



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	<u>ИУ7-73Б</u>	_____	<u>Лагутин Д. В.</u>
	(Группа)	(Подпись, дата)	(Фамилия И. О.)
Преподаватель		_____	<u>Рудаков И. В.</u>
		(Подпись, дата)	(Фамилия И. О.)

Цель работы

Целью работы является реализация программы для генерации псевдослучайных чисел с использованием табличного и алгоритмического методов. Также необходимо выбрать метрику для оценки случайности последовательности чисел и вычислить ее значение для сгенерированного набора.

Алгоритмы генерации псевдослучайных чисел

На практике применяют три основных способа генерации случайных чисел:

- аппаратный — случайная величина вырабатывается специальной электрической приставкой (генератор случайных чисел) как правило внешнее устройство компьютера не требует других устройств и операций кроме обращения к устройству;
- табличный — случайные числа оформляются в виде таблицы;
- алгоритмический — применяются специализированные алгоритмы.

Таблица 1 – Сравнение методов генерации псевдослучайных чисел

Способ	Достоинства	Недостатки
Аппаратный	Запас чисел неограничен	Неустойчивость к внешнему воздействию
	Мало вычислительных операций	Нельзя воспроизвести последовательность
	Не занимает место в памяти	Используются специальные устройства
		Необходимы меры стабилизации
Табличный	Можно воспроизвести последовательность	Запас чисел ограничен
	Однократная проверка на случайность	Занимает место в памяти и на диске
		Затраты на обращение к памяти
Алгоритмический	Однократная проверка на случайность	Запас чисел ограничен периодом
	Многократное воспроизведение последовательности	Существенные затраты вычислительных ресурсов
	Малые затраты памяти	
	Не требуются специальные устройства	

Линейный конгруэнтный метод

В качестве алгоритмического метода генерации псевдослучайных чисел был выбран линейный конгруэнтный метод. Для осуществления генерации чисел данным методом, необходимо задать 4 числа:

- $m > 0$ — модуль;
- $0 \leq a \leq m$ — множитель;
- $0 \leq c \leq m$ — приращение;
- $0 \leq X_0 \leq m$ — начальное значение.

Последовательность случайных чисел генерируется при помощи следу-

ющего рекуррентного соотношения:

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

В качестве значений коэффициентов были выбраны соответствующие значения для реализации функции `rand` из `glibc`:

$$m = 2^{32}$$

$$a = 1103515245$$

$$c = 12345.$$

Табличный метод

Табличный метод базируется на выборке и методе, описанных в ГОСТ 11.003-73

Генерация случайных чисел производится на основании 8 таблиц, содержащих 1024 цифры.

Выбор случайной последовательности из n l -разрядных чисел в диапазоне $[a, b]$ выполняется по следующим правилам:

- используя начальное смещение выбирается таблица, строка и столбец;
- относительно начальной позиции выбираются $n \cdot l$ цифр при движении слева направо, сверху вниз; если текущая таблица содержит меньшее требуемого количество цифр, то необходимо перейти к следующей таблице; в случае исчерпания последней таблицы, указатель переходит в начало первой таблицы;
- выбранные цифры объединяются в группы по l элементов, что является представлением l -разрядного числа;
- если число не подходит под указанный диапазон, оно — вычеркивается, и из таблицы выбирается следующее число.

Критерий проверки χ^2

Критерий χ^2 является критерием проверки простой непараметрической гипотезы о законе распределения дискретной случайной величины X .

Пусть X может принимать l различных значений a_1, \dots, a_l с вероятностями p_1, \dots, p_l соответственно ($\frac{1}{l}$ в случае равномерного распределения).

Тогда, если X_n — случайная выборка размера n , то для определения ее случайности может использоваться следующую статистику

$$\Delta(\vec{X}_n) = \sum_{i=1}^l \frac{(n_i(\vec{X}_n) - np_i)^2}{np_i},$$

где $n_i(\vec{X}_n)$ — случайная величина, принимающая для каждой реализации \vec{x}_n случайной выборки \vec{X}_n значение, равное числу компонент выборки, имеющих значение a_i . Малые значения данной статистики связаны с близостью фактических значений вероятности появления числа в последовательности с теоретическим значением $\frac{1}{l}$.

Доказано, что статистика $\Delta(\vec{X}_n)$ слабо сходится к случайной величине, имеющей распределение χ^2 с $l - 1$ степенями свободы, при $n \rightarrow \infty$.

Таким образом, значение $1 - \chi^2_{l-1}(\Delta(\vec{x}_n))$, где $\chi^2_{l-1}(\Delta(\vec{x}_n))$ — значение функции распределения χ^2 для значения $\Delta(\vec{x}_n)$, может быть использовано для оценки случайности последовательности, чем ближе значение метрики к 1, тем более случайна выборка, и наоборот, если значение близко к 0, то выборка — неслучайна.

Так как проверяется гипотеза о равномерности закона распределения генератора, то значение статистики может быть вычислено следующим образом

$$\Delta(\vec{X}_n) = \frac{l}{n} \sum_{i=1}^l n_i(\vec{X}_n)^2 - n,$$

Текст программы

Листинг 1 – Интерфейс генератора случайных чисел

```
1 from abc import ABC, abstractmethod
2
3 class RandomChecker(ABC):
4     @abstractmethod
5     def check(self, sequence: list[int]) -> float:
6         pass
7
8 class Randomizer(ABC):
9     @abstractmethod
10    def get(self) -> int:
11        pass
12
13 class RandomizerCreator(ABC):
14    @abstractmethod
15    def create(self, a: int, b: int) -> Randomizer:
16        pass
```

Листинг 2 – Класс, реализующий линейный конгруэнтный метод

```
1 from time import time
2
3 from src.randomizer import Randomizer, RandomizerCreator
4
5 class LinearCongruentRandomizer(Randomizer):
6     def __init__(self, a: int, c: int, m: int, i: int,
7                 min: int, max: int):
8         self.a = a
9         self.c = c
10        self.m = m
11
12        self.current = i % self.m
13
14        self.min = min
15        self.rng = max - min + 1
16
17    def get(self) -> int:
```

```

18         self.current = (self.a * self.current + self.c) \
19                         % self.m
20
21     return self.min \
22           + int(round(self.rng * (self.current
23                           / (self.m - 1))
24                   - 0.5,
25                   0))
26
27 class LinearCongruentRandomCreator(RandomizerCreator):
28     def __init__(self, a: int, c: int, m: int):
29         self.a = a
30         self.c = c
31         self.m = m
32
33     def create(self, a: int, b: int) -> Randomizer:
34         return LinearCongruentRandomizer(self.a, self.c,
35                                           self.m,
36                                           int(time() * 1e6),
37                                           a, b)

```

Листинг 3 – Класс, реализующий табличный метод

```

1 from time import time
2 from math import log10
3
4 from src.randomizer import Randomizer, RandomizerCreator
5
6 class TableRandomizer(Randomizer):
7     def __init__(self, values: list[int], seed: int,
8                 min: int, max: int):
9         self.values = values
10        self.current = seed % len(self.values)
11        self.min = min
12        self.max = max
13        self.l = int(log10(self.max)) + 1
14        self.limit = len(self.values)
15
16    def get(self) -> int:
17        num = self.min - 1
18
19        while self.min > num or self.max < num:

```

```

20         num = self.values[self.current]
21         self.current = (self.current + 1) % self.limit
22
23         for _ in range(self.l - 1):
24             num = num * 10 + self.values[self.current]
25             self.current = (self.current + 1) \
26                 % self.limit
27
28         return num
29
30 class TableRandomCreator(RandomizerCreator):
31     def __init__(self, filename: str):
32         file = open(filename, "r")
33         content = file.read()
34         file.close()
35         self.values = list(map(lambda x: int(x),
36                                content[:-1]))
37
38     def create(self, a: int, b: int) -> Randomizer:
39         return TableRandomizer(self.values,
40                                int(time() * 1e6), a, b)

```

Листинг 4 – Класс, реализующий критерий χ^2

```

1 from scipy.stats import chi2
2
3 from src.randomizer import RandomChecker
4
5 class Chi2RandomChecker(RandomChecker):
6     def check(self, sequence: list[int]) -> float:
7         values = {}
8         n = len(sequence)
9         range = max(sequence) - min(sequence) + 1
10
11         for i in sequence:
12             if i in values:
13                 values[i] += 1
14             else:
15                 values[i] = 1
16
17         sum_sqr = sum(map(lambda x : values[x] * values[x],
18                           values))

```



```

19         val = range / n * sum_sqr - n
20
21     return 1 - chi2.cdf(val, range - 1)

```

Результаты работы

Разрядность чисел 1		
Метод Линейный Конгруэнтный		
Сгенерировать выборку		
	Метод	Пользовательский ввод
1	2	0
2	1	0
3	3	0
4	3	0
5	9	0
6	5	0
7	4	0
8	9	0
9	6	0
10	6	0
Случайность	0.725829	0

Разрядность чисел 2		
Метод Линейный Конгруэнтный		
Сгенерировать выборку		
	Метод	Пользовательский ввод
1	95	0
2	30	0
3	75	0
4	62	0
5	22	0
6	2	0
7	36	0
8	63	0
9	80	0
10	62	0
Случайность	0.859291	0

Разрядность чисел 3		
Метод Линейный Конгруэнтный		
Сгенерировать выборку		
	Метод	Пользовательский ввод
1	996	0
2	772	0
3	801	0
4	366	0
5	984	0
6	283	0
7	215	0
8	593	0
9	111	0
10	644	0
Случайность	0.794586	0

Рисунок 1 – Результаты работы программы для линейного конгруэнтного метода

Разрядность чисел 1		
Метод Табличный		
Сгенерировать выборку		
	Метод	Пользовательский ввод
1	0	0
2	3	0
3	9	0
4	0	0
5	5	0
6	7	0
7	9	0
8	9	0
9	0	0
10	4	0
Случайность	0.807412	0

Разрядность чисел 2		
Метод Табличный		
Сгенерировать выборку		
	Метод	Пользовательский ввод
1	90	0
2	0	0
3	76	0
4	77	0
5	13	0
6	18	0
7	60	0
8	66	0
9	24	0
10	13	0
Случайность	0.704327	0

Разрядность чисел 3		
Метод Табличный		
Сгенерировать выборку		
	Метод	Пользовательский ввод
1	946	0
2	245	0
3	183	0
4	832	0
5	323	0
6	754	0
7	932	0
8	503	0
9	789	0
10	827	0
Случайность	0.684615	0

Рисунок 2 – Результаты работы программы для табличного метода

Вывод

В ходе выполнения лабораторной работы была разработана программа для генерации псевдослучайных чисел с использованием табличного и линейного конгруэнтного методов. Также была реализована возможность оценки случайности последовательности чисел с использованием критерия χ^2 .