Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

по лабораторной работе № 3

по курсу: «Моделирование»

Тема Генерация случайных чисел

Студент Якуба Д. В.

Группа ИУ7-73Б

Оценка (баллы) _____

Преподаватель Рудаков И.В.

Москва, 2021

1. Задание

Написать программу, которая генерирует псевдослучайную последовательность одноразрядных, двухразрядных и трёхразрядных целых чисел с использование табличного и алгоритмического способа.

Для каждой сгенерированной последовательности чисел вычислить количественный критерий оценки случайности.

Предусмотреть ввод десяти чисел для проверки работы программы.

2. Теория

Среди способов получения последовательности случайных чисел различают:

1) Аппаратный способ;
2) Табличный (файловый) способ;
3) Алгоритмический способ.

2.1 Аппаратный способ

Аппаратный генератор случайных чисел – это устройство, которое генерирует последовательность случайных чисел на основе измеряемых, хаотически изменяющихся параметров протекающего физического процесса.

В качестве устройства генерации может быть использовано любое внешнее устройство, причём реализация не требует дополнительных вычислительных операций по выработке чисел.

Среди достоинств аппаратного способа получения можно отметить: неограниченный запас чисел, отсутствие дополнительных вычислительных операций.

К недостаткам данного способа отнесены: периодические проверки, невозможность воспроизвести последовательность, использование специального устройства, необходимость принятия мер по обеспечению стабильности.

2.2 Табличный способ

Если случайные числа, оформленные в виде таблицы, помещать во внешнюю или оперативную память ЭВМ, предварительно сформировав из них соответствующий файл, то такой способ будет называться табличным. Однако этот способ получения случайных чисел при моделировании систем на ЭВМ обычно рационально использовать при сравнительно небольшом объёме таблицы и, соответственно, файла чисел, когда для хранения можно применять оперативную память. Хранение файла во внешней памяти при частном обращении в процессе статистического моделирования не рационально, так как вызывает увеличение затрат машинного времени при моделировании системы из-за необходимости обращения к внешнему накопителю.

Среди достоинств табличного способа получения можно отметить: возможность воспроизвести последовательность, отсутствие используемых внешних устройств.

К недостаткам данного способа отнесены: ограниченность запаса чисел последовательности периодом, существенные затраты машинного времени.

2.3 Алгоритмический способ

Алгоритмический способ – это способ получения последовательности случайных чисел, основанный на формировании случайных чисел в ЭВМ с использованием специальных алгоритмов и реализующих их программ.

Среди достоинств данного способа получения можно отметить: многократная воспроизводимость последовательности чисел, отсутствие внешних устройств и малое количество задействованной памяти.

К недостаткам данного способа отнесены: ограничение запаса чисел периодом последовательности, затраты машинного времени на вычисления.

2.4 Алгоритмический способ генерации случайных чисел, реализуемый в лабораторной работе

В качестве используемого метода генерации последовательности случайных чисел был выбран линейный конгруэнтный метод.

Суть данного метода заключается в вычислении последовательности случайных чисел, полагая:

$$X_{n+1} = (aX_n + c)\ mod\ m,$$

где X_{n+1} – это следующее число в последовательности, a – множитель, причём $0 \leq a < m$, c – приращение, причём $0 \leq c \leq m$, m – натуральное число, относительно которого вычисляется остаток от деления, причём $m \geq 2$.

При выборе значения m необходимо учитывать следующие условия:

1) Данное число должно быть довольно большим, так как период не может иметь более m элементов;
2) Данное значение должно быть таким, чтобы случайные значения вычислялись быстро.

Для улучшения статистических свойств числовой последовательности во многих генераторах используется только часть битов результата.

В качестве констант в данной работе используются следующие значения:

$$a = 1103515245$$

$$\mathrm{c} = 12345$$

$$m = 2^{32}$$

2.5 Табличный способ генерации случайных чисел, используемый в лабораторной работе

В данной работе используется таблица случайных чисел «A million random digits with 100,000 normal deviates».

2.6 Статистический критерий оценки случайности, используемый в лабораторной работе

К наиболее известным программным пакетам статистических тестов относятся: тесты NIST, TEST-U01, CRYPT-X, The pLab Project, DIEGARD, Dieharder, ENT.

Статистические тесты NIST – это пакет тестов, в состав которого входят 15 статистических тестов, целью которых является определение меры случайности двоичных последовательностей, порождённых либо аппаратными, либо программными генераторами случайных чисел. Эти тесты основаны на различных статистических свойствах, присущих только случайным последовательностям.

В качестве используемого теста был реализован частотный блочный тест. Суть данного теста заключается в определении доли единиц внутри блока некоторой длины. Цель – выяснить, действительно ли частота повторения единиц в блоке длины m бит приблизительно равна $\frac{m}{2}$, как можно было бы предположить в случае абсолютно случайной последовательности. Вычисленное в ходе теста значение вероятности p должно быть не меньше 0,01.

В данном тесте используется значение статистики $\chi^2(obs)$ в качестве меры того, насколько наблюдаемая доля единиц в блоке соответствует ожидаемой пропорции $\frac{1}{2}$.

Так называемое «p-value», которое является количественным критерием оценки случайности, вычисляется как значение неполной гамма-функции от значений $\frac{N}{2}$ (N – количество рассматриваемых блоков) и $\frac{\chi^2(obs)}{2}$, где $\chi^2(obs) = 4M\sum_{i=1}^{N}\left(\pi_i - \frac{1}{2}\right)^2$. Причём для каждого i-го блока $\pi_i = \frac{\sum_{j=1}^{M}\varepsilon_{(i-1)M+j}}{M}$, где ε – последовательность битов блока, M – длина рассматриваемого блока.

3. Выполнение

На рисунках 3.1-3.2 предоставлен интерфейс разработанного приложения.

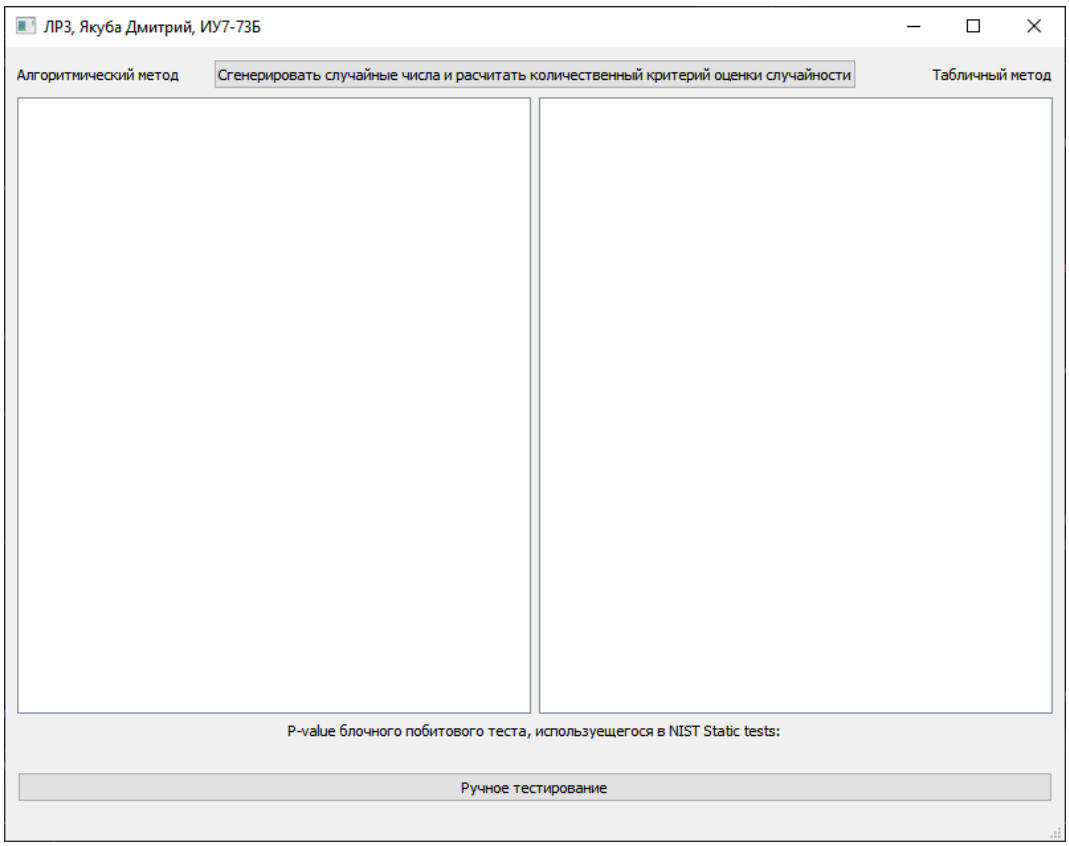

Рис. 3.1, Интерфейс разработанного приложения

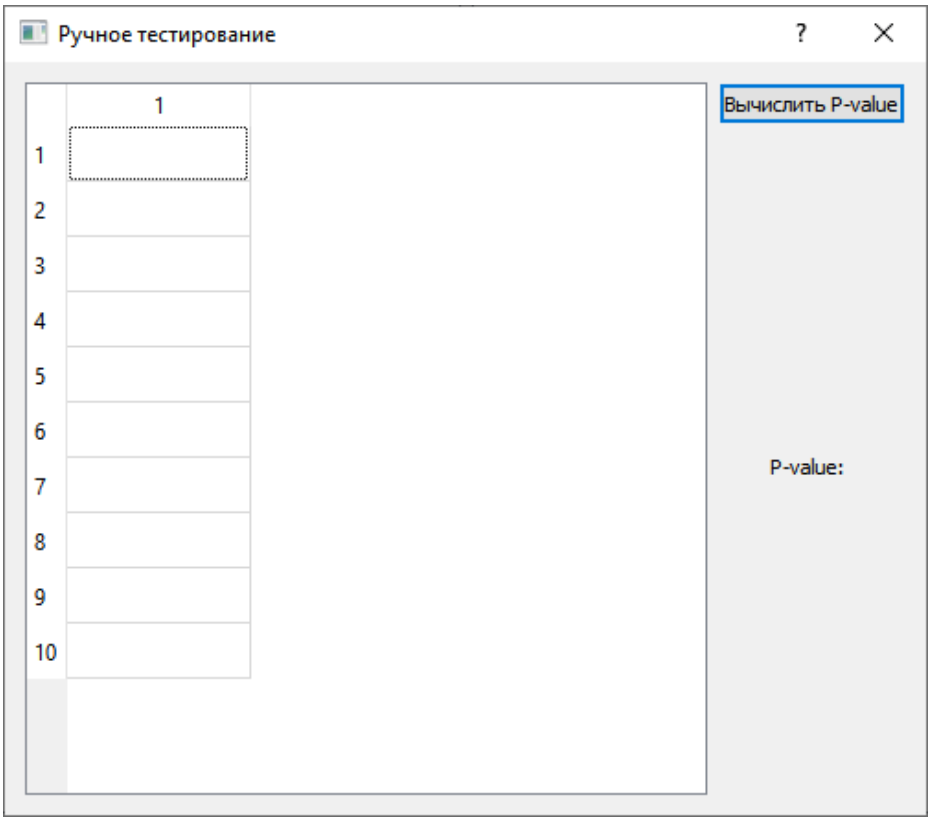

Рис. 3.2, Интерфейс разработанного приложения

3.1 Примеры работы

На рисунках 3.3-3.5 предоставлены примеры работы приложения.

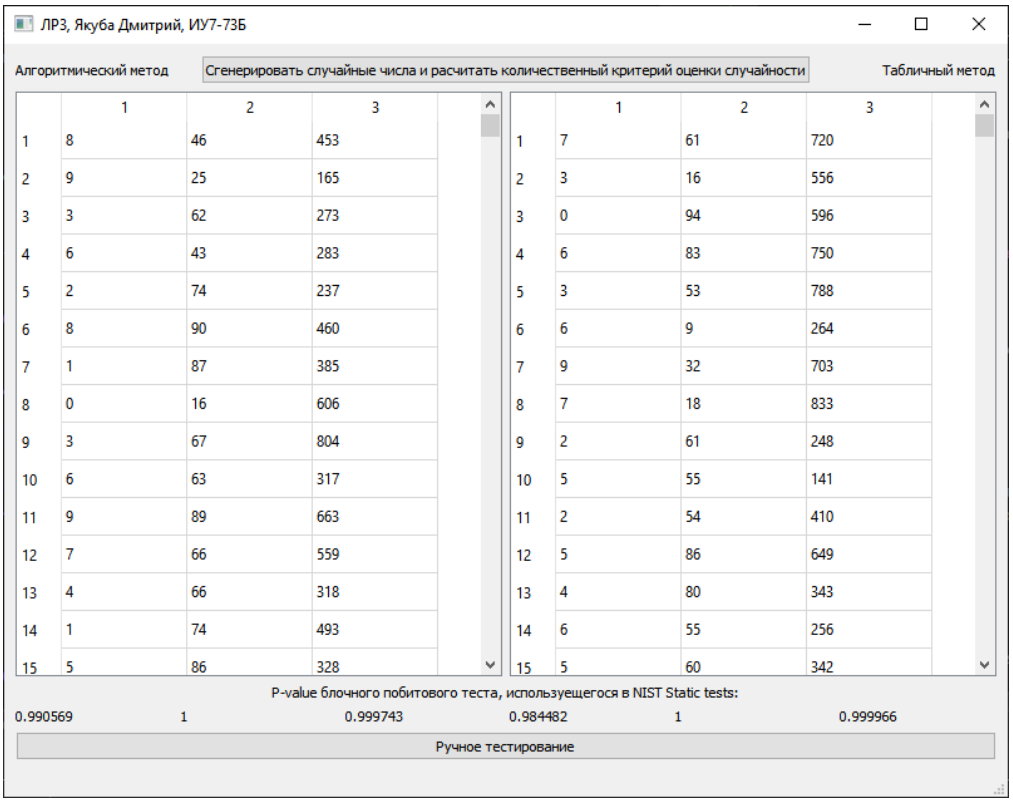

Рис. 3.3, Пример расчёта количественного критерия оценки случайности для двух способов генерации последовательности псевдослучайных чисел

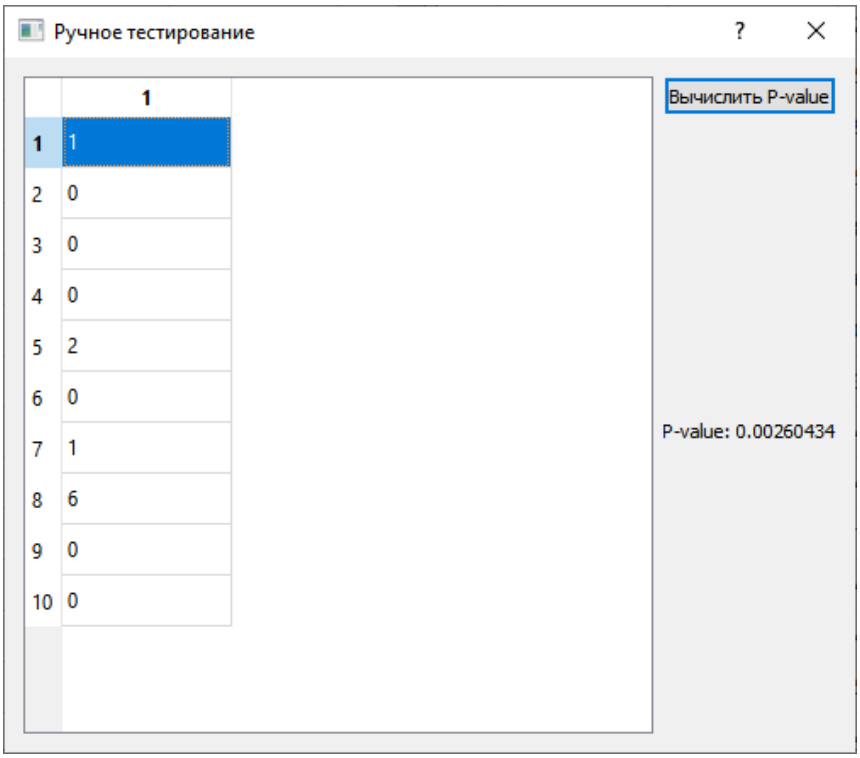

Рис. 3. 4, Пример расчёта количественного критерия оценки случайности для ручного ввода последовательности

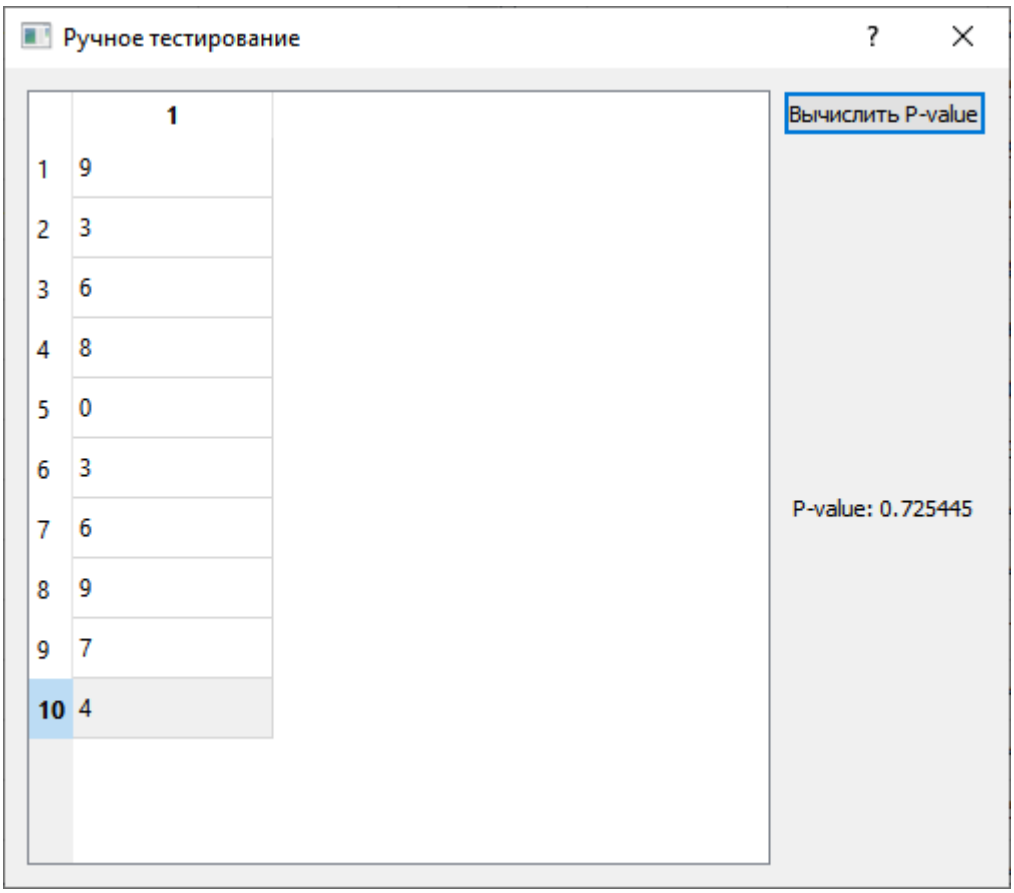

Рис. 3.5, Пример расчёта количественного критерия оценки случайности для ручного ввода последовательности

4. Выводы

Как видно из рисунка 3.3, в случае генерации одноразрядных чисел лучше всего себя показал линейный конгруэнтный метод. При этом, в случае генерации последовательности двухразрядных случайных чисел, оба метода отрабатывают одинаково хорошо. При генерации трёхразрядных чисел лучше себя показал табличный метод. Все предоставленные заключения опираются только на фактический набор данных, предоставленных на рисунке.

Для всех разрядностей каждый метод прошёл тест (критерий ≥ 0.01).

5. Листинг

В данном разделе предоставлены используемые методы для решения поставленной задачи (используемый ЯП – C++).

```
double FrequencyTest::getPValueOfSequence(QVector<long> sequence)
{
    QVector<std::bitset<LONG_SIZE_IN_BITS>> vectorOfLongsBits =
        prepareSequenceBits(sequence);

    int lengthOfEachBlock = findNumberOfSignificantDigits(sequence[0]);

    int numberOfAnalyzedBits = lengthOfEachBlock * vectorOfLongsBits.size();
```

```
    int numberOfBlocks = numberOfAnalyzedBits / lengthOfEachBlock;

    double sumForStatisic = 0;

    double blockSum, elementOfChiSquaredStat;
    for (int i = 0; i < numberOfBlocks; i++)
    {
        blockSum = 0;
        for (int j = lengthOfEachBlock - 1; j >= 0;
             blockSum += vectorOfLongsBits[i][j], j--)
        {}

        elementOfChiSquaredStat = blockSum / lengthOfEachBlock - 0.5;
        sumForStatisic += elementOfChiSquaredStat * elementOfChiSquaredStat;
    }

    double chiSquared = 4 * lengthOfEachBlock * sumForStatisic;

    return incompleteGammaFunction(numberOfBlocks / 2, chiSquared / 2);
}

QVector<long> LinearCongruentRandomizer::createRandomSequence(
    int numberOfRequiredDigits, int numberOfElements)
{
    if (numberOfElements < 1 || numberOfRequiredDigits < 1)
    {
        return QVector<long>();
    }

    QVector<long> sequence = QVector<long>();

    long requiredDigitsDivider = pow(10, numberOfRequiredDigits);
    long minAppendValue = requiredDigitsDivider / 10 - 1;
    long numberToAppend;
    for (int i = 0; i < numberOfElements; i++)
    {
        curElement = curElement * 1103515245 + 12345;
        numberToAppend = ((unsigned int)(curElement / 65536) % (RAND_MAX +
1)) % requiredDigitsDivider;

        if (numberToAppend >= minAppendValue)
        {
            sequence.append(numberToAppend);
        }
        else
        {
            i--;
        }
    }

    return sequence;
}
```