Lab 4 08227 Advanced Programming

This tutorial introduces the reader to pointers.

1.0 Basics

Download the Lab4.zip file and extract the files to G:/08227/Lab4/.

Create a new empty C++ project.

Add the **source.cpp** to the new project.

Located the following code in **source.cpp** file:

```
void exercise1() {
	int a = 10;
	int b = 20;
	int *p = &a;

	cout << "a= " << a << endl;
	cout << "b= " << b << endl;

	// Add your code here

	cout << "a= " << a << endl;
	cout << "b= " << b << endl;
}
```

The program initializes two integer variables **a** and **b**, with the values 10 and 20 respectively. A pointer **p** is initialized to point at **a**.

Compile and run the program.

Add a line of code, at the position indicated by the comment, to assign the value of 100 to **a**, by using only the pointer **p**.

Run the code to check the output.

Now set a breakpoint at the line

```
int a = 10;
```

Run the code to the breakpoint, then single step through the code whilst looking at the variables in the Local window.

Notice how **a** and **b** are initialized with values 10 and 20, and that pointer p is assigned a hexadecimal value. This value is the memory location of **a**.

Open a Memory window. Copy the value of **p** into the address field of the Memory window and confirm that your are looking at variable **a** in memory.

2.0 Poor assumptions

View **source.cpp** within Visual Studio and locate the main method. Comment out the call to exercise1 and uncomment the call to exercise 2.

```
void exercise2() {
     int a = 10;
     int b = 20;
     int c = 30;
     int *p = &b;

     cout << "a= " << a << endl;
     cout << "b= " << b << endl;
     cout << "c= " << c << endl;

     *p = 100;

     cout << "a= " << a << endl;
     cout << "b= " << b << endl;
     cout << "c= " << c << endl;
}
```

Compile and run the program.

Observe the result.

Now we'll attempt to do a quick hack and advance the pointer 4 bytes in memory from the location of variable **b** to the location of variable **c**

After line

```
*p = 100;
```

Add

```
p++;
*p = 200;
```

Compile and run the program.

Is this what you expected?

Set a break point and single step through the code whilst observing the Locals window.

The pointer does get advanced by 4 bytes, but the memory location is invalid. Just because we list variables **a**, **b**, and **c** sequentially in our programme, does not guarantee that the compiler places them contiguous in memory.

If you want to do this sort of pointer arithmetic then you need to guarantee the memory layout. Arrays are a way to achieve this. We'll look at these later in the module.

For now just be careful using pointer arithmetic. This time we were lucky and the C++ run time checking detected the error for us. You cannot rely on the run time finding more complex errors.

3.0 The crash

View **source.cpp** within Visual Studio and locate the main method. Comment out the call to exercise 2 and uncomment the call to exercise 3.

Compile and run the program.

The program crashes, why?

Set a break point at line

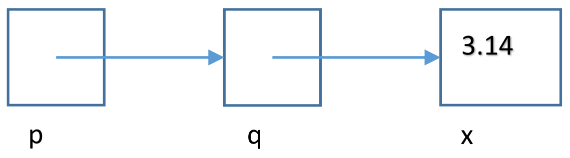
```
    unsigned int a = 0x00ff00ff;
```

Single step through the code and determine the reason for the crash.

The Windows operating system attempts to prevent applications from damaging other applications. This error message is from Windows telling you that your code has attempted to access a memory location outside of its permitted memory footprint.

4.0 Pointers to pointers

View **source.cpp** within Visual Studio and locate the main method. Comment out the call to exercise 3 and uncomment the call to exercise 4.



Add code, at the position identified by the comment, to implement the above pointer chain.

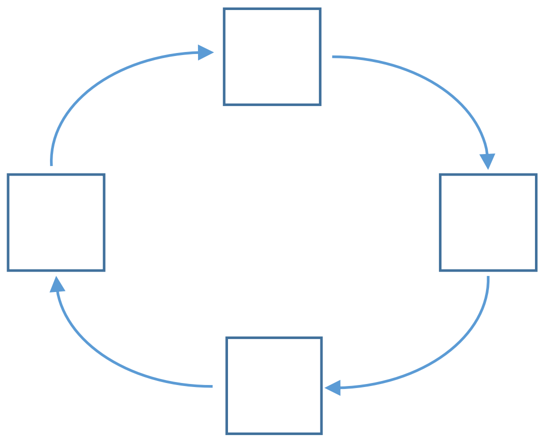
You will need to declare two new pointers **p** and **q**.

Then add the code to change the value of **x** by using only pointer **p**.

Compile and run the program. Checking your solution with the debugger and disassembler.

5.0 Pointer chains (optional)

How would you implement the following in C++?



Hint: You might find **void** useful.