1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «**Линейно Конгруэнтный Генератор (ЛКГ)**»
2. по дисциплине «Структуры данных»
3. Выполнил
4. студент гр.5151001/40001 Волошкевич М.А.

<*подпись*>

1. Преподаватель Семьянов П.В.

<*подпись*>

1. Санкт-Петербург
2. 2025 г.
3. **Цель работы**

Реализовать ЛКГ с собственными коэффициентами и проверить его качественность.

**2. Постановка задачи**

Написать программу, которая генерирует псевдослучайные числа. Вручную проверить мощность, период, разброс. Пользователь может самостоятельно проверить ЛКГ по двум критерия со своими данными:

-критерий хи-квадрат

-критерий по Пародоксу Дней Рождений

**3. Теоретические исследования**

Для выполнения работы нужно было разобраться, как именно работает линейно конгруэнтный генератор, какие коэффициенты на что влияют. Найти оптимальную конфигурацию.

**4. Описание решения**

Для выбора оптимальных параметров было принято решение воспользоваться именно этими критериями (из лекций по АиП и Кнуту):

- **m** следует брать 2^k, где k-длинная машинного слова.

- **a** следует брать такое что:

1) *0.01m < a < 0.99m*

2) *a mod 8 = 5*

3) *a – не регулярное (не красивое\*)*

- **c** следует брать нечетным.

Подобрав такие коэффициенты, мы теоретически добиваемся высокой мощности, при этом сохраняя большой период.

В нашем случае было выбрано:

m=2^64

a=92233720368547765

c=3737373737

**5. Тестирование и результаты работы программы**

**1)Проверка мощности:**

void power() {

int s = 1;

ull b1 = b;

for (int i = 0; i < 123; i++) {

b = b \* b1;

s++;

if (b == 0) {

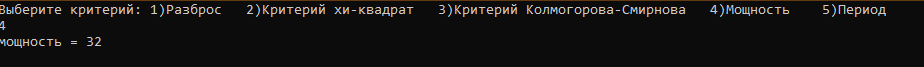
printf("мощность = %llu\n", s);

return;

}

}

}



**2)Проверка разброса:**

void razbrosik(ull x) {

ull r = 0;

printf("Введите правую границу: ");

scanf("%llu", &r);

ull n = 0;

printf("Количество измерений: ");

scanf("%llu", &n);

ull\* mass = new ull[r]{ 0 };

while (n--) {

mass[x % r]++;

x = randim(x);

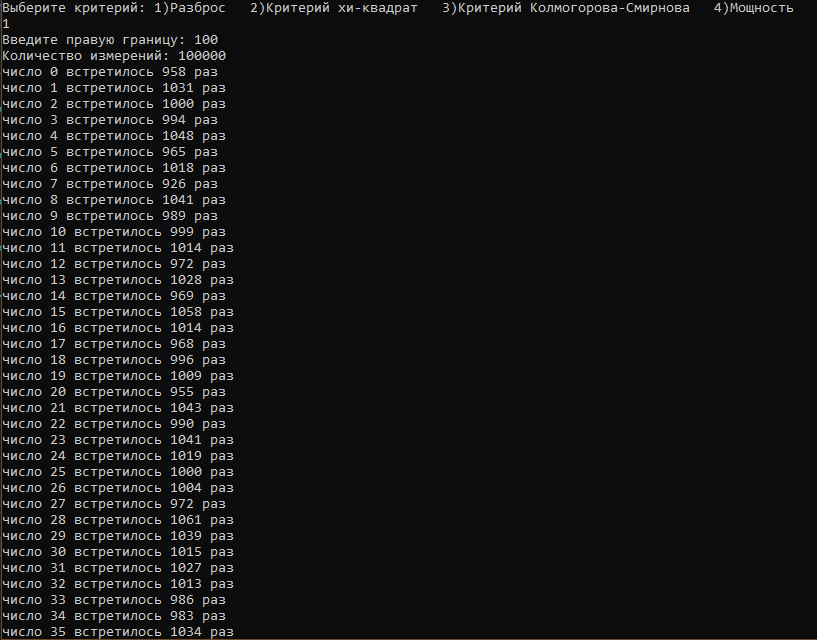
}

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

printf("число %llu встретилось %llu раз\n", i, mass[i]);

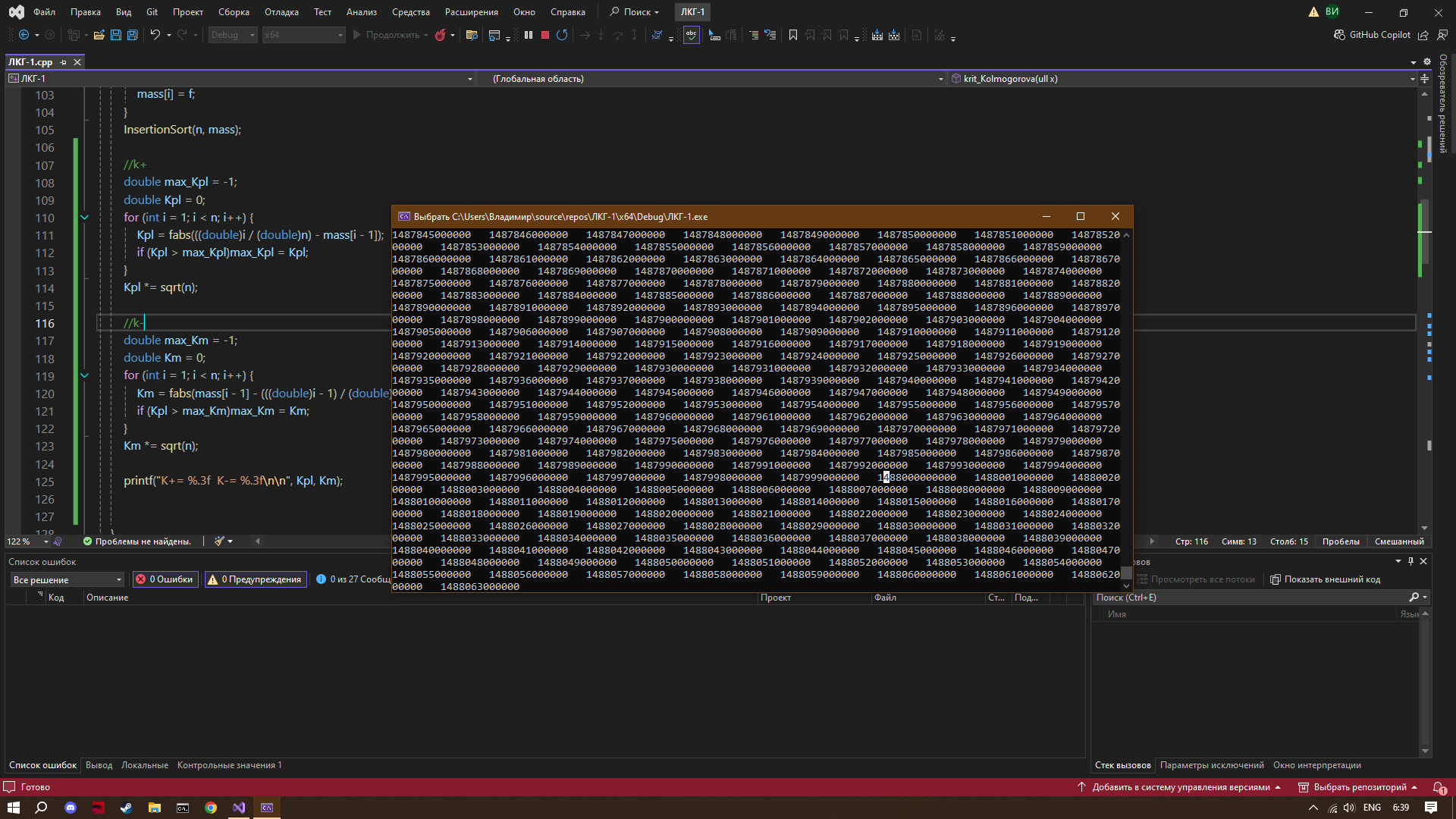
}

}



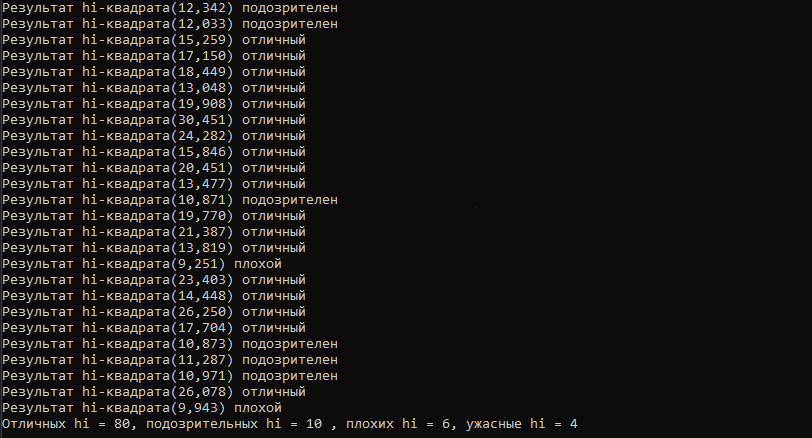
**3) Проверка периода:**

Для проверки периода программа была запущена пока последовательность не начнет повторяться. Теоретическое значение периода = 2^64, на практике за 12 часов было сгенерировано 1 488 063 000 000 случайных чисел, последовательность ни разу не повторилась. Можно предположить, что практическая длинна периода явно > 2^40.

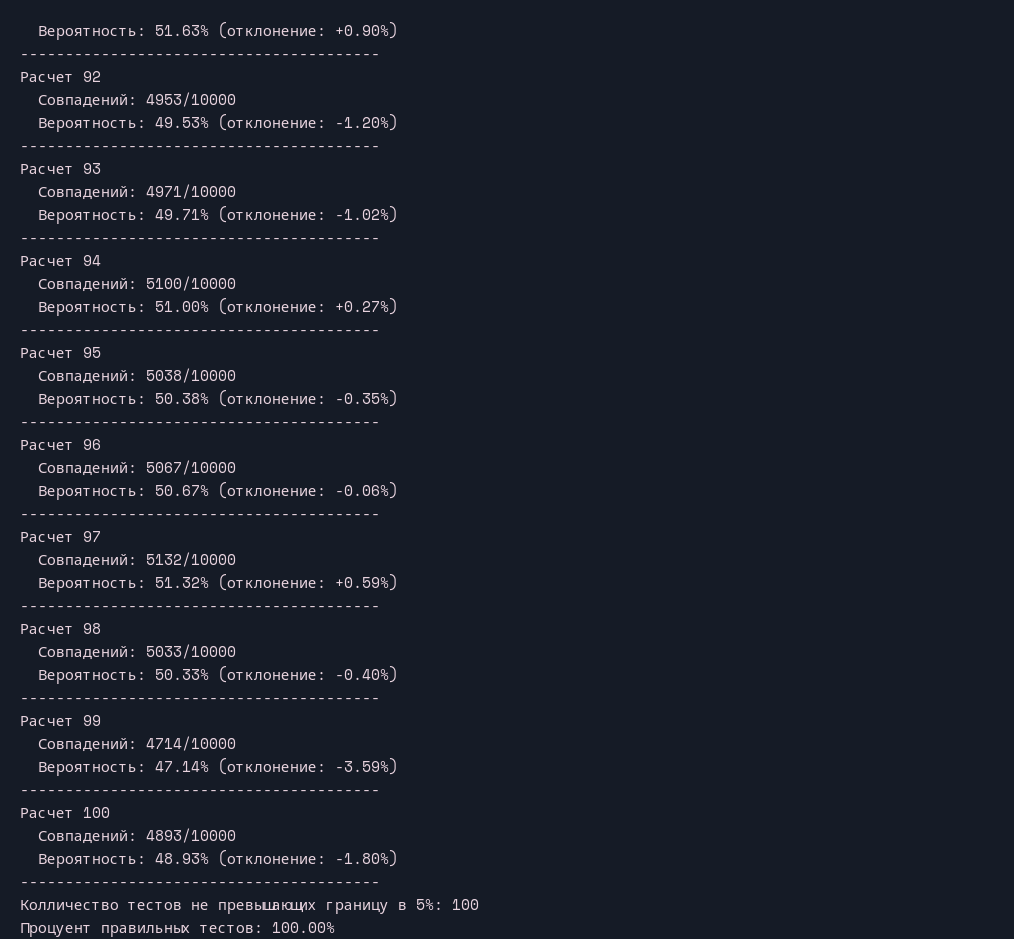


**4)Проверка Хи-квадрата:**

*Приложение 1.*



**5)Проверка критерия Парадокс Дней Рождений:**

*Приложение 2.*

**6. Вывод**

Были подобранны оптимальные коэффициенты, дающие максимальный период с высокой мощностью. ЛКГ прошёл два критерия:

Хи-квадрат- достигнуто процентное значение в 80% хороших результатов, что удовлетворяет условию в 2/3 должны быть хорошими значения.

Критерий Парадокса Дней Рождений - достигнуто необходимое соотношение хороших результатов к плохим в 93% (хорошие+отличные), что удовлетворяет условие, в 90% случаях расхождение K- и K+ должно быть минимальным и не быть подозрительным.

Однако чтобы утверждать, что этот ЛКГ в действительности может генерировать числа достаточной случайности и последовательность не будет иметь дефектов, необходимо проверить ЛКГ по еще 4 критериям (по Кнуту, необходимо в общей сложности пройти минимум 6 тестов).

**7. Приложения**

**Приложение1:**

**Приложение2:**

void birthday\_paradox(ull initial\_seed) {

ull num\_calculations = 0;

ull num\_people = 0;

ull counts\_of\_correct = 0;

int fringe\_diff = 5;

printf("Введите количество расчетов вероятности: ");

scanf("%llu", &num\_calculations);

printf("Введите количество людей в эксперименте (n): ");

scanf("%llu", &num\_people);

if (num\_calculations == 0 || num\_people == 0) {

printf("Количество расчетов и людей должно быть больше 0.\n");

return;

}

// Вычисляем теоретическую вероятность

double theoretical\_prob = 1.0;

for (ull i = 0; i < num\_people; ++i) {

theoretical\_prob \*= (365.0 - i) / 365.0;

}

theoretical\_prob = (1.0 - theoretical\_prob) \* 100.0;

printf("\nТеоретическая вероятность совпадения для %llu людей: %.2f%%\n\n",

num\_people, theoretical\_prob);

for (ull calc = 0; calc < num\_calculations; ++calc) {

ull collisions\_in\_100 = 0;

// Генерируем уникальный seed для каждого расчета

ull current\_calc\_seed = initial\_seed + calc;

ull experiment\_seed = current\_calc\_seed;

printf("Расчет %llu \n", calc + 1);

// 100 экспериментов с текущим seed

for (ull exp = 0; exp < 10000; ++exp) {

HashTable\* ht = create\_hashtable(365);

if (ht == NULL) {

perror("Ошибка создания хеш-таблицы");

return;

}

bool has\_collision = false;

ull current\_x = experiment\_seed;

for (ull person = 0; person < num\_people; ++person) {

current\_x = randim(current\_x);

ull birthday = (current\_x % 365) + 1;

if (insert\_hashtable(ht, birthday)) {

has\_collision = true;

break;

}

}

free\_hashtable(ht);

if (has\_collision) {

collisions\_in\_100++;

}

// Обновляем seed для следующего эксперимента

experiment\_seed = randim(experiment\_seed);

}

double experimental\_prob = (double)collisions\_in\_100 / 10000 \* 100.0;

double deviation = experimental\_prob - theoretical\_prob;

printf(" Совпадений: %llu/10000\n", collisions\_in\_100);

printf(" Вероятность: %.2f%% (отклонение: %+.2f%%)\n",

experimental\_prob, deviation);

printf("----------------------------------------\n");

if(fabs(deviation) < fringe\_diff) counts\_of\_correct++;

}

printf("Колличество тестов не превышающих границу в %d%%: %llu \n", fringe\_diff, counts\_of\_correct);

printf("Процуент правильных тестов: %.2f%% \n", (double)counts\_of\_correct / num\_calculations \* 100.0);

}

**Приложение код:**

#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#include <time.h>

#include "hash.h"

#include <math.h>

#define ull unsigned long long

ull a = 92233720368547765ULL;

ull b = 92233720368547764ULL;

ull c = 3737373737ULL;

ull M = 18446744073709551615ULL;

ull randim(ull x) {

return (x \* a + c);

}

void power() {

int s = 1;

ull b1 = b = 92233720368547764ULL;

for (int i = 0; i < 128; i++) {

b = b \* b1;

s++;

if (b == 0) {

printf("мощность = %d\n\n", s);

return;

}

}

}

void hi\_kvadrat(ull x) {

double v21[6] = { 2.558, 3.94, 4.865, 15.99, 18.31, 23.31 };

int bad = 0;

int good = 0;

int neg = 0;

ull r = 0;

printf("Введите диапозон: ");

scanf("%llu", &r);

ull n = 0;

printf("Количество чисел: ");

scanf("%llu", &n);

ull k = 11;

for (int i = 0; i < 100; i++)

{

ull Y\_i[11] = { 0 };

double np\_i = (double)n / k;

ull t = n;

ull chislo = 0;

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

{

x = randim(x);

chislo = x % r;

for (int i = 0; i < 11; i++)

{

if (i \* (r / 11) <= chislo && chislo < (i + 1) \* (r / 11))

{

Y\_i[i]++;

break;

}

}

}

double hi = 0;

for (int i = 0; i < k; i++)

{

double v0 = (double)Y\_i[i] - np\_i;

hi += ((v0 \* v0) / np\_i);

}

printf("Тест %d: hi-квадрат = %.3f, ", i + 1, hi);

if (hi <= v21[0] || hi >= v21[5]) {

printf("Результат плохой\n");

neg++;

}

else if (hi <= v21[1] || hi >= v21[4]) {

printf("Результат подозрителен\n");

bad++;

}

else if (hi <= v21[2] || hi >= v21[3]) {

printf("Результат почти подозрительный\n");

bad++;

}

else {

printf("Результат хороший\n");

good++;

}

}

printf("Отличных hi = %d, подозрительныx hi = %d , плохих hi = %d, ужасные hi = %d\n\n", good, bad, neg, 100 - good - bad - neg);

}

void birthday\_paradox(ull initial\_seed) {

ull num\_calculations = 0;

ull num\_people = 0;

ull counts\_of\_correct = 0;

int fringe\_diff = 5;

printf("Введите количество расчетов вероятности: ");

scanf("%llu", &num\_calculations);

printf("Введите количество людей в эксперименте (n): ");

scanf("%llu", &num\_people);

if (num\_calculations == 0 || num\_people == 0) {

printf("Количество расчетов и людей должно быть больше 0.\n");

return;

}

// Вычисляем теоретическую вероятность

double theoretical\_prob = 1.0;

for (ull i = 0; i < num\_people; ++i) {

theoretical\_prob \*= (365.0 - i) / 365.0;

}

theoretical\_prob = (1.0 - theoretical\_prob) \* 100.0;

printf("\nТеоретическая вероятность совпадения для %llu людей: %.2f%%\n\n",

num\_people, theoretical\_prob);

for (ull calc = 0; calc < num\_calculations; ++calc) {

ull collisions\_in\_100 = 0;

// Генерируем уникальный seed для каждого расчета

ull current\_calc\_seed = initial\_seed + calc;

ull experiment\_seed = current\_calc\_seed;

printf("Расчет %llu \n", calc + 1);

// 100 экспериментов с текущим seed

for (ull exp = 0; exp < 10000; ++exp) {

HashTable\* ht = create\_hashtable(365);

if (ht == NULL) {

perror("Ошибка создания хеш-таблицы");

return;

}

bool has\_collision = false;

ull current\_x = experiment\_seed;

for (ull person = 0; person < num\_people; ++person) {

current\_x = randim(current\_x);

ull birthday = (current\_x % 365) + 1;

if (insert\_hashtable(ht, birthday)) {

has\_collision = true;

break;

}

}

free\_hashtable(ht);

if (has\_collision) {

collisions\_in\_100++;

}

// Обновляем seed для следующего эксперимента

experiment\_seed = randim(experiment\_seed);

}

double experimental\_prob = (double)collisions\_in\_100 / 10000 \* 100.0;

double deviation = experimental\_prob - theoretical\_prob;

printf(" Совпадений: %llu/10000\n", collisions\_in\_100);

printf(" Вероятность: %.2f%% (отклонение: %+.2f%%)\n",

experimental\_prob, deviation);

printf("----------------------------------------\n");

if(fabs(deviation) < fringe\_diff) counts\_of\_correct++;

}

printf("Колличество тестов не превышающих границу в %d%%: %llu \n", fringe\_diff, counts\_of\_correct);

printf("Процуент правильных тестов: %.2f%% \n", (double)counts\_of\_correct / num\_calculations \* 100.0);

}

void razbrosik(ull x) {

ull r = 0;

printf("Введите правую границу: ");

scanf("%llu", &r);

ull n = 0;

printf("Количество измерений: ");

scanf("%llu", &n);

ull\* mass = (ull\*)calloc(r, sizeof(ull));

int y = n;

while (y--) {

mass[x % r]++;

x = randim(x);

}

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

printf("число %llu встретилось %llu раз\n", i, mass[i]);

}

free(mass);

}

void period(ull x) {

ull x1, x2, x3;

x1 = randim(x);

x2 = randim(x1);

x3 = randim(x2);

x = x3;

ull n1, n2, n3;

ull h = 0;

while (1) {

n1 = randim(x);

x = n1;

n2 = randim(n1);

n3 = randim(n2);

if (h % 1000000 == 0)printf("%llu ", h);

if (n1 == x1 && n2 == x2 && n3 == x3) {

break;

}

h++;

}

}

int main(){

setlocale(LC\_ALL, "Rus");

int g = 0;

ull seed = 0;

ull x = 0;

while (1) {

switch (g)

{

default:

printf("Выберите критерий: 1)Разброс 2)Критерий хи-квадрат 3)Критерий по парадоксу дней рождений 4)Мощность 5)Период \n");

scanf("%d", &g);

break;

case 1:

seed = time(NULL);

x = randim(seed);

razbrosik(x);

g = 0;

break;

case 2:

seed = time(NULL);

x = randim(seed);

hi\_kvadrat(x);

g = 0;

break;

case 3:

seed = time(NULL);

x = randim(seed);

birthday\_paradox(x);

g = 0;

break;

case 4:

seed = time(NULL);

x = randim(seed);

power();

g = 0;

break;

case 5:

seed = time(NULL);

x = randim(seed);

period(x);

g = 0;

break;

}

}

return 0;

}