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 Колмогорова-Смирнова

**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) {

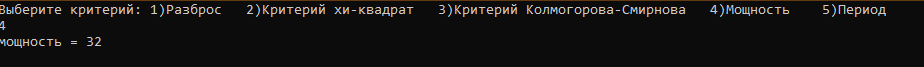
printf("мощность = %llu\n", s);

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);

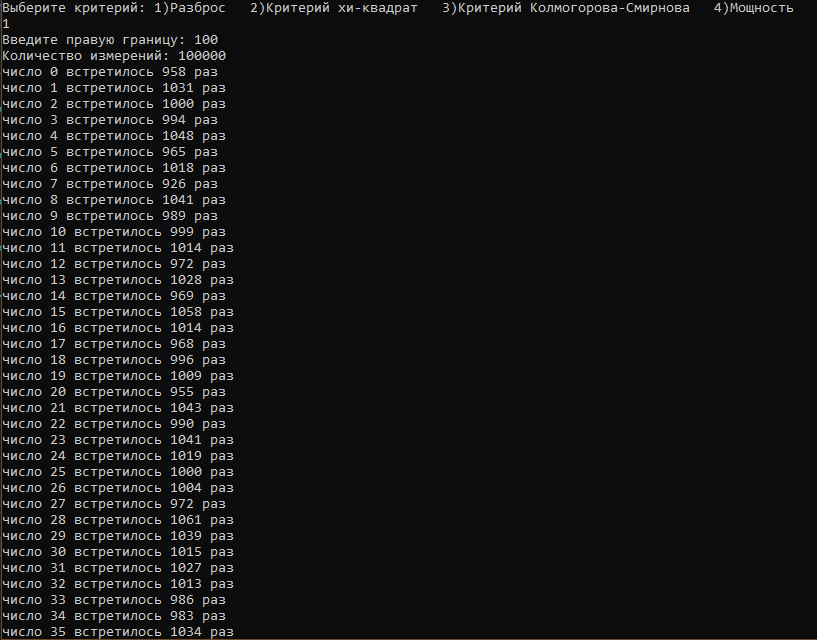
}

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

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

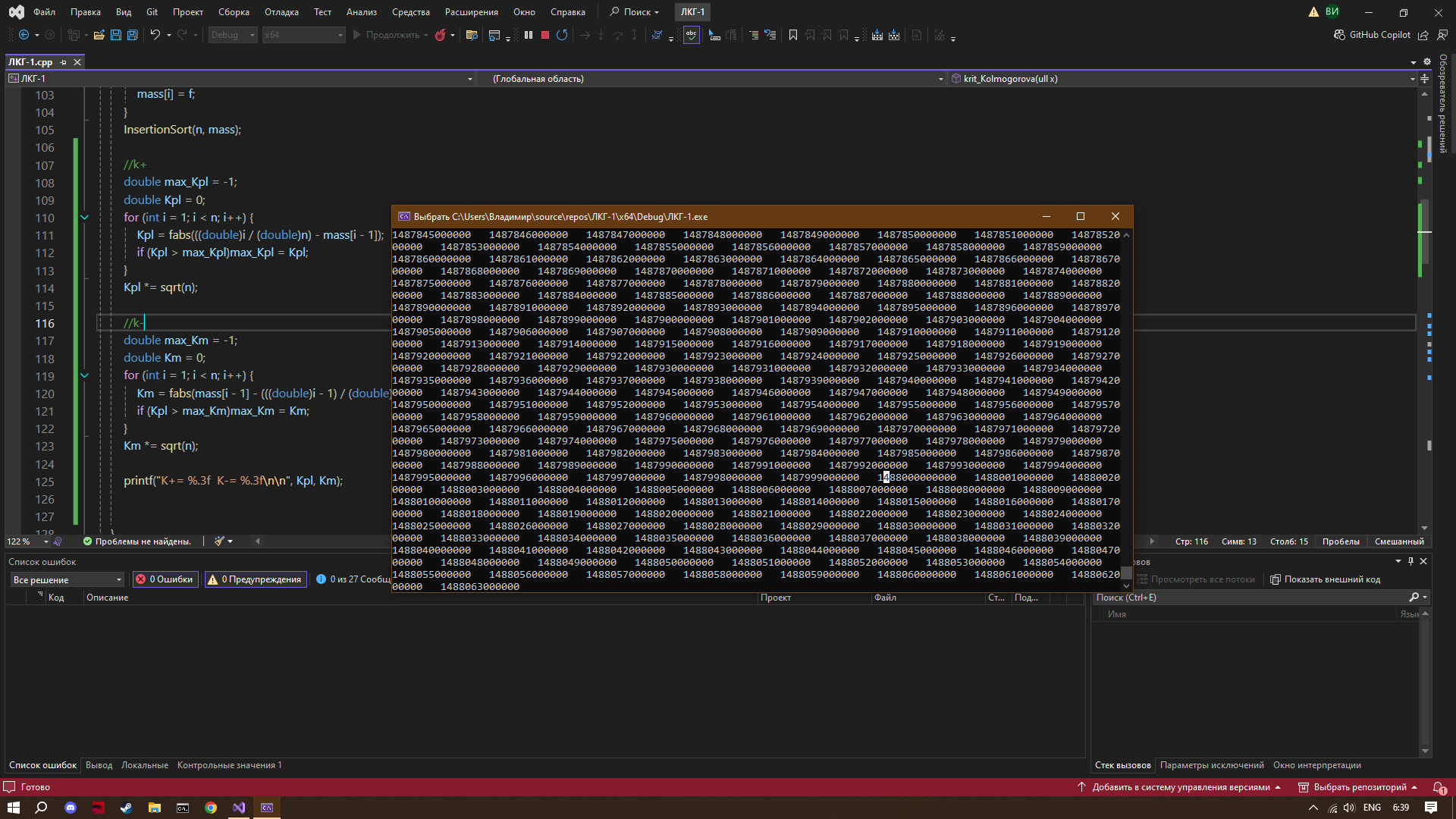
}

}



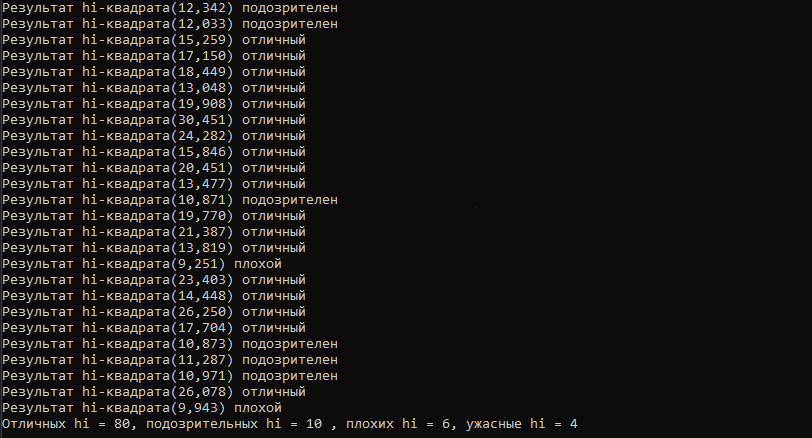
**3) Проверка периода:**

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.



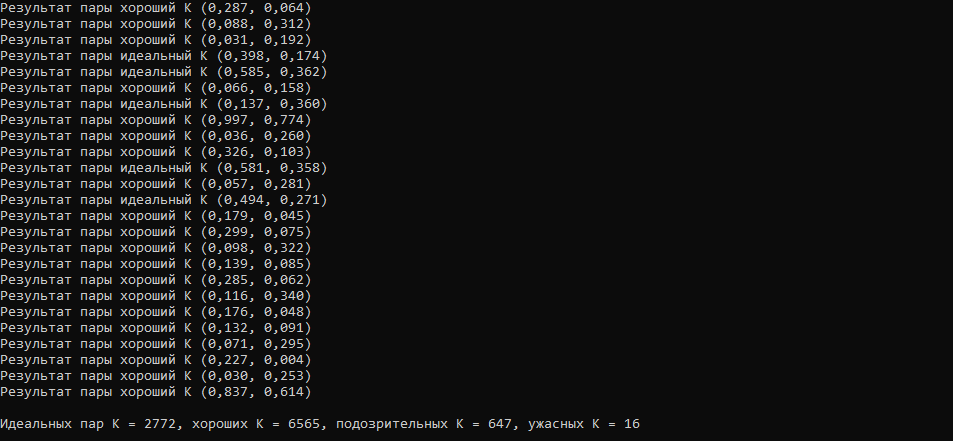
**4)Проверка Хи-квадрата:**

*Приложение 1.*



**5)Проверка критерия Колмогорова-Смирнова(по Кнуту):**

*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 Колмагорова-Смирнова- 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:**

void hi\_kvadrat(ull x) {

double v21[8] = { 8.260, 10.85, 12.45, 19.34, 28.412, 31.41, 37.57 };

int bed = 0;

int good = 0;

int neg = 0;

ull r = 0;

printf("Введите правую границу: ");

scanf("%llu", &r);

ull n = 0;

printf("Количество измерений: ");

scanf("%llu", &n);

ull k = 21;

int p = 100;

while (p--) {

//double np\_i = n / k;

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

//int y = n;

//while (y--) {

// mass[x % r]++;

// x = randim(x);

//}

//ull\* Y\_i = new ull[k]{ 0 };

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

// for (int j = 0; j < r / k; j++) {

// Y\_i[i] += mass[i \* j + j];

//

// }

//}

ull Y\_i[21] = { 0 };

double np\_i = n / k;

ull t = n;

ull chislo = 0;

while (t--) {

x = randim(x);

chislo = x % r;

for (int i = 0; i < 21; i++) {

if (i \* (r / 21) <= chislo && chislo < (i + 1) \* (r / 21)) {

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);

}

if (v21[0] <= hi && hi <= v21[1]) { printf("Результат hi-квадрата(%.3f) плохой\n", hi); neg++; }

else if (v21[1] <= hi && hi <= v21[2]) { printf("Результат hi-квадрата(%.3f) подозрителен\n", hi); bed++; }

else if (v21[2] <= hi && hi <= v21[5]) {printf("Результат hi-квадрата(%.3f) отличный\n", hi); good++;}

else if (v21[5] <= hi && hi <= v21[6]){ printf("Результат hi-квадрата(%.3f) подозрителен\n", hi); bed++;}

else if (v21[6] <= hi && hi <= v21[7]){ printf("Результат hi-квадрата(%.3f) плохой\n", hi); neg++;}

}

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

}

**Приложение2:**

void hi\_kvadrat(ull x) {

double v21[8] = { 8.260, 10.85, 12.45, 19.34, 28.412, 31.41, 37.57 };

int bed = 0;

int good = 0;

int neg = 0;

ull r = 0;

printf("Введите правую границу: ");

scanf("%llu", &r);

ull n = 0;

printf("Количество измерений: ");

scanf("%llu", &n);

ull k = 21;

int p = 100;

while (p--) {

//double np\_i = n / k;

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

//int y = n;

//while (y--) {

// mass[x % r]++;

// x = randim(x);

//}

//ull\* Y\_i = new ull[k]{ 0 };

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

// for (int j = 0; j < r / k; j++) {

// Y\_i[i] += mass[i \* j + j];

//

// }

//}

ull Y\_i[21] = { 0 };

double np\_i = n / k;

ull t = n;

ull chislo = 0;

while (t--) {

x = randim(x);

chislo = x % r;

for (int i = 0; i < 21; i++) {

if (i \* (r / 21) <= chislo && chislo < (i + 1) \* (r / 21)) {

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);

}

if (v21[0] <= hi && hi <= v21[1]) { printf("Результат hi-квадрата(%.3f) плохой\n", hi); neg++; }

else if (v21[1] <= hi && hi <= v21[2]) { printf("Результат hi-квадрата(%.3f) подозрителен\n", hi); bed++; }

else if (v21[2] <= hi && hi <= v21[5]) {printf("Результат hi-квадрата(%.3f) отличный\n", hi); good++;}

else if (v21[5] <= hi && hi <= v21[6]){ printf("Результат hi-квадрата(%.3f) подозрителен\n", hi); bed++;}

else if (v21[6] <= hi && hi <= v21[7]){ printf("Результат hi-квадрата(%.3f) плохой\n", hi); neg++;}

}

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

}

void krit\_Kolmogorova(ull x) {

ull r = 1;

//printf("Введите правую границу(при границе = 1, критерий Смирнова будет использован с модификацией Кнута) : ");

//scanf("%llu", &r);

if (r == 1) {

ull n = 20;

double n20[8] = { 0.03807, 0.1298, 0.3461, 0.5547, 0.7975, 1.1839, 1.2698 };

int u = 10000;

int ideal = 0;

int good = 0;

int bed = 0;

while (u--) {

double\* mass = new double [n] { 0 };

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

double f = x / (double)M;

x = randim(x);

mass[i] = f;

}

InsertionSort(n, mass);

//k+

double max\_Kpl = -1;

double Kpl = 0;

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

Kpl = fabs(((double)i / (double)n) - mass[i - 1]);

if (Kpl > max\_Kpl)max\_Kpl = Kpl;

}

Kpl \*= sqrt(n);

//k-

double max\_Km = -1;

double Km = 0;

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

Km = fabs(mass[i - 1] - (((double)i - 1) / (double)n));

if (Kpl > max\_Km)max\_Km = Km;

}

Km \*= sqrt(n);

if (n20[2] <= Km && Km <= n20[3] || n20[2] <= Kpl && Kpl <= n20[3]) {printf("Результат пары идеальный K (%.3f, %.3f)\n", Kpl, Km); ideal++;}

else if (n20[1] <= Km && Km <= n20[4] || n20[1] <= Kpl && Kpl <= n20[4]) { printf("Результат пары хороший K (%.3f, %.3f)\n", Kpl, Km); good++;}

else if (n20[0] <= Km && Km <= n20[5] || n20[0] <= Kpl && Kpl <= n20[5]) { printf("Результат пары подозрительных K (%.3f, %.3f)\n", Kpl, Km); bed++; }

}

printf("\nИдеальных пар K = %d, хороших K = %d, подозрительных K = %d, ужасных К = %d\n\n", ideal, good, bed, 10000-ideal-good-bed);

}

else {

ull n = 0;

printf("Количество измерений: ");

scanf("%llu", &n);

ull k = 0;

printf("Количество блоков: ");

scanf("%llu", &k);

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

int y = n;

while (y--) {

mass[x % r]++;

x = randim(x);

}

ull\* chastot = new ull[k]{ 0 };

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

for (int j = 0; j < r / k; j++) {

chastot[i] += mass[i \* j + j];

}

//printf("chastota=%d\n", chastot[i]);

}

double\* nakop\_otn\_chast\_emp = new double[k] { 0 };

nakop\_otn\_chast\_emp[0] = (double)chastot[0] / n;

//printf("nakop=%.3f\n", nakop\_otn\_chast\_emp[0]);

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

nakop\_otn\_chast\_emp[i] = nakop\_otn\_chast\_emp[i - 1] + (double)chastot[i] / n;

//printf("nakop=%.3f\n", nakop\_otn\_chast\_emp[i]);

}

double chastot\_teor = n / k;

double\* nakop\_otn\_chast\_teor = new double[k] { 0 };

nakop\_otn\_chast\_teor[0] = 1 / (double)k;

//printf("%.3f\n", nakop\_otn\_chast\_teor[0]);

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

nakop\_otn\_chast\_teor[i] = nakop\_otn\_chast\_teor[i - 1] + 1 / (double)k;

//printf("teor=%.3f\n", nakop\_otn\_chast\_teor[i]);

}

//разность накопительных отн. частот по модулю

double maxim\_emp = 0;

double A\_emp = 0;

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

//N\_эмп-N\_теор

A\_emp = fabs(nakop\_otn\_chast\_emp[i] - nakop\_otn\_chast\_teor[i]);

if (maxim\_emp < A\_emp) maxim\_emp = A\_emp;

}

A\_emp = maxim\_emp;

int g = 0;

printf("1)Ур.доверия = 0.99 2)Ур.доверия = 0.95 3)Ур.доверия = 0.90\n");

scanf("%d", &g);

double A\_cr = 0;

if (g == 1) {

A\_cr = (1.63 / (double)sqrt(n));

}

else if (g == 2) {

A\_cr = (1.36 / (double)sqrt(n));

}

else if (g == 3) {

A\_cr = (1.22 / (double)sqrt(n));

}

printf("\nA\_эмп = %.3f A\_кр=%.3f\n", A\_emp, A\_cr);

if (A\_emp < A\_cr)printf("Отвергать, что данная выборка могла быть получена из равномерного распределения R[0, %llu), нельзя\n", r);

else if (A\_emp > A\_cr)printf("Данная выборка могла быть получена не из равномерного распределения R[0, %llu)\n", r);

}

}

**Приложение код:**

#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 a = 92233720368547765ULL;

ull b = 92233720368547764ULL;

ull c = 3737373737ULL;

//ull seed = 424234373327333ULL;

ull M = 18446744073709551615ULL;//M=m-1

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 InsertionSort(int n, double mass[])

{

double newElement;

int location;

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

{

newElement = mass[i];

location = i - 1;

while (location >= 0 && mass[location] > newElement)

{

mass[location + 1] = mass[location];

location = location - 1;

}

mass[location + 1] = newElement;

}

}

void hi\_kvadrat(ull x) {

double v21[8] = { 8.260, 10.85, 12.45, 19.34, 28.412, 31.41, 37.57 };

int bed = 0;

int good = 0;

int neg = 0;

ull r = 0;

printf("Введите правую границу: ");

scanf("%llu", &r);

ull n = 0;

printf("Количество измерений: ");

scanf("%llu", &n);

ull k = 21;

int p = 100;

while (p--) {

//double np\_i = n / k;

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

//int y = n;

//while (y--) {

// mass[x % r]++;

// x = randim(x);

//}

//ull\* Y\_i = new ull[k]{ 0 };

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

// for (int j = 0; j < r / k; j++) {

// Y\_i[i] += mass[i \* j + j];

//

// }

//}

ull Y\_i[21] = { 0 };

double np\_i = n / k;

ull t = n;

ull chislo = 0;

while (t--) {

x = randim(x);

chislo = x % r;

for (int i = 0; i < 21; i++) {

if (i \* (r / 21) <= chislo && chislo < (i + 1) \* (r / 21)) {

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);

}

if (v21[0] <= hi && hi <= v21[1]) { printf("Результат hi-квадрата(%.3f) плохой\n", hi); neg++; }

else if (v21[1] <= hi && hi <= v21[2]) { printf("Результат hi-квадрата(%.3f) подозрителен\n", hi); bed++; }

else if (v21[2] <= hi && hi <= v21[5]) {printf("Результат hi-квадрата(%.3f) отличный\n", hi); good++;}

else if (v21[5] <= hi && hi <= v21[6]){ printf("Результат hi-квадрата(%.3f) подозрителен\n", hi); bed++;}

else if (v21[6] <= hi && hi <= v21[7]){ printf("Результат hi-квадрата(%.3f) плохой\n", hi); neg++;}

}

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

}

void krit\_Kolmogorova(ull x) {

ull r = 1;

//printf("Введите правую границу(при границе = 1, критерий Смирнова будет использован с модификацией Кнута) : ");

//scanf("%llu", &r);

if (r == 1) {

ull n = 20;

double n20[8] = { 0.03807, 0.1298, 0.3461, 0.5547, 0.7975, 1.1839, 1.2698 };

int u = 10000;

int ideal = 0;

int good = 0;

int bed = 0;

while (u--) {

double\* mass = new double [n] { 0 };

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

double f = x / (double)M;

x = randim(x);

mass[i] = f;

}

InsertionSort(n, mass);

//k+

double max\_Kpl = -1;

double Kpl = 0;

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

Kpl = fabs(((double)i / (double)n) - mass[i - 1]);

if (Kpl > max\_Kpl)max\_Kpl = Kpl;

}

Kpl \*= sqrt(n);

//k-

double max\_Km = -1;

double Km = 0;

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

Km = fabs(mass[i - 1] - (((double)i - 1) / (double)n));

if (Kpl > max\_Km)max\_Km = Km;

}

Km \*= sqrt(n);

if (n20[2] <= Km && Km <= n20[3] || n20[2] <= Kpl && Kpl <= n20[3]) {printf("Результат пары идеальный K (%.3f, %.3f)\n", Kpl, Km); ideal++;}

else if (n20[1] <= Km && Km <= n20[4] || n20[1] <= Kpl && Kpl <= n20[4]) { printf("Результат пары хороший K (%.3f, %.3f)\n", Kpl, Km); good++;}

else if (n20[0] <= Km && Km <= n20[5] || n20[0] <= Kpl && Kpl <= n20[5]) { printf("Результат пары подозрительных K (%.3f, %.3f)\n", Kpl, Km); bed++; }

}

printf("\nИдеальных пар K = %d, хороших K = %d, подозрительных K = %d, ужасных К = %d\n\n", ideal, good, bed, 10000-ideal-good-bed);

}

else {

ull n = 0;

printf("Количество измерений: ");

scanf("%llu", &n);

ull k = 0;

printf("Количество блоков: ");

scanf("%llu", &k);

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

int y = n;

while (y--) {

mass[x % r]++;

x = randim(x);

}

ull\* chastot = new ull[k]{ 0 };

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

for (int j = 0; j < r / k; j++) {

chastot[i] += mass[i \* j + j];

}

//printf("chastota=%d\n", chastot[i]);

}

double\* nakop\_otn\_chast\_emp = new double[k] { 0 };

nakop\_otn\_chast\_emp[0] = (double)chastot[0] / n;

//printf("nakop=%.3f\n", nakop\_otn\_chast\_emp[0]);

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

nakop\_otn\_chast\_emp[i] = nakop\_otn\_chast\_emp[i - 1] + (double)chastot[i] / n;

//printf("nakop=%.3f\n", nakop\_otn\_chast\_emp[i]);

}

double chastot\_teor = n / k;

double\* nakop\_otn\_chast\_teor = new double[k] { 0 };

nakop\_otn\_chast\_teor[0] = 1 / (double)k;

//printf("%.3f\n", nakop\_otn\_chast\_teor[0]);

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

nakop\_otn\_chast\_teor[i] = nakop\_otn\_chast\_teor[i - 1] + 1 / (double)k;

//printf("teor=%.3f\n", nakop\_otn\_chast\_teor[i]);

}

//разность накопительных отн. частот по модулю

double maxim\_emp = 0;

double A\_emp = 0;

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

//N\_эмп-N\_теор

A\_emp = fabs(nakop\_otn\_chast\_emp[i] - nakop\_otn\_chast\_teor[i]);

if (maxim\_emp < A\_emp) maxim\_emp = A\_emp;

}

A\_emp = maxim\_emp;

int g = 0;

printf("1)Ур.доверия = 0.99 2)Ур.доверия = 0.95 3)Ур.доверия = 0.90\n");

scanf("%d", &g);

double A\_cr = 0;

if (g == 1) {

A\_cr = (1.63 / (double)sqrt(n));

}

else if (g == 2) {

A\_cr = (1.36 / (double)sqrt(n));

}

else if (g == 3) {

A\_cr = (1.22 / (double)sqrt(n));

}

printf("\nA\_эмп = %.3f A\_кр=%.3f\n", A\_emp, A\_cr);

if (A\_emp < A\_cr)printf("Отвергать, что данная выборка могла быть получена из равномерного распределения R[0, %llu), нельзя\n", r);

else if (A\_emp > A\_cr)printf("Данная выборка могла быть получена не из равномерного распределения R[0, %llu)\n", r);

}

}

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]);

}

}

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 (true) {

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 (true) {

switch (g)

{

default:

printf("Выберите критерий: 1)Разброс 2)Критерий хи-квадрат 3)Критерий Колмогорова-Смирнова 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);

krit\_Kolmogorova(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;

}