Programming 1: Tutorial 7

Introduction

For this tutorial you are going to be using arrays in context of basic programming problem solving.

You can either over write previous question solutions with new ones or you can have a project for each of the exercises – don't forget to make them blank applications.

Hopefully you remember the basic layout of a C++ program? If not refer to one of the previous tutorials, last weeks maybe more appropriate as you may wish to add functions to your programs.

It may also help if you go through a design stage for each exercise where you plan and puzzle through the logic before actually coding.

To complete these exercises you may need to reference the notes, either online or use the printed ones provided. Doing this isn't a bad thing; most programmers have to look up the syntax to make sure there code is correct.

There is sample code at the end of the tutorials, typing and running this code may help you when it comes to solving the exercises.

It is also important to understand what is happening with the questions, rather than just typing it into the compiler. If you look at it, run it and don't understand then just ask.

Array Errors

We have looked at the four types of error. I have also mentioned in the lecture how some these kinds of error, run-time and logical can occur when using arrays. We are now going to look at some array code with errors, try inputting and running the code to see what messages are generated.

The following code should result in a random garbage integer to be printed during the 6th pass of the loop, this is because it is randomly accessing some memory and printing out the value.

```
#include <iostream>
#include <string>

using namespace std;

int main()
{     // declare and initialise array
      // of 5 integers to 0
```

```
	int array[5] = {0};
	// print the values off to make sure
	// but overstep the array index causing
	// the program to crash
	for(int i = 0; i < 6; i++)
	{
		cout << array[i] << endl;
	}

	return 0;
}
```

This can result in some strange logical errors, especially if you are reading in the value from that location and then assigning, rather than just printing out.

Attempts to accessing and change the data stored in another location will result in a runtime crash:

```
#include <iostream>
#include <string>

using namespace std;

int main()
{	// declare and initialise array
	// of 5 integers to 0
	int array[5] = {0};
	// print the values off to make sure
	// but overstep the array index causing
	// the program to crash
	for(int i = 0; i < 6; i++)
	{
		array[i] = 10;
	}

	return 0;
}
```

Questions

1) What happens if you write to element 25 in a 24-member array?

2) What is an un-initialised array element?

Exercises

The majority of these exercises require you to use a loop (or two) with the array to solve the problem.

1) Write a program that initialises an array of 25 integers loops through assigning a value of 0 to 24 to the array elements e.g. element 0 = 0, element 1 = 1. The program should then print out the array:

Hints:

Arrays are declared in the following manner:

```
data_type       array_name[array_size];
```

Array elements are accessed in the following way:

```
array_name[/* index */];
```

2) Write a program that defines an array of 26 char and initialises its elements to the characters a, b, c, … z, it then assigns the letter M to the elements containing the characters a, g, p and z.

Finally display the values of the elements of the array.

3) Complete this program. It may be worth compiling it first before moving on to completing it, so you get a better feel for the program.

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

const int MAX = 10;

int main ()
{
    int  numbers[MAX];
    int  index;

    for (index = 0; index < MAX; index++)
    {
        cout << "Please enter an integer number: " << endl;
       // FILL IN Code to store values in the numbers array
    }

    for (index = MAX - 1; index >= 0; index--)
   // FILL IN Code to write values of the numbers array to the screen


    return 0;
}
```

4) Write a function that search's arrays to find the smallest number. For this code an array that contains the following values 2, 3, 5, 7, 9. This array then needs to

be searched to see if it contains a certain value. Test to see if it contains the value 3 first, then alter your code so that the user enters a value and the array is searched to see if it contains that number.

Hints:
You can initialise the array like this:
`array_name = {value1, value2...};`
Array values can be compared just like the data type they contain.

Sample Programs

Basic Initialisation Example 1

```
// This code demonstrates local initialisation
// using curly brackets {}
#include <iostream>

using namespace std;

int main()
{
	const int MAX_SIZE = 10;
	// first number stored should 1
	// rest should be 0
	int X[MAX_SIZE] = {1};

	for(int i = 0; i < MAX_SIZE; i++)
	{
		cout << X[i] << endl;
	}

	return 0;
}
```

Basic Initialisation Example 2

```
// This code demonstrates global initialisation
// using no assignment (=) operators
// To check the difference between local
// and global in this way try moving the
// global array declaration to in main()
#include <iostream>

using namespace std;

const int MAX_SIZE = 10;
// the numbers should be initialised to 0
int X[MAX_SIZE];

int main()
{
	for(int i = 0; i < MAX_SIZE; i++)
	{
		cout << X[i] << endl;
	}

	return 0;
}
```

Basic Initialisation Example 3

```
// This code demonstrates local initialisation
```

```
// using a for loop
#include <iostream>

using namespace std;

int main()
{
	const int MAX_SIZE = 10;
	// the numbers should be initialised to 0
	int X[MAX_SIZE];

	for(int i = 0; i < MAX_SIZE; i++)
	{	// initialise X[index of i] to
		// the value of i
		X[i] = i;
		// print out the value stored in X[index of i]
		cout << X[i] << endl;
	}

	return 0;
}
```