Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

По дисциплине «Криптографические методы защиты информации»

Тема: «Генерирование равномерно распределенных псевдослучайных последовательностей»

Выполнил:

Студент 2 курса

Группы ИИ-23

Макаревич Н.Р.

Проверил:

Хацкевич А. С.

Брест 2024

Задание:

1. Согласно варианту реализовать приложение, генерирующие псевдослучайную равномерно распределенную последовательность произвольной длины из заданного алфавита.
2. Подобрать параметры данного генератора таким образом, чтобы период последовательности имел максимальное значение.
3. Используя статистическое тестирование, проверить гипотезу о том, что сгенерированная последовательность действительно имеет равномерное распределение (см. приложение 2).

3	Генератор Эйхенауэра – Лена с обращением	N=20

Ход работы:

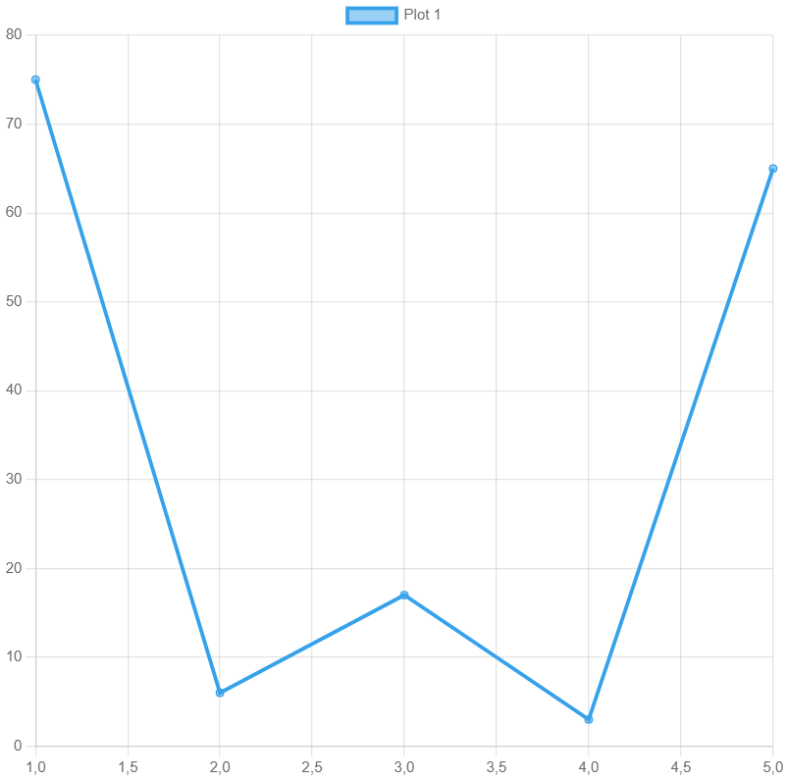
CBCTable.h:

```
#pragma once
#include <vector>
#include <iostream>
class RandomGenerator
{
private:
	int a,c,N;
	int extendedEuclidean(int a, int b, int& x, int& y);
	int findMultiplicativeInverse(int x);
	int calculateNewX(int x0);
public:
	RandomGenerator(int a, int c, int N);
	std::vector<int> generate(int x0, int size);
};
```

CBCTable.cpp:

```
#include "RandomGenerator.h"

int RandomGenerator::extendedEuclidean(int a, int b, int& x, int& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }

    int x1, y1;
    int gcd = extendedEuclidean(b, a % b, x1, y1);

    x = y1;
    y = x1 - (a / b) * y1;

    return gcd;
}

int RandomGenerator::findMultiplicativeInverse(int x) {
    int inv, y;
    int gcd = extendedEuclidean(x, N, inv, y);

    if (gcd != 1)
        return -1;

    inv = (inv % N + N) % N;
```

```
    return inv;
}

RandomGenerator::RandomGenerator(int a, int c, int N){
    this->a = a;
    this->c = c;
    this->N = N;
}

int RandomGenerator::calculateNewX(int x0) {
      if (x0 == 0)
            return c;
    else {
        int inverse = findMultiplicativeInverse(x0);
        std::cout << x0 << " inverted: " << inverse << "\n";
        if (inverse != -1)
            return (a * inverse + c) % N;
        else
            return (a * x0 + c) % N;
    }
}

std::vector<int> RandomGenerator::generate(int x0, int size) {
      int x = x0;
      std::vector<int> result;
      for (int i = 0; i < size; i++) {
            result.push_back(x);
        x = calculateNewX(x);
      }
      return result;
}
```



Вывод: в ходе лабораторной работы я освоил алгоритмы генерирования равномерного распределения псевдослучайных последовательностей.