



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: «Генераторы псевдослучайных чисел»

Студент:

Ле Ни Куанг

Группа:

ИУ7И-76Б

Преподаватель:

Рудаков И. В.

Москва

2021

1 Задание

Изучить и реализовать генератор псевдослучайных чисел программным и табличным методом. Разрядность чисел должна быть равна 1, 2, 3. Сравнить методы по определенному критерию и сделать выводы.

2 Теоритическая часть

2.1 Программный генератор

Программный генератор формирует псевдослучайные числа. Каждое последующее число в такой последовательности зависит от предыдущего. Сгенерированные числа статистически случайны, несмотря на то, что это полностью детерминированный и повторяемый процесс.

2.1.1 Линейный конгруэнтный метод

Генератор определяется рекуррентным соотношением:

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

где: $m > 0$, модуль
 $0 \leq a \leq m$, множитель
 $0 \leq c \leq m$, приращение
 $0 \leq X_0 \leq m$, начальное число

2.2 Табличный генератор

Табличный генератор использует таблицу проверенных некоррелированных цифр в качестве источника случайных чисел.

2.3 Критерий равномерности

Пусть имеется последовательность целых чисел X_n , $0 \leq X_i < d$.

При оценке равномерности для каждого r ($0 \leq r < d$) подсчитывается количество случаев, когда элемент последовательности $X_i = r$.

После этого применяется критерий χ^2 , в котором вычисляется статистика:

$$\chi^2 = \sum_{i=0}^{k-1} \frac{(x_i - m_i)^2}{m_i} \sim \chi^2(k-1)$$

Где $m_i = np_i = n/d$

Рассчитать значение p для уровня погрешность $\alpha = 0.05$ допускать $0.05 < p < 0.95$.

3 Результаты

Lab 3 - Random number generator

10000 Генерировать

	Программный метод:			Табличный метод:			
	1p	2p	3p	1p	2p	3p	?p
	7	39	372	6	91	407	11
	1	26	720	7	15	433	99
	9	8	76	2	3	967	66
	0	43	535	4	24	107	28
	7	27	436	5	74	812	82
	7	45	846	5	94	422	44
	5	94	837	7	56	934	55
	6	0	881	7	3	153	
	5	28	493	3	60	940	
	9	63	403	9	21	998	
m:	4.5	49.54	506.76	4.49	49.51	494.55	55.0
p:	0.831	0.386	0.854	0.893	0.255	0.581	0.651

Послед. число 11 99 66 28 82 44 55 Добавить

Lab 3 - Random number generator

100000 Генерировать

	Программный метод:			Табличный метод:			
	1p	2p	3p	1p	2p	3p	?p
	6	47	173	3	28	713	33
	7	4	568	4	34	186	80
	8	79	943	4	46	887	47
	0	41	234	0	38	653	11
	2	77	661	0	46	986	46
	2	14	673	4	63	12	77
	8	21	866	6	96	895	36
	9	46	560	9	46	893	95
	7	32	527	8	8	796	86
	6	51	809	6	95	495	
m:	4.52	49.53	500.33	4.49	49.43	499.45	56.778
p:	0.661	0.175	0.255	0.639	0.944	0.214	0.704

Послед. число 33 80 47 11 46 77 36 95 86 Добавить

При уровня погрешность $\alpha = 0.05$, $0.05 < p < 0.95$ в большинстве случаев. Чем больше количество генерируемых случайных чисел, тем равномернее они распределены. Но для большого количества чисел какая-то закономерность повторяется.

4 Листинг кода

Листинг 1 – программная реализация генерации псевдослучайных чисел программным и табличным методом

```
import enum
from datetime import datetime
from random import randrange

from scipy.stats import chisquare

class RandomGenerator:
    class Method(enum.Enum):
        Standard = 1
        Table = 2
        LinearCongruential = 3

    def __init__(self, table_path='digits.txt'):
        self.digits = ''
        with open(table_path) as table:
            for line in table:
                self.digits += line[:-1]

        self.table_len = len(self.digits)
        self.seed = int(datetime.now().microsecond)
        self.table_idx = self.seed % self.table_len
        self.table_idx = 0

    def __standard(self, n_digits):
        return randrange(10 ** n_digits)

    # https://en.wikipedia.org/wiki/Linear\_congruential\_generator
    def __linear_congruential(self, n_digits, modulus=2e31, a=1664525,
                              c=1013904223):
        self.seed = (a * self.seed + c) % modulus
        return int(self.seed / modulus * (10 ** n_digits))

    def __table(self, n_digits):
        num = ''
        for _ in range(n_digits):
            num += self.digits[self.table_idx]
            self.table_idx = (self.table_idx + 1) % self.table_len
        return int(num)

    def rand(self, method: Method, n_digits, n_numbers):
        res = []
        if method == RandomGenerator.Method.Standard:
```

```

        res = [self.__standard(n_digits) for _ in range(n_numbers)]
    elif method == RandomGenerator.Method.Table:
        res = [self.__table(n_digits) for _ in range(n_numbers)]
    elif method == RandomGenerator.Method.LinearCongruential:
        res = [self.__linear_congruential(n_digits) for _ in
                range(n_numbers)]
    return res

class RandomCriteria:
    @staticmethod
    def p_of_chi_square_test(sequence, n_digits):
        d = 10 ** n_digits
        f_obs = [0] * d
        for i in sequence:
            f_obs[i] += 1

        # Got same result: y == chisq
        # y = 0
        # mi = len(sequence) / d
        # for i in range(d):
        #     y += (f_obs[i] - mi) ** 2
        # y /= mi

        chisq, p = chisquare(f_obs)
        return p

```