Random Numbers

1. Pseudo random number generator

The function `rand()` is used to generate pseudo-random integer numbers in the range 0 to `RAND_MAX` where `RAND_MAX` is just a constant defined in the system. Each time you call `rand()`, the function will return a (pseudo) random positive integer number.

A typical way to generate random numbers in a determined range using `rand()` is to use the modulo (%) operation. Recall that the modulo operation returns the remainder of a division of two integer numbers. When you do a division, the remainder will always be a number between 0 and the denominator minus 1. For example, doing a division with 4 gives a remainder between 0 and 3 as shown below:

\[
\begin{align*}
0 \mod 4 &= 0, \\
1 \mod 4 &= 1, \\
2 \mod 4 &= 2, \\
3 \mod 4 &= 3, \\
4 \mod 4 &= 0, \\
5 \mod 4 &= 1, \\
6 \mod 4 &= 2, \\
&\text{etc.}
\end{align*}
\]

So if you want to simulate the roll of a die, you will want to generate random numbers between 1 and 6. If you do a modulo 6 with any integer number, you will get a number between 0 and 5. You then need to add a 1 to move the range to between 1 and 6. The resulting expression is:

\[
\text{rand()} \mod 6 + 1
\]

Earlier, I said that `rand()` returns a pseudo random number. It is not a true random number because `rand()` uses an equation to calculate a sequence of numbers one after the other. The next number in the sequence is calculated based on the current number. Thus, if you know the equation and the current number, then you can always calculate what the next number is going to be. This also implies that `rand()` needs an initial number to start the sequence.

The function `srand(seed)` is used to initialize the random number generator, where the seed is an integer number. If you use the same seed each time, you will get the same sequence of random numbers with the `rand()` function. That is why it is called a “pseudo” random number generator because you will get the same sequence of numbers. However, it is random in the sense that if you do statistical analysis to all of the numbers generated, the numbers will have an equal probability of appearing. If you give it a different seed, you will get a different sequence of numbers. Usually, we use the current system time as the seed because each time you run the program the time will always be different.
int main (int argc, const char * argv[]) {

    int i;

    // initialize the random number generator
    // using the current time as the seed
    srand((int) time(NULL));

    cout << "Here are 20 random numbers... " << endl;

    for(i=1; i<=20; i++){
        // rand() generates a random number
        cout << rand() << endl;
    }

    cout << "Here are 20 random numbers between 1 and 6... " << endl;

    for(i=1; i<=20; i++){
        // generates a random number between 1 and 6
        cout << rand() % 6 + 1 << endl;
    }

    return 0;
}

Exercises

1. Type in the above program. What is the difference between the first set of 20 numbers and the second set of 20 numbers? Do the numbers look random?

2. Replace the srand() statement in the program above with just

   srand(3);

   and run the program several times. What do you notice about the numbers?

3. Change the seed to another number (such as 8) and see what happens.

4. Write a program to simulate the roll of a die 14 times.

5. Write a program to generate 10 random numbers in the range 1 to 13.

6. Write a program to generate 12 random numbers in the range 4 to 9.

7. Write a program to simulate the roll of two dice 9 times.

8. Write a program to simulate the flip of a coin 10 times.