The ability to generate random numbers is extremely useful. It has applications in many domains: computer simulations, games, statistical sampling, etc. Monte Carlo methods are a class of algorithms that rely upon random numbers to obtain numerical results. Monte Carlo methods are extensively deployed to solve a host of very complex physical problems that are otherwise infeasible to solve.

**Generating pseudo random numbers between 0 and RAND_MAX**

C++ supports random number generation. The following program uses `rand()` method to generate uniformly sampled random integers between 0 to `RAND_MAX`. `RAND_MAX` is a constant whose value may vary between different machines.

**By uniform sampling we mean to say that each integer between 0 and RAND_MAX is equally likely.**

```cpp
#include <iostream>
#include <time.h>
#include <cstdlib>
// header needed for rand()

using namespace std;

int main()
{
    srand(time(NULL)); // setting the seed of
    // the random number to current
    // time, so different at each
    // run

    for (int i=0; i<10; ++i) {
        cout << "Random number " << i << " = " << rand() << endl;
    }

    return 0;
}
```

If you compile and run the above program, it will print a different sequence of 10 random numbers each time.

**Output**

```
$ g++ rand-1.cpp
$ ./a.out
Random number 0 = 1320409651
Random number 1 = 28996259
Random number 2 = 2008820791
```
Random number 3 = 1660619850
Random number 4 = 1340342538
Random number 5 = 33579136
Random number 6 = 1723823238
Random number 7 = 595279389
Random number 8 = 1881863197
Random number 9 = 335598963
$ ./a.out
Random number 0 = 1320426458
Random number 1 = 311471508
Random number 2 = 1483987217
Random number 3 = 498079861
Random number 4 = 336967821
Random number 5 = 503790408
Random number 6 = 1824850782
Random number 7 = 2053130267
Random number 8 = 1193157473
Random number 9 = 195353025

Generating the same sequence of random numbers at each sequence

Sometimes it is desireable to generate the same sequence of random numbers every time the program runs. This can be achieved by specifying a seed for the random number generator as shown below.

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

int main()
{
    srand(0); // setting the seed of
                // random number generator
                // to a specific value

    for (int i=0; i<n; ++i) {
        cout << "Random number " << i << " = " << rand() << endl;
    }

    return 0;
}
```

This program generates the same sequence of random numbers at each run.

Generating a psuedo random number between 0.0 and 1.0

The following program generates 10 random numbers drawn from a uniform distribution between 0 and 1.

```
#include <iostream>
#include <time.h>
#include <cstdlib>
using namespace std;

int main()
{
```
srand(time(NULL));

for (int i=0; i<10; ++i) {
    cout << "Random number " << i << " = " << ((double) rand())/RAND_MAX << endl;
}

return 0;
}

Generating a pseudo random number between 0 and some number \( N \) less than \( \text{RAND\_MAX} \)

The following program generates random numbers between 0 and \( N \).

```cpp
#include <iostream>
#include <time.h>
#include <cstdlib>    // header needed for rand()

using namespace std;

int main()
{
    const int N=3;

    srand(time(NULL));

    for (int i=0; i<10; ++i) {
        cout << "Random number " << i << " = " << rand()%(N+1) << endl;
    }

    return 0;
}
```

References

- cstdlib
- ctime