1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. **Институт прикладной математики и механики**
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ЛАБОРАТОРНАЯ РАБОТА №2**

1. по дисциплине «Структуры данных»
2. Выполнил
3. студентка гр. 13636/2 Ивченкова Н. С

1. Проверил Алексеев И.В
2. ассистент

1. Санкт-Петербург
2. 2018
3. **Цели:**

Написать программу, выдающую на выход последовательность псевдослучайных чисел.

1. **Задачи:**

Высчитать оптимальные параметры для ЛКГ, проверить выходную псевдослучайную последовательность тестом на случайность.

1. **Ход работы.**

Теоретические сведения:

**Линейный конгруэнтный генератор (ЛКГ)** – это последовательность чисел от 0 до m −1 , удовлетворяющая следующему рекуррентному выражению X(k+1)=(X( k) a + с)modm.

X 0 - начальное значение, a – множитель, с – приращение, m – модуль.

Период максимален в том случае, когда параметры a, m и c обладают следующими свойствами:

1.  Числа c и m взаимно просты.

2.  Число a-1 кратно для каждого простого p, являющегося делителем m.

3. Число a-1 кратно 4, если  m кратно 4.

**1) модуль m**  
а) Необходимо, чтобы было довольно большим.

б) m – длина слова компьютера (2^k)  
**2)множитель a**

а)<=m

б)потенциал s >=4. (a-1)^s mod m=0, s- минимальное.

**3)приращение c**

а)не равно нулю

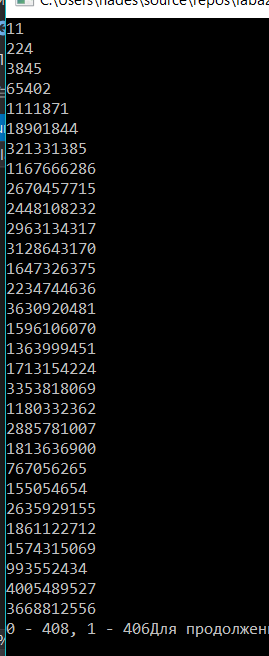
б)c mod 8 = 5

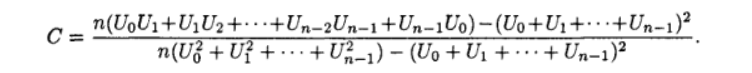
Рисунок 1 – результаты побитовой проверки

Практика

1. Используя данные свойства, подберем параметры ЛКГ(программа перебирает все возможные множители и приращения, которые бы удовлетворяли параметры, описанные выше.
2. С помощью частотного побитового теста проверяем наш ЛКГ на случайность.

1)Побитовый. Суть данного теста заключается в определении соотношения между нулями и единицами во всей двоичной последовательности. Цель — выяснить, действительно ли число нулей и единиц в последовательности приблизительно одинаковы, как это можно было бы предположить в случае истинно случайной бинарной последовательности. Тест оценивает, насколько близка доля единиц к 0,5. Таким образом, число нулей и единиц должно быть примерно одинаковым.

На рисунке 1 виден результат такого подсчета при вводе в качестве первого элемента, например, 11(В зависимости от выбора этого элемента, результат может меняться). И, как видно, что количество нулей и единиц практически совпадает, а значит, выбранные параметры дают хорошие показатели.

2) Сериальная корреляция.   
Необходимо рассчитать коэффициент сериальной корреляции по данной формуле 

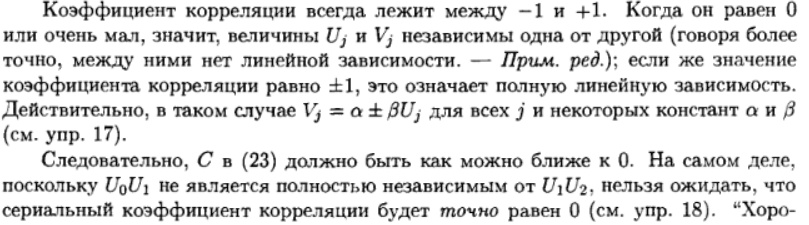
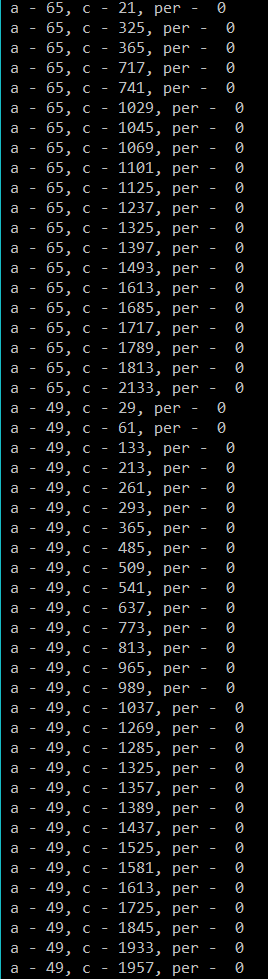
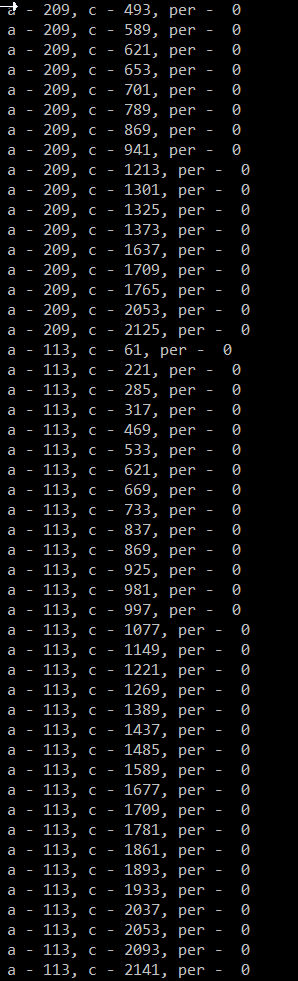
А принцип описан в книге Кнута. Фрагмент: 

Рисунок 2 Описание Сериальной корреляции

Итог: получили такие параметры ЛКГ: 

1. **Выводы:** При выполнении всех условий для получения максимального периода ЛКГ есть высокая вероятность действительно получить максимальный период. Также можно заметить, что, хоть предполагается введение нулевого элемента произвольным, проверка теста на случайность показывает, что выбор и этого элемента может повлиять на результат.
2. **Код:**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

int prov[2]; long long parametr[3]; long long c = 13; long long a; long long parametr\_a[5] = { 0 }; int max = 4; int ai = 0; long long parametr\_c[500] = { 0 }; int ci = 0;

long double prost(long long c, long long m) {

long long div = 2; int i = 0; long long prost[100] = { 0 };

while (c >= div)

{

while (c % div == 0)//раскладываем на ПРОСТЫЕ

{

c = c / div;

prost[i] = div; i++;

}

div++;

}

i = 0; div = 2;

while (m >= div)

{

while (m % div == 0)//раскладываем на ПРОСТЫЕ

{

m = m / div;

if (prost[i] == div) return 0;

if ((prost[i] < div) && prost[i] != 0) i++;

}div++;

}return 1;

}

long long proverka\_c(long long m)

{

for (c; c < m / 2000000; c++) {

if (c % 8 == 5) {

if (prost(c, m) == 1) {

parametr\_c[ci] = c; ci++;

}

}

}

}

int simple(long long n) {

for (long long i = 2; i <= n / 2; i++) if ((n%i) == 0) return 0;

return 1;

}

void mnozitel(long long m)

{

while (a != 10) {

long long b = a - 1;//3 условие

if (b % 4 == 0) { // 2 условие

for (long long p = 2; p <= m / 1000000; p++) {

if (simple(p) == 1) { // если р простое

if (m%p == 0) { // и к тому же является делителем m

if (b%p == 0) {// и если b кратно этому р

for (int s = 11; s > 4; s--) {

long long k = pow(b, s);//проверка на мощность

if ((k%m == 0) && k > 0) {

if (max < s) {

max = s; parametr\_a[ai] = a; ai++;//ПРОВЕРИМ ВСЕ А И ЗАПИШЕМ В МАССИВ ПОДХОДЯЩИЕ ВАРИАНТЫ

}

}

}

}

}

}

}

}a--;

}

}

int proverka2(long long x[], int n) {//СЕРИАЛЬНАЯ КОРРЕЛЯЦИЯ

long long sum1, sum2, sum3; sum1 = 0; sum2 = 0; sum3 = 0;

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

sum2 = sum2 + x[i];

sum3 += pow(x[i], 2);

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

sum1 = sum1 + x[i] \* x[i + 1];

long double res;

res = (n\*sum1 - pow(sum2, 2)) / (n\*(sum3 - pow(sum2, 2)));

return res;

}

void proverka(long long n)//ПОБИТОВЫЙ

{

long long l;

long long a2, a3;

long long two[31];

a3 = n; int one, zero = 0;

a2 = n;

int k = 0;

while (a2 > 1)

{

a2 = a2 / 2;

k++;

}

for (l = 31; l >= 31 - k; l--)

{

if ((a3 % 2) == 1)

prov[1]++;

else

prov[0]++;

a3 = a3 / 2;

}

}

void main()

{

prov[1] = 0; prov[0] = 0; long long per = 0;

//scanf("%lld", &x[0]);

long long i = 1; double prosent = 0; long long \*x = NULL;

long long m = pow(2, 32); a = m / 20000000;

mnozitel(m);

proverka\_c(m);

for (int i1 = 0; i1 < ai; i1++) {

for (int i2 = 0; i2 < ci; i2++) {

x = (long long\*)malloc(20000000 \* sizeof(long long)); x[0] = 17;

while (i != 20000000) {

x[i] = (parametr\_a[i1] \* x[i - 1] + parametr\_c[i2]) % m;

if (i == 10) {

proverka(x[i]);

}

i++;

} i = 1; double zero = prov[0];

double one = prov[1];

prov[0] = 0; prov[1] = 0;

prosent = zero / one;

if ((prosent < 1.04) && (prosent > 0.96)) {

long double res = proverka2(x, 5);

if ((res < 1.000000000) && (res >= 0.000000000000)) {

}for (i = 2; i < 20000000; i++) {

if (x[1] == x[i])per = i - 1;

}

printf(" a - %lld, c - %lld, per - %lld\n", parametr\_a[i1], parametr\_c[i2], per);

}free(x);

}

}

system("pause");

}