Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное   
образовательное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

Лабораторная Работа №2.   
Генераторы псевдослучайных величи

Отчет по лабораторной работе №2 по учебной дисциплине   
«Методы защиты информации»

по специальности 09.03.01 Информатика и вычислительная техника

|  |  |
| --- | --- |
|  | Руководитель  / Т. В. Жгун  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |
|  | Студент группы 3091  / Р. А. Михайлов  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |

Оглавление

[Линейный конгруэнтный генератор (*LCG*) 3](#_Toc182509155)

[Генератор псевдослучайных чисел (*BBS)* 6](#_Toc182509156)

[Линейный рекуррентный генератор (*LFSR*) 9](#_Toc182509157)

[Приложение А 12](#_Toc182509158)

[Приложение Б 18](#_Toc182509159)

# Линейный конгруэнтный генератор (*LCG*)

Генератор определяется рекуррентным соотношением:

*X n + 1 = (a X n + c) mod m*

У последовательности, созданной с помощью *LCG* и определённой целыми параметрами *m, a, c* и *X0,* период максимален и равен числу *m,* когда выполняются следующие условия:

* *c* и *m* – взаимно просты:    
  (наибольший общий делитель *c* и *m* равен 1).
* *b = а - 1* – кратно любому простому числу, являющемуся делителем *m*, значением
* если *m* кратно 4, тогда *b* также кратно 4.

Популярным выбором *m* является степень двойки *2n.* Очень часто в качестве *m* выбирают одно из простых чисел Мерсенна .

В моей программе я установил следующие значения:

1. с = 1013904223
2. m = 2147483647 ()
3. a = 1664525
4. seed = 0

При помощи этого генератора я сгенерировал три битовых строки, из 50, 100 и 1000 битов. Битовые строки разделил на восьмибитные отрезки. Полученные восьмибитные значения перевел в десятичные для удобства отображения на гистограмме. Полученные гистограммы представлены ниже, полученные битовые строки, сгенерированные LCG генератором в Приложении Б.1.

Изображение выглядит как линия, диаграмма, График, Параллельный

Автоматически созданное описание

Рисунок 1 – Гистограмма 8-битных отрезков из 50-битной строки LCG

Изображение выглядит как линия, диаграмма, График, Параллельный

Автоматически созданное описание

Рисунок 2 – Гистограмма 8-битных отрезков из 100-битной строки LCG

Изображение выглядит как линия, снимок экрана, диаграмма, График

Автоматически созданное описание

Рисунок 3 – Гистограмма 8-битных отрезков из 1000-битной строки LCG

Генератор псевдослучайных чисел   
*BBS* (L. Blum, M. Blum, M. Shub) (генератор с квадратичным остатком)

1. Вначале выбираются два больших простых числа *p* и *q:*

p ≡ 3 mod 4, q ≡ 3 mod 4

1. Вычисляем число M = p q, называемое целым числом Блюма.
2. Выбирается случайное целое число *х*, взаимно простое с М.

(x, M) = 1

1. Вычисляем стартовое число генератора *х*0:

Псевдослучайная последовательность вырабатывается по правилу:

Результатом *n* - го шага является **один младший** **бит** числа *хn+1*. (Иногда в качестве результата принимают бит чётности). Для целей криптографии этот метод предложен в 1986 году.

В моей программе я установил следующие значения:

1. p = 3571
2. q = 3559
3. M = 12709189
4. seed = 255

При помощи этого генератора я сгенерировал три битовых строки, из 50, 100 и 1000 битов. Битовые строки разделил на восьмибитные отрезки. Полученные восьмибитные значения перевел в десятичные для удобства отображения на гистограмме. Полученные гистограммы представлены ниже, полученные битовые строки, сгенерированные BBS генератором в Приложении Б.2.

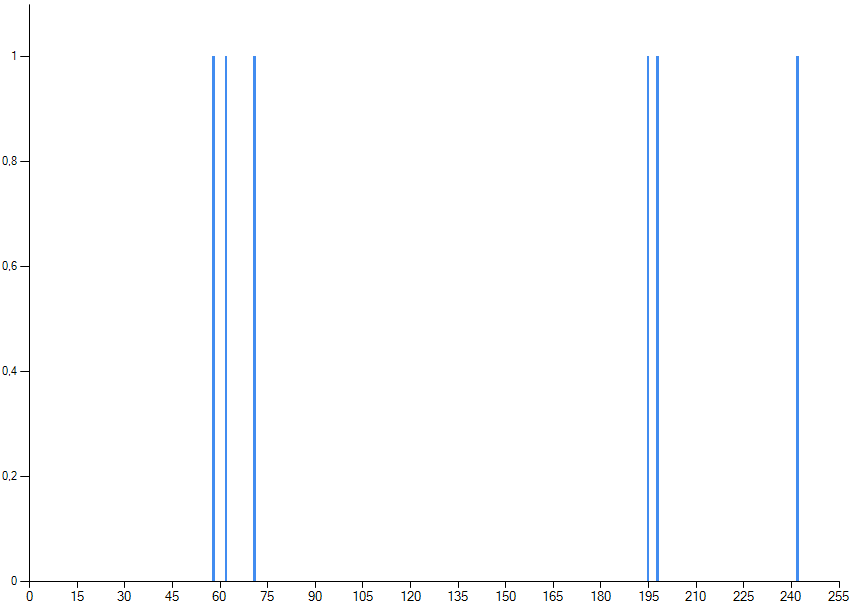


Рисунок 4 – Гистограмма 8-битных отрезков из 50-битной строки BBS

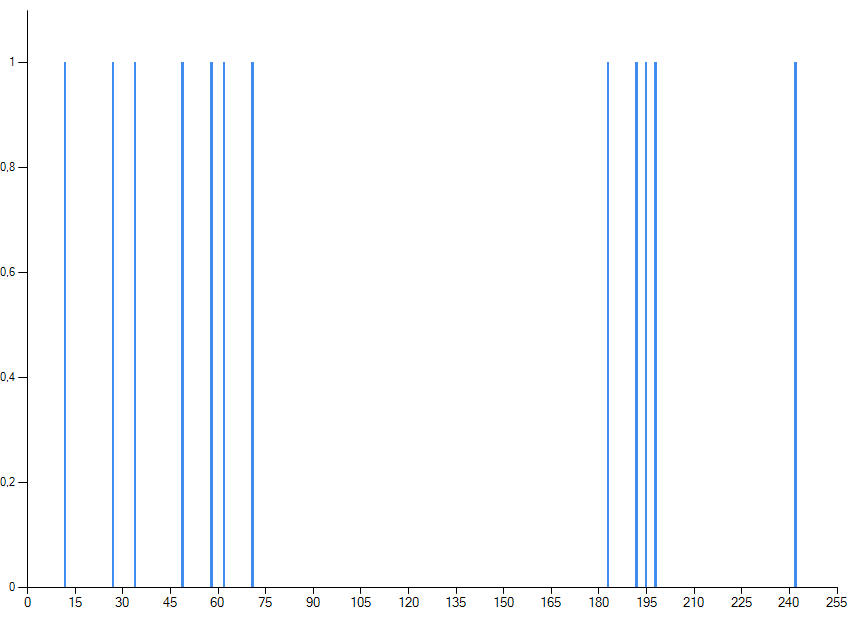


Рисунок 5 – Гистограмма 8-битных отрезков из 100-битной строки BBS

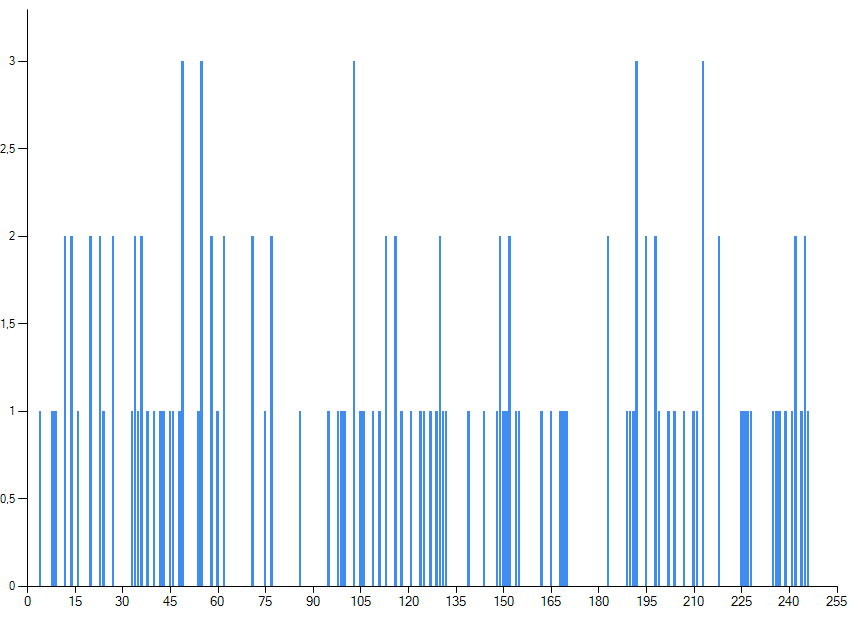


Рисунок 6 – Гистограмма 8-битных отрезков из 1000-битной строки BBS

# Линейный рекуррентный генератором (*LFSR*)

Работа генератора в поле моделируется уравнением

Генератор вырабатывает последовательность максимальной длины, если матрица А имеет вид: ,   
(an, an-1, …, a0) – коэффициенты неприводимого в поле многочлена. Соответствующий полином – x8 + x5 + x4 + x3 + 1.

В моей программе я установил следующие значения:

1. X = 181 (10110101)
2. A =

Такой выбор матрицы обусловлен тем, что при использовании матрицы из методических указаний, период генерации минимален (8 бит), а при использовании матрицы, представленной выше, период генерации больше (168 бит).

При помощи этого генератора я сгенерировал три битовых строки, из 50, 100 и 1000 битов. Битовые строки разделил на восьмибитные отрезки. Полученные восьмибитные значения перевел в десятичные для удобства отображения на гистограмме. Полученные гистограммы представлены ниже, полученные битовые строки, сгенерированные BBS генератором в Приложении Б.3.

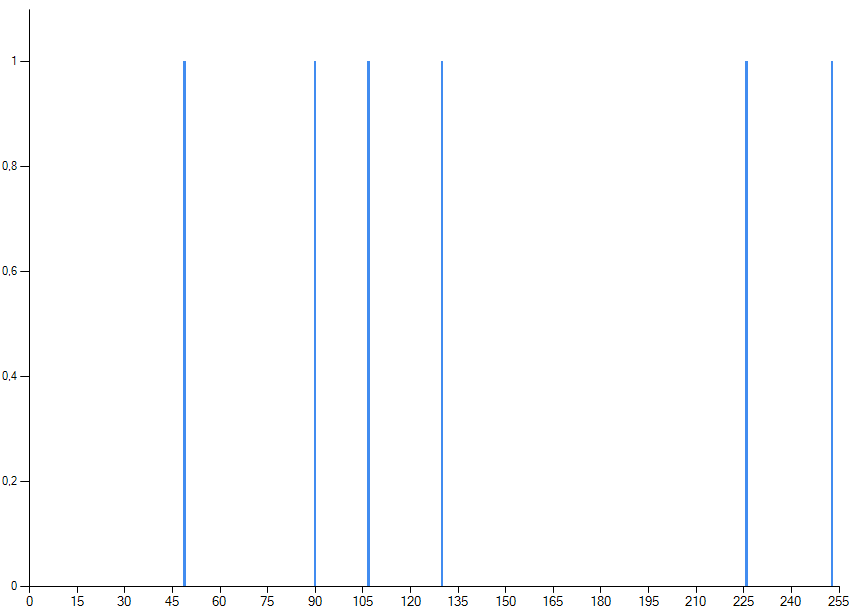


Рисунок 7 – Гистограмма 8-битных отрезков из 50-битной строки LFSR

Изображение выглядит как текст, линия, Параллельный, снимок экрана

Автоматически созданное описание

Рисунок 8 – Гистограмма 8-битных отрезков из 100-битной строки LFSR

Изображение выглядит как снимок экрана, линия, Параллельный, График

Автоматически созданное описание

Рисунок 9 – Гистограмма 8-битных отрезков из 1000-битной строки LFSR

# Приложение А

Листинг файла LCG.h

#pragma once

#include <vector>

#include <limits>

#include <stdexcept>

#define MAX std::numeric\_limits<uint>::max()

typedef unsigned int uint;

class LCG {

public:

LCG() {}

bool generate() {

last\_generated = (a \* last\_generated + c) % m;

return std::bitset<1>(last\_generated)[0];

}

private:

bool is\_coprime(uint a, uint b) {

while (b != 0) {

uint temp = b;

b = a % b;

a = temp;

}

return a == 1;

}

uint seed = 0;

uint m = 2147483647; // Число Мерсена 2^31 - 1 (2^p - 1)

uint a = 1664525;

uint c = 1013904223;

uint last\_generated = seed % m;

};

Листинг файла BBS.h

#pragma once

#include <vector>

#include <limits>

#include <stdexcept>

class BBS {

public:

BBS() : seed(255), p(3571), q(3559), M(p\* q), last\_generated((seed\* seed) % M) {

if (!is\_coprime(M, seed))

throw std::invalid\_argument("M и seed не взаимнопростые\n");

}

bool generate() {

last\_generated = (last\_generated \* last\_generated) % M;

return std::bitset<1>(last\_generated)[0];

}

private:

bool is\_coprime(size\_t a, size\_t b) {

while (b != 0) {

size\_t temp = b;

b = a % b;

a = temp;

}

return a == 1;

}

size\_t seed;

size\_t p;

size\_t q;

size\_t M;

size\_t last\_generated;

};

Листинг файла LFSR.h

#pragma once

#include <vector>

#include <stdexcept>

class LFSR {

public:

LFSR() : last\_generated(181) {}

std::vector<std::bitset<8>> generate(uint quantity) {

std::vector<std::bitset<8>> gen(quantity);

for (uint i = 0; i < quantity; i++) {

gen[i] = generate();

}

return gen;

}

bool generate() {

std::bitset<8> res(0);

for (int i = 0; i < 8; i++) {

for (int j = 0; j < 8; j++) {

res[i] = res[i] ^ (matrix[i][j] & last\_generated[j]);

}

}

last\_generated = res;

return last\_generated[0];

}

private:

std::bitset<8> last\_generated;

std::vector<std::vector<bool>> matrix =

{ { 0, 1, 0, 0, 0, 0, 0, 0 },

{ 0, 0, 1, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 1, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 1, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 1, 0, 0 },

{ 0, 0, 0, 0, 0, 0, 1, 0 },

{ 0, 0, 0, 0, 0, 0, 0, 1 },

{ 1, 0, 0, 1, 1, 1, 0, 1 } };

};

Листинг файла MZI\_2.h

#include <iostream>

#include <fstream>

#include <conio.h>

#include <string>

#include <bitset>

#include "LCG.h"

#include "BBS.h"

#include "LFSR.h"

typedef unsigned int uint;

int LCG\_init();

int BBS\_init();

int LFSR\_init();

int main()

{

setlocale(0, "");

unsigned char ch = 0;

while (ch != '4') {

system("cls");

std::cout << "МЕНЮ\n\n1. LCG\n2. BBS\n3. LFSR\n4. Выход\n>> ";

ch = \_getch();

if (ch < '4') {

switch (ch) {

case '1': LCG\_init(); break;

case '2': BBS\_init(); break;

case '3': LFSR\_init(); break;

}

}

}

return 0;

}

int LCG\_init() {

system("cls");

uint n = 0, seed = 0, a = 0, c = 0, m = 0;

std::cout << "LCG generator\n\nВведите требуемое количество битов\n>> ";

std::cin >> n;

std::ofstream out("generated\_LCG.bin", std::ios::binary);

if (!out.is\_open()) {

std::cout << "File was not opened\n";

return 1;

}

LCG LCG;

std::vector<bool> binary;

std::vector<uint8\_t> decimal;

std::bitset<8> b;

system("cls");

std::cout << "Сгенерированная LCG последовательность\n\nВ виде битовой строки:\n";

for (int i = 0; i < n + 1; i++) {

binary.push\_back(LCG.generate());

std::cout << binary.back();

if (i % 8 == 0 && i != 0) {

decimal.push\_back(b.to\_ulong());

out.write(reinterpret\_cast<const char\*>(&decimal.back()), sizeof(uint8\_t));

}

b[i % 8] = binary.back();

}

std::cout << std::endl << "\nВ виде 8-битных чисел:\n";

for (int i = 0; i < decimal.size(); i++) {

std::cout << (int)decimal[i] << " ";

}

std::cout << std::endl << std::endl;

out.close();

system("pause");

return 0;

}

int BBS\_init() {

system("cls");

uint n = 0, seed = 0, a = 0, c = 0, m = 0;

std::cout << "BBS generator\n\nВведите требуемое количество битов\n>> ";

std::cin >> n;

std::ofstream out("generated\_BBS.bin", std::ios::binary);

if (!out.is\_open()) {

std::cout << "File was not opened\n";

return 1;

}

BBS BBS;

std::vector<bool> binary;

std::vector<uint8\_t> decimal;

std::bitset<8> b;

system("cls");

std::cout << "Сгенерированная LCG последовательность\n\nВ виде битовой строки:\n";

for (int i = 0; i < n + 1; i++) {

binary.push\_back(BBS.generate());

std::cout << binary.back();

if (i % 8 == 0 && i != 0) {

decimal.push\_back(b.to\_ulong());

out.write(reinterpret\_cast<const char\*>(&decimal.back()), sizeof(uint8\_t));

}

b[i % 8] = binary.back();

}

std::cout << std::endl << "\nВ виде 8-битных чисел:\n";

for (int i = 0; i < decimal.size(); i++) {

std::cout << (int)decimal[i] << " ";

}

std::cout << std::endl << std::endl;

out.close();

system("pause");

return 0;

}

int LFSR\_init() {

system("cls");

uint n = 0, seed = 0, a = 0, c = 0, m = 0;

std::cout << "LFSR generator\n\nВведите требуемое количество битов\n>> ";

std::cin >> n;

std::ofstream out("generated\_LFSR.bin", std::ios::binary);

if (!out.is\_open()) {

std::cout << "File was not opened\n";

return 1;

}

LFSR LFSR;

std::vector<bool> binary;

std::vector<uint8\_t> decimal;

std::bitset<8> b;

system("cls");

std::cout << "Сгенерированная LFSR последовательность\n\nВ виде битовой строки:\n";

for (int i = 0; i < n + 1; i++) {

binary.push\_back(LFSR.generate());

std::cout << binary.back();

if (i % 8 == 0 && i != 0) {

decimal.push\_back(b.to\_ulong());

out.write(reinterpret\_cast<const char\*>(&decimal.back()), sizeof(uint8\_t));

}

b[i % 8] = binary.back();

}

std::cout << std::endl << "\nВ виде 8-битных чисел:\n";

for (int i = 0; i < decimal.size(); i++) {

std::cout << (int)decimal[i] << " ";

}

std::cout << std::endl << std::endl;

out.close();

system("pause");

return 0;

}

# Приложение Б

Приложение Б.1 LCG

Битовая строка из 50 битов

100110111110110100111010011110000000001110010110101

Битовая строка из 100 битов

10011011111011010011101001111000000000111001011010101101100111100111011111011001100101101101000110101

Битовая строка из 1000 битов

10011011111011010011101001111000000000111001011010101101100111100111011111011001100101101101000110101101110111110101111000110011100111000110000001001101001110000110110011011011000011100101000000000000111111001010111011000101100001100010100011001111000001100010111010001101100100110010101111110010100011010000010010000010000110100001111110110010000000101000001000111001111001010100010110111101000100101101110101100011100100010011010111000110111001011011100000001110100111110011101111011010001101000101010000001101101100010100011100101100001100011011000111001011111001110011111100001011010100010100000100001110011011101111101100100010110101111011001001110110001010101010011101100100100010001001110011010110111111101010001011011100101010000110111010100010011100100110100111111010010011011111010000000001011110110100001110111001100111001011111011111011110001110011110000100110100000010000111010001000100101101011100110001111001001001111001100011111111101101111011111000011100100010011000010110001100110110

Приложение Б.2 BBS

Битовая строка из 50 битов

110000110111110001001111111000100110001101011100001

Битовая строка из 100 битов

11000011011111000100111111100010011000110101110000110000111011010100010000000011100011001101100011101

Битовая строка из 1000 битов

11000011011111000100111111100010011000110101110000110000111011010100010000000011100011001101100011101100010110110100011001011001001000010100010101010100000110011011001011010100011100001000000101101001000011001110011011101001001010001010111100110111010000011100011110001110110000011110001101101111010101011010100110000111010000010010011010111110100001001011010010001110001001001110100000011001111010000010111011000110111110100111010010001111111100110010111000010000011100000010011111010001011011100011001100010100111111100111110100101111100100001110110001010110001010011001010111010010110010110000100001001011000000110010000011000100000110001101100110011110000010010101001101101010101111010011110011111101001001001010101111110111001010001001011000111110111001100001010111100110110101111010100110101011101101101010101110001100101011111010010101100100101100100110110001000111111101101011011111000011011111000100111111100010011000110101110000110000111011010100010000000011100011001101100011101100010110110

Приложение Б.3 LFSR

Битовая строка из 50 битов

010110100100011101000001101111111000110011010110100

Битовая строка из 100 битов

01011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110

Битовая строка из 1000 битов

01011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100011001101011010010001110100000110111111100