COM4013 – Introduction to Software Development
Week 2 – Functions

Lab Exercises

Always refer to the lecture notes for examples and guidance.
Also look at your previous work, many exercises are similar.

There are two "stages" marked in the lab sheet. Stage 1 is the *absolute minimum* point you should reach. Hopefully, you can reach it in the lab session. Reaching stage 2 suggests you are on target for a top-class mark for the module. Set your goals sensibly - do not just aim for the minimum or you may struggle to pass the module.

I use the term integer, string and float, and Boolean to infer what the input/output looks like. Unlike many other languages C++ does expect a data type before the variable names.

So if I ask that you declare an integer num1 to value of 1, then do as follows:

```
int num1 = 1;
```

For declaring a string and assigning the value hello

```
string greeting = "hello";
```

For characters we can declare it like this

```
char character = ‘R’; // single quotation mark
```

For floats we can declare it like this

```
float money = 2.50;
```

For doubles (more accurate floating-point number) we can declare it like this

```
double money = 2.50;
```

For Booleans (bool) we can declare it like this

```
bool isHeavy = true;
```

Setting up the C++ development environment and starting a new project

- See the worksheet for week 1 for details about how to do this. Name it week 12.

Baby Function

- Write a function named "MyFunction". The function should simply output the message "function called".
- Invoke (call) the function. You should see the message written to screen.

- Invoke (call) the function a second time. You should see the message written to screen twice.

- Delete your two previous invocations of the function.
- Write a *for* loop to invoke the function 5 times.
- You should see the message being written 5 times to the screen.

Single Parameter

- Write a new function named "SingleParameter". The function should output the message "function SingleParameter called".
- The function receives a single integer parameter. The function should also output the value of the integer.
- Invoke the function. You should see the message and number written to screen.

- The variable used to maintain the count in the *for* loop is called the index variable. Use the index variable of a for loop as the calling parameter to the function. Invoke the "SingleParameter" function 4 times.
- You should see the message being written 4 times to the screen and a number being displayed each time the function is called. The number will count upward to 4 (if it doesn't then fix the for loop).

- Write another for loop, call SingleParameter, but print the numbers 27 to 5 instead.

Output String

- Write a new function called OutputString.
- The function receives a single *string* parameter.
- The function should also output the value of the string.

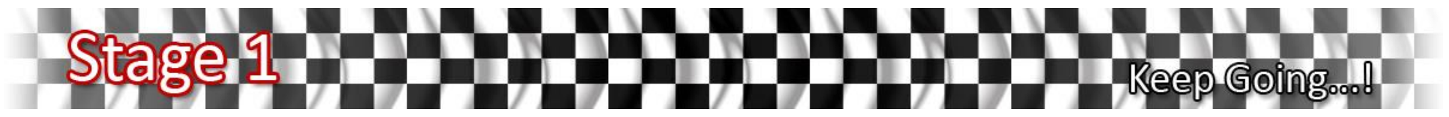
- Write a for loop to print this out 11 times in main.

Output String and Integer

- Write a new function called OutputStringAndInteger.
- The function receives two parameters. The first is *string* parameter. The second is an integer parameter.
- The function should **return a string variable** which should include the integer value.
    - To do this you will need to import the string library
    - #inlcude <string>
    - Use the to_string(integer) method to change the int to a string.
- Using console out (cout), invoke the function and print out the string and the integer.
    - Output example: Hello: 1

- Using a **while loop**, counter integer, number integer. Invoke the while loop **51** times using the count variable to keep track of the loops. Set the number variable to 250 and decrement this number by 5 on each loop of the while loop.
    - Use the shorthand for incrementing the counter and decrementing the number variable.

- Do the same (as above) with a for loop 😊.
    - An advance task (optional): Only use the counting **varibale(s)** in the for loop to do this operation.



Print File Contents to the Screen

- There is a text file called "input.txt" on Moodle. Download it.
- ⚠ Make sure that you place the file in the same directory as your project.
- Read the input file character by character. See the code below.

```
#include <iostream>

#include <fstream>
using namespace std;

int main( )
{
	ifstream infile("input.txt");
	if( !infile )
	{
		cout << "ERROR: ";
		cout << "Can't open input file\n";
	}

	infile >> noskipws;
	while( !infile.eof() )
	{
		char ch;
		infile >> ch;
	}
}
```

- It is important that you read it by character - you've not yet done anything about accessing the characters from a string in C++.
- After each character has been read, display it to screen.
- All of the characters are upper case. (This is important, we'll come back to this later.

Stage 2

Print File Contents to the Screen Function

- This introduces a function to your program.
- Write a function that has a single parameter. The parameter is the input file stream. See the lecture notes for passing a filestream over to a function. The function should open the input file. (In effect you are moving part of the code you wrote for exercise **Print File Contents to the Screen** into the function.)
- Write a second function which has a single parameter. The parameter is a single character.
- The body of the function contains a single line of code which simply outputs the character to the screen.
- Your program reads a character from file and outputs it to screen. Modify your program so that instead of outputting it to screen, it is passed over to the function. The function will be doing the work of outputting the character.

Advanced Tasks – (PYTHON PORTFOLIO PIECE 2)

Caeser Cipher Tool

- I want you to implement a Caeser Cipher console-based encryption program. This is a simple encryption algorithm in which the values of letters are increased by a fixed amount, say 3. So 'A' becomes 'D', 'B' becomes 'E' and so on. At the end of the alphabet return to the beginning so 'X' becomes 'A', 'Y' becomes 'B' and 'Z' becomes 'C'.

- Modify your function so that it implements Caeser's encryption. The easiest way to manipulate the character is recognise that every character is actually implemented as a number. An upper case 'A' has an ACSII numerical value of 65, 'B' has a value of 66, 'C' has a value of 67 and so on all the way up to 'Z' with a value of 90.

- Modify your program to use an output file. Write the encrypted characters to the output file.
- You can view your file in Textedit.

- Write a second program to decrypt your encrypted file.

- Combine all your functions into one program.
- Write a menu that allows the user to select encryption or decryption.

- Write a function which copies the contents of text file into a new file. The function has two parameters: the name of the existing file and the name of the new file.

Expand on the functionality of this tool – This assessment will also test your

Advanced Tasks – (PYTHON PORTFOLIO PIECE 3)

Draw more Shapes

Create an interactive program that allows users to print console-based shapes along with their areas.

- Use nested for loops to draw the shapes that requires them.
- Provide a menu to help users
- Feel free to allow users to:

  - input the height of the shapes as an integer. Then update your code so each shape is drawn as the desired size.
  - Offer the option to see the outlines of shapes.
  - Offer the area of shapes based on user input.
  - Feel free to implement classes for this logic.
- Add to this program in whatever way you feel is necessary (must still be text based at the very end).
- Marks will be awarded for your own ingenuity and creativity.

```
Example shapes:

X
XX
XXX
XXXX
XXXXX
XXXXXX
XXXXXXX
XXXXXXXX
XXXXXXXXX
XXXXXXXXXX


       X
      XXX
     XXXXX
    XXXXXXX
   XXXXXXXXX
  XXXXXXXXXXX
 XXXXXXXXXXXXX
XXXXXXXXXXXXXXX



XXXX        XXXX
 XXXX      XXXX
  XXXX    XXXX
   XXXX  XXXX
    XXXXXXXX



   XXXXX
  XXXXXXX
 XXXXXXXXX
XXXXXXXXXXX
XXXXXXXXXXX
XXXXXXXXXXX
XXXXXXXXXXX
XXXXXXXXXXX
 XXXXXXXXX
```

```
  XXXXXXX
   XXXXX




      X
     XXX
    XXXXX
   XXXXXXX
  XXXXXXXXX
 XXXXXXXXXXX
XXXXXXXXXXXXX
XXXXXXXXXXXXXXX
XXXXXXXXXXXXX
 XXXXXXXXXXX
  XXXXXXXXX
   XXXXXXX
    XXXXX
     XXX
      X
```