Object-Oriented Programming Language

03/12/2020

Lab Assignment 2

(**Hint**: if you are not familiar with random number generation in C++, take a quick look of the note below. You can skip it and jump directly to Exercise 1 if you are familiar with the topic)

Random number generation

Computers are often used to generate random numbers. To do so in C++, you need to include the cstdlib header and call the random number generation function rand. Each time rand is called, it returns a random integer in the range of 0 through RAND_MAX (RAND_MAX is a compiler-dependent constant defined in cstdlib). The following code generates ten random numbers.

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

int main()
{
   int i;
   for (i=0; i<10; i++)
      cout << rand() << ' ';
   cout << endl;
   return 0;
}
```

Q: If we want to simulate throwing a dice ten times, will the above code work? If not, how to modify the code for this scenario? You can use the modulus operator %.

A:

```
   for (i=0; i<10; i++)
      cout << rand() % 6 + 1 << ' ';
```

All random numbers are called pseudo-random. This means that **they are part of a predefined set of numbers in the language.** For you to be able to mix up the set each time you run a program, you need to **seed** the generator by making use of the srand() function. A standard method of doing this is to use the computer's internal clock.

The library file ctime contains a function time that uses the computer's internal clock to return an integer. For our purpose it is sufficed to provide an argument of 0 to the function.

```
#include <ctime>

...
```

```
srand(time(0));
...
```

This will seed the generator to the current time object. Essentially, this will make the random number's generator change it's set every time you run the program.

The final code looks like the following:

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

int main()
{
   srand(time(0));
    int i;
   for (i=0; i<10; i++)
      cout << 1.0*rand()/RAND_MAX << ' ';
   cout << endl;
   return 0;
}
```

Exercise 1 (15%): Suppose you want to develop a program to display a sequence of numbers. You will ask the user to input the amount of numbers, as well as the lower and upper bound. Your program will then print out those numbers. The following are sample runs:

```
Please give how many random numbers you want: 20
Please give the lower and upper bounds: 0 5
3 2 1 1 2 3 2 5 1 1 5 1 1 4 5 0 0 1 0 5
```

Exercise 2 (15%): Write a C++ program to output Celsius and Fahrenheit temperature equivalents. Ask the users to enter the range of Celsius temperatures and process the operation in a one-degree interval. The following is the conversion equation just in case you've forgotten.

$$T_f = \frac{9}{5} \cdot T_c + 32$$

Below is a sample run:

```
Please input the range of Celsius temperature values: 1 12
Celsius Fahrenheit
1       33.8
2       35.6
3       37.4
4       39.2
5       41
6       42.8
7       44.6
8       46.4
9       48.2
10      50
11      51.8
12      53.6
```

Exercise 3 (15%): Write a C++ program that requests from the user two integers to be stored in variables *a* and *b* respectively. Please have two integer pointers named *ptrA* and *ptrB* to point to variables *a* and *b* respectively. Please then ask the user to input two new integers to be stored at memory locations pointed by *ptrA* and *ptrB*. Print all four values out.

```
Please input two integers: 1 2
(a,b) = (1,2)
Please input two new integers: 3 4
(a,b,*ptrA,*ptrB) = (3,4,3,4)
end of assignment 3
```

Exercise 4 (15%): Write a C++ program that inputs three vertices then determines if they can form a triangle. You should define the following `struct` in proper header files with header guards. You should then include these two separate header files in your client code.

```
struct Vertex {
    double x;
    double y;
};
```

```
struct Triangle {
    Vertex A;
    Vertex B;
    Vertex C;
};
```

The following are 3 sample runs:

```
Please input the x & y coordinates of the 1st vertex: 1 1
Please input the x & y coordinates of the 2nd vertex: 2 2
Please input the x & y coordinates of the 3rd vertex: 3 3
These points cannot form a triangle!!
```

```
Please input the x & y coordinates of the 1st vertex: 2 2
Please input the x & y coordinates of the 2nd vertex: 2 2
Please input the x & y coordinates of the 3rd vertex: 2 2
These points cannot form a triangle!!
```

```
Please input the x & y coordinates of the 1st vertex: 1 1
Please input the x & y coordinates of the 2nd vertex: 2 2
Please input the x & y coordinates of the 3rd vertex: 1 2
These points can form a triangle!!
```