Nasumično?

Generatori slučajnih brojeva

- PRNGs (Pseudorandom number generators) –
  - Generišu “nasumičnu” sekvencu brojeva.
  - Obično je grade preko zadate formule od klice(seed).
  - Brzi su(složenost formule).
- URNGs (Uniform random number generators) –
  - Uniformno generišu brojeve na intervalu generisanja.
  - Svaki broj sa “istom” verovatnoćom.

Pseudoslučajni generatori brojeva

- Linearan kongruentan generator(LKG)-
  - $X_n = (aX_{n-1} + c)\ \%\ m$,
  - $m > 0$, *moduo*
  - $m > a \geq 0$, *množilac*
  - $m > c \geq 0$, *inkrement*
  - X_0 , *klica*
  - *(Donald Knuth) – Perioda ovog generator je m ako su:*
    - c *I* m *uzajamno prosti*
    - $a - 1$ *deljivo svim prostim činiocima od* m
    - $a - 1$ *je umnožak 4, ako je* m *umnožak 4.*

Pseudoslučajni generatori brojeva

- Mersenne twister generator –
  - Proizvodi brojeve u intervalu $[0, 2^w - 1]$
  - Standarna primena u MT19937 generatoru, gde je $w = 32 (\text{Velika perioda} = 2^{19937} - 1)$.
  - Više informacija na http://www.cplusplus.com/reference/random/mersenne_twister_engine/

$UIntType\ Mask = (1u << w) - 1, \quad UMask = (Mask << r)\ \&\ Mask, \quad LMask = (\sim UMask)\ \&\ Mask,$
$Y = (x[i]\ \&\ UMask)\ |\ (x[(i+1)\ \%\ n]\ \&\ LMask);$ // gornja bitovi x[i] I donji x[i+1]
$x[i] = x[(i+m)\ \%\ n]\ \hat{}\ (Y >> 1)\ \hat{}\ ((Y\&1) * a);$ // mersenne twister transformacija

- Postoji još mnogo vrsta pseudoslučajnih generatora, ova dva ćemo koristiti u primerima koji slede.

Funkcija $rand()$

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main() {
    srand(time(NULL));
    for (int i = 0; i < 16; ++i) {
        printf("%d ", rand() % 100);
    }
    printf("\n");
}
```

Proizvodimo random brojeve u intervalu [0,99]

Šta rade srand(time(NULL)) i rand()?

Šta je loše sa kodom?

Konverzija kalendarskog vremena u 32bit int

LKG u rand() – loš kvalitet

Nije uniformna raspodela!!

rand() range [0, 32767]

Zašto izbegavati $rand()$?

```
int src = rand(); // Pretpostavimo da je P(src = k) ista za sve k ϵ [0, 32767]
int dst = src % 100;
//        [0, 99] src ➔ [0, 99] dst
//     [100, 199] src ➔ [0, 99] dst
// ...
// [32700, 32767] src ➔ [0, 67] dst
```

Očigledno brojevi [0, 67] imaju veću verovatnoću da se pojave pa mod u ovom slučaju nije uniforman.

Da bi bio uniforman potrebno je da moduo deli RAND_MAX.

Zašto izbegavati $rand()$?

Drugi način na koji možemo da mapiramo uniformno generisane brojeve [0,32767] na [0,99] je:

```
int src = rand();
int dst = static_cast<int>((src * 1.0 / RAND_MAX) * 99);
```

Da li je ovo uniformno?

Primetimo da je $dst = 99$ akko $src = 32767$.

Ako su src i RAND_MAX tipa long long -> problem sa mantisom?

Popravka?

```
int src = rand();
int dst = static_cast<int>((src * 1.0 / (RAND_MAX + 1)) * 100)
```

Ispostavlja se da ni ovako ne dobijamo uniformnu raspodelu. Neke vrednosti se češće pojavljuju.

Uniformno generisanje slučajnih brojeva

```
#include <iostream>
#include <random>
int main() {
    std::mt19937 mt(1729);
    std::uniform_int_distribution<int> dist(0, 99);
    for (int i = 0; i < 16; ++i) {
        std::cout << dist(mt) << " ";
    }
    std::cout << std::endl;
}
```

C++11

mt – Mersenov twister generator pseudoslučajnih brojeva

1729 random klica

mt19937 - engine

dist – uniformna distribucija na intervalu [0, 99]

dist(mt) uniformno rasporedjuje

Uniformno generisanje slučajnih brojeva

Umesto klice 1729 na prethodnom slajdu može se koristiti random_device iz <random> biblioteke.

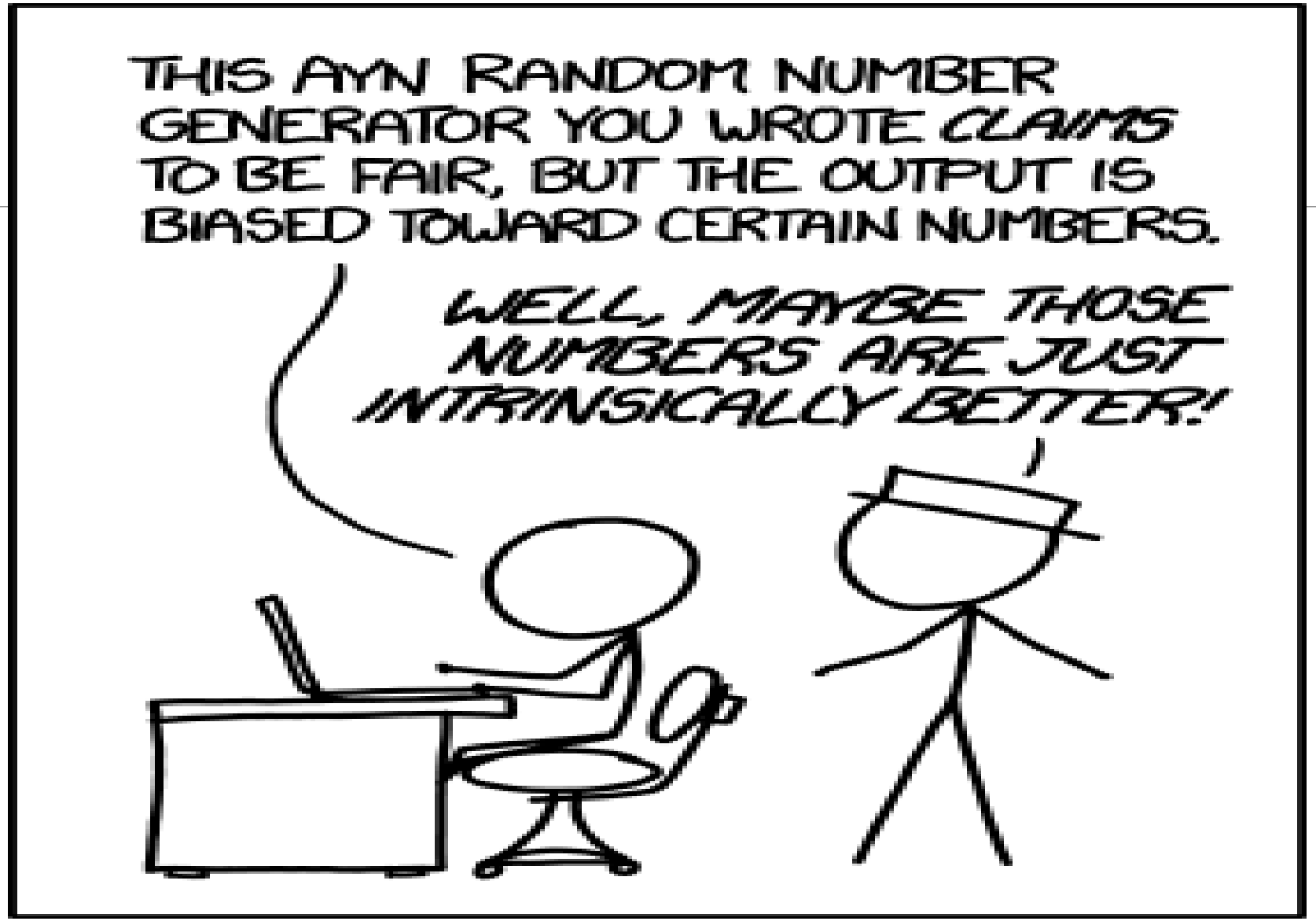
```
#include <iostream>
#include <random>
int main() {
    std::random_device rd;
    std::mt19937 mt(rd());
    std::uniform_int_distribution<int> dist(0, 99);
    for (int i = 0; i < 16; ++i) {
        std::cout << dist(mt) << " ";
    }
    std::cout << std::endl;
}
```

Sada je klica ne-deterministička(stohastička)

Random device je baziran na stohastičkim procesima

Više mogućih prelaza iz jednog stanja u naredno

http://en.cppreference.com/w/cpp/numeric/random/random_device


THIS AYN RANDOM NUMBER GENERATOR YOU WROTE CLAIMS TO BE FAIR, BUT THE OUTPUT IS BIASED TOWARD CERTAIN NUMBERS.
WELL, MAYBE THOSE NUMBERS ARE JUST INTRINSICALLY BETTER!