Laboratory V-A

CS-102

Spring 2022

Laboratory V-A – Part 1

- Program 5-21 shows how to write a file using a loop.
- Program 5-24 shows how to read a file using a loop.
- These are two of the most frequent operations that we do with loops.
- Type in both of these programs.
- Use Program 5-21 to type in the sales for five days of business, creating a file called Sales.txt.
- Use Program 5-24 to display the results of Sales.txt.

Laboratory V-A – Part 2

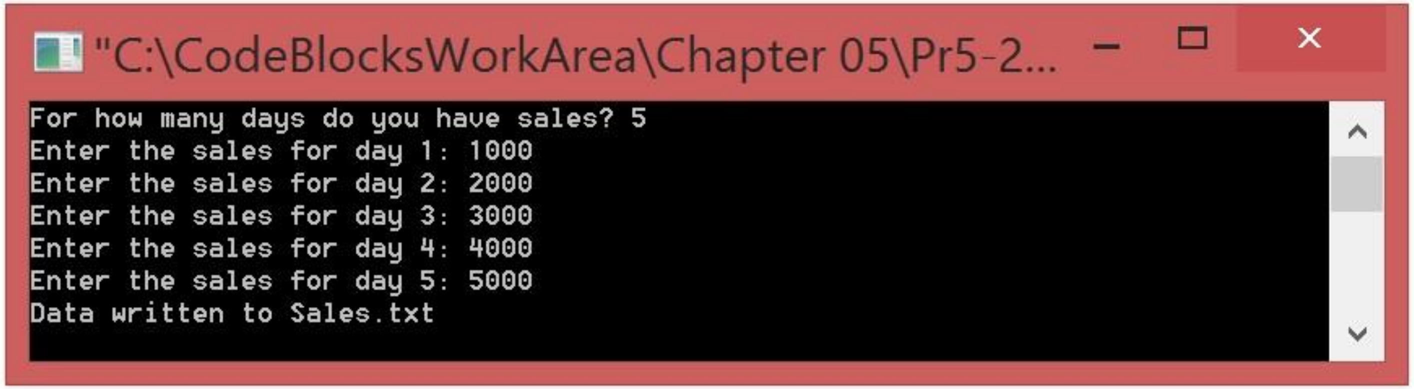
- Now modify Program 5-21 so that it accepts strings instead of numbers.
- Likewise modify Program 5-24 so that it also accepts strings.
- Use your modified version of Program 5-21 to type in the planets in our solar system (plus the dwarf planet Pluto).
    - Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto
    - Call your file: Planets.txt .
- Call your program: *YourName*-LabVA-2a.cpp
- Use the modified version of Program 5-24 to display the file Planets.txt.
- Call your program: *YourName*-LabVA-2b.cpp
- If you are doing LabVA synchronously, show the instructor the results of running these programs.
- If you are doing LabVA asynchronously, submit *YourName*-LabVA-2.cpp and Planets.txt to Canvas.

Program5-21: Using For Loop to control length of data entry

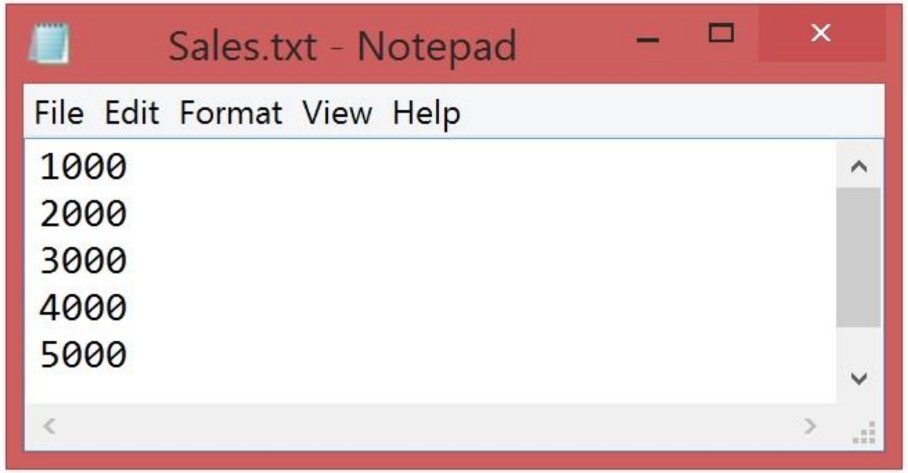
```
// This program writes user input to a file.
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
        ofstream outputFile;  // File stream object
        int numberOfDays;     // Number of days of sales
        double sales;         // Sales amount for a day
        // Get the number of days.
        cout << "For how many days do you have sales? ";
        cin >> numberOfDays;
        // Open a file named Sales.txt.
        outputFile.open("Sales.txt");
```

Program5-21 Concluded

```
        // Get the sales for each day and write it to the file.
        for (int count = 1; count <= numberOfDays; count++)
        {
                // Get the sales for a day.
                cout << "Enter the sales for day "
                << count << ": ";
                cin >> sales;
                // Write the sales to the file.
                outputFile << sales << endl;
        }
        // Close the file.
        outputFile.close();
        cout << "Data written to Sales.txt\n";
        return 0;
}
```



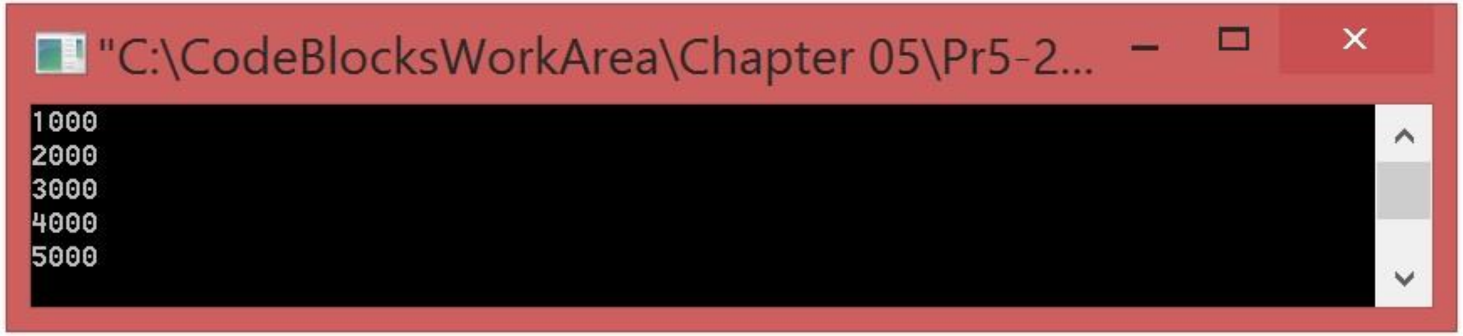
```
For how many days do you have sales? 5
Enter the sales for day 1: 1000
Enter the sales for day 2: 2000
Enter the sales for day 3: 3000
Enter the sales for day 4: 4000
Enter the sales for day 5: 5000
Data written to Sales.txt
```



```
1000
2000
3000
4000
5000
```

Program 5-24 Using a While Loop to Detect the End of a File

```
// This program reads data from a file of unkown length.
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
        ifstream inputFile;
        int number;
        // Open the file.
        inputFile.open(“Sales.txt");
        // Read the numbers from the file and
        // display them.
        while (inputFile >> number)
         {
                cout << number << endl;
         }
        // Close the file.
        inputFile.close();
        return 0;
}
```



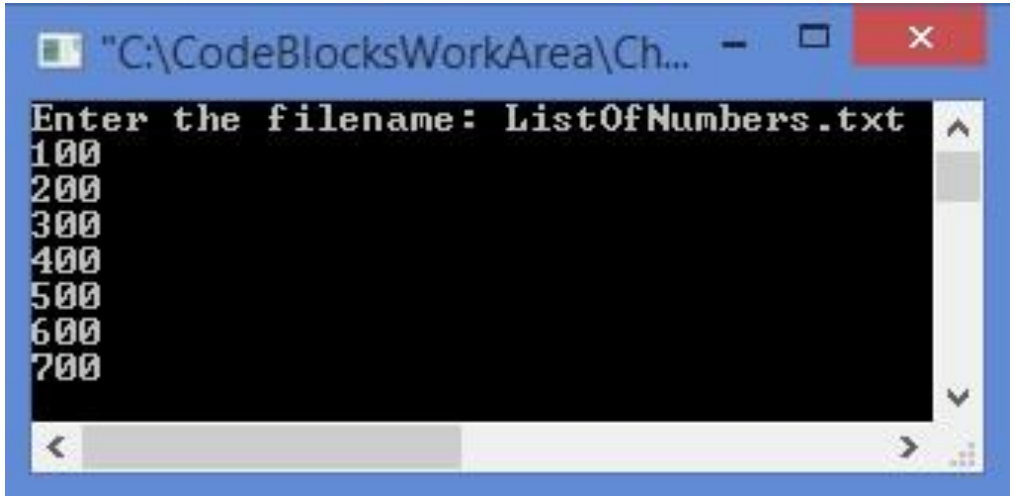
```
1000
2000
3000
4000
5000
```

Letting the User Specify a Filename in Program 5-24

```
// This program lets the user enter a filename.
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

int main()
{
	ifstream inputFile;
	string filename;
	int number;

	// Get the filename from the user.
	cout << "Enter the filename: ";
	cin >> filename;

	// Open the file.
	inputFile.open(filename);      // Without C++ 11 compiler you can use: inputFile.open(filename.c_str());
```

Letting the User Specify a Filename in Program 5-24 Concluded

```
          // If the file successfully opened, process it.
          if (inputFile)
           {
                    // Read the numbers from the file and
                    // display them.
                    while (inputFile >> number)
                    {
                              cout << number << endl;
                    }

                    // Close the file.
                    inputFile.close();
           }
          else
          {
                    // Display an error message.
                    cout << "Error opening the file.\n";
          }
          return 0;
}
```



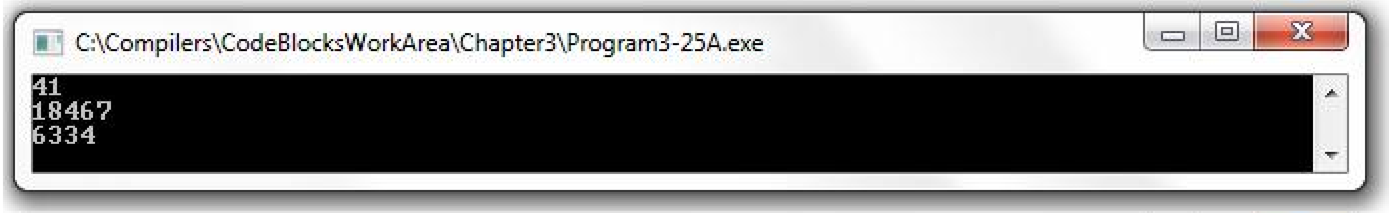
```
Enter the filename: ListOfNumbers.txt
100
200
300
400
500
600
700
```

Lab V-A – Part 3: The problem with Pseudo-Random Number Generators

- To generate pseudo-random numbers in C++, you need to include the **<cstdlib>** library.
- The random number is generated by **rand()**
- Indeed, if you run the rand() function three times, you will get 3 different pseudo-random numbers.
- But each time you run the program, you'll get the same three pseudo-random numbers.

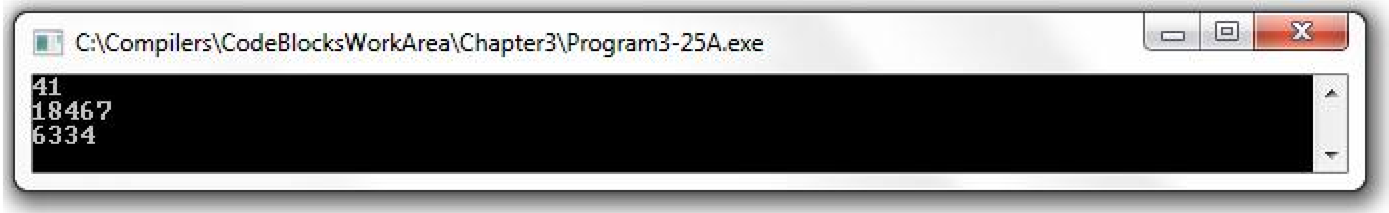
Problem with generating Pseudo-Random Numbers

```
// This program demonstrates pseudo random numbers.
#include <iostream>
#include <cstdlib>      // For rand
using namespace std;
int main()
{
   // Display three random numbers.
   cout << rand() << endl;
   cout << rand() << endl;
   cout << rand() << endl;
   return 0;
}
```



C:\Compilers\CodeBlocksWorkArea\Chapter3\Program3-25A.exe

```
41
18467
6334
```



C:\Compilers\CodeBlocksWorkArea\Chapter3\Program3-25A.exe

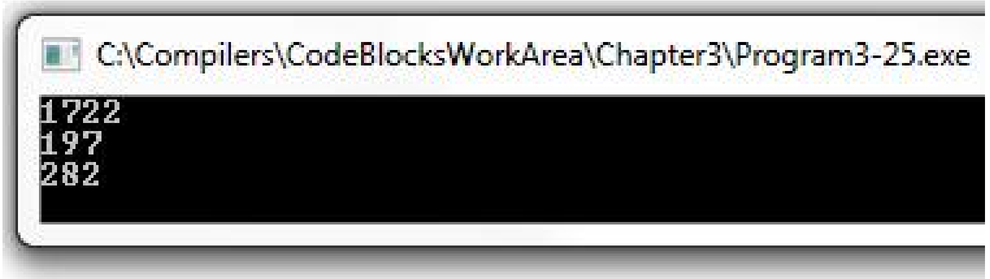
```
41
18467
6334
```

Seeding the Pseudo-Random number generator solves this problem

- The pseudo-random number generator generates a new random number by taking the previous random number it generated and performing some randomizing process to it to generate the new number.
- If you don't give it an initial number or "seed" to begin with, it will choose its own, by default – and it will always choose the same number.
- So we use the srand() function to set the initial value of the pseudo-random number generator to a particular seed value.
- To ensure that we get a different number, each time we use the pseudo-random number generator, we use a value that we are sure will be different every time.
- What better value to use than the exact time of day, which will be different each time you examine it.
- The **time(0)** function generates the number of seconds that have passed since January 1, 1970.
- However, it changes its value about every 12 seconds. So if you call for a new value of time(0) before 12 seconds has elapsed you are likely to get the same seed as you did previously.
- There are better time generators out there than time(0) if you want more precise clocks.

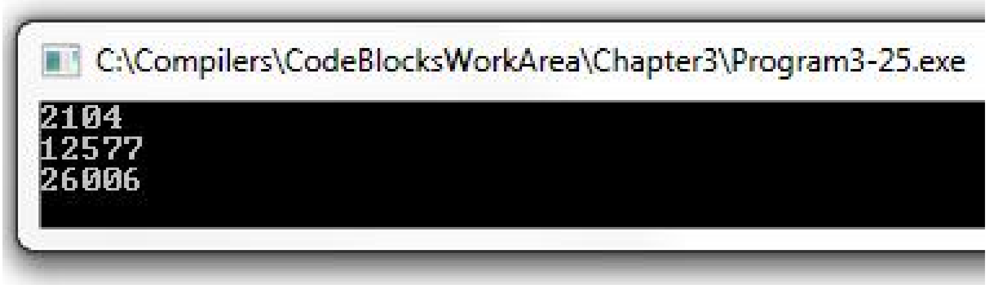
Generating Unique Pseudo-Random #'s

```
// This program demonstrates pseudo-random numbers.
#include <iostream>
#include <cstdlib>     // For rand and srand
#include <ctime>       // For the time function
using namespace std;
int main()
{
   // Get the system time.
   unsigned seed = time(0);
   // Seed the random number generator.
   srand(seed);
   // Display three random numbers.
   cout << rand() << endl;
   cout << rand() << endl;
   cout << rand() << endl;
   return 0;
}
```



C:\Compilers\CodeBlocksWorkArea\Chapter3\Program3-25.exe

```
1722
197
282
```



C:\Compilers\CodeBlocksWorkArea\Chapter3\Program3-25.exe

```
2104
12577
26006
```

Suppose you want to define a range for the Pseudo-Random #'s

- Suppose you wanted to generate a pseudo-random number between 1 and 100 you could use:

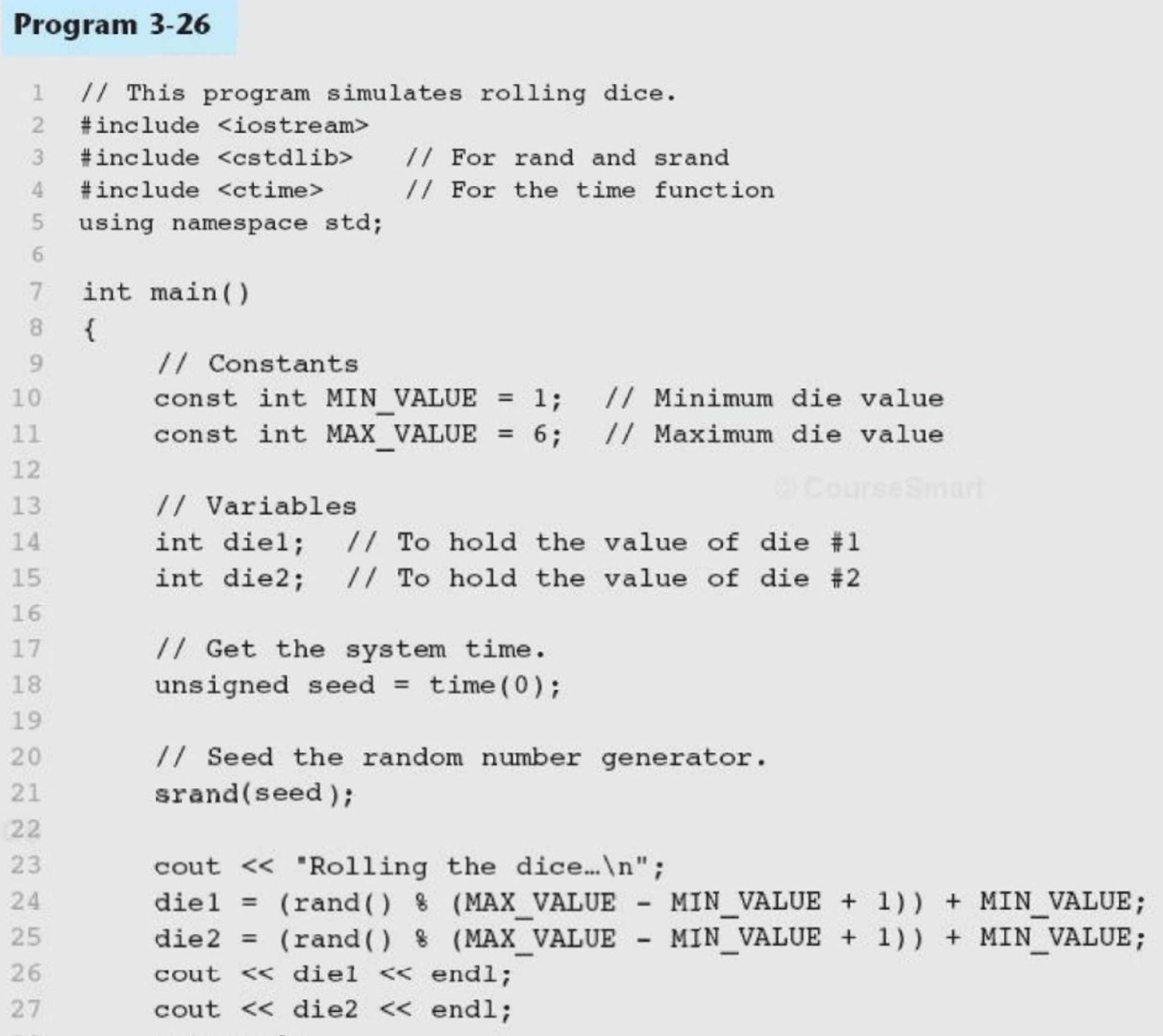
```
y = 1  +  rand()  %  100;
```

- This will divide whatever the pseudo-random number is by 100, and then take remainder, which will be less than 100.
  - Since you wanted the range to start at 1, not zero (and zero is what you'd get if rand() returned a value of 5000, for example), you must add a 1 to the result.
- In the general case, if you wanted a pseudo-random number between 1 and some maxRange, then you'd use:

```
y = 1  +  rand()  %  maxRange;
```

Lab V-A – Part 3

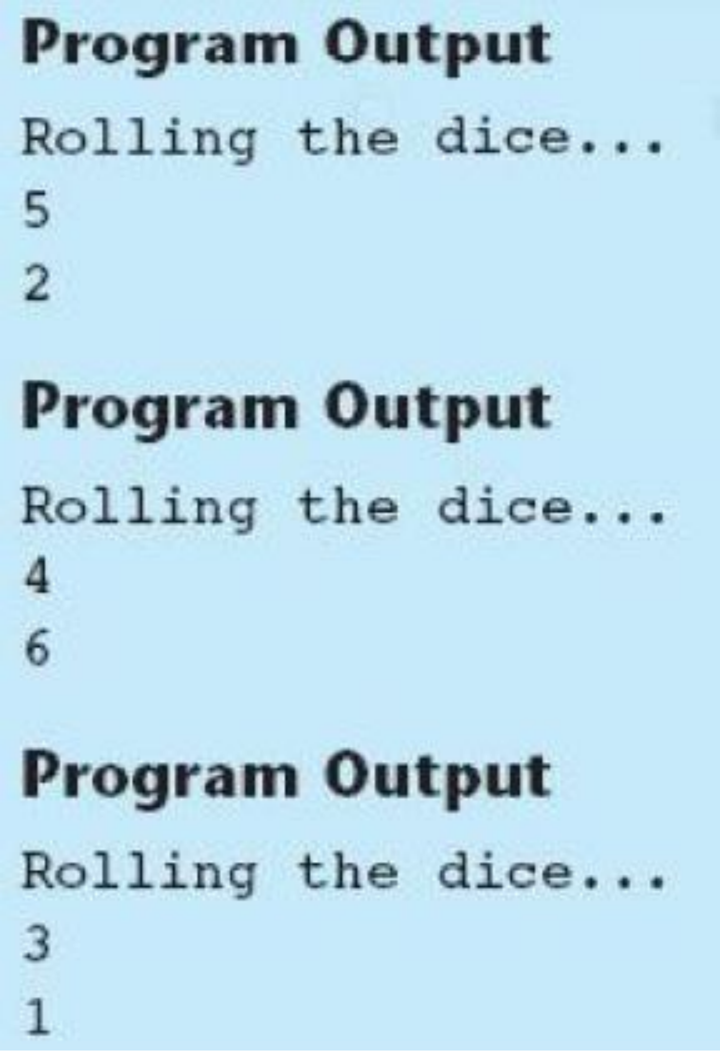
- Generate two files called “randnums1.txt” and “randnums2.txt” that will have 1000 integers each in the range of 1 to 10,000.
- Call your program: *YourName*-LabVA-3a.cpp
- Write a program which reads randnums1.txt as well as randnums2.txt.
- You will call this program: *YourName*-LabVA-3b.cpp
- If you are doing LabVA synchronously, show the instructor the results that you get for <u>two</u> tries at determining the mean.
- If you are doing LabVA asynchronously, submit: *YourName*-LabVA-3a.cpp and *YourName*-LabVA-3b.cpp to Canvas.



Program 3-26

```
// This program simulates rolling dice.
#include <iostream>
#include <cstdlib>   // For rand and srand
#include <ctime>     // For the time function
using namespace std;

int main()
{
     // Constants
     const int MIN_VALUE = 1;  // Minimum die value
     const int MAX_VALUE = 6;  // Maximum die value

     // Variables
     int die1;  // To hold the value of die #1
     int die2;  // To hold the value of die #2

     // Get the system time.
     unsigned seed = time(0);

     // Seed the random number generator.
     srand(seed);

     cout << "Rolling the dice…\n";
     die1 = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
     die2 = (rand() % (MAX_VALUE - MIN_VALUE + 1)) + MIN_VALUE;
     cout << die1 << endl;
     cout << die2 << endl;
```

Program 3-26 Runs



Program Output

```
Rolling the dice...
5
2
```

Program Output

```
Rolling the dice...
4
6
```

Program Output

```
Rolling the dice...
3
1
```

Lab V-A – Part 4

- Write a program that will determine the largest number in the file, **randnums1.txt** or **randnums2.txt**, that you produced in part 3.
- Now modify that program so that it will determine, in addition, the smallest number in **randnums1.txt** or **randnums2.txt** as well as the largest number in the file.
- Call your program: *YourName*-LabVA-4.cpp
- If you are doing LabVA synchronously, show the instructor the results that you get for both the largest as well as the smallest number in the file.
- If you are doing LabVA asynchronously, submit: *YourName*-LabVA-4.cpp, to Canvas.

Lab V-A – Part 5

- Now write a program so that it will determine, the average number in the files **randnums1.txt** or **randnums2.txt**.
    - The program should also display the number of numbers in the file.
- Call your program: *YourName*-LabVA-4.cpp
- If you are doing LabVA synchronously, show the instructor the results that you get for computing the average of all the numbers in either the file **randnums1.txt or randnums2.txt**.
- If you are doing LabVA asynchronously, submit: *YourName*-LabVA-5.cpp, to Canvas.