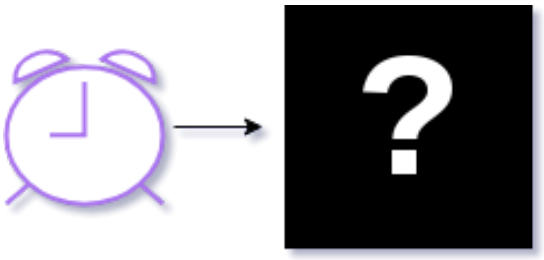
Pseudo-random Numbers



In order to create "random" numbers, you are going to use two built-in functions **srand()** and **rand()**. These numbers are known as pseudo-random, because a mathematical algorithm is in place to calculate the final result. Therefore, it is not truly random. The **srand()** function is to obtain a starting value (seed) for the pseudo-random number processes. Inserting a number into this function will produce the same set of numbers each and every time. For example:

```
srand(10); //If you do not call srand(), the default value is srand(1)
for(int i=0; i<10; i++) {
    cout << rand() << endl; }
```

As you can see even though we seed through **srand()**, the result comes from the **rand()** function. This will produce the same ten numbers onto your screen every single time you run this program. However, if we seed using the current system time, this will constantly change the starting seed and produce different results every time the program is executed (as long as at least one second passes between executions). The preprocessor directive for time is **#include <ctime>**. Now you can invoke the **srand()** function with the argument **time(NULL)** or **time(nullptr)** to look like **srand(time(nullptr));** Now your seed is ever-changing and **rand()** will output something relatively unique.

You will want to generate random numbers between a particular range. Usually it is between 0 and some number. This is where the % (modulo) operator comes into play. If you do **cout << rand() % 101;** you will display a number between 0 - 100 inclusive as the modulo operator is the remainder result by using elementary school long division. If you need your number to be in a range from 1 - some number, just add 1 after the modulo operation, such as **cout << (rand() % 100) + 1;** If you would like a range between two numbers, such as a random number between 50 and 100 inclusive, the general formula is:

```
(rand() % (max - min + 1)) + min
```

Take the difference between your maximum number and your minimum number. Add 1 to that result and you now have the number of possible outcomes. For this example 100-50 = 50 + 1 = 51 possible outcomes. Now, **rand() % 51** will result in a number between 0 - 50 inclusive. Add the minimum allowed number, which is 50 in our example, to the previous result to obtain a number between 50 and 100 inclusive.

```
srand(time(nullptr));
cout << (rand() % 51) + 50; //Can be stored in an integer
```