Difference in Algorithm:

Pseudo

Pseudo-Random generates sequence of numbers through a formula. And it heavily relies on the seed to get the random output. The algorithm for getting the random output is very similar to the Linear Congruential Generator (LCG) algorithm.

Algorithm for the LCG:

let x[i] = seed let m = modulo or the range of numbers

let a = multiplier let c = increment

x[i]+1 = (a\* x[i] + c) % m

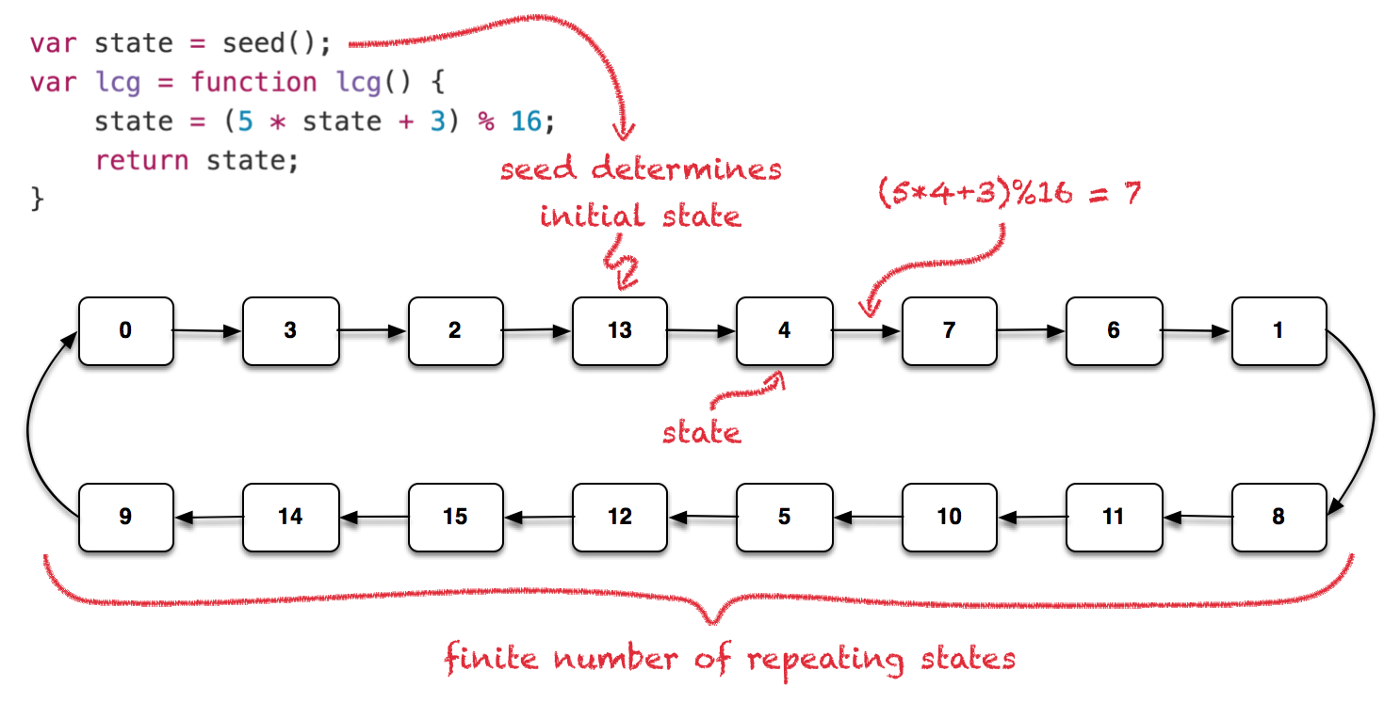
An example is shown below:

let x[i]=13 // initial seed

x[i]+1 = (5 \* x[i] + 3) % 16;

with x[i] is equal to 4 and so on…

Refer to the image below for the representation of LCG:



Though it follows a formula the outputs can be looped if the seed is predefined. We now have the idea on how LCG works, let us move on to Pseudo-Random.

We refer to the code Pseudo and get the method **rand().** This is the most known for using the fast and deterministic RNG.

Let us see the how the rand() works:

//Initialization

int seed=1; // this is the variable for the seed that goes through inside of srand() || we define the seed as 1 for now

int holder; // a variable holder for the algorithm which will be the next seed

//Method

holder = (214013 \* seed + 2531011); // holder then stores the value for the NEXT seed when looped.



Every compiler has their own multiplier and increment. We are using the Microsoft Visual C++ source. Refer to Figure? for the list of sources.

seed = ((holder) >> 16) & 0x7fff; // we take the holder value and shift it to right by 16 after that we calculate the bits and 0x7fff (32767 or RAND\_MAX) || Alternative without shifting is dividing the holder to 65535.

When we run it through a code, this is how it displays:

Holder: 2745024

Random Output: 41

When the seed is not changed, the holder will then be the new seed.

Holder: -937167229

Random Output: 18467

|  |
| --- |
| Seed: 1 |
| 41 |
| 18467 |
| 6334 |
| … |

It is the same as when call the function rand() and looped it.

Uniform

Uniform does not follow an algorithm unlike Pseudo-Random or LCG. But it gets its random outputs through probabilities. However, a seed is called to not produce the same output as before.

We refer to the code Uniform.

//Initialization

default\_random\_engine gen() // this is a engine to hold our seed in.

uniform\_int\_distribution<>distribution(1, 6) // this will initialize the range of numbers right now it is 1 to 6

//Method

distribution(gen); // we call distribution with the parameter of the gen() which then give the random output through 1 to 6

float pro = 1.0/(distribution.max() - distribution.min() +1); // this calculates the probabilities of the numbers that will be the output || assume that 1.0 is 100% and we divide it with the value of total possible numbers

When we run it through a code, this is how it is displays:

Output: 1

Total Possible Numbers: 6

Probabilities: 0.166667%

When default\_random\_engine does not change, the output will be the same.

|  |  |
| --- | --- |
| Possible Numbers: 6 | Probability of each number: 0.166667% |
| Possible Numbers: 17 | Probability of each number: 0.0588235% |
| Possible Numbers: 25 | Probability of each number: 0.04% |
| Possible Numbers: 54 | Probability of each number: 0.0185185% |
| … | … |

Hence the name Uniform Distribution, it distributes the chances in every number.