Programming 2: Tutorial 2

The aims for this tutorial are simple.
There is a small exercise that aims to introduce you to random numbers, once you have finished this you are then advised to start on Assessment 1.

Examples

Look at the example code supplied at the end of this tutorial.
Compile and run the examples and make sure you understand what is occurring.

Exercises

To complete these exercises you may need to reference the notes, either online or use the printed ones provided.
There is also sample code available, it has been included in hard copy format at the back of the tutorial notes.

Exercises

Essentially, the computer's memory is made up of bytes. Each byte has a number, an address, associated with it.

- An address is a memory location and will look like a large number when printed out.

1) Type in this code and see what happens:

```
#include <iostream>

using namespace std;

int main()
{
	double	fMine = 4.76;

	cout << "fMine = " << fMine << endl;
	cout << "fMine's Address = " << &fMine << endl;

	return 0;
}
```

And this code:

```
#include <iostream>

using namespace std;

int main()
{
	double	fMine = 4.76;
	double	*iAddy = &fMine;
```

```
	cout << "fMine = " << fMine << endl;
	cout << "fMine's Address = " << iAddy << endl;

	return 0;
}
```

- Question: What is this code doing?

2) Modify your code as shown below and see what happens:

```
#include <iostream>

using namespace std;

int main()
{
	double	fMine = 4.76;
	double	*iAddy = &fMine;

	cout << "fMine = " << fMine << endl;
	cout << "iAddy's Contents = " << *iAddy << endl;

	return 0;
}
```

- Question: What is this code doing?

3) Why do we need & and *

We have now seen 2 distinct steps occur when accessing a variable, and that we can make those steps occur separately.
Alter your code again so it looks like the code below, this will hopefully give some insight as to why they are useful:

```
#include <iostream>

using namespace std;

void Example_Func(double fVar);

int main()
{
	double	fMine = 4.76;

	Example_Func(fMine);

	cout << "fMine = " << fMine << endl;

	return 0;
}

void Example_Func(double fVar)
{
	fVar = 9.99;
```

```
}
```

- Question: What happens and why?

4) Change the code as shown below and answer the question?

```
#include <iostream>

using namespace std;

void Example_Func(double *fPtr);

int main()
{
	double	fMine = 4.76;
	double	*iAddy = &fMine;

	Example_Func(iAddy);

	cout << "fMine = " << fMine << endl;

	return 0;
}

void Example_Func(double *fPtr)
{
	*fPtr = 9.99;
}
```

- Question: What happens and why?

5)

```
#include <iostream>

using namespace std;

int main()
{

	int a = 10;
	int &b = a;
	int *c = &b;

	cout << &a << endl;
	cout << &b << endl;
	cout << &(*c) << endl;

	return 0;
}
```

- Question: What happens and why? Explain what is going inside the code

6) Write a program with a pointer to a pointer to a string object. Use the pointer to the pointer to call the `size()` member function of the `string` object.

7) Write a Boolean valued function which returns "True" if its first string argument is alphabetically smaller than its second string argument, "False" otherwise. You may assume that the two strings contain only lower case letters, and no blanks or other non-alphabetic characters. Test your function with a suitable main program. When you are satisfied it works properly, convert the function to pointer arithmetic syntax, and check that it still behaves in the same way.

9) Write a program that passes an array to a function, the function then randomly accesses an array using pointers to get at an array element. The function then returns the address of the array element and prints out the value stored.

Sample Programs

Pointer Examples

The following code shows many of the ways pointers can be used with variables:

```
// Demonstrates using pointers
#include <iostream>
#include <string>

using namespace std;

int main()
{
    int* pScore = 0;   //declare and initialize a pointer

    int score = 1000;
    //assign pointer pScore address of a variable score
    pScore = &score;

    cout << "Assigning &score to pScore\n";
      //address of score variable
    cout << "&score is: " << &score << "\n";
    cout << "pScore is: " << pScore << "\n";
      //address stored in pointer
    cout << "score is: " << score << "\n";
      //value pointed to by pointer
    cout << "*pScore is: " << *pScore << "\n\n";

    cout << "Adding 500 to score\n";
    score += 500;
    cout << "score is: " << score << "\n";
    cout << "*pScore is: " << *pScore << "\n\n";

    cout << "Adding 500 to *pScore\n";
    *pScore += 500;
    cout << "score is: " << score << "\n";
    cout << "*pScore is: " << *pScore << "\n\n";

    cout << "Assigning &newScore to pScore\n";
    int newScore = 5000;
    pScore = &newScore;
    cout << "&newScore is: " << &newScore << "\n";
    cout << "pScore is: " << pScore << "\n";
    cout << "newScore is: " << newScore << "\n";
```

```
    cout << "*pScore is: " << *pScore << "\n\n";

    return 0;
}
```

Pointers and Strings

The value of RAND_MAX varies between compilers and can be as low as 32767, which would give a range from 0 to 32767 for rand(). To find out the value of RAND_MAX for your compiler run the following small piece of code:

```
// Demonstrates using pointers with strings
#include <iostream>
#include <string>

using namespace std;

int main()
{
    //declare and initialize a pointer
    int* pScore = 0;

    int score = 1000;
      //assign pointer pScore address of a variable score
    pScore = &score;

    cout << "Assigning &str to pStr\n";
    string str = "score";
    string* pStr = &str;   //pointer to string object
    cout << "str is: " << str << "\n";
    cout << "*pStr is: " << *pStr << "\n";
    cout << "(*pStr).size() is: " << (*pStr).size() << "\n";
    cout << "pStr->size() is: " << pStr->size() << "\n";

    return 0;
}
```

Swapping using pointers

This demonstrates the lecture code where we saw how two integers could be swapped.

```
#include <iostream>

using namespace std;

void swapScores(int* const pX, int* const pY);

int main()
{
```

```
    int myScore = 150;
    int yourScore = 1000;
    cout << "Original values\n";
    cout << "myScore: " << myScore << "\n";
    cout << "yourScore: " << yourScore << "\n\n";

    cout << "Calling swapScores()\n";
    swapScores(&myScore, &yourScore);
    cout << "myScore: " << myScore << "\n";
    cout << "yourScore: " << yourScore << "\n";

    return 0;
}

void swapScores(int* const pX, int* const pY)
{
    //store value pointed to by pX in temp
    int temp = *pX;
    //store value pointed to by pY in address pointed to by pX
    *pX = *pY;
//store value originally pointed to by pX in address pointed to by pY
    *pY = temp;
}
```