Министерство науки и высшего образования Российской ФедерацииФедеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имениН.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 псевдослучайных последовательностей

Студент Климов И.С.

Группа ИУ7-72Б

Оценка (баллы) _____

Преподаватель Рудаков И.В.

Москва, 2022 г.

Задание

Используя алгоритмический и табличный метод сгенерировать последовательность из 100 одноразрядных, двухразрядных и трёхразрядных чисел. Отобразить первые 10 в графическом интерфейсе. Составить статистический критерий оценки случайности последовательностей и вывести его значение для каждой последовательности. Также предусмотреть возможность получения оценки для 10 чисел, введённых пользователем.

Методы получения последовательности случайных чисел

Существует три метода получения последовательности случайных чисел:

- аппаратный (физический);
- табличный (файловый);
- алгоритмический (программный).

В данной лабораторной работе рассмотрены и реализованы два последних.

Табличная схема

Случайные числа оформляются в виде таблицы и помещаются во внешнюю или оперативную память. В лабораторной работе файлы были сгенерированы при помощи стандартной библиотеки `random` языка программирования `Python`. Начальное значение генерируется таким же способ и используется для смещения относительно начала файла.

Алгоритмический способ

Способ основан на формировании случайных чисел с помощью специальных алгоритмов. Очередное полученное значение используется для генерации последующих чисел.

В данной работе был использован мультипликативным конгруэнтный метод. Последовательность вычисляется по формуле:

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

при этом $c = 0$.

Самым известным генератором подобного рода является так называемый минимальный стандартный генератор случайных чисел, предложенный Стивеном Парком и Кейтом Миллером в 1988 году. Для него $a = 16807$, $m = 2147483647$.

В данной лабораторной работе в качестве начального значения X_1 используется текущее время в секундах.

Критерий оценки

В качестве критерии оценки случайности последовательности взят критерий знаково-рейтинговый критерий Холлина, основанный на статистике:

$$r = \frac{1}{k(n-1)} \sum_{i=2}^{n} \delta\Big[(x_i - \tilde{x})(x_{i-1}\tilde{x})\Big] R_i R_{i-1},$$

где

- k –коэффициент, зависящий от объема выборки (значения подбираются по таблице);
- $\tilde{x}$ – медиана вариационного ряда $x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$;
- R_i – ранг величины $z_i = |x_i - \tilde{x}|$ в упорядоченном по возрастанию ряду значений $z_1 \leq z_2 \leq \ldots z_n$;
- $\delta = \begin{cases} 1, & y > 0; \\ -1, & y < 0 \\ 0, & y = 0. \end{cases}$

Ряд значений x_i признается случайным, еслиr $|r| < r_\alpha$. Критические значения представлены на рисунке 1.1. При этом если $r = 0$, то ряд не будет являться случайным.

Критические значения r_α знаково-рангового критерия Холлина [556]

n	Уровень значимости α			n	Уровень значимости α		
	0,10	0,05	0,025		0,10	0,05	0,025
5	0,667	0,816	0,915	28	0,250	0,317	0,374
6	0,608	0,749	0,834	29	0,245	0,311	0,367
7	0,548	0,681	0,778	30	0,241	0,306	0,361
8	0,501	0,625	0,720	32	0,233	0,296	0,349
9	0,467	0,587	0,682	34	0,225	0,287	0,339
10	0,421	0,553	0,642	36	0,219	0,279	0,320
11	0,415	0,522	0,610	38	0,213	0,271	0,320
12	0,394	0,498	0,583	40	0,207	0,264	0,312
13	0,380	0,479	0,558	42	0,202	0,257	0,304
14	0,361	0,455	0,535	44	0,197	0,251	0,297
15	0,349	0,442	0,518	46	0,192	0,246	0,291
16	0,338	0,428	0,502	48	0,188	0,240	0,285
17	0,325	0,412	0,485	50	0,184	0,235	0,279
18	0,317	0,402	0,472	55	0,175	0,224	0,266
19	0,306	0,388	0,459	60	0,168	0,214	0,254
20	0,298	0,378	0,446	65	0,161	0,206	0,244
21	0,292	0,370	0,436	70	0,155	0,198	0,235
22	0,283	0,361	0,425	75	0,150	0,191	0,227
23	0,276	0,350	0,413	80	0,145	0,185	0,220
24	0,270	0,343	0,404	85	0,140	0,180	0,213
25	0,265	0,338	0,400	90	0,136	0,174	0,207
26	0,260	0,330	0,388	95	0,133	0,170	0,202
27	0,255	0,323	0,323	100	0,129	0,165	0,197

Рисунок 1.1 - Критические значения для критерия Холлина

Текст программы

Ниже представлен текст программы, написанной на языке программирования Python.

```
import random as r
import time


class AlgorithmGenerator:
    def __init__(self, n_min: int, n_max: int):
        self.n_min: int = n_min
        self.n_max: int = n_max

    def get_random_numbers(self, n: int, seed: int = -1) -> list[int]:
        if seed == -1:
            seed = round(time.time())
        a = 16807
        m = 0x7fffffff

        numbers: list[int] = []
        for i in range(n):
            seed = seed * a % m
            numbers.append(seed % (self.n_max - self.n_min) + self.n_min)

        return numbers


class TableGenerator:
    def __init__(self, n_min: int, n_max: int):
        self.n_min: int = n_min
        self.n_max: int = n_max
        self.random_cnt: int = (self.n_max - self.n_min) * 1000

    def fill_file_by_random(self, filename: str):
        with open(filename, 'w') as f:
            for i in range(self.random_cnt):
                f.write(str(r.randint(self.n_min, self.n_max)) + '\n')

    def get_random_numbers(self, filename: str, n: int, seed: int = -1) -> list[int]:
        if seed == -1:
            seed = r.randint(0, self.random_cnt)

        numbers: list[int] = []
        with open(filename) as f:
            all_numbers: list[int] = [int(x) for x in f.read().split()]

        for i in range(seed, seed + n):
            numbers.append(all_numbers[i % self.random_cnt])

        return numbers
```

```python
class HollinCriterion:
    @staticmethod
    def count(x: list[int]) -> float:
        def R(idx: int) -> float:
            indices: list[int] = [i + 1 for i, value in enumerate(var_series_z) if value
== z[idx]]
            return sum(indices) / len(indices)

        def sign(y: int) -> int:
            if y > 0:
                return 1
            elif y < 0:
                return -1
            else:
                return 0

        def get_median() -> int | float:
            if n % 2:
                return var_series[(n - 1) // 2]
            else:
                return (var_series[n // 2 - 1] + var_series[(n // 2)]) / 2

        def get_k() -> float:
            def k_func(n1, n2, k1, k2) -> float:
                return (n - n1) * (k2 - k1) / (n2 - n1) + k1

            if 5 <= n < 10:
                return k_func(5, 10, 10.11, 36.95)
            elif 10 <= n < 20:
                return k_func(10, 20, 36.95, 140.62)
            elif 20 <= n <= 50:
                return k_func(20, 50, 140.62, 851.62)
            elif 50 <= n <= 100:
                return k_func(50, 100, 851.62, 3370)
            elif 100 <= n < 200:
                return k_func(100, 200, 3370, 13407)
            elif 200 <= n <= 400:
                return k_func(200, 400, 13407, 53480)
            else:
                raise ValueError(f'There is not value for n = {n}')

        n: int = len(x)
        var_series: list[int] = list(sorted(x))  # вариационный ряд
        median: int | float = get_median()
        k: float = get_k()
        z: list[int | float] = [abs(x[i] - median) for i in range(n)]
        var_series_z: list[int | float] = list(sorted(z))
        series: = [sign((x[i] - median) * (x[i - 1] - median)) * R(i) * R(i - 1)
                   for i in range(1, n)]
        return 1 / (k * (n - 1)) * sum(series)
```

Результат

В результате разработана программа, позволяющая получить случайные последовательности чисел табличным и алгоритмическим способами, вводить с клавиатуры и получать значения статистики, на которой основан критерий Холлина. На рисунке 2.1 представлен интерфейс. Последняя строка в каждом столбце – значение статистики.

Рисунок 2.1 - Интерфейс разработанной программы

На рисунках 2.2 – 2.5 представлены результаты работы программы для различных значений с клавиатуры. При этом при каждом запуске генерировалась новая последовательность случайных чисел алгоритмическим и табличным способами.

Рисунок 2.2 - Результат для одинаковых чисел

Рисунок 2.3 - Результат для возрастающей последовательности

Рисунок 2.4 - Результат для убывающей послежовательности

Рисунок 2.5 - Результат для чередующихся чисел

Можно увидеть, что для всех тестов критерий показывает правильный результат в соответствие с таблицей – для случайных последовательностей значения по модулю низкие, для неслучайных – высокие или равны 0.