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»

на тему: «Генерация псевдослучайных чисел»

Студент ИУ7-71Б _____ Мицевич М. Д.
(Группа) (Подпись, дата) (И. О. Фамилия)

Преподаватель _____ Рудаков И. В.
(Подпись, дата) (И. О. Фамилия)

2022 г.

1 Теоретический раздел

Существует три основных способа получения последовательностей случайных чисел:

1. аппаратный;
2. табличный (файловый);
3. алгоритмический.

В рамках лабораторной работы рассмотрены алгоритмический и табличный способы.

1.1 Алгоритмический способ

Этот подход основан на базе специальных алгоритмов. К ним относятся:

- метод серединных произведений;
- метод перемешивания;
- линейный конгруэнтный метод.

Было принято решение взять последний для генерации последовательности псевдослучайных чисел.

В этом методе каждое следующее число рассчитывается на основе предыдущего по формуле (1.1).

$$R_{n+1} = (a \cdot R_n + b) \; mod \; N, \; n \geq 1 \tag{1.1}$$

где a, b – коэффициенты, N – модуль.

Для качественного генератора требуется подобрать подходящие коэффициенты. Например, в таблице 1.1 приведены некоторые из них.

1.2 Табличный способ

В качестве источника случайных чисел используют специально заранее составленные таблицы, содержащие проверенные данные.

Таблица 1.1 – Примеры коэффициентов

a	**b**	**N**
106	1283	6075
430	2531	11979
84589	15989	217728
1103515245	12345	2^{31}
...	...	...

1.3 Критерий оценки

В рамках лабораторной работы был выбран собственный критерий, который вычисляется следующим образом:

1. массив с размером равным длине интервала выборки случайного значения заполняется значениями, равными числу выпадений каждого числа интервала;

2. у полученной последовательности считаются выборочное среднее и выборочная дисперсия;

3. считается дисперсия последовательности, полученной в случае, когда все значения, полученные с помощью алгоритма генерации псевдослучайных чисел, были одинаковые;

4. результат получается по формуле $res = 1 - \frac{freq\_disp}{max\_disp}$, где числитель дроби получен на шаге 2, а знаменатель – на шаге 3.

Чем ближе полученный результат к 1, тем более случайной является входная последовательность.

2 Практическая часть

На листинге 2.1 представлен код для моделирования работы системы.

Листинг 2.1 – Критерий для оценки случайности

```
#include "mycriteirium.h"
#include <vector>

MyCriteirium::MyCriteirium() {}

double MyCriteirium::findPValue(const QVector<long>& results,
                                long min,
                                long max) {
  // min max args
  // step = 1
  // freq
  std::vector<long> freqs(max - min + 1);
  for (auto res : results)
    freqs[res - min] += 1;

  // mid freq
  double freq_mean = 0;
  for (auto freq : freqs) {
    freq_mean += (double)freq / freqs.size();
  }
  // disp max disp
  double freq_disp = 0;
  for (auto freq : freqs) {
    freq_disp +=
        (double)((freq - freq_mean) * (freq - freq_mean)) / freqs
           .size();
  }
  // max disp = (n / size - size) ^ 2 + (size - 1) * (n / size) ^
     2
  double max_mean = (double)results.size() / freqs.size();
  double max_disp = ((max_mean - freqs.size()) * (max_mean -
     freqs.size()) +
                     (freqs.size() - 1) * max_mean * max_mean) /
                    freqs.size();

  // 1 - disp / max disp
  return 1 - freq_disp / max_disp;
```

```
}
```

На листинге 2.2 представлен код алгоритмического способа.

Листинг 2.2 – Алгоритмический способ

```
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;
  if (numberOfRequiredDigits == 1)
    minAppendValue = 0;
  long numberToAppend;
  for (int i = 0; i < numberOfElements; i++) {
    curElement = curElement * 84589 + 15989;
    numberToAppend =
        ((unsigned int)curElement % 217728) %
           requiredDigitsDivider;

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

  return sequence;
}
```

На листинге 2.3 представлен код табличного способа.

Листинг 2.3 – Табличный способ

```
QVector<long> TableRandomizer::getRandomSequence(int
   numberOfRequiredDigits,
                                                 int
                                                    numberOfElements
                                                    ) {
```

```
  if (numberOfRequiredDigits > 5 || numberOfElements < 1) {
    return QVector<long>();
  }

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

  long requiredDigitsDivider = pow(10, numberOfRequiredDigits);
  long minAppendValue = requiredDigitsDivider / 10;
  if (requiredDigitsDivider == 1)
    minAppendValue = 0;
  long numberToAppend;
  int addedElements = 0;

  for (; fileToRead >> numberToAppend, addedElements <
     numberOfElements;
       addedElements++) {
    numberToAppend = numberToAppend % requiredDigitsDivider;
    if (numberToAppend >= minAppendValue) {
      sequence.append(numberToAppend);
    } else {
      addedElements--;
    }
  }

  return sequence;
}
```

Результат работы алгоритмического и табличного способов и критерий для их проверки представлены на рисунке 2.1.

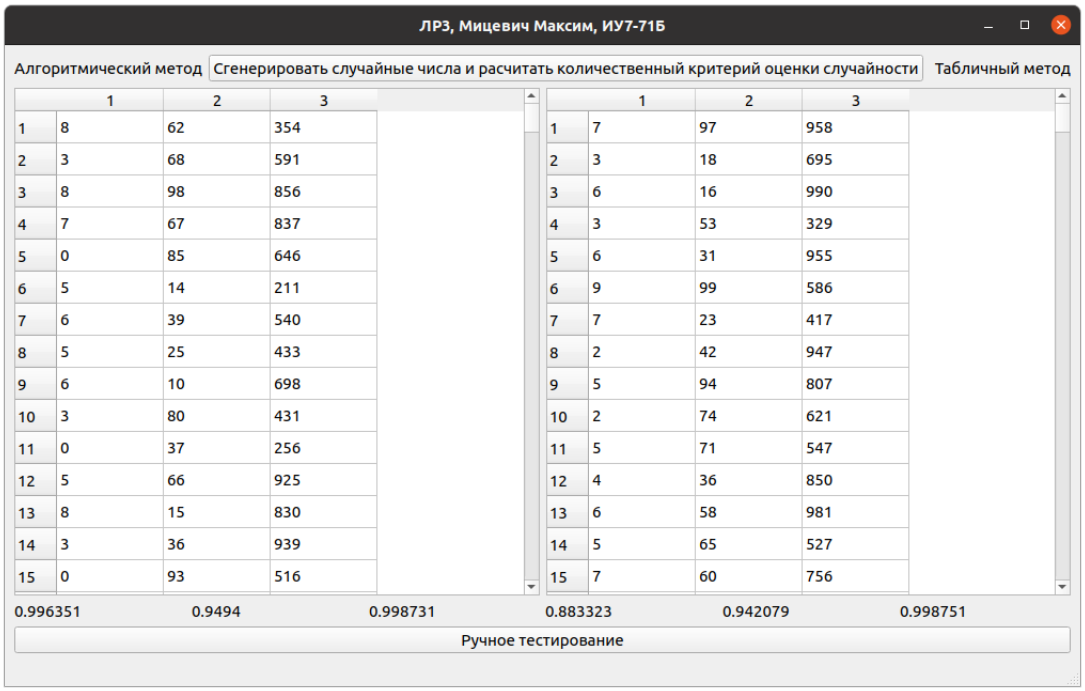

Рисунок 2.1 – Табличный и алгоритмический способы

Результат для последовательности от 0 до 9 представлен на рисунке 2.2.

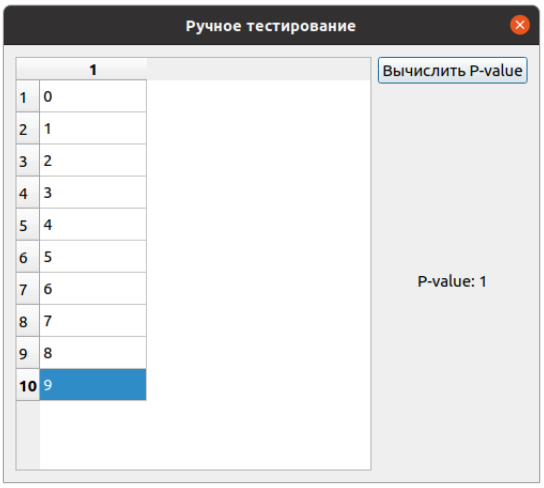

Рисунок 2.2 – Последовательность от 0 до 9

Результат для последовательности, состоящей из одних единиц, представлен на рисунке 2.3.

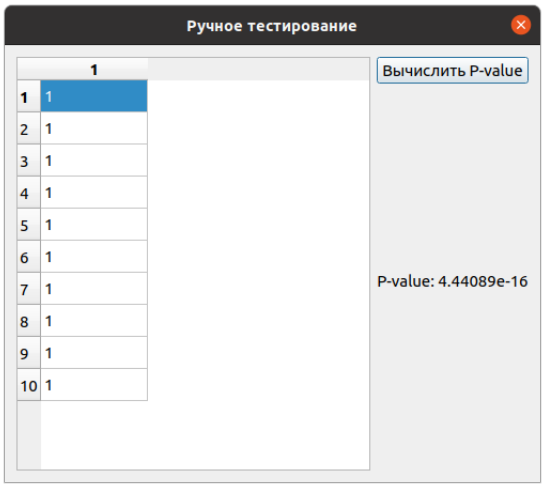

Рисунок 2.3 – Последовательность из одних единиц

Результат для последовательности чисел 1 и 9 представлен на рисунке 2.4.

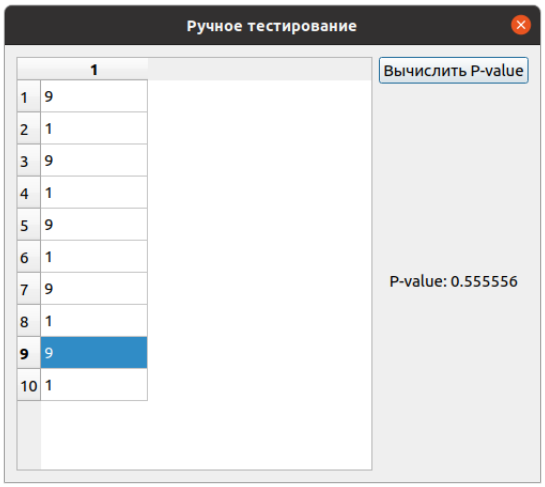

Рисунок 2.4 – Последовательность чисел 1 и 9