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: «Псевдослучайные числа»

Студент ИУ7-72Б (Группа) _____ (Подпись, дата) Т. М. Сучкова (И. О. Фамилия)

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

2022 г.

1 Задание

Реализовать критерий оценки случайности последовательности.

Последовательности требуется получить алгоритмическим и табличным способами для одно-, двух- и трехразрядных целых чисел (по 5000 чисел в каждой). Также предусмотреть ввод последовательности из собственных чисел в количестве 10 значений.

Каждую из полученных последовательностей оценить с помощью выбранного критерия.

2 Теоретическая часть

2.1 Способы генерации последовательности псевдослучайных чисел

2.1.1 Алгоритмический способ

В качестве алгоритмического способа рассмотрим линейный конгруэнтный метод.

В данном методе каждой следующее число рассчитывается на основе предыдущего по формуле (2.1).

$$g_{n+1} = (k \cdot g_n + C) \; mod \; N, \; n \geq 1 \tag{2.1}$$

где k, C – коэффициенты, N – модуль.

2.1.2 Табличный способ

В данном способе последовательность случайных чисел получают из заранее подготовленной таблицы (файла), данные в которой являются числами, не зависящими друг от друга.

2.2 Критерий оценки

За критерий оценки был принят критерий монотонности с опорой на критерий χ^2.

2.2.1 Критерий χ^2

Данный критерий относится к самым известным из статистических критериев и является основным методом, который используют в сочетании с другими критериями.

Критерий χ^2 позволяет выяснить, удовлетворяет ли генератор случайных чисел требованию равномерного распределения.

Используется статистика, представленная формулой (2.2).

$$V = \frac{1}{n} \sum_{s=1}^{k} \left(\frac{Y_s^2}{p_s} \right) - n \tag{2.2}$$

где n – количество независимых испытаний, k – количество категорий, Y_s – число наблюдений, которые относятся к категории S, p_s – вероятность того, что каждое наблюдение относится к категории S.

2.2.2 Критерий монотонности

Данный критерий используется для проверки распределения длин монотонных подпоследовательностей в последовательностях вещественных чисел.

Рассмотрим следующий пример.

Пусть дана выборка:

0.7, 0.03, 0.4, 0.17, 0.24, 0.55, 0.33, 0.64

Найдем в ней отрезки возрастания при условии, что смежные отрезки не являются независимыми, а значит, необходимо «выбросить» элемент, который следует непосредственно за серией. Таким образом, когда X_j больше $X_j + 1$, начнем следующую серию с X_{j+2}.

В данной последовательности найдено 4 отрезка возрастания:

[0.7], [0.4], [0.24, 0.55], [0.64].

Таким образом, в данной последовательности имеется три отрезка длиной 1 и один отрезок длиной 2.

В таком случае, после подсчета количества отрезков возрастания с различной длиной, можем использовать критерий χ^2 со следующими вероятностями:

$$\begin{cases} p_r = \frac{1}{r!} - \frac{1}{(r+1)!}, & r < t \\ p_t = \frac{1}{t!}, & r \geq t \end{cases} \tag{2.3}$$

3 Результат

На рисунках 3.1 и 3.2 представлены примеры результата работы программы для разных начальных значений генератора случайных чисел (имеет значение для алгоритмического способа).

На рисунках 3.3 и 3.4 представлены гистограммы входных данных для примеров работы программы 1 и 2 соответственно.

	Собственный	Табличный метод			Алгоритмический метод		
№	1 разряд	1 разряд	2 разряд	3 разряд	1 разряд	2 разряд	3 разряд
0	1	9	97	450	5	37	791
500	9	4	98	736	5	53	675
1000	1	1	98	206	3	45	319
1500	9	2	72	560	1	13	335
2000	1	4	17	891	3	29	219
2500	9	7	17	287	9	93	619
3000	1	4	93	870	9	25	599
3500	9	5	97	386	7	95	807
4000	1	4	21	130	7	51	415
4500	9	9	84	962	1	73	611
коэф	34.0	552.7287689269256	1208.223140495868	1177.7677371172517	1585.3421474358975	1459.5796078431372	1055.3016105417278

Рисунок 3.1 – Пример 1 (seed = 150)

	Собственный	Табличный метод			Алгоритмический метод		
№	1 разряд	1 разряд	2 разряд	3 разряд	1 разряд	2 разряд	3 разряд
0	1	8	15	539	7	59	717
500	9	2	77	250	9	69	757
1000	1	7	27	333	5	73	557
1500	9	6	21	585	3	53	945
2000	1	6	79	194	3	99	589
2500	9	2	63	515	7	85	857
3000	1	4	85	586	7	83	417
3500	9	8	45	717	7	21	745
4000	1	7	84	603	1	43	905
4500	9	8	65	388	7	95	861
коэф	34.0	481.73865110246425	1147.0104011887074	1277.6230007616145	1419.8529182879379	1728.0425880425882	1198.0892723180796

Рисунок 3.2 – Пример 2 (seed = 100)

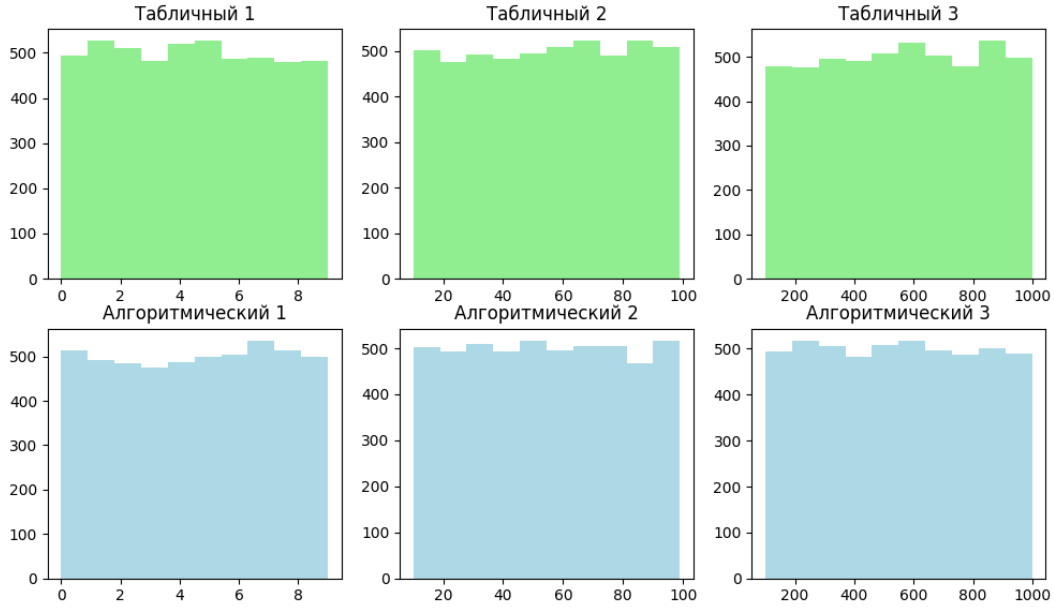

Рисунок 3.3 – Гистограммы для примера 1

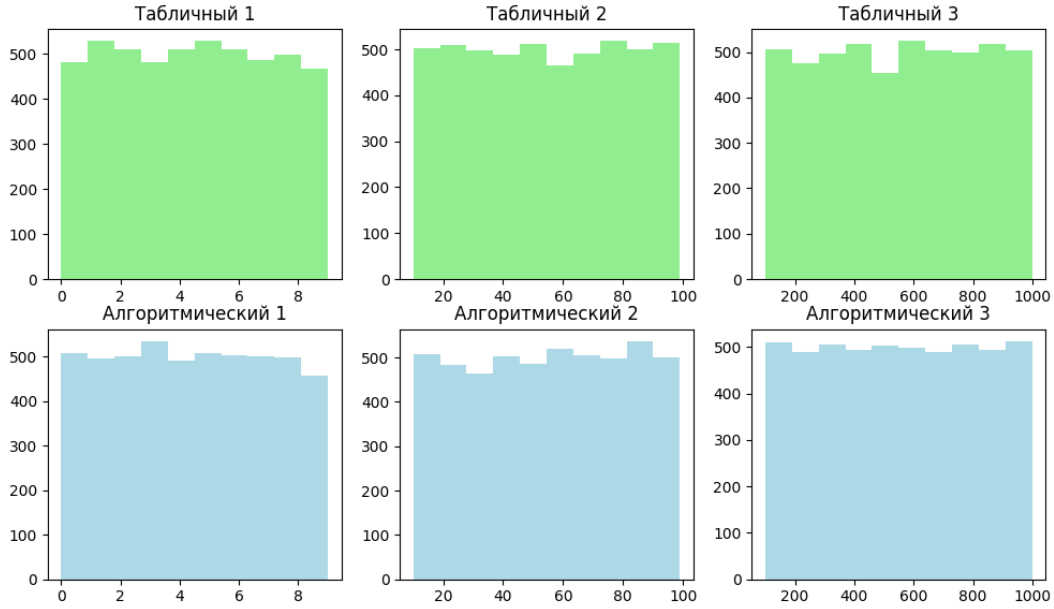

Рисунок 3.4 – Гистограммы для примера 2

4 Код программы

В листингах 4.1–4.2 представлен основной код программы.

Листинг 4.1 – Класс генератора случайных чисел

```
# Randomizer
# To generate random numbers – linear congurent method


# consts
# K             C           N
# 4096          150889      714025
# 36261         66037       312500
# 84589         45989       217728

# 1664525       1013904223  2^32
# 22695477      1           2^32
# 1103515245    12345       2^31

class Randomizer:
    def __init__(self, seed = 10):
        self.current = seed

        self.N = 217728
        self.C = 45989
        self.K = 84589

    def get_number(self, low = 0, high = 100) -> int:
        self.current = (self.K * self.current + self.C) % self.N
        res = int(low + self.current % (high – low))

        return res
```

Листинг 4.2 – Основной код программы

```
from Randomizer import Randomizer
from itertools import islice
from math import factorial
import numpy as np
from prettytable import PrettyTable
import matplotlib.pyplot as plt

NUMB_COUNT = 5000

def read_table_data(filename, count) -> set[int]:
    numbers = set()

    with open(filename) as file:
        line_num = 0
        lines = islice(file, line_num, None)

        for l in lines:
            numbers.update(set(l.split(" ")[1:-1]))

            if len(numbers) >= count + 1:
                break

            line_num += 1

        numbers.remove("")
        numbers = list(numbers)[:count]

    return numbers


def tabular_method(filename, count = NUMB_COUNT):
    numbers = read_table_data(filename, 3 * count)

    single_digit = [int(i) % 10 for i in numbers[:count]]
    double_digit = [int(i) % 90 + 10 for i in numbers[count:count *
        2]]
    three_digit = [int(i) % 900 + 100 for i in numbers[count * 2:3
       * count]]

    return single_digit, double_digit, three_digit
```

```
39
40
41 def algorithmic_method(seed = 10, count = NUMB_COUNT):
42     rnd = Randomizer(seed)
43
44     single_digit = [rnd.get_number(0, 10) for i in range(count)]
45     double_digit = [rnd.get_number(10, 100) for i in range(count)]
46     three_digit = [rnd.get_number(100, 1000) for i in range(count)]
47
48     return  single_digit, double_digit, three_digit
49
50
51 def get_hi(arr, n) -> int:
52     r = 0
53
54     arr_len = len(arr)
55
56     for i in range(arr_len):
57         if i == arr_len - 1:
58             p = (1 / factorial(i + 1))
59         else:
60             p = (1 / factorial(i + 1) - 1 / factorial(i + 1 + 1))
61
62         r += arr[i] * arr[i] / p
63
64     r = r / n - n
65
66     return r
67
68
69 def monotonicity_cr(arr):
70     tabs = np.zeros(6, dtype='int64')
71
72     i = 0
73     length = 1
74
75     while i < len(arr):
76         if (i == len(arr) - 1) or (arr[i] > arr[i + 1]):
77             j = 5 if length >= 6 else length - 1
78             tabs[j] += 1
79
```

```
80              i += 1
81              length = 0
82
83          i += 1
84          length += 1
85
86      n = sum(i * tabs[i] for i in range(len(tabs)))
87
88      return get_hi(tabs, n)
89
90
91  def draw_arr(arr, axis, i, j, name, clr = "blue"):
92      axis[i][j].set_title(f"{name} {j+1}")
93      axis[i][j].hist(arr, color = clr)
94
95
96  def draw_all(tbls, algs):
97      fig, axis = plt.subplots(2, 3, figsize = (12,7))
98
99      for i in range(len(tbls)):
100         draw_arr(tbls[i], axis, 0, i, "Табличный", clr = "
              lightgreen")
101
102     for i in range(len(algs)):
103         draw_arr(algs[i], axis, 1, i, "Алгоритмический", clr = "
              lightblue")
104
105     plt.show()
106
107
108 def main():
109     step = int(NUMB_COUNT / 10)
110     numbers = [i for i in range(0, NUMB_COUNT, step)]
111
112     io_arr = [1, 9, 1, 9, 1, 9, 1, 9, 1, 9]
113     single_tbl, double_tbl, three_tbl = tabular_method("table_data.
          txt")
114     single_alg, double_alg, three_alg = algorithmic_method(seed =
          100, count = NUMB_COUNT)
115
116     table_tbl = PrettyTable()
```

```
117
118     table_tbl.add_column("N", numbers)
119
120     table_tbl.add_column('1 разряд', io_arr)
121
122     table_tbl.add_column('1 разряд', single_tbl[::step])
123     table_tbl.add_column('2 разряд', double_tbl[::step])
124     table_tbl.add_column('3 разряд', three_tbl[::step])
125
126     table_tbl.add_column('1 разряд', single_alg[::step])
127     table_tbl.add_column('2 разряд', double_alg[::step])
128     table_tbl.add_column('3 разряд', three_alg[::step])
129
130     table_tbl.add_row(['коэф',
131                        monotonicity_cr(io_arr),
132                        monotonicity_cr(single_tbl),
133                        monotonicity_cr(double_tbl),
134                        monotonicity_cr(three_tbl),
135                        monotonicity_cr(single_alg),
136                        monotonicity_cr(double_alg),
137                        monotonicity_cr(three_alg)])
138
139     print("       Собственный|\t\t\tТабличный метод\t\t\t\t |   Алг
        оритмический метод")
140     print(table_tbl)
141
142     draw_all([single_tbl, double_tbl, three_tbl], [single_alg,
        double_alg, three_alg])
143
144 if __name__ == '__main__':
145     main()
```