

Check that all programs can run under both Visual studio and GCC.

Demonstrate the operation of your code to your group mates and discuss with them the answers to the questions set above. Write some general comments about your progress in your logbook and the space available on myplace section.