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
по курсу «Моделирование»

«Генерация случайных чисел»

Студент: ИУ7-73Б _____ М. Д. Маслова
(группа) (подпись, дата) (И. О. Фамилия)

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

2022 г.

СОДЕРЖАНИЕ

1 Задание

Разработать программное обеспечение, предоставляющее возможность генерации последовательности случайных чисел алгоритмическим и табличным спобом, а также возможность расчета коэффициента критерия случайности по полученным последовательностям.

Реализовать графический интерфейс, позволяющий пользователю ввести последовательность для проверки ее случайности.

2 Теоретическая часть

2.1 Методы получения последовательности случайных чисел

Для генерации случайных чисел может применяться один из следующих способов:

- ***аппаратный***, в основе которого лежит какой-либо физический эффект (не реализуется в данной работе);
- ***табличный***, при использовании которого заранее полученные и проверенные случайные числа оформлены в виде таблице в памяти ЭВМ;
- ***алгоритмический***, с помощью которого формируются детерминированные последовательности чисел, где каждое число зависит от предыдущего, но для стороннего наблюдателя такие последовательности выглядят случайными, из-за чего называются псевдослучайными.

2.1.1 Алгоритмический способ

В данной работе реализуется ***квадратичный когруэнтный метод***, в котором последовательность чисел формируется следующим образом:

$$y_{n+1} = (Ay_n^2 + By_n + C) \bmod m, \tag{2.1}$$

где $m = 2^l$.

Если $l \geq 2$, то наибольшее значение периода квадратического конгруэнтного датчика составляет $T_{\max} = 2^l$, что достигается при четном A, нечетном C и если нечетное B удовлетворяется условию $B \bmod 4 = (A + 1) \bmod 4$.

2.1.2 Табличный способ

В данной работе для генерации случайных чисел табличным способом используются цифры из части таблицы ***«A Million Random Digits with 100,000 Normal Deviates»***, опубликованной в 1955 году.

Данная таблица сохранена в виде текстового файла. Для генерации чисел выбирается начальная позиция в файле, читаются следущие n цифр, где n — количество разрядов в генерируемом числе, и из стороковая последовательность преобразуется в число. Для генерации следующиего числа происходит переход к следующей строке таблице с сохранением номера столбца. При

невозможности перейти к следующей строке в связи с окончанием файла позиция переводится на первую строку, а номер столбца увеличивается на единицу. Если цифр в строке не хватает для формирования числа, они берутся из начала следующей строки.

2.2 Критерий случайности

Для оценки случайности был использован критерий на основе углов между векторами, координаты начала и конца которых составляются из двух соседних пар последовательности с одним общим числом.

3 Практическая часть

3.1 Текст программы

На листинге 3.1 представлена реализация квадратичного когруэнтного метода генерации случайных чисел.

Листинг 3.1 – Реализация квадратичного когруэнтного метода генерации случайных чисел

```
class Generator:

    def __init__(self, normGen, lower=0, upper=100):
        self.normGen = normGen
        self.lower = lower
        self.upper = upper

    def GenerateNumber(self):
        return round(self.normGen.GetNumber01()
                     * (self.upper - self.lower) + self.lower)

    def GenerateSequence(self, length):
        return [self.GenerateNumber() for _ in range(length)]


class QuadraticGenerator(Generator):

    def __init__(self, lower, upper):
        super().__init__(QuadraticRandom(), lower, upper)


class QuadraticRandom(NormalizedRandom):

    def __init__(self):
        self.current = 4001
        self.A = 6
        self.B = 7
        self.C = 3
        self.m = 8192

    def GetNumber01(self):
        self.current = (self.A * self.current * self.current
                        + self.B * self.current
                        + self.C) % self.m
        return self.current / self.m
```

На листинге 3.2 представлена реализация табличного способа получения последовательности случайных чисел.

Листинг 3.2 – Реализация табличного способа генерации случайных чисел

```
from datetime import datetime

PAGES_NUMBER = 7
ROWS_PER_PAGE = 50
COLS_PER_ROW = 50

SYMBOLS_PER_PAGE = ROWS_PER_PAGE * COLS_PER_ROW
SYMBOLS_NUMBER = SYMBOLS_PER_PAGE * PAGES_NUMBER

class TabularGenerator:

    def __init__(self):
        page   = datetime.now().microsecond % PAGES_NUMBER
        row    = datetime.now().microsecond % ROWS_PER_PAGE
        column = datetime.now().microsecond % COLS_PER_ROW

        self.position = SYMBOLS_PER_PAGE * page + COLS_PER_ROW *
           row + column


    def GenerateNumber(self, digits=1):
        num = -1
        with open("randseq/data/digits.txt", "r") as f:
            notRead = True

            while notRead:
                f.seek(self.position, 0)
                num = int(f.read(digits))

                if num // (10 ** (digits - 1)) >= 1:
                   notRead = False

                self.position += COLS_PER_ROW

                if self.position > SYMBOLS_NUMBER - 1:
                    self.position %= SYMBOLS_NUMBER
                    self.position += 1

        return num
```

На листинге 3.3 представлена функция вычисления коэффициента описанного выше критерия оценки случайности.

Листинг 3.3 – Реализация функции расчета коэффициента критерия оценки случайности

```
class RandomnessCriterion:

    def __init__(self):
        pass


    def GetCoefficient(self, sequence):
        return sum(sequence)
```

3.2 Полученный результат