ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»

***Отчёт***

По лабораторной работе №1

“Моделирование генераторов случайных чисел с равномерным

законом распределения вероятности”

Выполнил студент:

Группы А-04-23

Осипов А.А.

Москва, 2025

**Лабораторная работа 1**

**Цель работы**

Построить генератор случайных чисел с равномерным законом распределения на интервале [0,1) по методу середин квадратов или мультипликативного датчика в соответствии с заданием. Написать и отладить программу, реализующую генератор на языке С++. Получить выборку неповторяющихся псевдослучайных чисел объемом не меньше 20 тыс. Определить период генератора случайных чисел. Если он меньше 10000, то изменить исходные данные. Провести анализ качества последовательности случайных чисел по критерию Пирсона.

**Описание метода**

Вариант №7: Метод для генератора случайных чисел - мультипликативный датчик.

В этом методе очередное случайное число xi+1 получается как остаток от деления двух больших целых чисел. Это случайное число определяется следующей формулой xi+1 = (axi+b) mod M, где a, b, M - заданные целые числа. xi – предыдущее целое случайное число.

Для получения "хорошего" со статистической точки зрения распределения рекомендуется выбирать параметры M, a и b исходя из следующих соображений:

1. M — большое целое число, желательно простое;
2. a — целое число порядка корня квадратного из M, желательно простое, M не должно делиться на a;
3. a и M выбираются такими, чтобы их произведение не выходило за границы разрядной сетки для целых переменных;
4. b — целое число того же порядка, что и a.

В ходе вычислительных экспериментов замечено, что хорошие в статистическом смысле реализации мультипликативного датчика получаются, если целое число a выбрано так, чтобы при делении на 8 в остатке оставалось бы 3 или 5. Слагаемое b используется, чтобы устранить потенциальную опасность получения непрерывной последовательности нулей, начиная с определенного шага. Если M и a — простые числа, то такой опасности не существует.

**Анализ параметров x, a, b, m**

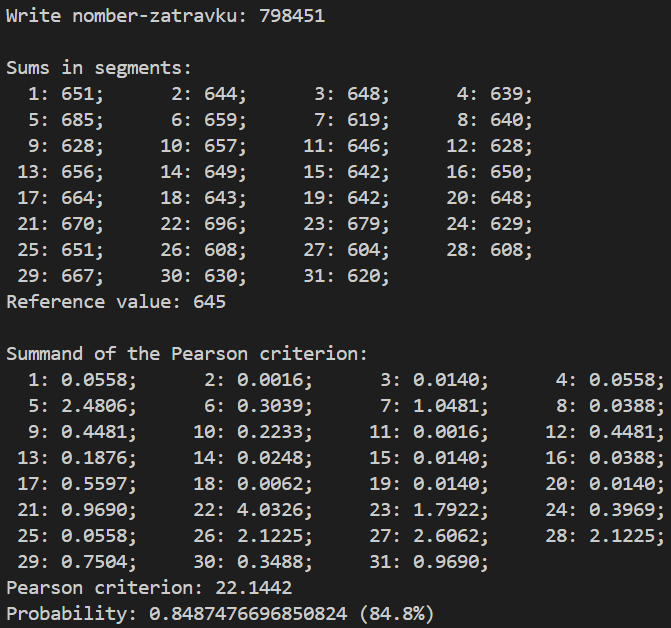
По условию необходим период последовательности случайных чисел не менее 10000. Значит диапазон числа x должен быть не менее [0, 10000], иначе нельзя будет заполнить промежуток в 10000 неповторяющихся значений. Но в таком случае, при достижении первых 10000 элементов, последовательность начнет повторяться, ведь при достижении 10000 элемента поставить на место 10001 элемента можно будет лишь тот, который стоит на 1 месте из-за условия уникальности. Следует увеличить диапазон минимум на 1-2 разряда.

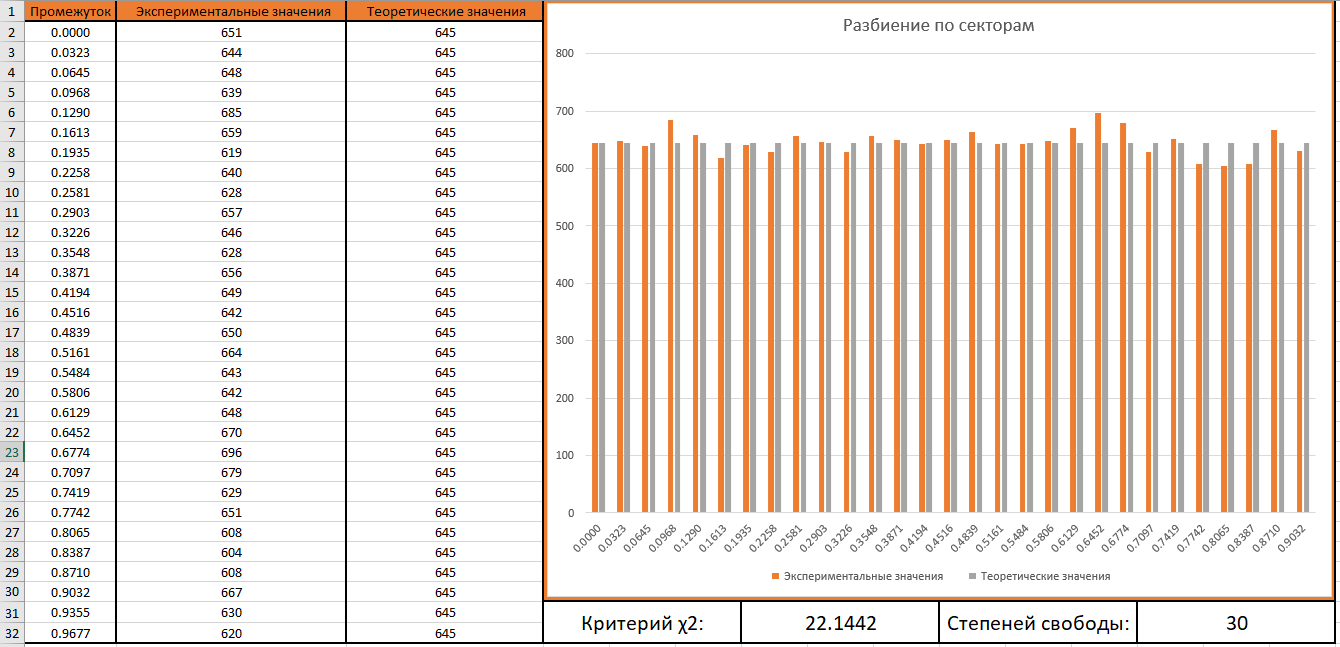
Максимальный целый тип переменных в с++ - long long (+-9,223,372,036,854,775,807) – 13 неполных разрядов, будем считать, что это 12 целых. При работе с методом есть операция (axi+b), произведение будет увеличивать разрядность ответа, при подборе констант следует это учесть.

Параметр M контролирует размер xi+1 после того, как оно возросло за счет умножения. По условию M – простое число, но если взять его как простое, то не будет равномерного распределения. Допустим, M = 500, тогда при любом x > m будут остатки . В 1м промежутке все хорошо: числа будут начинаться с каждой цифры равновероятно, но во 2м промежутке числа могут начинаться только с 1, 2, 3 и 4. Это неправильно с точки зрения равномерного распределения: вероятность получения числа, начинающегося с 1-4 будет выше чем у 5-9. Получается M должно быть вида 10k, k > 5 (из-за T > 10000). Пусть k = 7.

Если M = 107, то a и b должны быть 3 или 4 порядка. Пусть а = 2503, b = 2509.

**Анализ работы программы**

****



Если вероятность правильности статистической гипотезы мала (обычно меньше 70%), то результат нельзя считать хорошим. Аналогично обстоит дело и с очень высокой вероятностью правильности статистической гипотезы (обычно больше 95%). В этом случае, возможно, что последовательность чисел является не случайной, а является специально подобранным набором чисел.

В результате программы видно, что вероятность выдвинутой статистической гипотезы равна 0.84874 (84,8%). Это можно считать неплохим результатом, т.к. . Теоретическая модель хорошо согласуется с экспериментальными данными. Наблюдаемые отклонения являются случайными.

**Код программы**

**main.cpp:**

#include <iostream>

#include <cmath>

#include <stdio.h>

#include <cstdio>

using namespace std;

#include "Headers/Rundom.h"

using namespace Pseudorandom\_numbers;

int main() {

    const int n = 20000;        // Кол-во случайных чисел

    const int m = 31;           // Кол-во разбиений (20-50)

    vector <int> split (m, 0);  // Сумма чисел по секторам

    //! Начальное условие

    Rundom::set\_a(2503);

    Rundom::set\_b(2509);

    Rundom::set\_m(10000000);

    //> Генерация ряда случайных чисел

    FILE\* file\_nombers = fopen("Output/numbers.scv", "w");

    if (!file\_nombers) {

        cout << "File opening error!" << endl;

        return 0;

    }

    for (int i = 0; i < n; i++) {

        long long rund = Rundom::creat\_rundom\_number(); //? start: 798451

        double u = double(rund / pow(10, int(log10(rund) + 1)));

        split[int((u - 0.1) / 0.9 \* m)]++;

        fprintf(file\_nombers, "%8d    ", rund);

        if ((i + 1) % 7 == 0) fprintf(file\_nombers, "\n");

    }

    //> Распределение чисел

    int const w = n/m;

    FILE\* file\_sectors = fopen("Output/sectors.scv", "w");

    if (!file\_sectors) {

        cout << "File opening error!" << endl;

        return 0;

    }

    cout << endl << "Sums in segments: " << endl;

    fprintf(file\_sectors, "Промежуток; Экспериментальные значения; Теоретические значения\n");

    for (int i = 0; i < m; i++) {

        printf("%3d: %d;    ", i+1, split[i]);

        fprintf(file\_sectors, "%1.4f; %d; %d\n", double(i)/m, split[i], w);

        if ((i+1) % 4 == 0) cout << endl;

    }

    //> Оценка качества последовательности

    double pearson = 0;

    cout << endl <<"Reference value: " << w << endl << endl << "Summand of the Pearson criterion:" << endl;

    for (int i = 0; i < m; i++) {

        double mini\_pearson = pow(split[i] - w, 2) / w;

        pearson += mini\_pearson;

        printf("%3d: %2.4f;    ", i+1, mini\_pearson);

        if ((i+1) % 4 == 0) cout << endl;

    }

    cout << endl << "Pearson criterion: " << pearson << endl << "Probability: ";

    system(("py Scripts/get\_probability.py " + to\_string(pearson) + " " + to\_string(m-1)).c\_str());

    fclose(file\_sectors);

    fclose(file\_nombers);

    return 0;

}

**Rundom.h:**

#include <vector>

namespace Pseudorandom\_numbers {

    class Rundom {

    public:

        static void set\_a(int a);

        static void set\_b(int b);

        static void set\_m(int m);

        static long long creat\_rundom\_number();

        static long long set\_start\_number();

        static void print\_all\_numbers();

        static int print\_count\_of\_numbers();

    private:

        static int a, b;

        static long long m;

        static std::vector<long long> random\_row;

        static const int limit\_T;

        static long long multiplicative\_sensor(long long x);

    };

}

**Rundom.cpp:**

#include <algorithm>

#include <iostream>

using namespace std;

#include "../Headers/Rundom.h"

using namespace Pseudorandom\_numbers;

int Rundom::a = 0;

int Rundom::b = 0;

long long Rundom::m = 0;

const int Rundom::limit\_T = 10000;

vector<long long> Rundom::random\_row {};

long long Rundom::multiplicative\_sensor(long long x) {

    return (a \* x + b) % m;

}

void Rundom::set\_a(int a\_outer) {

    Rundom::a = a\_outer;

    return;

}

void Rundom::set\_b(int b\_outer) {

    Rundom::b = b\_outer;

    return;

}

void Rundom::set\_m(int m\_outer) {

    Rundom::m = m\_outer;

    return;

}

long long Rundom::set\_start\_number() {

    cout << "Write nomber-zatravku: ";

    long long x = 0;

    cin >> x;

    random\_row.push\_back(x);

    return x;

}

long long Rundom::creat\_rundom\_number() {

    if (random\_row.size() == 0) {

        return set\_start\_number();

    }

    long long new\_rand\_nom = multiplicative\_sensor(random\_row.back());

    if (find(random\_row.end() - min(limit\_T, int(random\_row.size())), random\_row.end(), new\_rand\_nom) == random\_row.end()) {

        random\_row.push\_back(new\_rand\_nom);

    } else {

        //print\_all\_numbers();

        return set\_start\_number();

    }

    return new\_rand\_nom;

}

void Rundom::print\_all\_numbers(){

    for (int i = 0; i < random\_row.size(); i++){

        cout << random\_row[i] << ' ';

    }

    cout << endl;

    return;

}

int Rundom::print\_count\_of\_numbers(){

    return random\_row.size();

}

**get\_probability.py:**

from scipy.stats import chi2

import sys

pearson, freedom\_numbers = map(float, sys.argv[1:3])

probability = chi2.sf(pearson, freedom\_numbers)

print(probability, "(" + str(int(probability\*1000)/10) + "%)")

**Вывод**

В данной лабораторной работе был создан генератор псевдослучайных мультипликативного датчика с законом равномерным законом распределения на полуинтервале от [0,1). Была создана последовательность из 20000 псевдослучайных чисел с учетом периода больше 10000. В последних пунктах задания была произведена проверка качества данной последовательности по критерию Пирсона. Вероятность того, что данная последовательность распределена по нормальному закону равна 84,8% - исходя из условий, это хороший результат.