

CST8110: Introduction to Computing: Generating Random Numbers

Overview

To make game play interesting, you must introduce some element of randomness. To do that, you'll need to use the `rand()` function. Here's some sample code to get you started.

Details: The C++ Code

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

#include <stdlib.h> // for rand() and srand()
#include <time.h> // for time()
#include <conio.h> // for _getch()

void main()
{
    // Seed the random number generator with constantly changing value
    srand(unsigned(time(NULL)));
    cout << "Loop 1\n";
    cout << "Each keystroke creates a new random number between 0 and 32767.\n";
    cout << "Use 'q' to quit the loop.\n";
    int nNumber;
    do {
        nNumber = rand(); // rand() returns a number between 0 and 32767.
        cout << nNumber << " ";
    } while (_getch() != 'q');

    cout << "\n\nLoop 2\n";
    cout << "Each keystroke creates a new random number between 0 and 99.\n";
    cout << "Use 'q' to quit the loop.\n";
    do {
        nNumber = rand(); // rand() returns a number between 0 and 32767.
        cout << nNumber%100 << " "; // % operator shrinks the range to 0-99 inclusive.
    } while (_getch() != 'q');

    cout << "\n\nLoop 3\n";
    cout << "Each keystroke will result in a new random number between -10 and 10.\n";
    cout << "Use 'q' to quit the loop.\n";
    do {
        nNumber = rand(); // rand() returns a number between 0 and 32767.
        cout << nNumber%21 - 10 << " "; // % operator shrinks the range to 0-20 inclusive.
        // Subtract 10. Range now -10 to +10 inclusive.
    } while (_getch() != 'q');
}
```

The preceding C++ code is available on my website. The key function is `rand()`.

Every time this function is called, it returns a different number that will range between 0 and 32767 (inclusive).¹ The three different loops show methods of adjusting the range generated by `rand()`.

Details: What is Really Random?

The `rand()` function generates pseudo-random numbers. True random numbers have to be generated using different techniques.

The sequence of random numbers generated by this pseudo-random number generator is repeatable. That is, `rand()` starts with an initial *seed value*; it then spawns and returns a newly calculated pseudo-random number on each function call. But everytime you run the program, the precise sequence is determined by the *seed value*. You can change the initial *seed value* by making another function call in advance of calling

`rand()`. The function is called `srand()`; of course, the *s* stands for *seed*. But you have to be careful how you use it. For example, if you add the code:

```
srand(999);
```

It will seed the random number generator with *999*, and you'll end up with a sequence based on that number. But every time you run the program, you'll always start with exactly the same seed, *999*. So you really need to seed the random number generator with a value that is constantly changing.

So . . . what's constantly changing on the computer?

```
srand(time(NULL));
```

Here, I have not sent a specific number to `srand()`. I have asked the computer's clock the current time and sent this to seed the random number generator. The `time()` function returns a long integer that increments once per second².

¹ This maps to the binary numbers: 0000000000000000 to 1111111111111111 (in each case, 15 binary digits). The function `rand()` is returning a 16-bit integer, but it only uses 15 of those 16 bits. That way the 16th bit is guaranteed to be 0, thus the number is always positive.

² And the beginning of **time** is January 1, 1970.