# COM4013 – Introduction to Software Development
# Week 14 – Arrays, Stacks, and Stack Class

*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 14 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.

**Nested 'for' Loops (From semester 1 – Week 5)**

- Try to do these on your own first...
- Write a 'for' loop that displays 10 X's in a row.

```
XXXXXXXXXX
```

♦ Add a newline print (in the same cell), ***nest*** this 'for' loop in another 'for' loop to display a 10x20 box of X's. **Again, this is shown in the lecture notes in Python**.

*Note: You must use nested for loops in all these exercises.* ***Do not try to solve these problems with many repeated print() statements!***

```
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXX
```

**Create an Array**

- Copy the following constant (go above and outside main) and array into the project:

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

const int SIZE = 4;

int main()
{
    int array[SIZE] = { 2, 5, 8, 3 };
}
```

- Output the first element of the array.  The array index begins at 0.
- Output the last element of the array.  Remember that the last index of the last element of the array will be *1 less* than the size of the array

- Use a *for* loop to output the contents of the array.

- Declare an array (with no size), of type integer and initialise the array in the declaration to the values:

  5 7 9 3

- Print the number of elements in the array:
  sizeof(arrayVarName)/sizeof(arrayVarName[0]
- Print the contents in **reverse** using a **for loop.**

**Rainfall**

- Declare an array of type *int*, of size 6 and call the array rainfall.
- When you declare the array, initialise the elements of the array to the values:
- 4, 2, 0, 5, 1, 3.
- Use a *for* loop to go through the array.
- Calculate the lowest (min) rainfall, the highest (max) rainfall and the total (sum) amount of rainfall.
- Print to the console.

**Nested For Loop Array**

- Write a program that creates and the following 3 by 3 matrix array:

  1 2 3

  4 5 6

  7 8 9

- Use a nested loop to display all of the elements of the array. (See the lecture slide to help you with this).
- Write another loop to find the sum of all of the elements in the array.


**Empty Array Fill**

- Write a program that fills an empty four-element integer array with multiples of two and then displays the contents of each array element.

- The output should look like:

  2 4 6 8

**Summing Two Arrays**

- Write a program to create **three** arrays each of size 3.
- Initialise the first array to:

  1 3 5
- Initialise the second array to:

  2 4 3
- Add the first and second array together and place the result in the third array. You need to add the arrays together element by element. The values held within the third array at the end should be:

  3 7 8

**Array Search**

- Create an integer array of size 6 and initialise it to the values: 4 6 8 5 2 3.
- Write a loop to go through each of the elements of the array.
- Test each of the elements of the array to see if it is equal to a value of 5.
- If the array element is equal to a value of 5 then output "found" and the *index* of the element.
- Test your code. You should find an element with a value of 5. The index of the variable is 3.


**Print Array**

- Write a function called PrintArray. The function accepts a single dimensional integer array of size 10 in its parameter list.
- The function should output all of the elements of any array which is sent to it (see the lecture slides to help with this).
- Initialise an array of size 10 to the values: 8 6 5 4 3 1 2 5 7 8.

- Pass the array to the function PrintArray.

**Set Array Element Values to 0**

- Write a new function called SetToZero. The function has a single parameter: an integer array of size 10.
- The function will set each element to of the array passed to a value of 0, i.e. assign a value of 0 to each element of the array.
- Test your function using SetToZero and PrintArray.

**Print Array**

- Complete the stack class – look at the lecture slides to find the last few methods names
- Test it in main