C++ Random

```
`#include<iostream>`
`#include<cstdlib>`
`#include<ctime>`
```

The `rand()` and `srand()` functions in C++ are used for generating random numbers. Here's how they work:

1. `rand()`

- `rand()` generates a **pseudo-random** integer between `0` and `RAND_MAX` (which is at least 32767).
- Since it generates the same sequence every time the program runs, we usually use `srand()` to introduce randomness.

2. `srand(seed)`

- `srand(seed)` is used to **seed** the random number generator.
- If the seed is constant, the sequence of random numbers remains the same.
- If we use `srand(time(0))`, the seed is based on the current time, ensuring different results in each run.

Example 1: Basic `rand()` Usage

```
#include <iostream>
#include <cstdlib>
```

```
// rand() using namespace std;
int main() {
cout << "Random numbers without seeding:" << endl;
for (int i = 0; i < 5; i++) {
    cout << rand() << endl;
}
return 0;
}
```

Output (Same Every Time)

`Random numbers without seeding: 1804289383 846930886 1681692777 1714636915 1957747793`

- Since `srand()` is not used, the same sequence is generated in every execution.

Example 2: Using `srand()` to Seed Random Numbers

```
#include <iostream>
#include <cstdlib> // rand(), srand()
#include <ctime> // time()
using namespace std;
int main() {
srand(time(0));  // Seed the random number generator with current time
cout << "Random numbers after seeding:" << endl;
for (int i = 0; i < 5; i++) {
    cout << rand() << endl;
}
```

```
return 0;
}
```

Output (Different Every Time)

`Random numbers after seeding: 135792468 789123654 147258369 369258147 987654321`

- Since `srand(time(0))` changes every second, a new sequence is generated in each execution.

Example 3: Generating Random Numbers in a Specific Range

To generate a random number in the range [min, max], use:

random number=min+(rand()%(max−min+1))\text{random number} = \text{min} + (\text{rand}() \% (\text{max} - \text{min} + 1))random number=min+(rand()%(max−min+1))

Code

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main() {
srand(time(0));
int min = 10,
max = 20;
cout << "Random numbers between " << min << " and " << max << ":" << endl;
```

```
for (int i = 0; i < 5; i++) {
    cout << min + (rand() % (max - min + 1)) << endl;
}
return 0;
}
```

Possible Output

```
Random numbers between 10 and 20: 14 19 11 16 18
```

- The output values are **always between 10 and 20**.

Example 4: Generating Random Floating-Point Numbers

Since `rand()` only returns integers, we can generate floating-point numbers like this:

Code

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;
int main() {
srand(time(0));
double min = 1.5, max = 5.5;
cout << "Random floating-point numbers between " << min << " and " << max
<< ":" << endl;
for (int i = 0; i < 5; i++) {
```

```
    double randomFloat = min + static_cast<double>(rand()) / RAND_MAX *
(max - min);
    cout << randomFloat << endl;
}
return 0;
}
```

**

`Random floating-point numbers between 1.5 and 5.5: 3.7468 2.1123 5.2398 1.8954 4.6759`

- `static_cast<double>(rand()) / RAND_MAX` ensures the result is between `0.0` and `1.0`, which is then scaled to the desired range.

Conclusion

- `rand()` generates pseudo-random integers.
- `srand(seed)` sets the random seed to produce different results each run.
- `time(0)` ensures randomness across executions.
- To control ranges, use modulo (`%`) for integers and scaling for floating points.