Министерство образования и науки Российской Федерации

1. Санкт-Петербургский Политехнический Университет Петра Великого
2. —
3. Институт компьютерных наук и кибербезопасности

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «**Линейный конгруэнтный генератор**»
2. по дисциплине «Структуры данных»
3. Выполнил
4. студент гр. 5151001/40001 Кириллов Д.А

<*подпись*>

1. Преподаватель
2. асс. преподавателя Семьянов П.В

<*подпись*>

1. Санкт-Петербург
2. 2025

# Цель работы

Разработать программу, которая реализует линейный конгруэнтный генератор псевдослучайных чисел, и провести тесты этого генератора.

# Постановка задачи

1. Выбрать правильные параметры ЛКГ.
2. Проверить характеристики полученного ЛКГ, а именно период, мощность и разброс.
3. Проверить ЛКГ критерием хи-квадрат.
4. Проверить ЛКГ критерием перестановок из 2 тома “Искусство программирования ЭВМ” Д.Кнута.

# Теоретические исследования

## Характеристики ЛКГ

Линейно конгруэнтный генератор – один из методов генерации псевдослучайных чисел. Применяется в простых случаях и не обладает криптографической стойкостью.

Суть метода заключается в вычислении последовательности случайных чисел , пологая, что: .

При выборе числа m нужно учитывать следующие условия:

1. Число m должно быть довольно большим, так как период не может иметь больше m элементов.
2. Значение числа m должно быть таким, чтобы вычислялось быстро.

При реализации метода используют чаще всего выбирают 2^k - 1, где k-число битов в машинном слове.

Так как линейно конгруэнтный генератор определен числами m, a, c, то от выбора данных параметров будут зависеть его характеристики. Чтобы период ЛКГ был равен числу m, нужно, чтобы выполнялись следующие условия:

1. Числа c и m взаимно простые.
2. кратно p для каждого простого p, являющегося делителем числа m.
3. b кратно 4, если m кратно 4.
4. - произвольное число.
5. должны быть такими, что: и c – нечетное.

В зависимости от выбранных параметров можно рассматривать такие характеристики ЛКГ:

* Мощность – это зависимость коэффициентов a и m. По формуле можем определить i - мощность.
* Разброс. Если числа из промежутка [a;b] появляются с одинаковой вероятностью, то разброс ЛКГ является хорошим.
* Период. Каждое число последовательности, созданной генератором псевдослучайных чисел, повторится через какой-то количество чисел. Это количество числе и называется периодом.

## Критерий «Хи-квадрат

Этот критерий является одним из самых известных. Каждое число должно генерироваться с какой-то одинаковой вероятностью. В данном методе сравнивается теоретическая вероятность, с которой появляется определенное число, с практической. В данной лабораторной работе этот критерий был реализован так: промежуток от 0 до максимального числа делится на группы с равным количество чисел. Тогда теоретическая вероятность того, что число попадет в одну из групп: . Далее рассчитывается число V по формуле: , где k - количество групп, – количество чисел, попавших в группу, n-количество чисел в последовательности.

Коэффициент V был получен для 100 последовательностей, каждая из которых была разделена на 31 промежуток и соответственно v = 30 степеней свободы. По таблице для метода хи-квадратов из книги Кнута «Искусство программирования, 2 том» в ряду с числом v найти число, близкое к V. Посмотреть значение p. Если диапазон для V: p < 1% или p > 99%, то данная последовательность считается очень плохой, т.к там недостаточно случайные числа, 1% < p < 5% или 95% < p < 99%, то последовательность считается подозрительной, 5% < p < 10% или 90% < p < 95%, то последовательность считается почти подозрительной, p > 10% или p < 90%, то последовательность считается отличной. Т.к генерируется несколько последовательностей, то генератор считается хорошим, если не более 2/3 из них подозрительные (или недостаточно случайные).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 1 – Распределение Хи-квадрат

# Описание решения и тестирование программы

## Параметры ЛКГ

Учитывая условия из теоретического условия, были подобраны следующие параметры: seed = rand(), , a = 6364136223846793005,

c = 1442695040888963407.

В программе есть меню, благодаря которому можно узнать характеристики ЛКГ или сделать проверку последовательности псевдослучайных чисел.

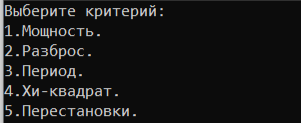


Рисунок 2 – Меню программы

## Характеристики ЛКГ

### Мощность

По формуле можем рассчитать мощность ЛКГ. Это осуществляется благодаря циклу, если на i-итерации цикла число будет равно 0, значит число i – мощность. Это осуществляется за счет переполнения переменной unsigned long long, поэтому не выполняется действие сравнения по модулю (mod m).

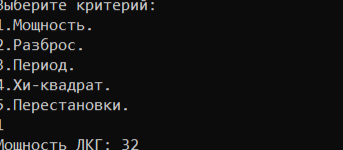


Рисунок 3 – Мощность ЛКГ

### Период

Период высчитывается за счет цикла. В переменную count записывается псевдослучайное сгенерированное число, а далее генерируются числа до тех пор, пока не найдется число, равное по значению переменной count. За несколько часов генерации не нашлось такого числа, которое было бы равным по значению переменной count

### Разброс

Количество чисел было разбито на 100 равных групп. Отклонение в каждой группе от предполагаемого значения не превышало 1%.

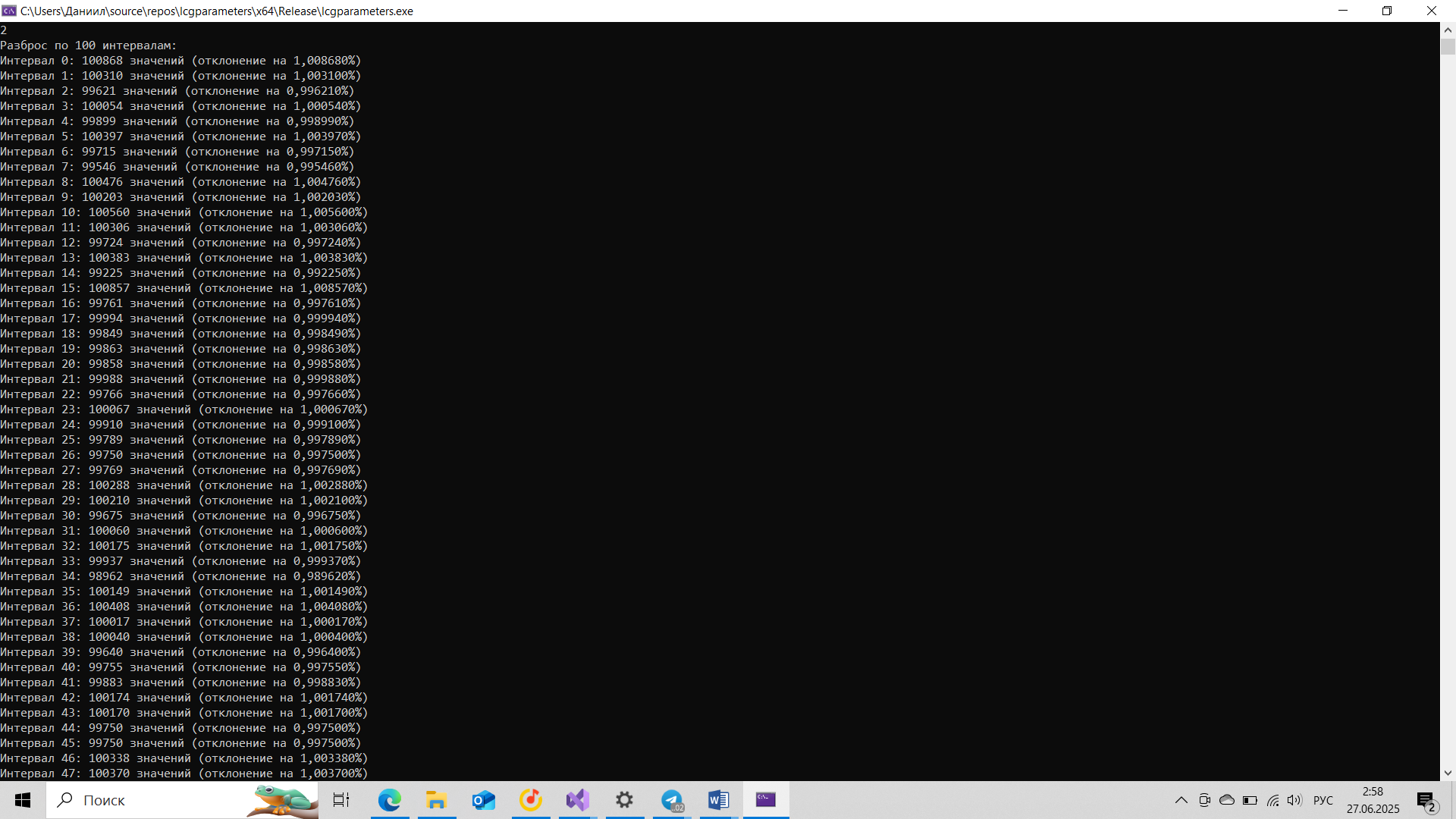


Рисунок 4 – разброс по 100 промежуткам

## Хи-квадрат

Числа были разделены на 31 группу. С помощью циклов рассчитывается, сколько чисел попадает в первую группу, во вторую и так далее. По формуле высчитывается значение V для каждой последовательности чисел (было взято 100 последовательностей). – это количество чисел, которые попали в данную группу, – количество чисел в последовательности, деленное на количество групп. Определяется тип последовательности по попаданию в интервалы критических значений хи-квадрата. После вычисляется количество всех описанных ранее последовательностей и если подозрительных и недостаточно случайных из них меньше 2/3, то генератор считается хорошим.

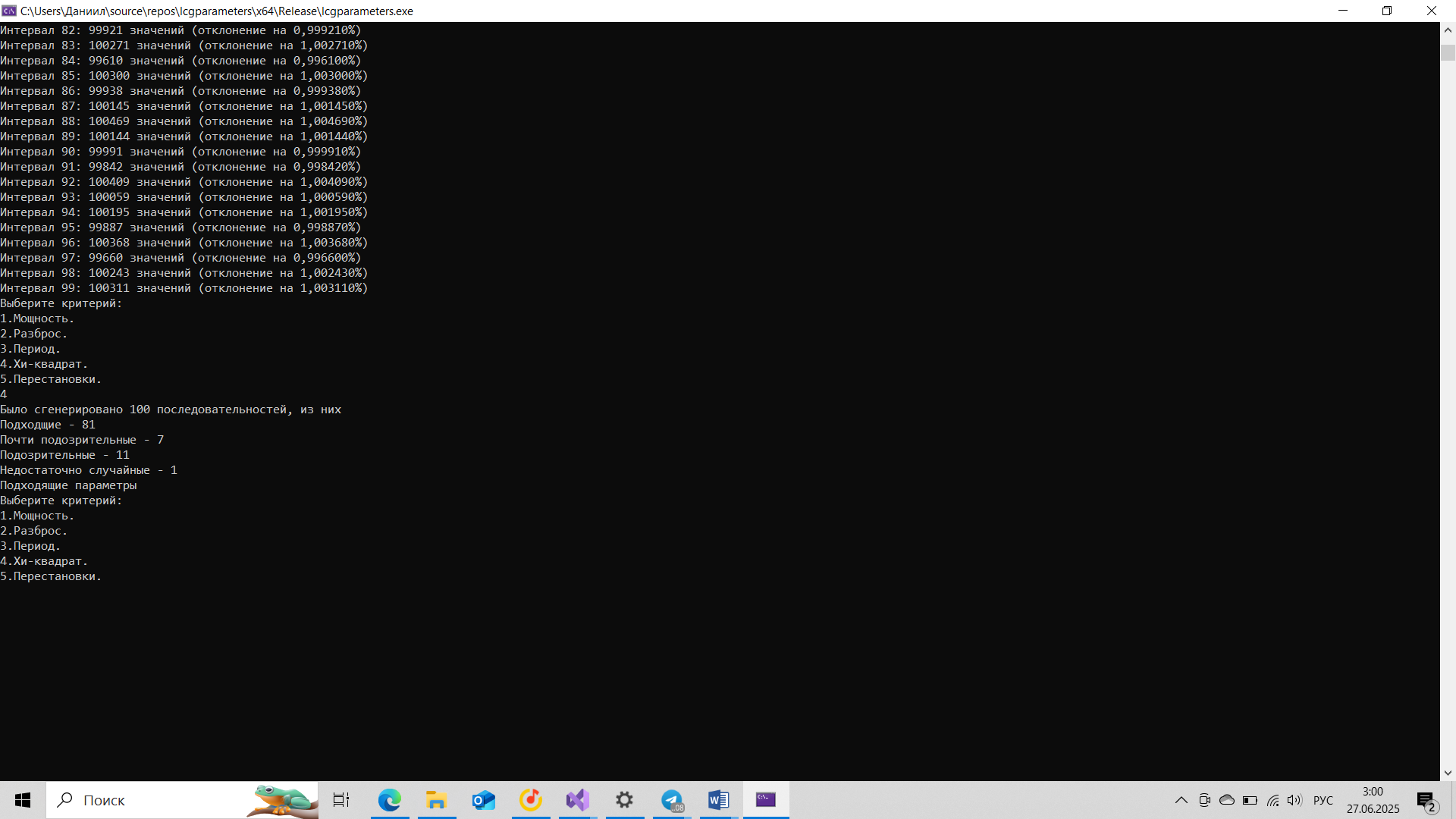


Рисунок 5 – Проверка Хи-квадрат

## Проверка критерием перестановок.

Разделим последовательность на выходе на n групп по t (в моем случае 3) элементов к каждой. Элементы в этой группе можно упорядочить t! различными способами. Подсчитывается число групп с любыми возможными перестановками и применяется критерий хи-квадрат к k=t! возможным категориям с вероятностью 1/t! для каждой. Проделывается это для нескольких последовательностей и делается вывод как в предыдущем пункте.

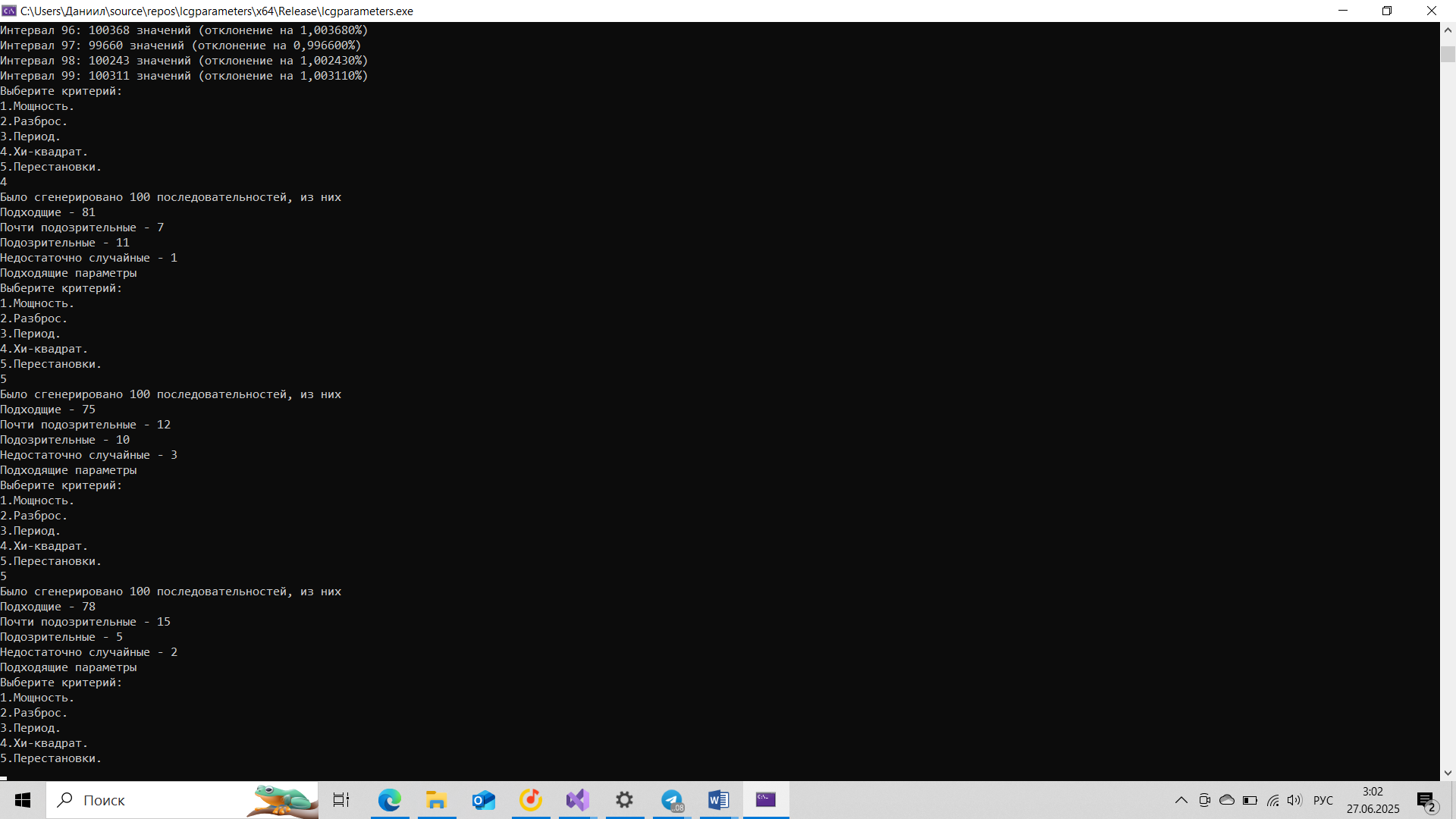


Рисунок 7 – Проверка перестановок.

# Выводы

В результате выполнения лабораторной работы был изучен ЛКГ и подобраны оптимальные параметры для него. Данный генератор прошел проверку на различные критерии, а полученные характеристики ЛКГ показывают, что генератор является хорошим и его можно использоваться для генерации псевдослучайных чисел.

Список используемых источников

1. Д. Кнут - Искусство программирования том 2

ПРИЛОЖЕНИЕ

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <locale.h>

#include <ctime>

#define ull unsigned long long

ull size = 10000000;

ull a = 6364136223846793005;

ull m = 18446744073709551615;

ull c = 1442695040888963407;

// Функция для вычисления факториала

ull factorial(int n) {

unsigned long long result = 1;

for (int i = 2; i <= n; i++) {

result \*= i;

}

return result;

}

// Функция для вычисления индекса перестановки

int permutation\_index(ull\* sequence, int t) {

int f = 0;

for (int r = t; r > 1; r--) {

int max\_index = 0;

for (int i = 1; i < r; i++) {

if (sequence[i] > sequence[max\_index]) {

max\_index = i;

}

}

f = r \* f + max\_index;

ull temp = sequence[max\_index];

sequence[max\_index] = sequence[r - 1];

sequence[r - 1] = temp;

}

return f;

}

//Функция, создающая массив для будущего нахождения значения хи квадрата

int\* make\_mass(ull\* sequence, int\* counts,int k) {

for (ull i = 0; i < size; i++) {

double normalized = (double)sequence[i] / m;

int index = (int)(normalized \* k);

counts[index]++;

}

return counts;

}

//Функция, находящая хи квадрат последовательности

double find\_hi2(int\* counts, int num, ull Size) {

double hi2 = 0.0;

double np = (double)Size / num;

for (int s = 0; s < num; s++) {

hi2 += pow((counts[s] - np), 2) / np;

}

return hi2;

}

void print\_result(double hi2, double\* crit\_value, int\* result) {

if (hi2 <= crit\_value[0] || hi2 >= crit\_value[5]) {

//printf("Недостаточно случайные числа %f\n", hi2);

result[0]++;

}

else if ((crit\_value[0] < hi2 && hi2 <= crit\_value[1]) || (crit\_value[4] <= hi2 && hi2 < crit\_value[5])) {

//printf("Подозрительные числа %f\n", hi2);

result[1]++;

}

else if ((crit\_value[1] < hi2 && hi2 <= crit\_value[2]) || (crit\_value[3] <= hi2 && hi2 < crit\_value[4])) {

//printf("Почти подозрительные числа %f\n", hi2);

result[2]++;

}

else if (crit\_value[2] < hi2 && hi2 < crit\_value[3]) {

//printf("Подходящие числа %f\n", hi2);

result[3]++;

}

}

unsigned long long lcg(ull\* seed, ull a, ull c, ull m) {

return (\*seed \* a + c) % m;

}

void Power() {

ull b = a - 1;

ull g = a - 1;

for (int i = 1; i <= 64; ++i) {

if (b % m == 0) {

printf("Мощность равна: %d\n", i);

return;

}

b \*= (a - 1);

}

}

void Scatter() {

ull seed = rand()\*1234;

ull\* sequence = (ull\*)malloc(size \* sizeof(ull));

sequence[0] = seed;

for (ull i = 1; i < size; i++) {

sequence[i] = lcg(&sequence[i - 1], a, c, m);

}

const int bins = 100;

int counts[bins];

for (int i = 0; i < bins; i++) counts[i] = 0;

for (ull i = 0; i < size; i++) {

double normalized = (double)sequence[i] / m;

int index = (int)(normalized \* bins);

counts[index]++;

}

printf("Разброс по %d интервалам:\n", bins);

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

float otkl = float(counts[i]) / (size / bins);

printf("Интервал %d: %d значений (отклонение на %f%%)\n", i, counts[i],otkl);

}

}

void Period() {

unsigned long long count = 0;

long long period = 1;

printf("Выполняется подсчет периода...\n");

ull x = rand();

count = lcg(&x,a,c,m);

x = lcg(&x, a, c, m);

while (count != lcg(&x,a,c,m)){

x = lcg(&x, a, c, m);

period++;

//printf("\n%llu", period);

}

printf("Период равен %llu\n", period);

}

void Hi2() {

double crit\_value[6] = { 14.953, 18.493, 20.599,40.256, 43.773, 50.892 };

int n = 100;

int k = 31;

int result[4] = {0};//i = 0 - недостаточно случайные числа, i = 1 - подозрительные, i = 2 - почти подозрительные, i = 3 - подходящие

while (n) {

ull\* sequence = (ull\*)malloc(size \* sizeof(ull));

sequence[0] = rand();

for (ull i = 1; i < size; i++) {

sequence[i] = lcg(&sequence[i - 1], a, c, m);

}

int counts[100] = {0};

make\_mass(sequence, counts, k);

double hi2 = find\_hi2(counts,k, size);

print\_result(hi2, crit\_value, result);

free(sequence);

n--;

}

printf("Было сгенерировано 100 последовательностей, из них\n");

printf("Подходщие - %d\nПочти подозрительные - %d\nПодозрительные - %d\nНедостаточно случайные - %d\n", result[3], result[2], result[1], result[0]);

if ((result[2] + result[1] + result[0]) / 100 > 2 / 3) { printf("Плохие параметры, т.к частота подозрительных результатов больше 2/3(%d)\n", (result[2] + result[1] + result[0]) / 100); }

else { printf("Подходящие параметры\n"); }

}

void Permutations() {

double crit\_value[6] = { 0.554, 1.145, 1.610, 9.236, 11.070, 15.086 };

int iter = 100;

int t = 3;

int result[4] = { 0 };

while (iter) {

ull\* sequence = (ull\*)malloc(size \* sizeof(ull));

sequence[0] = rand();

for (ull i = 1; i < size; i++) {

sequence[i] = lcg(&sequence[i - 1], a, c, m);

}

ull n = size / t;

int k = factorial(t);

int counts[7] = { 0 };

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

ull group[4] = { 0 };

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

group[j] = sequence[i \* t + j];

}

int index = permutation\_index(group, t);

counts[index]++;

}

double hi2 = find\_hi2(counts,k, n);

print\_result(hi2, crit\_value, result);

iter--;

free(sequence);

}

printf("Было сгенерировано 100 последовательностей, из них\n");

printf("Подходщие - %d\nПочти подозрительные - %d\nПодозрительные - %d\nНедостаточно случайные - %d\n", result[3], result[2], result[1], result[0]);

if ((result[2] + result[1] + result[0]) / 100 > 2 / 3) { printf("Плохие параметры, т.к частота подозрительных результатов больше 2/3(%d)\n", (result[2] + result[1] + result[0]) / 100); }

else { printf("Подходящие параметры\n"); }

}

int main() {

setlocale(LC\_ALL, "rus");

while (1) {

int action;

printf("Выберите критерий:\n1.Мощность.\n2.Разброс.\n3.Период.\n4.Хи-квадрат.\n5.Перестановки.\n");

scanf("%d", &action);

switch (action)

{

case 1:

Power();

break;

case 2:

Scatter();

break;

case 3:

Period();

break;

case 4:

Hi2();

break;

case 5:

Permutations();

break;

default:

printf("Неверный номер критерия.\n");

break;

}

}

return 0;

}