Generacja liczb pseudolosowych

Sprawozdanie z laboratorium 12 – Piotr Sarna LK1

Cel ćwiczenia

Podczas zajęć zapoznaliśmy się z metodą generacji liczb pseudolosowych za pomocą Liniowego Generatora Kongruentnego (LCG) oraz go zaimplementowaliśmy.

Wstęp teoretyczny

Liczby losowe są szeroko wykorzystywane w symulacjach, grach wideo czy kryptografii.

Z założenia, liczba losowa powinna być otrzymywana bez względu na poprzednie liczby w generowanej sekwencji oraz czynniki zewnętrzne, w praktyce jest to jednak niemożliwe do wykonania przez komputery. Dlatego mówiąc o liczbach losowych, generowanych komputerowo mamy na myśli liczby pseudolosowe.

Podczas zajęć zapoznaliśmy się z „liniowym generatorem kongruentnym”, służącym do generacji sekwencji liczb pseudolosowych.

Jest on dany wzorem:

$$x_i = (a * x_{i-1} + c) \, mod \, m$$

Gdzie ‘a’, ‘c’ oraz ‘m’ to dowolne współczynniki spełniające podane założenia:

- ‘m’ musi być większe niż generowany zakres
- ‘c’ i ‘m’ są względnie pierwsze, czyli GCD(c, m) == 1. Iloczyn unikalnych czynników pierwszych ‘m’ jest zawsze podzielny przez każdy czynnik ‘m’. Dodając 1 można zagwarantować, że ‘c’ nie jest podzielne przez żaden z nich
- ‘a - 1’ ma być podzielne przez wszystkie czynniki pierwsze ‘m’

Spełniając te założenia, przyjmujemy najlepszy możliwy scenariusz dla LCG, który dzięki temu generuje wszystkie liczby w zadanym zakresie.

Wartość startowa algorytmu, czyli jego ziarno to x_o.

Opis algorytmu

Algorytm przed rozpoczęciem generowania pseudolosowej sekwencji ustala odpowiednie wartości dla ‘m’, ‘c’ oraz ‘a’ w następujący sposób:

‘m’ to maksymalna liczba z zakresu + 1, ziarno x_o ustawiam na 0.

```
int m = maxNumber + 1;
int x0 = 0;
```

W celu ustawienia wartości ‘c’ wykorzystuję funkcję zwracającą niepowtarzające się czynniki pierwsze liczby ‘m’.

```
std::vector<int> uniqueFactors = getUniquePrimeFactors(m);
int c = 1;
for (int factor : uniqueFactors) {
    c *= factor;
}
c += 1;
```

W celu ustawienia wartości ‘a’ wykorzystuję funkcję zwracającą wszystkie czynniki pierwsze liczby ‘m’.

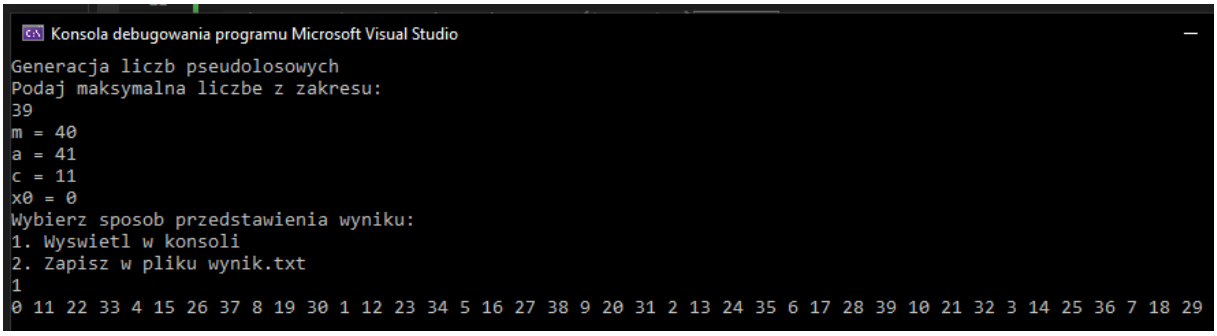
```
std::vector<int> allFactors = getPrimeFactors(m);
int a = 1;
for (int factor : allFactors) {
    a *= factor;
}
a += 1;
```

Następnie obliczam wszystkie wartości pseudolosowe za pomocą podanego wcześniej wzoru.

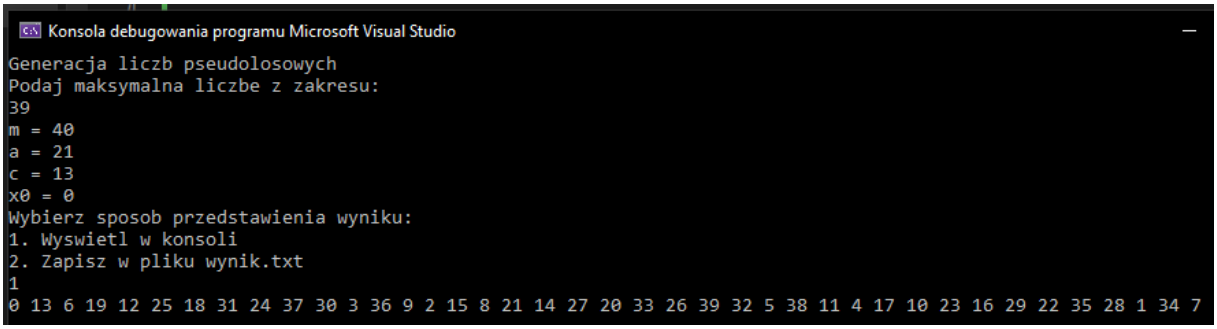
```
std::vector<int> pseudoRandomNumbers;
pseudoRandomNumbers.push_back(x0);
for (int i = 1; i < m; i++) {
    pseudoRandomNumbers.push_back((a * pseudoRandomNumbers[i - 1] + c) % m);
}
```

Prezentacja działania mojej implementacji w C++

Dla maksymalnej liczby z zakresu 39, automatycznie obliczając wartości ‘m’, ‘a’ oraz ‘c’ według uprzednich kryteriów otrzymujemy następujące wyniki:



Wykorzystując przykładowe wartości podane na „Delcie”, otrzymujemy następujące wyniki:



Wnioski

Zarówno funkcje getPrimeFactors jak i getUniquePrimeFactors cechuje złożoność O(√m), gdzie m = maxNumber + 1. Następnie wyznaczanie sekwencji cechuje złożoność O(m). W sumie końcowa złożoność funkcji generateLCG wynosi

O(2√m + m).

Bibliografia

http://www.algorytm.org/liczby-pseudolosowe/generator-lcg-liniowy-generator-kongruentny.html

https://eduinf.waw.pl/inf/utils/010_2010/0213.php