Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

# ОТЧЕТ
по лабораторной работе № 1

“Методы генерации псевдослучайных чисел”

Выполнил: И.А. Григорик

Проверил: И.Г. Алексеев

МИНСК 2023

## 1 ЦЕЛЬ

Изучить основные способы создания последовательностей случайных чисел с заданными законами распределения вероятности.

## 2 ЗАДАНИЕ

Написать приложение для моделирования генераторов псевдослучайных чисел. Использовать 64-битную арифметику целых чисел без знака.

1. Генератор на основе алгоритма Лемера
2. Генератор на основе метода серединных произведений
3. Генератор на основе регистра сдвига с обратными связями

Подобрать параметры для генераторов с целью получения наилучших результатов для каждого генератора (для выборок N = {10000, 500000, 100000000}).

Вывести гистограмму, матожидание, корреляцию и результаты 1-8 тестов NIST для лучшего случая.

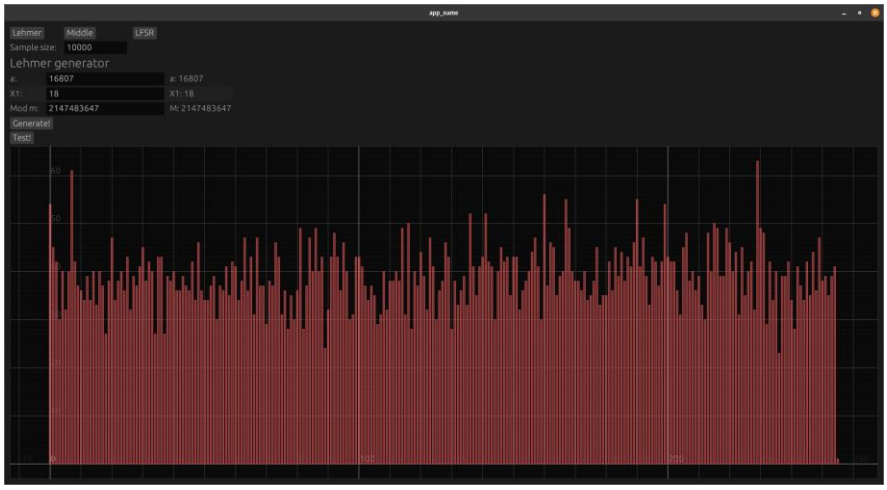
Сравнить полученные результаты с встроенным генератором.

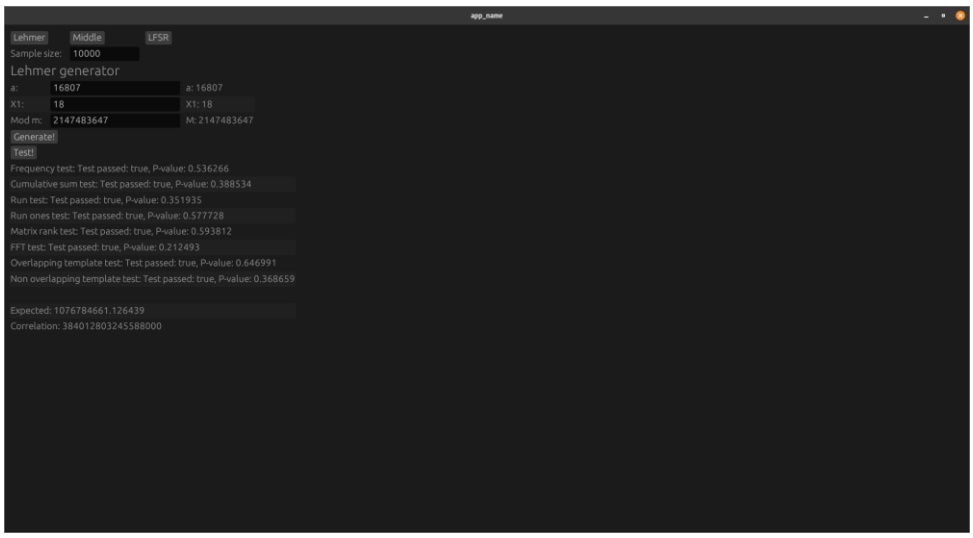
## 3 ХОД РАБОТЫ

### 3.1 Генератор Лемера

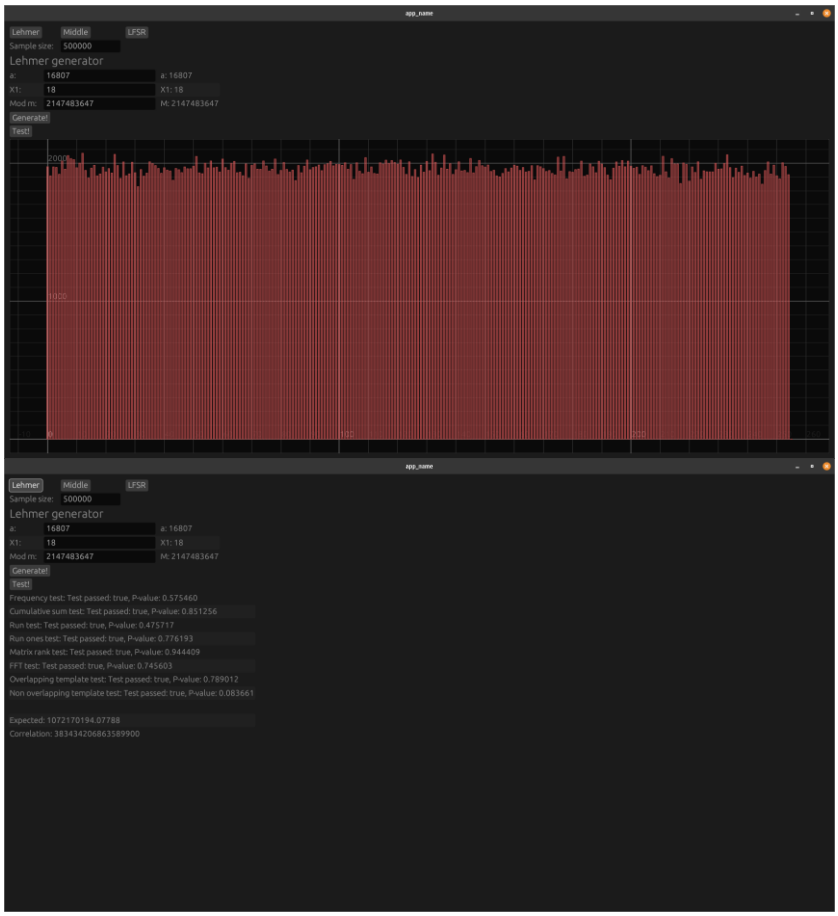
Генератор на основе алгоритма Лемера.

Генератор с выборкой равной n = 10000

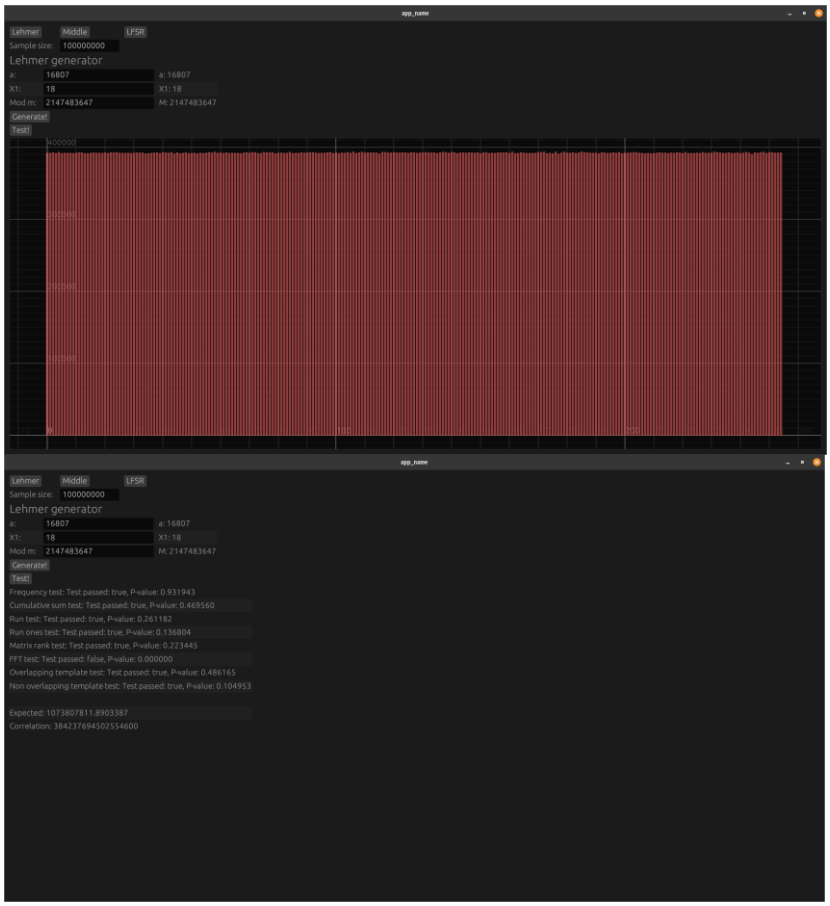





Генератор с выборкой равной n = 500000



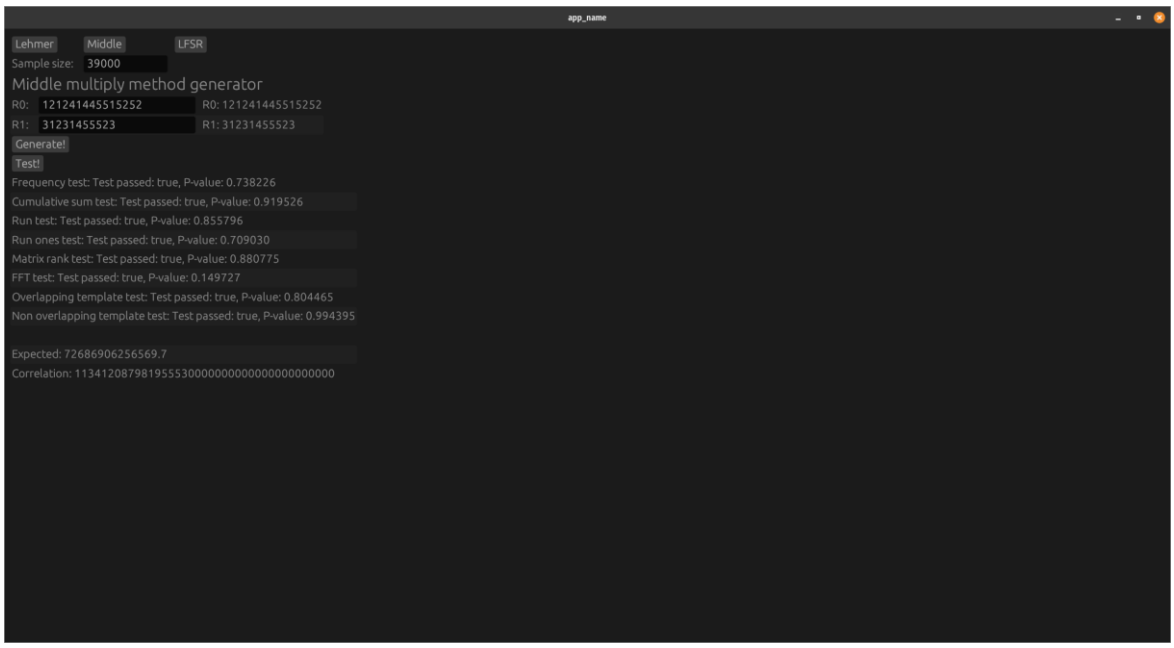
Генератор с выборкой равной n = 100000000



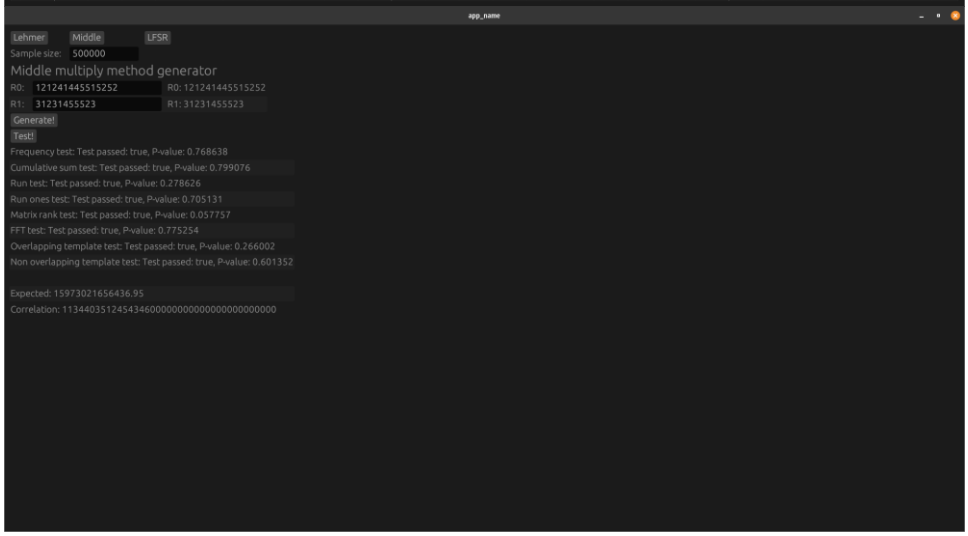

### 3.2 Генератор серединных произведений

Генератор на основе алгоритма серединных произведений.

Генератор с выборкой равной n = 10000




Генератор с выборкой равной n = 500000

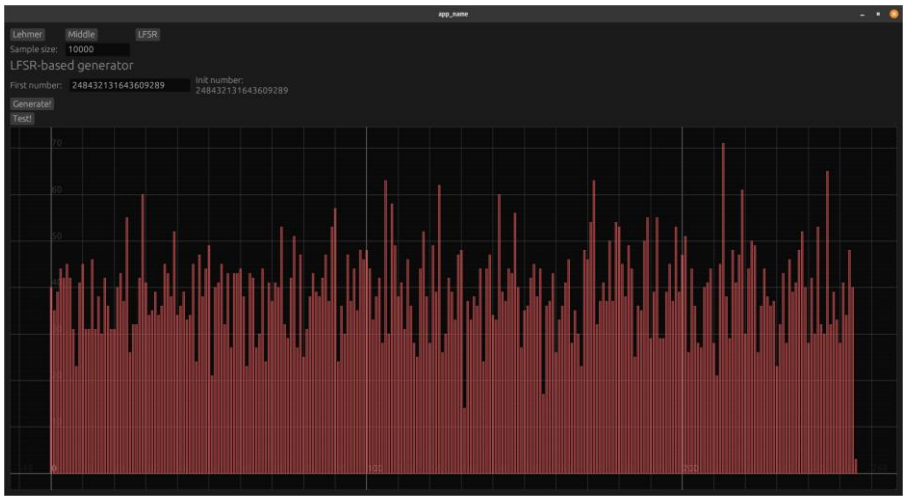



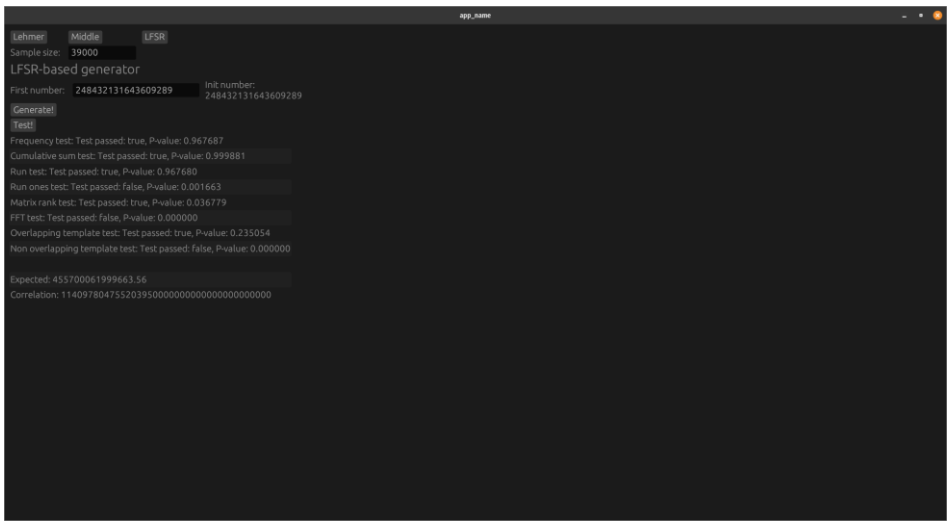
Генератор с выборкой равной n = 100000000


### 3.3 Генератор, основанный на методе регистров обратной связи

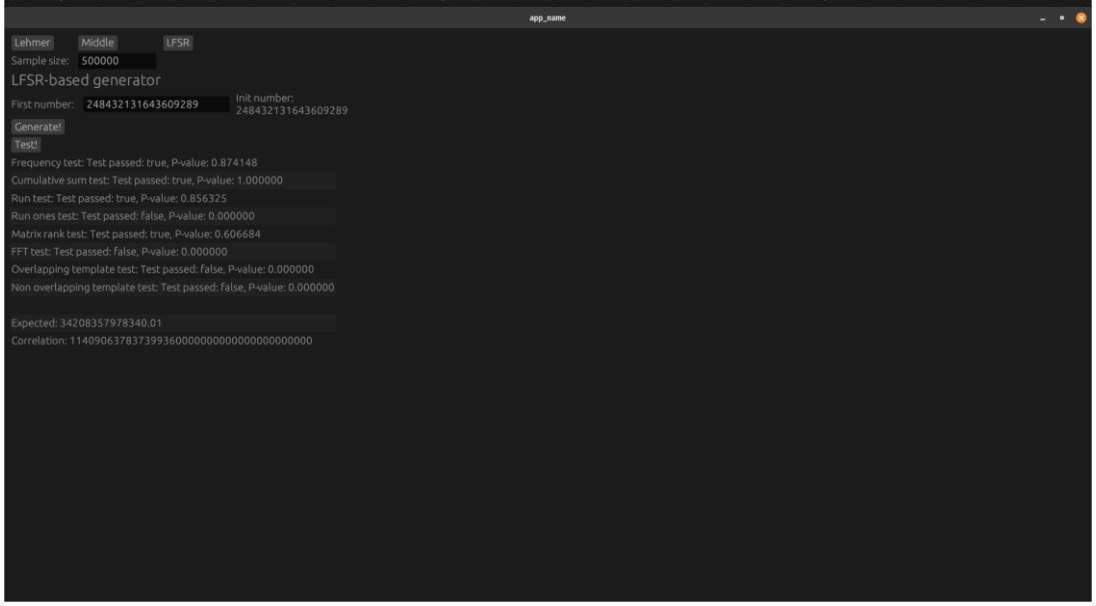
Генератор на основе алгоритма серединных произведений.

Генератор с выборкой равной n = 10000






Генератор с выборкой равной n = 500000

Генератор с выборкой равной n = 100000000

## 4 Листинг кода

Данная лабораторная работа была выполнена на языке Rust

Генератор Лемера

```
fn generate_lehmer(&mut self) {
    let a = self.a_buffer as u128;
    let x1 = self.x1_buffer as u128;
    let m = self.m_buffer as u128;
    self.random_numbers.push(x1 as u64);

    for i in 1..self.count {
        let t = a * self.random_numbers[i - 1] as u128;
        let _tmp = t % m;
        self.random_numbers.push(_tmp as u64);
    }
}
```

Генератор серединных произведений

```
fn generate_middle(&mut self) {
    let mut r0 = self.r0_buffer as u128;
    let mut r1 = self.r1_buffer as u128;

    let mask: u128 = 0xFFFFFFFFFFFFFFFF00000000; // mask to identify
middle part of the number
    for _ in 0..self.count {
        let mut new_r = r0 * r1;

        new_r &= mask;
        new_r >>= 32;

        self.random_numbers.push(new_r as u64);

        r0 = r1;
        r1 = new_r;
    }
}
```

Генератор, основанный на методе регистров обратной связи

```
fn generate_shift(&mut self) {
    let mut num = self.shift_buffer;
    for _ in 0..self.count {
        // Identify two last bits bit
        let b_0 = num & 1;
```

```
        let b_1 = (num & 2) >> 1;
        // Generate new last bit
        let n_bit = (b_0 ^ b_1) << 63;

        // Shift our number and put the last bit
        num >>= 1;
        num |= n_bit;
        self.random_numbers.push(num);
    }
}
```