

COM4013 – Introduction to Software Development

Week 13 – Structures and Logic

Always select “Enable Editing” if prompted by Microsoft Word

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 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;
```

- Create a new Visual Studio Console Application project called **Lab 13 Exercises**. Refer to Week 11's lecture/worksheet/video if you have forgotten how to create a new project/file/program or use the shell code.

Fahrenheit to Celsius (From semester 1 – Week 2)

- Write a function called CelciusToFahrenheit (returns a float) that convert temperature in Celsius to Fahrenheit using the formula:

$$\text{Fahrenheit} = \text{Celsius} * (9.0/5.0) + 32.0$$

Watch the brackets. Display the result as such (25C = 77.00F)

- Test the function in main, by asking the user for the temperature in Celsius
- Make sure the answer is to tow decimal places

A Simple 'while' Loop (From semester 1 – Week 3)

- Rewrite this Python code to C++.

```
counter = 1
while (counter <= 10):
    print( counter )
    counter+=1
```

- Run the code and see that it works.
- Place commas between each number... i.e., 1, 2, 3, 4, etc...

A more complex 'while' Loop (From semester 1 – Week 3)

- Now add a completely new while loop that displays **the even numbers only** from 0 to 100. This will be a very similar piece of code.
 - Be careful about declaring the same variable twice.
 - Assign a new value to the `counter` variable
- Display it on a single line, separated by commas like this...


```
0, 2, 4, 6, 8, 10, 12, 14...
```

- Write a **for loop to reverse the count** from, (inclusive) 100, 98, 96... 58 (inclusive)

Increment, Decrement and Assignment Operators (From semester 1 - Week 3)

- Rewrite this Python code to C++ and write the following code into the main function:
- DO NOT COPY AND PASTE IT INTO C++ - TYPE IT IN!

```

userNumber = int(input("Enter a number, any number: "))
startNumber = userNumber

print("Now add 1...")
userNumber = userNumber + 1
print(f"You now have { userNumber }... ")

print("Double it...")
userNumber = userNumber * 2
print(f"You now have { userNumber }... ")

print("Now add 4...")
userNumber = userNumber + 4
print(f"You now have { userNumber }... ")

print("Now divide by 2...")
userNumber = userNumber / 2
print(f"You now have { userNumber }... ")

print("Now take away the number you first thought of...")
userNumber = userNumber - startNumber
print(f"You now have three : { userNumber }")

```

- Run the code, it's a simple maths trick - the result is always three.
- Update the code to use operators introduced in week 11:

++ += -= *= /=

Make sure that the trick still works after your changes

Create a Structure

- Use the code below to create a SDetails struct variable called myDetails and fill out your details using the dot operator method.

```

#include <iostream>
#include <string>
using namespace std;

struct SDetails
{
    string name;
    int age;
};

int main()
{
    SDetails myDetails;


```

}

- Create a second variable using the SDetails structure. Call it “person2”. This time **initialise** it to the values “Susan” and 17.
- Create a third variable using the SDetails structure. Call it “person3”. This time **initialise** it to the values “Fred” and 21.
 - Output the values of person 3 in order to demonstrate that it works.
- Assign the contents of “person2” to “person3” using the dot operator, i.e. assign individual data members.
- Output the values of the “person3” variable in order to demonstrate that it works. “person3” should now have the values “Susan” and 17.
- Assign the contents of “myDetails” to “person3”. Do not use the dot operator, instead do this just using the method of assigning one structure variable to the other.
- Output the values of the “person3” variable in order to demonstrate that it works. “person3” should now have your name and age.



Employee Records

- Imagine a company with employees. Define a structure called SEmployee to store the name, national insurance number, and salary of the employees.
- Write a constructor for the structure
- Create one variable to store the data for an employee.
- Enter some data into the structure variable using the constructor.
- Output the data in the structure variable.
- Write a **function** outside of the struct to **output the data for a single employee** to the screen. You will need to pass the employee over in the parameter list to the function.
 - ( Advanced: use a **const reference parameter** to gain a speed advantage)
- Demonstrate the function being used.
- Format this output well

- Write a **function** to **read in the data for a single employee**. You will need to pass the employee over in the parameter list to the function. You will have to use a **reference parameter** to change the values of the original variable.
- Demonstrate the function being used.

Stage 2

Write Data to File

- Write a function that **writes** the contents of a single SEmployee structure variable to a file “store.txt”.
- Check that the function has worked by examining the contents of the file.
- Write a function that **reads** the contents of a single structure variable from the file “store.txt” into a single structure variable.

- It is possible to create an array using structures. Look at the code below for an example of syntax:

```
#include <iostream>
#include <string>
using namespace std;

struct SDetails
{
    string name;
    int age;
};

int main()
{
    SDetails manyDetails[4];
}
```

- The code provided above creates a size 4 array. Note that the syntax for creating an array in C++ is slightly different from that of Python.
- Create a function to read in the details of 4 people. You should use a *for* loop to implement this. (If you are familiar with the concept of “magic numbers” then ensure that your program does not use magic numbers, i.e. replace the ‘4’ with a suitable constant and use this constant throughout your program.
- Create a function to output the details of all of the people in the array.
- Create a function to write the details of all of the people in the array to file.
- Create a function to read the details of all of the people in the array to file. Note: this is comparatively easy if you assume that there are the details of exactly 4 people in the file. Increase the size of the array to 100. Can you rewrite the function so that it will deal with an unknown number of people in the file.

- See if you can include appropriate validation and error checking as well.

Advanced Tasks – (PORTFOLIO PIECE)

Advanced Tasks – (PYTHON PORTFOLIO PIECE 4)

Logical Calculator

- I want you to implement a console-based [Truth Table Generator](#) and/or Logic Calculator.
- Use this week's PowerPoint to help you with figuring out the steps needed to create this tool.
- In logic we have the five basic operators: conjunction, disjunction, implication, biconditional, and negation
- Remember that each option is binary 1 meaning true, and 0 meaning false.
- Let the users know if the complex proposition is a tautology.
- Think about how you want users to interact with the text-based tool, how many terms you want to allow (begin with one (maybe p) and make your way up.
 - Create a menu maybe?

Advanced Tasks – (PYTHON PORTFOLIO PIECE 5)

Task Manager

Create a console-based task manager that allows users to manage their tasks efficiently. The program should provide functionality for adding tasks, marking tasks as complete, listing tasks, and removing tasks.

- Add whatever features you feel are needed.
- A menu as well as responses.
- Users should be able to save their list of tasks to file and reload them on resumption of the tool.
- Ensure that the tool is user centric in its design.

Advanced Tasks – (PYTHON PORTFOLIO PIECE 6)

Wordsearch

Create a wordsearch. Read in two text files and search them for words using your very own (algorithms).

- Copy the text below into two text files and name them accordingly.
- Use a two-dimensional array to store the wordsearch, line by line.
- Store the words in another array.
- Write an algorithm to search for the words in the forwards, backwards, orthogonal directions.
 - I do not expect a binary search algorithm for this or any other searching algorithm – you can make your own by learning to traverse the arrays.
- Print out the words found and their index to screen.

wordsearch.txt

WSUNSHINESOWANEYF
AEUCEALTPCONEOTNW
TUIINYTOLEONTHIEON
EHSTNURALREGOLALI
ROVNCYNRRRAINBOWOI
FTUENREARTHQUAKE
ALTSEROFCLHOPENEL
LEORIRVHEAHOEYRAW
LMIWANIATNUOMDEGF
GRNOEOATETEMAEVLL
NHNANREIEOAYLIIEG
BUTTERFLYTSNUHRE

searchterms.txt

SUNNY
RAINBOW
WATERFALL
BUTTERFLY
EAGLE
MOUNTAIN
FOREST
OCEAN
DAYS
FLOWER
RIVER
LION
EARTHQUAKE
SUNSHINE