**Chapter 3 Inserts**

**Insert 129-A**

The C++ library provides everything you need to generate random numbers in your programs. To use these capabilities, you will need the following directive:

#include <random>

To generate random numbers, you will need to define two objects in your program:

* A random number engine
* A distribution object

The *random number engine* generates a sequence of random bits. The *distribution* *object* reads the random bits that are generated by the random number engine and produces random numbers of a specified data type, within a specified range.

The C++ library provides several random number engines that you can use, depending on the needs of your application. We will primarily use the random\_device engine, which is adequate for most applications. Here is an example of how you define a random\_device engine named myEngine:

random\_device myEngine;

Next, you must define a distribution object. If you want random integers, you can define a uniform\_int\_distribution object. Here is an example:

uniform\_int\_distribution<int> randomInt(0, 100);

This statement defines a distribution object named randomInt. Notice the word int that appears inside angled brackets (<>). This specifies that the object will produce numbers of the int data type. You can specify any of the C++ integer types (short, int, long, etc.). The numbers 0 and 100 that appear inside the parentheses specify the minimum and maximum values for the random number. In this example, the distribution object will generate numbers that are in the range 0 through 100.

Assuming we have defined a random number engine named **myEngine** and a distribution object named **randomInt**, the following statement will generate a random number and assign it to the number variable:

int number = randomInt(myEngine);

The following statement will generate a random number and display it on the screen:

cout << randomInt(myEngine) << endl;

You can also generate random floating-point numbers by defining a uniform\_real\_distribution object. Here is an example:

uniform\_real\_distribution<double> randomReal(0.0, 1.0);

This statement defines a distribution object named randomReal. Notice the word double that appears inside angled brackets (<>). This specifies that the object will produce numbers of the double data type. You can specify any of the C++ floating-point types (float, double, or long double). The numbers 0.0 and 1.0 that appear inside the parentheses specify the minimum and maximum values for the random number. In this example, the distribution object will generate numbers that are in the range 0.0 through 1.0.

Assuming we have defined a random number engine named myEngine and a distribution object named randomReal, the following statement will generate a random floating-point number and assign it to the number variable:

double number = randomReal(myEngine);

The following *In the Spotlight* section demonstrates how to use random integers to simulate rolling dice.

**Insert 130-A**

1 // This program simulates rolling dice.  
 2 #include <iostream>  
 3 #include <random>  
 4 using namespace std;  
 5   
 6 int main()  
 7 {  
 8 // Constants  
 9 const int MIN = 1; // Minimum dice value  
10 const int MAX = 6; // Maximum dice value  
11   
12 // Random number engine  
13 random\_device engine;  
14   
15 // Distribution object  
16 uniform\_int\_distribution<int> diceValue(MIN, MAX);  
17   
18 cout << "Rolling the dice...\n";  
19 cout << diceValue(engine) << endl;  
20 cout << diceValue(engine) << endl;  
21 return 0;  
22 }

**Insert 132-A**

**Checkpoint: [COMP: This replaces Checkpoint 3.35 currently in Revel]**

Assume the following statements appear in a program:

random\_device engine;

uniform\_int\_distribution<int> randInt(1, 99);

The minimum random number that will be produced by the randInt object is \_\_\_\_\_\_\_\_\_\_\_.

**Checkpoint: [COMP: This replaces Checkpoint 3.36 currently in Revel]**

Assume the following statements appear in a program:

random\_device engine;

uniform\_int\_distribution<int> randInt(1, 99);

The maximum random number that will be produced by the randInt object is \_\_\_\_\_\_\_\_\_\_\_.

**Checkpoint: [COMP: This replaces Checkpoint 3.37 currently in Revel]**

Assume the following statements appear in a program:

random\_device engine;

uniform\_int\_distribution<int> randInt(1, 99);

Write a statement that uses the randInt object to get a random number and assigns that number to a variable named randNumber.

**In MPL**

**Ex- no: 00016**