**Recitation Activity: Data Generation Using the Random Number Generator**

**Objectives:** In this recitation activity, students will:

* learn how to use the Random Number Generator (RNG)
* create data to test and/or simulate a system
* prepare for Assignment #6

**Motivation & Background:** Data Generation is always seen to be a complex problem, especially for new systems (*software applications*). This process may require some study on how people will use the system. If you are replacing an existing system, or automating a task that people have been doing manually, you would not need to generate data because you can easily collect “real” inputs. The problem is when you try to simulate/test new systems which nobody has tried/used before. Examples of applications which might require data generation are:

1. Assume, you were asked to write a program to play a special coin-flipper-game, which is played by two or more players. The players take turn throwing 5 different coins. In each trial, the number of heads and tails are counted. The player who first scores 50,000 heads or tails wins the game.
2. Another example is [playing a card shuffler](https://www.random.org/playing-cards/) game – in which, each player will draw cards from 5 shuffled decks. Please note that each deck has 52 different cards. Total number of possible combinations is 2,598,960 (*52 choose 5*), which is a huge number.
3. A third example is rolling dice to play a board game – where each die has 6 faces. Board games are usually played with multiple players. These type of games require rolling dice many times until one of the players wins the game. How would you create this data to test your software?
4. Creating data for a nationwide/worldwide exam is also a very challenging task. For standardized exams like (SAT, IQ, IELTS, TOEFL, *etc*.) - which are administered and scored in a standard manner, the monthly attempts are usually huge. Assume the number of students attempted the TOEFL test this month is 500,000 students, and the number of question in each exam is 80 multiple choice questions (A, B, C, D, and E) and 35 TRUE/FALSE. How would you create the data for this test?
5. Other examples for creating data include eye exam at an eye doctor’s office, describing various products (*shape, size, color, weight, texture, …*) in a factory, *etc*.

All these examples require creating a tremendous amount of data – impossible to create manually. Automatic creation of data is possible in C++ using the random number generator function rand() from the <cstdlib> library. Please note that the random numbers created by computer programs are not a 100% random, however they are pseudo random, which means they are generated in a predictable fashion. Even when using complicated mathematical formula to calculate these number, they cannot be truly random. The randomness can be enhanced by seeding the rand() function.

**Activity:** In this activity we will create numbers between 16 and 35. To start, open Code::Blocks and create a new project for this activity. Next, clean your file to look like:

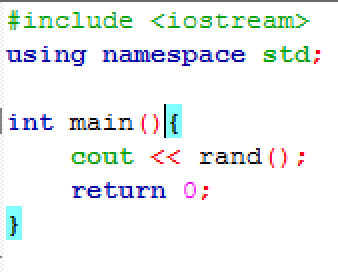
#include <iostream>

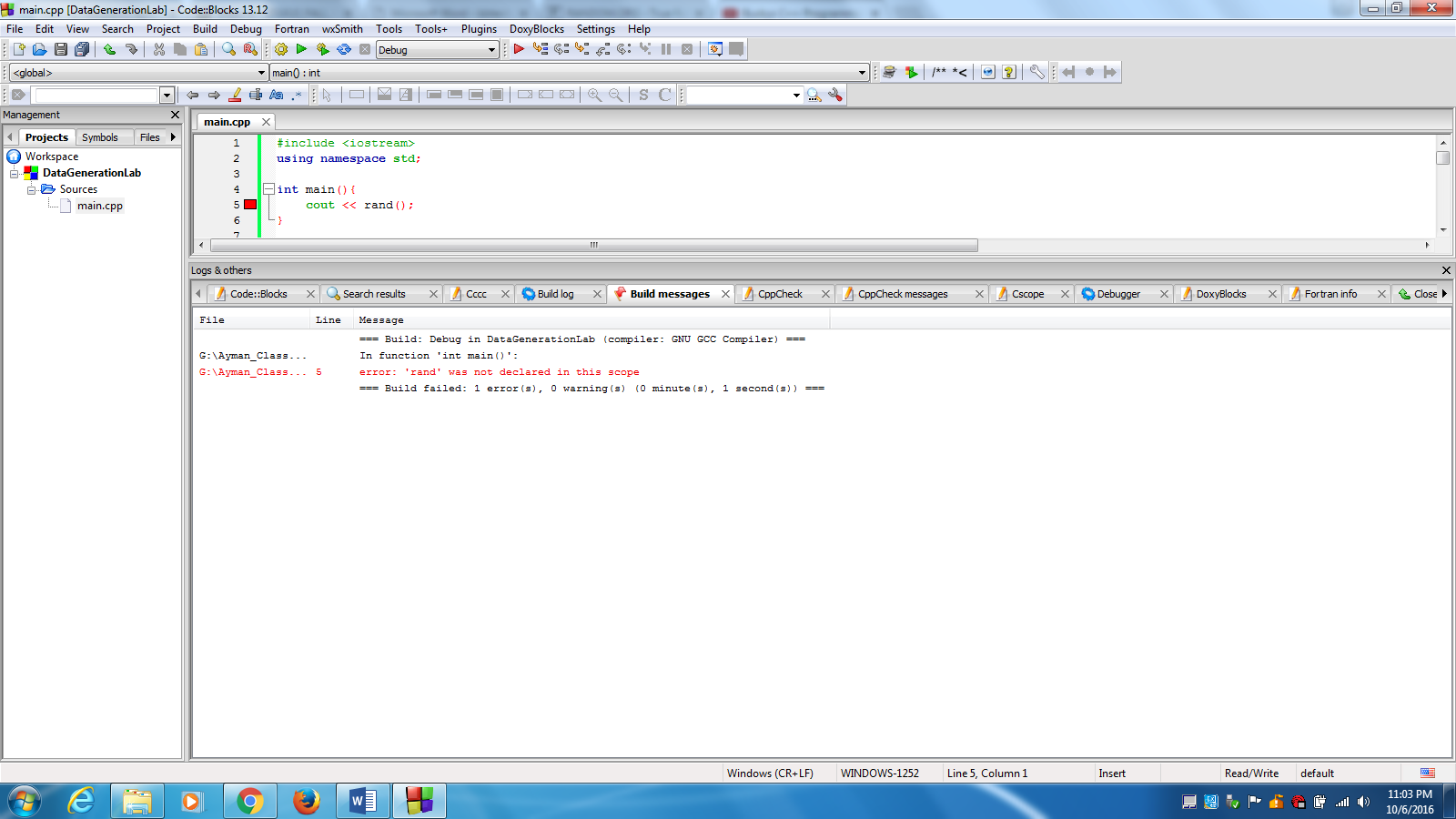
using namespace std;

int main(){

return 0;

}

Next, call the function rand() to display a random number, as follows:



* **Q:** Why does the compiler generate this type of error? How to fix it?
  + **Ans.:** The function rand() is one of the standard library functions, which requires the <cstdlib> library.

After fixing this error and recompilation, a random number should be displayed. Run your executable file multiple times.

* **Q:** What number is displayed in each run?
* **Ans.:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

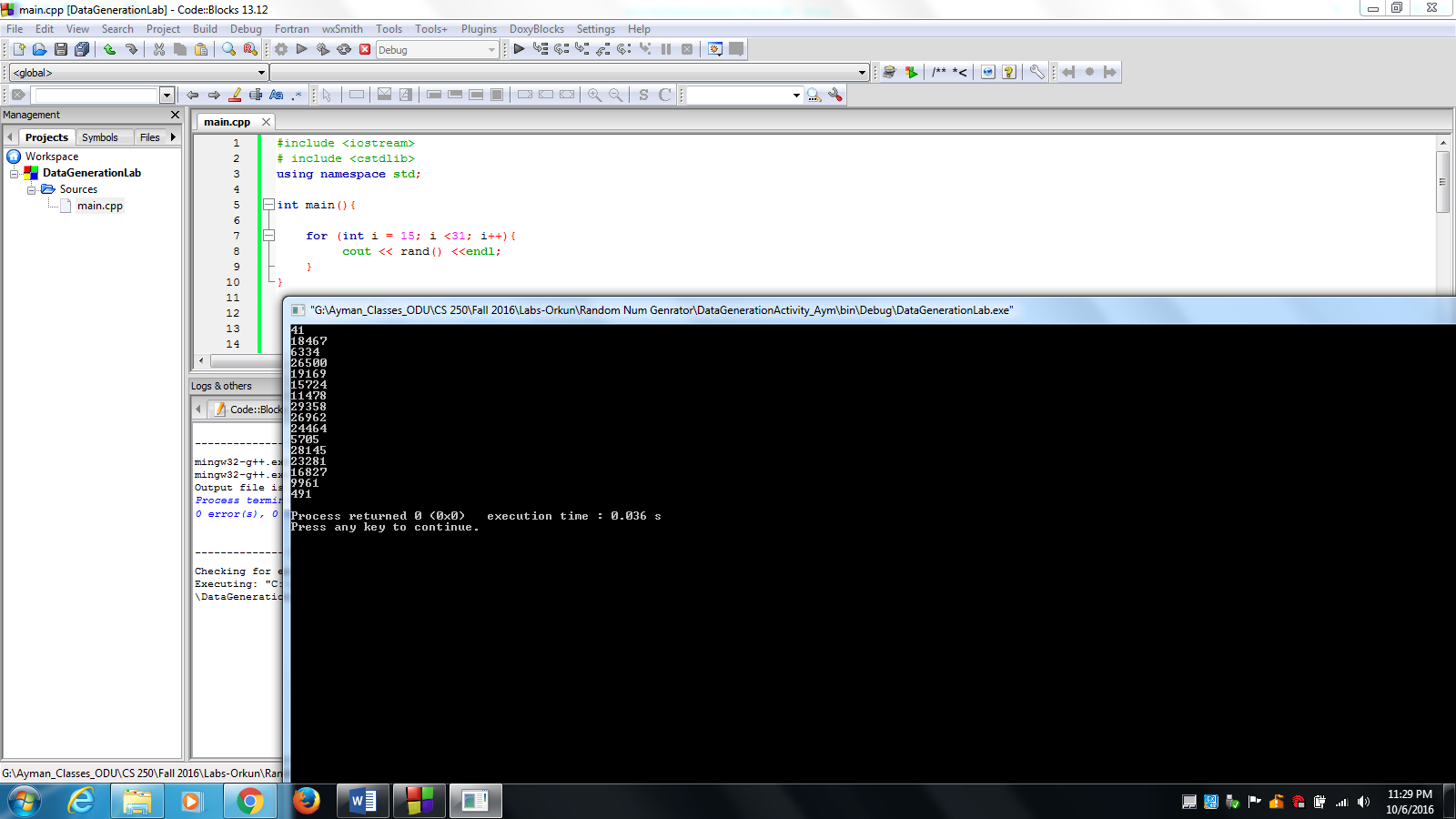
Re-compile the program and run again.

* **Q:** Has the generated number changed? Why?
* **Ans.:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Since we need to create numbers between 16 and 35, let us call the rand() function from a loop.

* **Q:** what numbers are displayed?
* **Ans.:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

You should be getting a set of number like this:



These numbers seem random, are not they? Is this a good thing? **YES/NO**

But, the created numbers are not between 16 and 35 as I wanted. How can I fix that?

To fix that, you will need to think about the numbers which you want to generate. In fact, we need to display 20 random numbers from 16 to 35. This fix will be implemented in two steps:

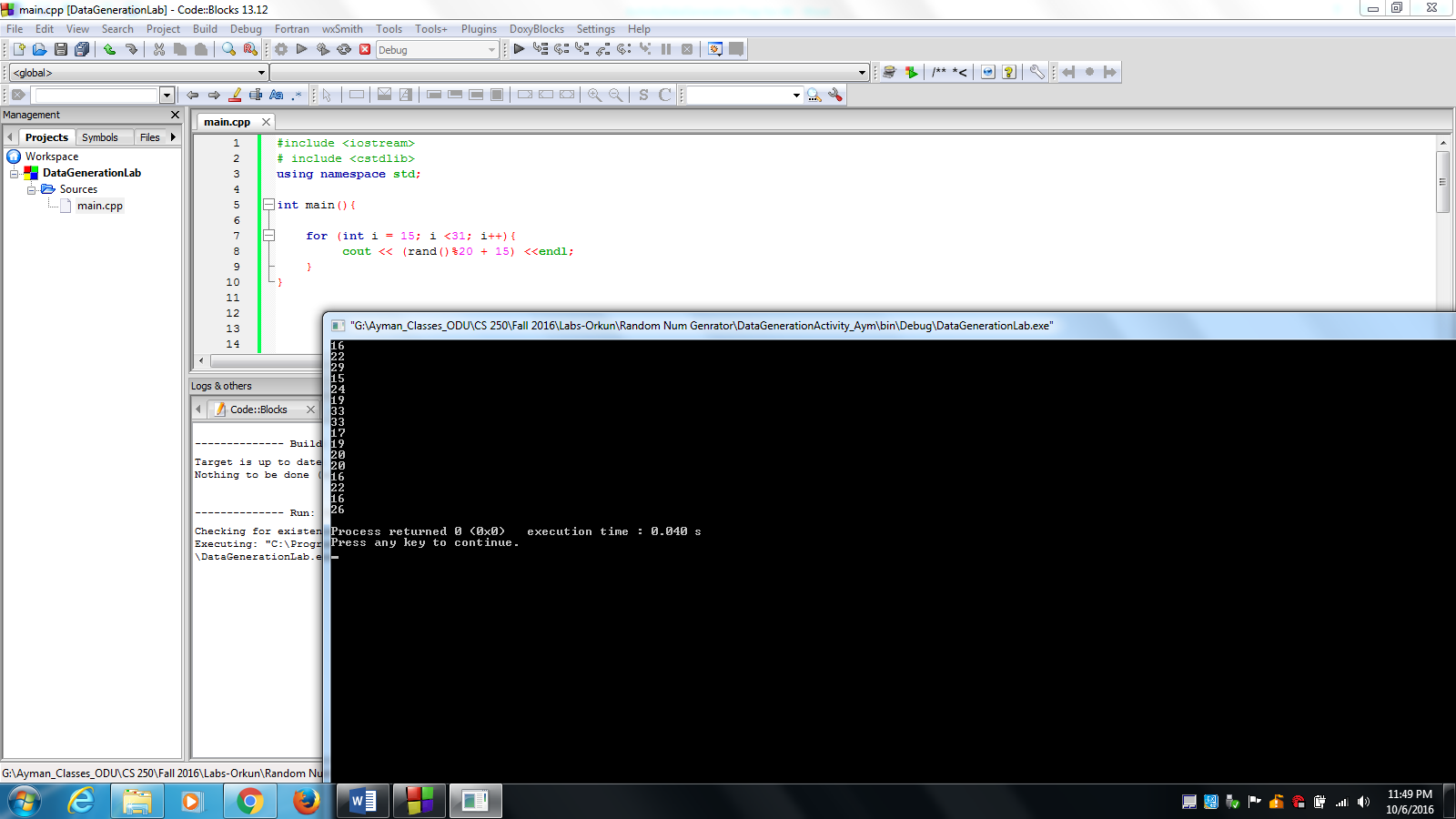
* ***Step-1:*** To display 20 different numbers, we will need to calculate the remainder of the division by 20. This process will generate numbers between 1 and 20. Compile multiple time and display your output to display numbers between 1 and 20.
* ***Step-2:*** we need to shift the numbers to be between 15 and 35. How?
  + **Ans.:** add 15 to the randomly generated numbers, as follow:

for (int i = 16; i <35; i++){

cout << (rand()%20 + 15) <<endl;

}

The number displayed should be as follow:

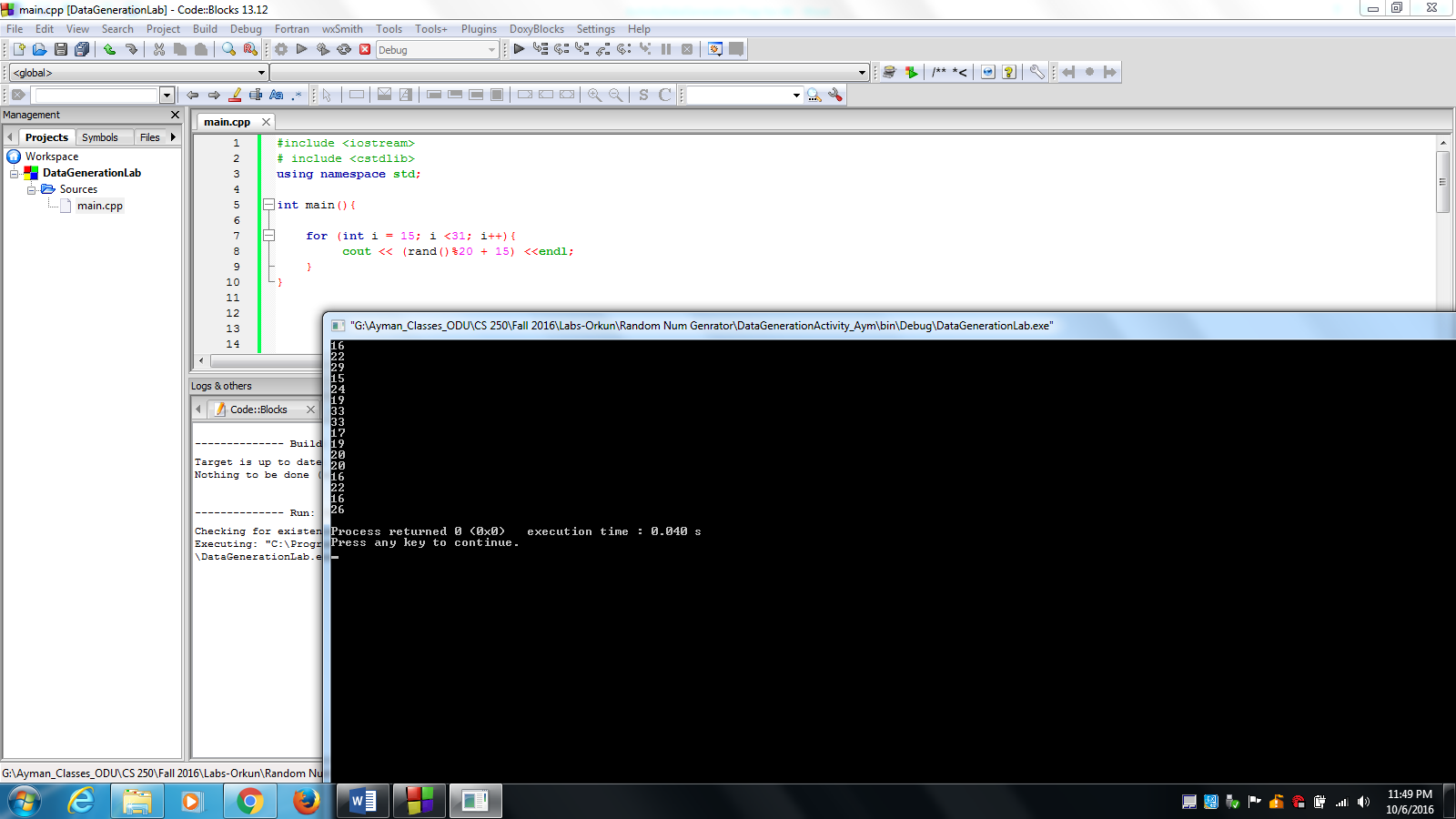
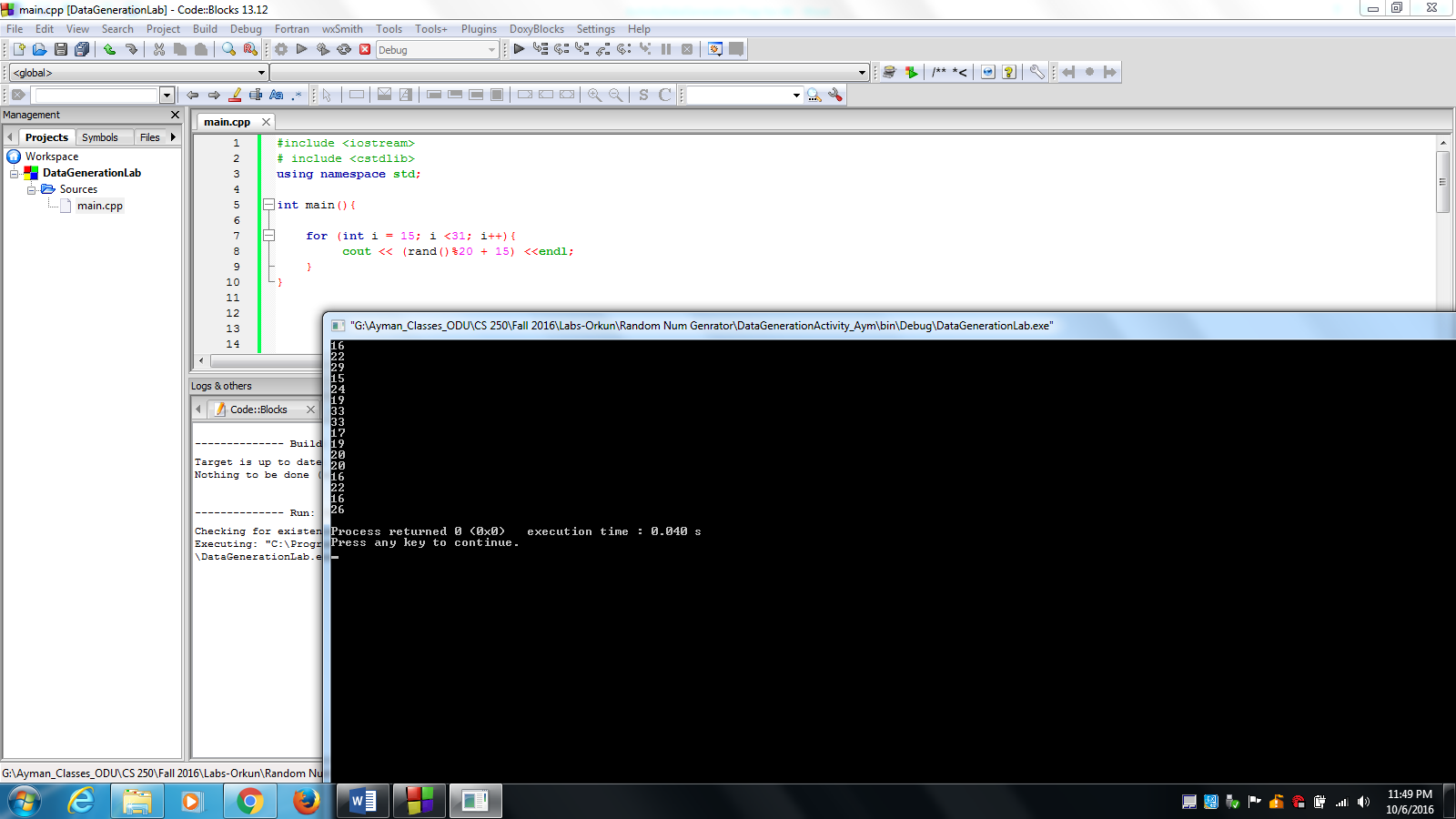
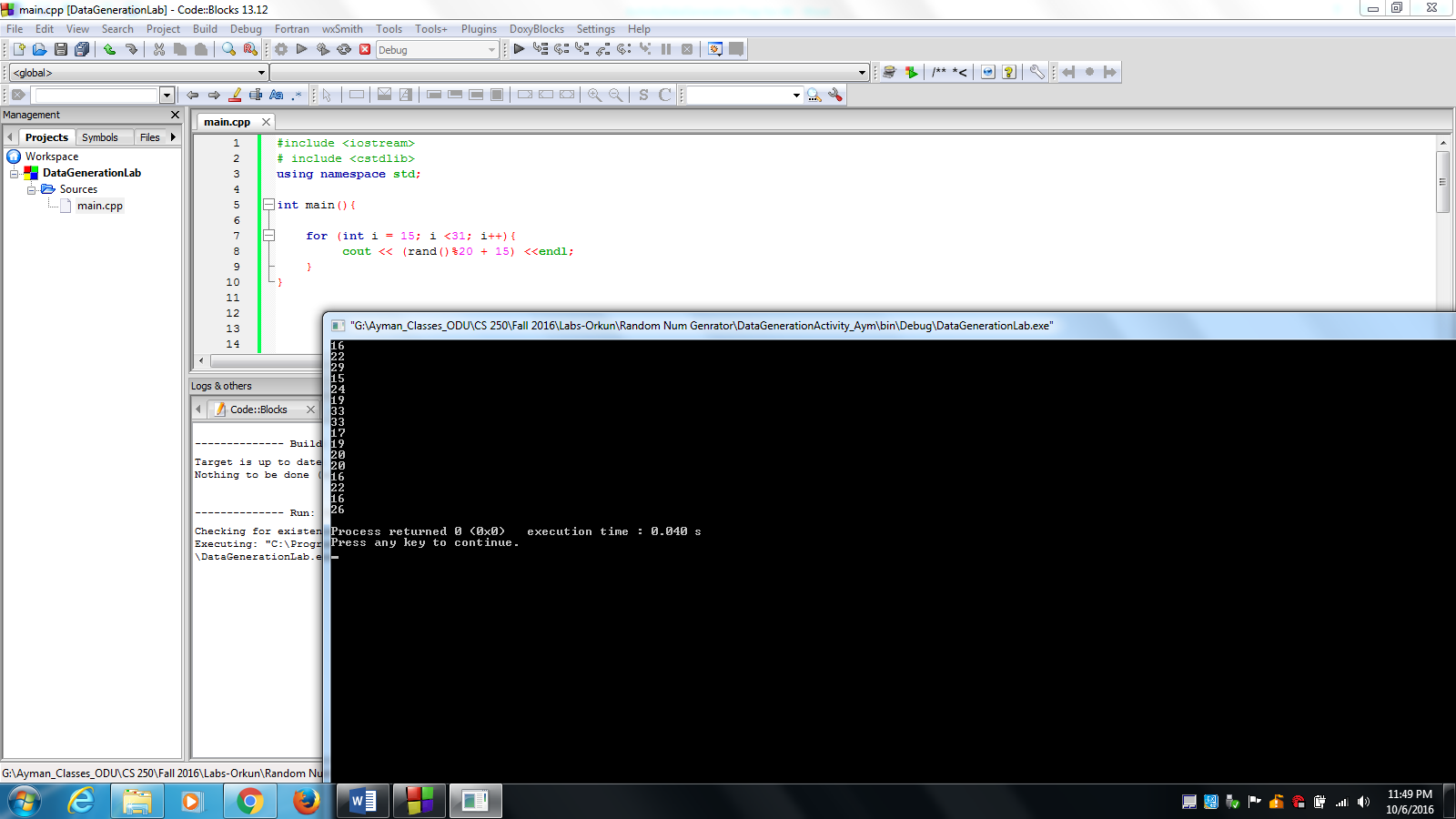
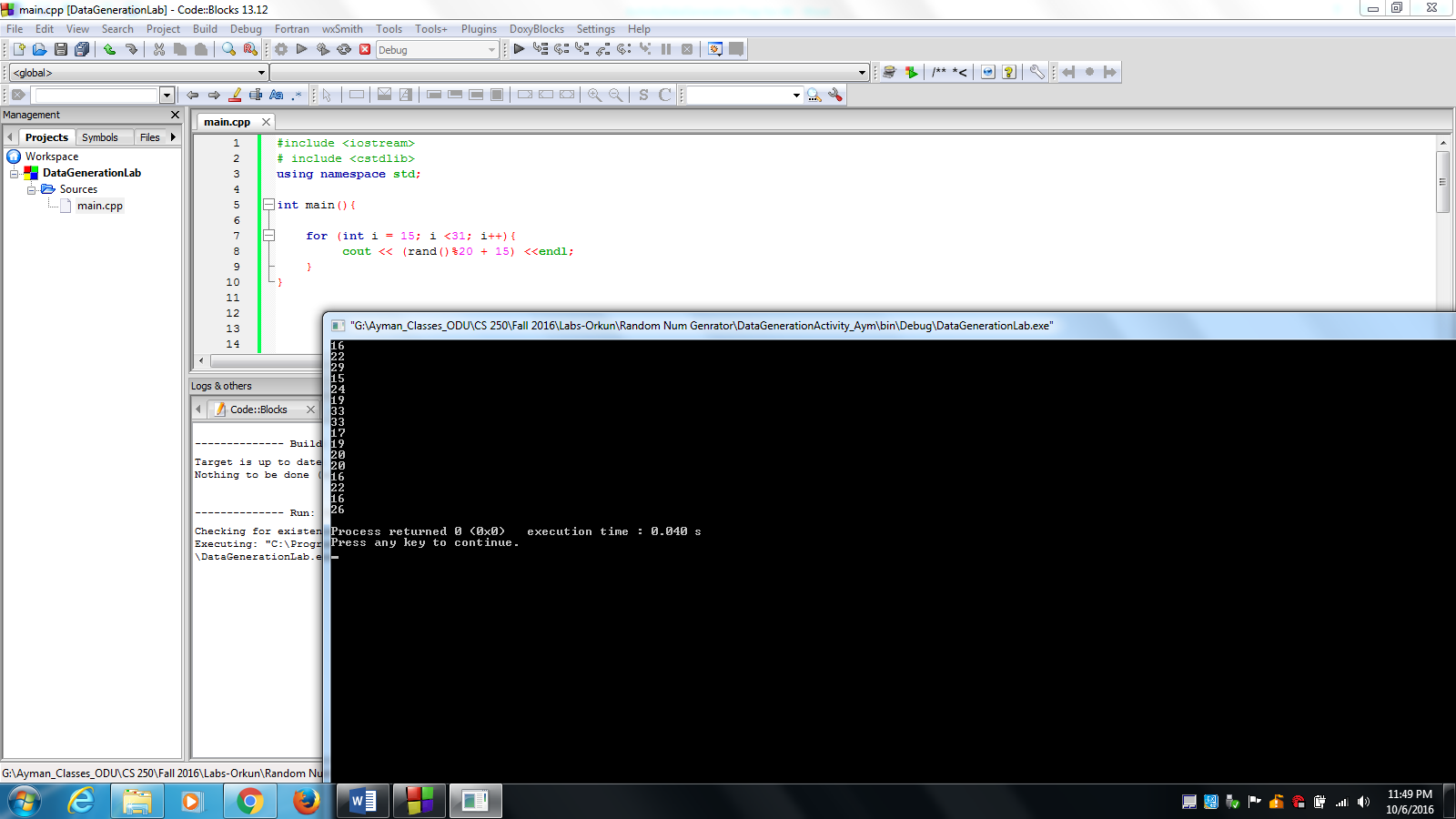
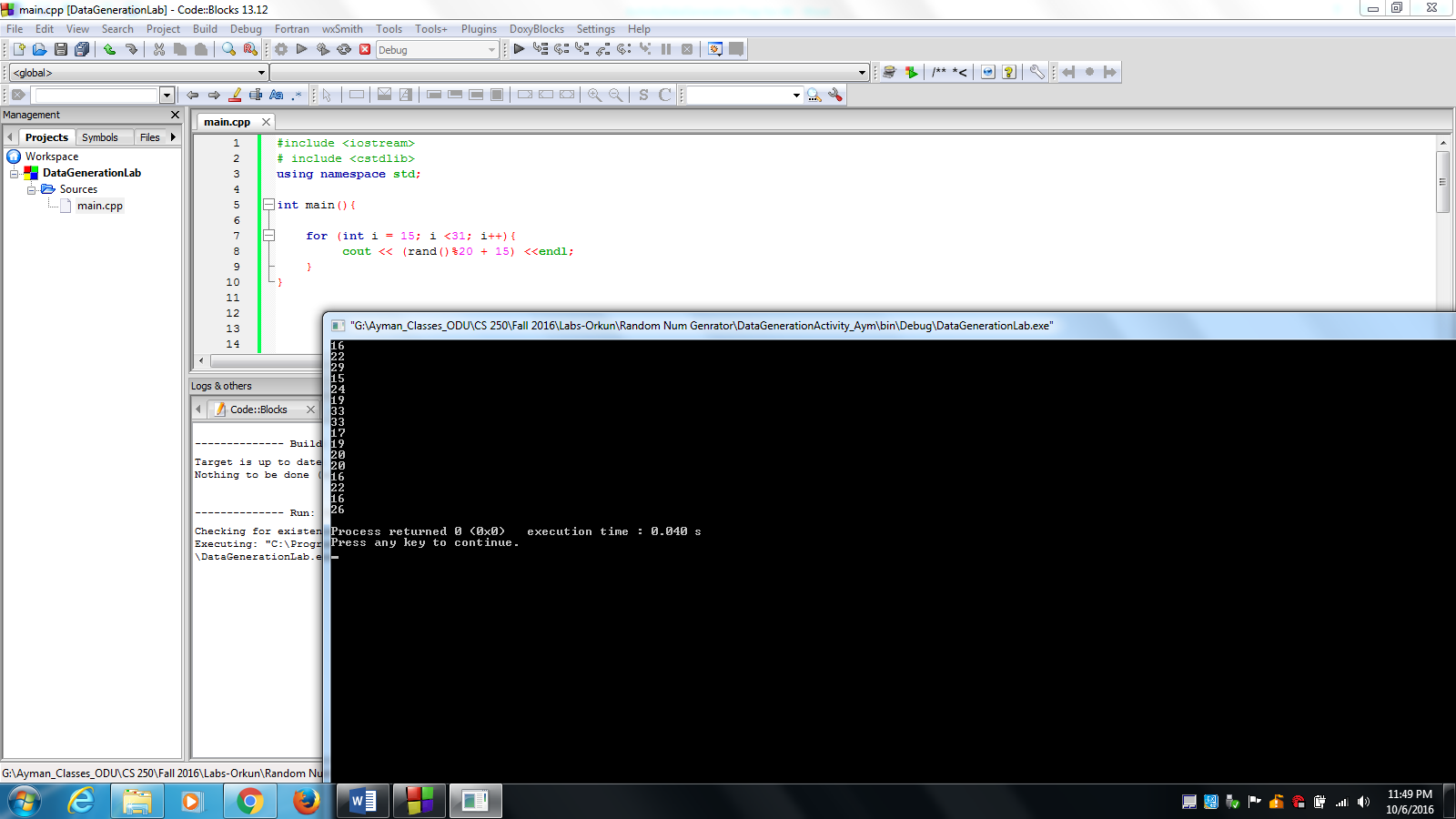
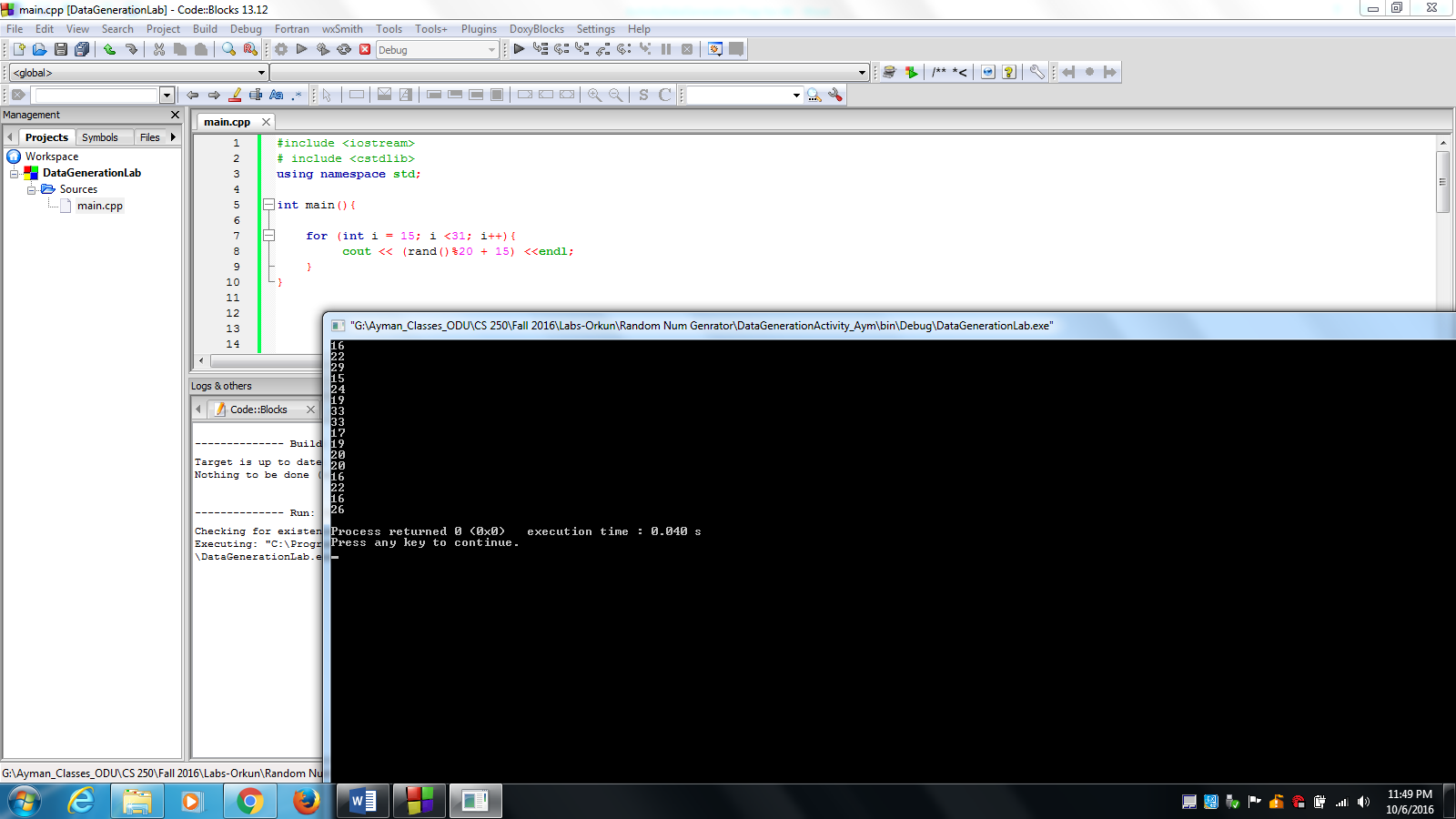
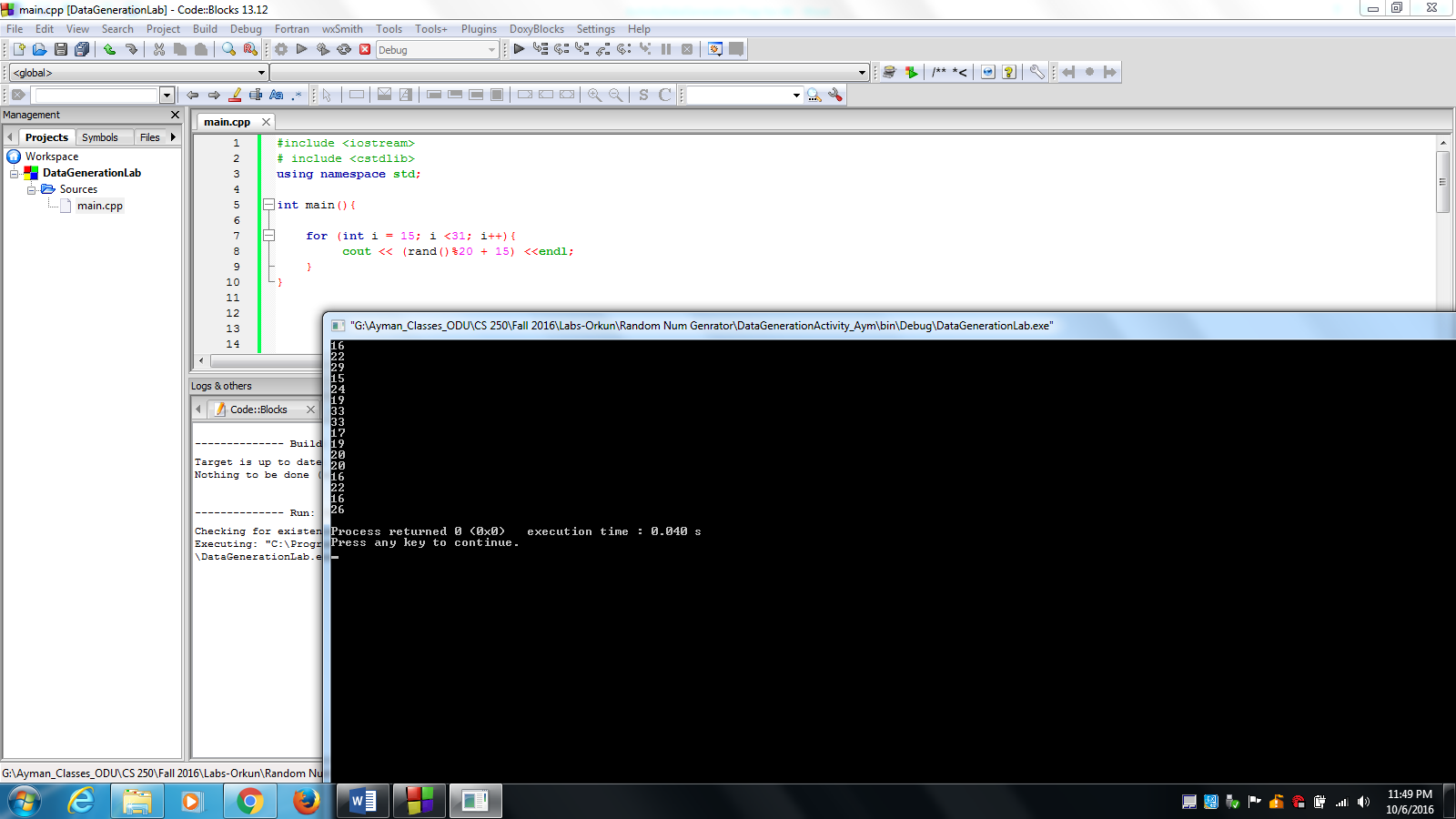
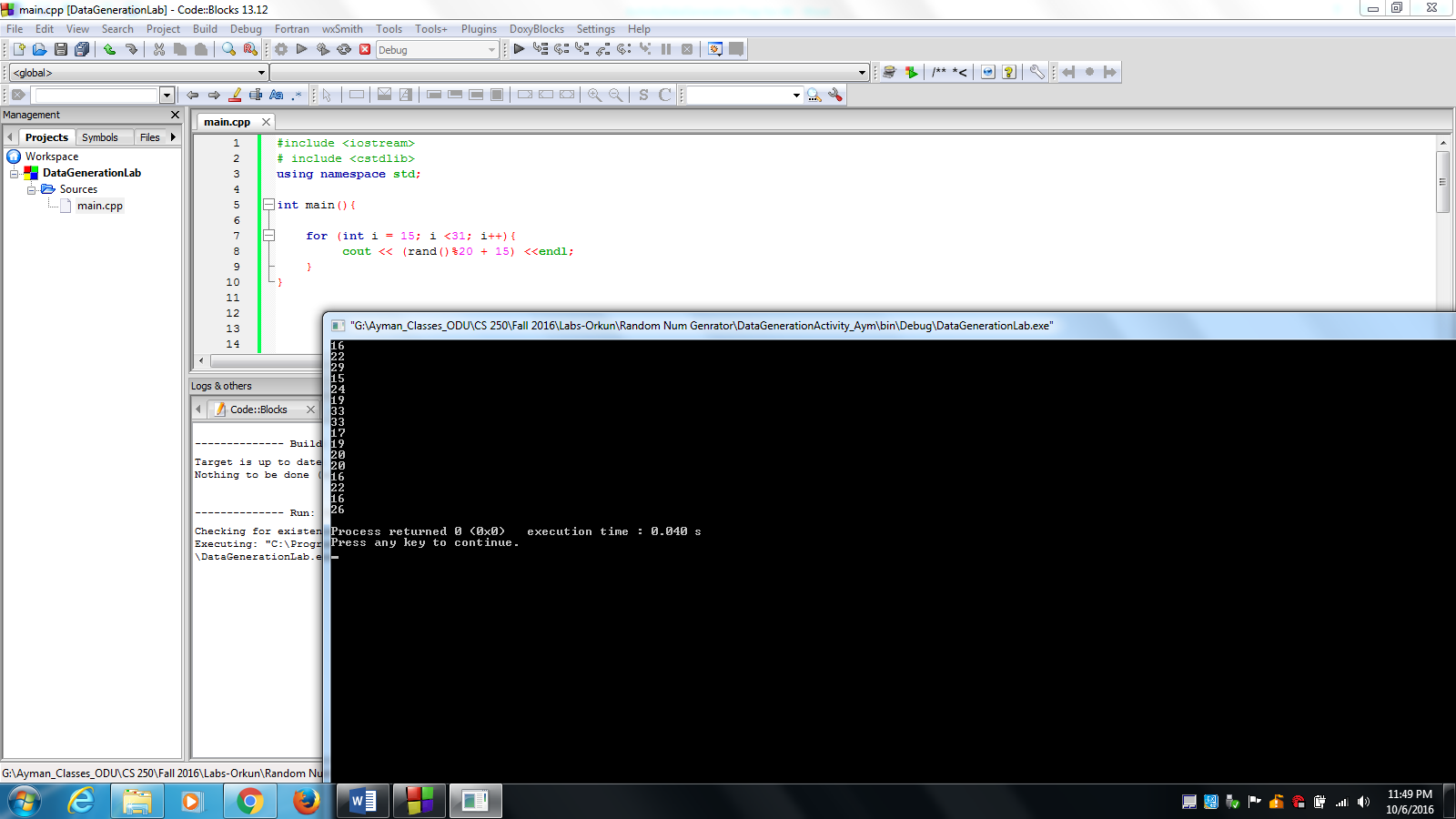
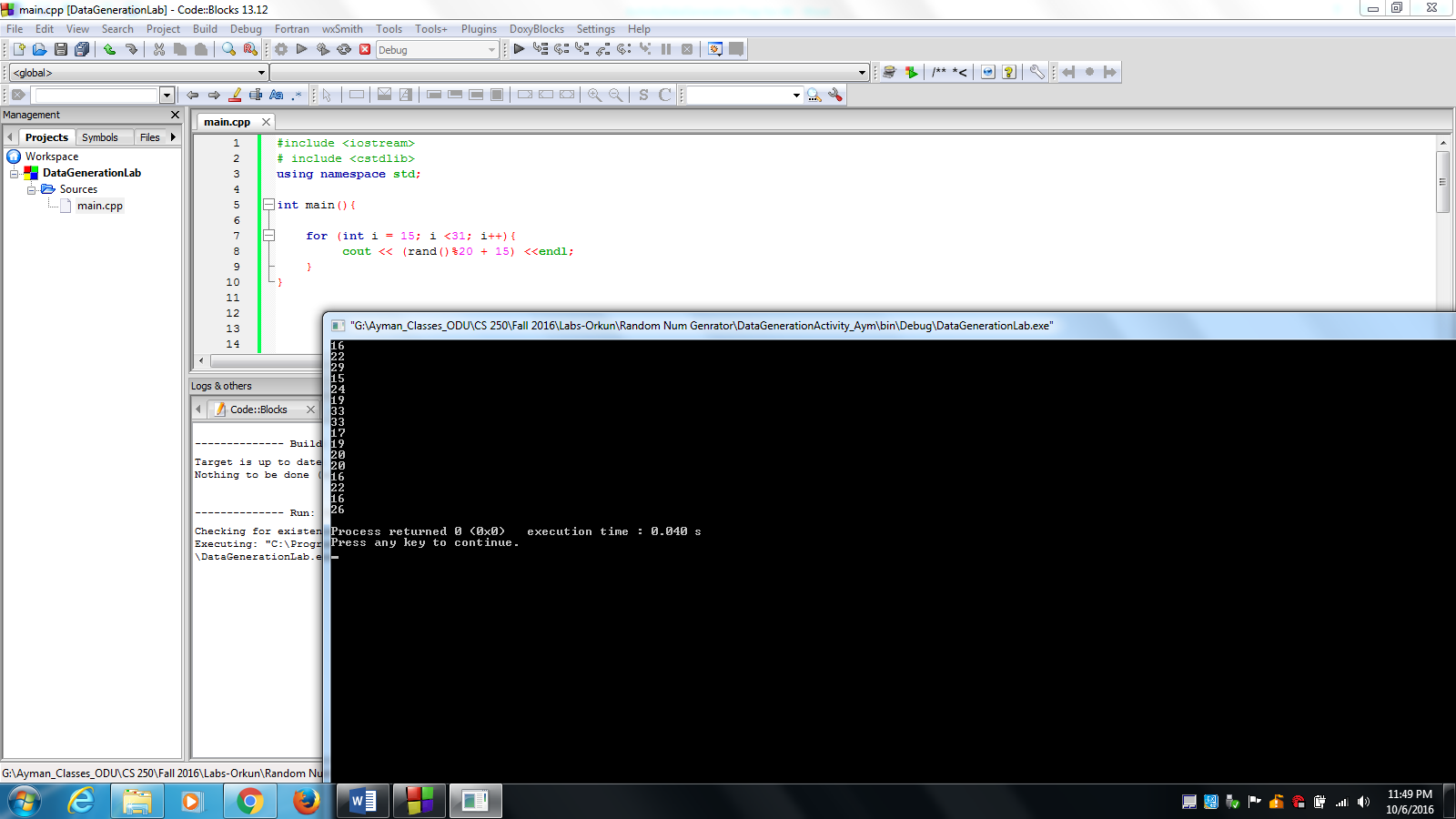
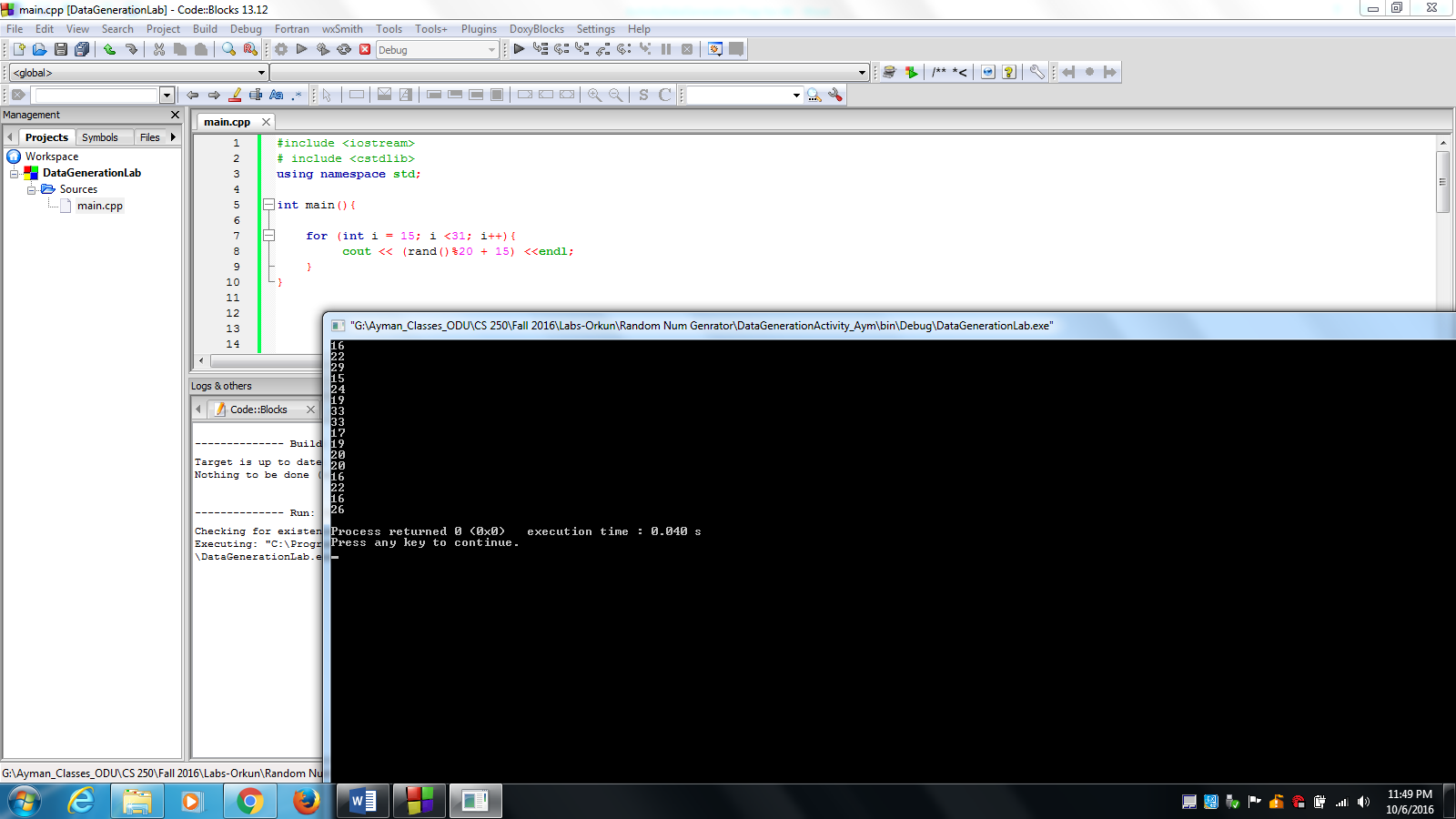
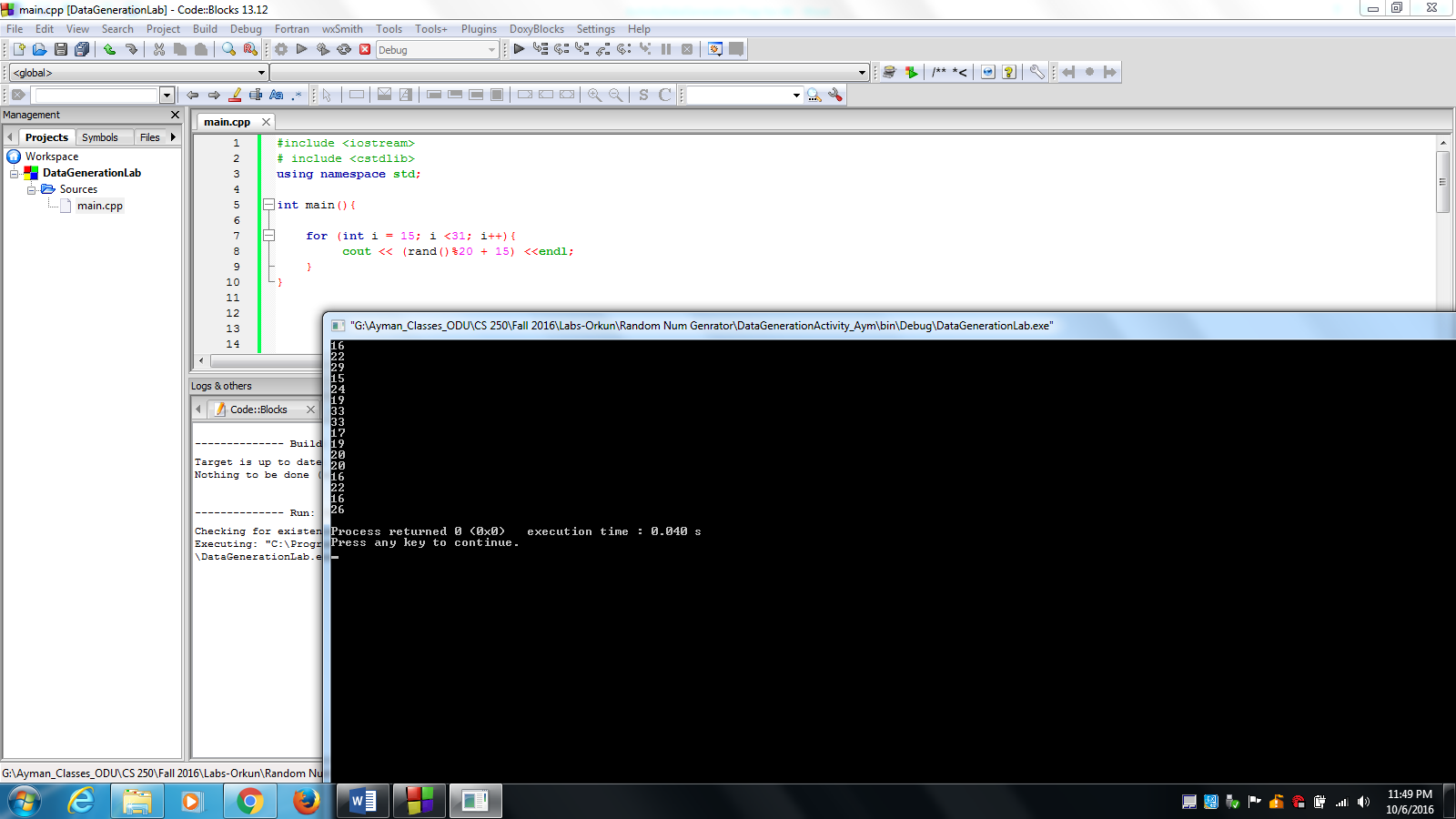
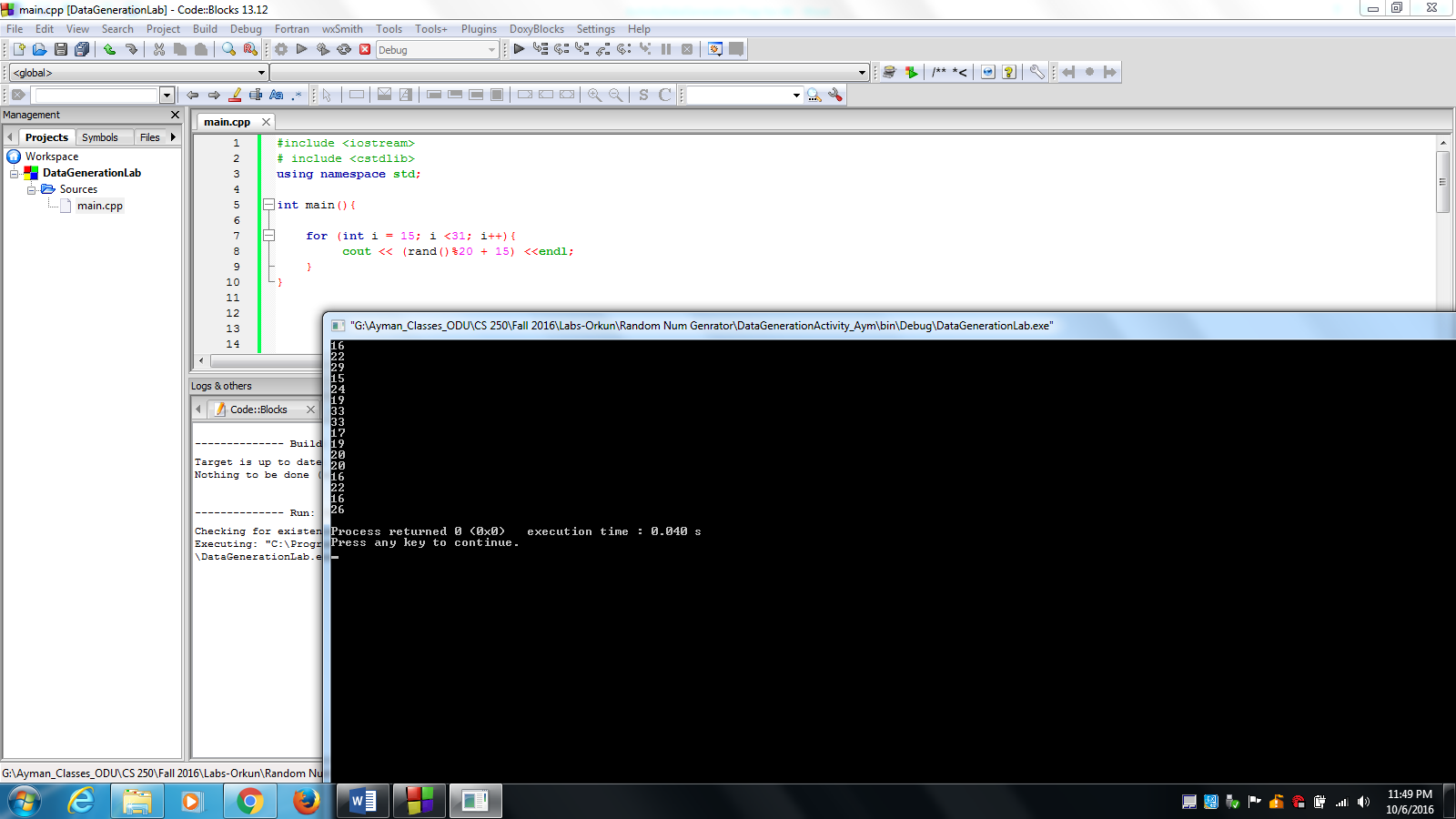
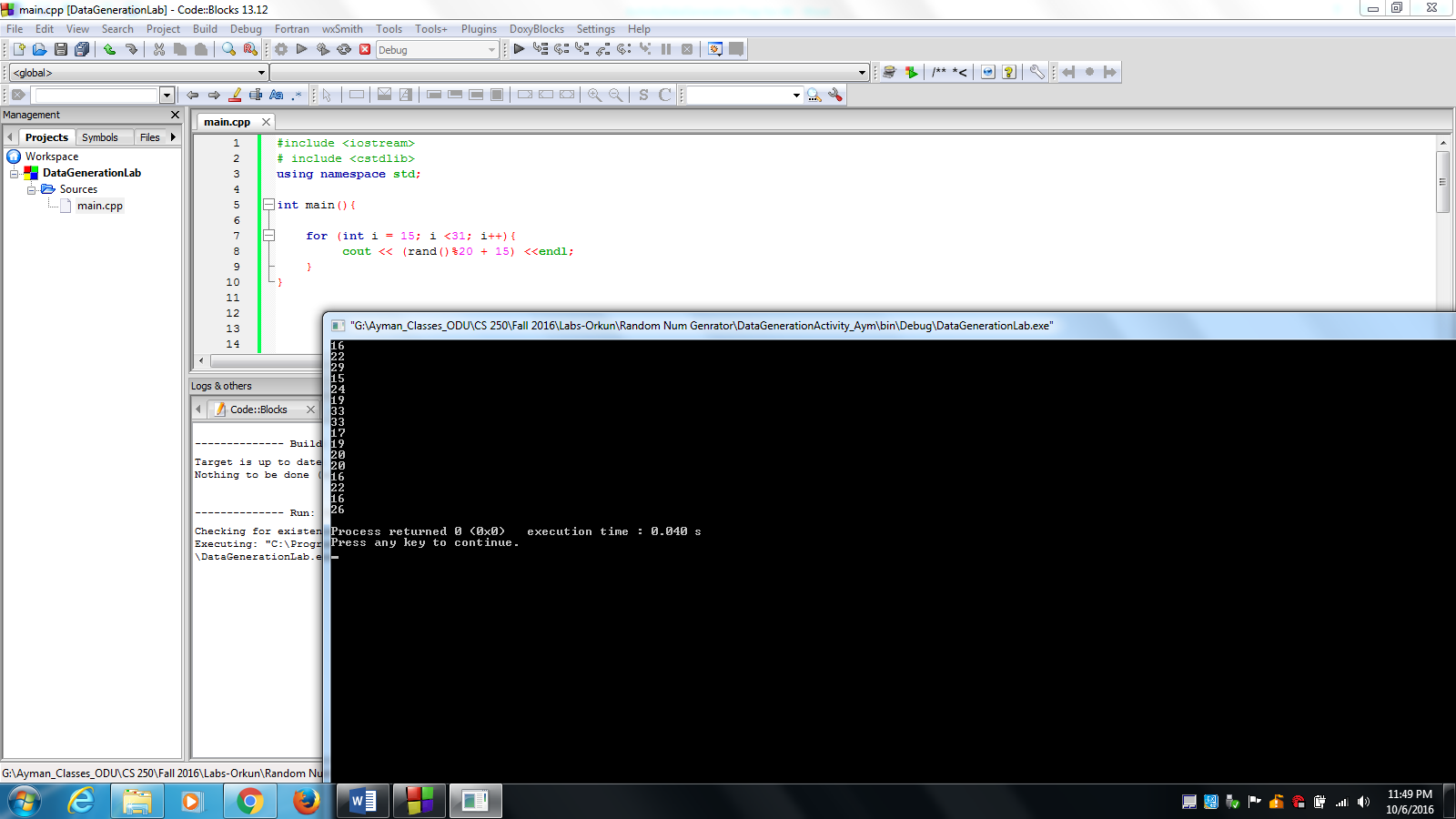
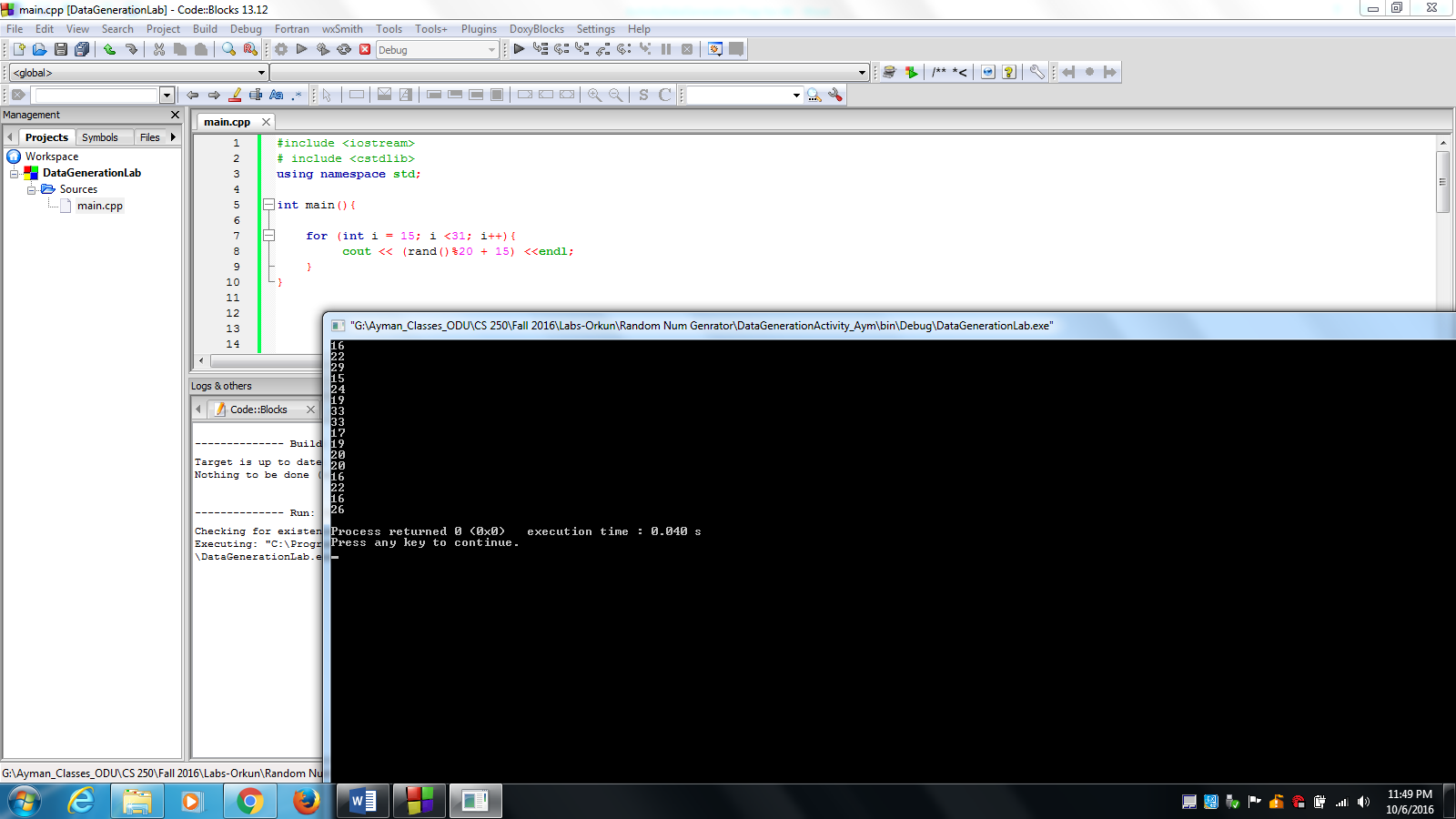
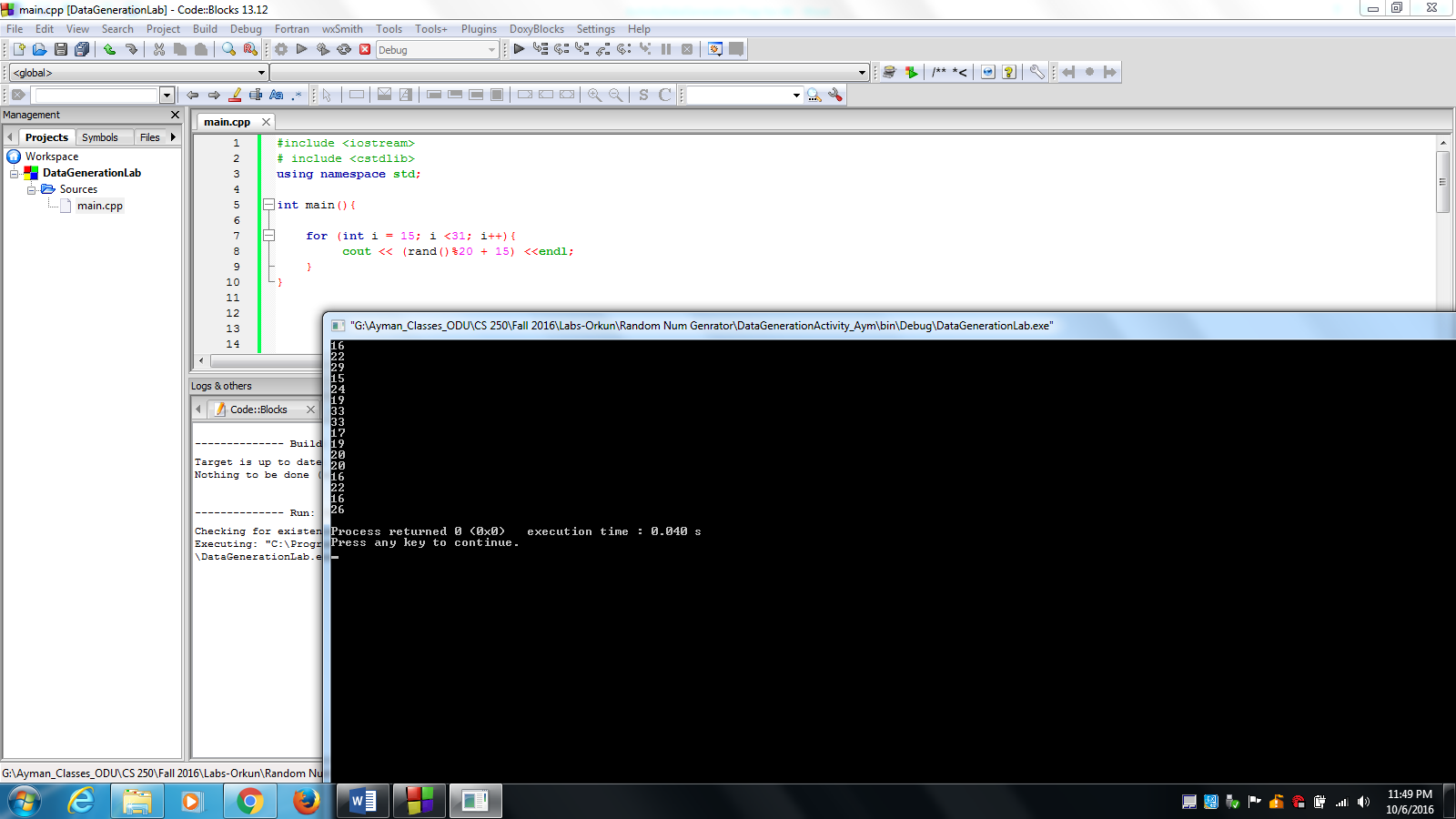


Run your executable file multiple times.

* **Q:** What numbers are displayed in each run?
* **Ans.:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Re-compile the program and run again.

* **Q:** Will the generated number be different each time? Why?
* **Ans.:** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

It seems that the exact set of numbers will be repeated each time, you run the pogrom. The reason for that is the random numbers created by computer programs are not a 100% random, however they are pseudo random, which means they are generated in a predictable fashion. To solve this problem we must seed the rand() function with the computer time – **how?**

* **Ans.:** using the srand() function from <cstdlib> library. And seed with the computer time by calling the function time(0) from the <ctime> library.

srand(time(0));

for (int i = 16; i <31; i++){

cout << (rand()%20 + 15) <<endl;

}

Recompile and run. You should get different random numbers in each run between 16 and 35.

