



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» (ИУ7)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.04 Программная инженерия

**О Т Ч Е Т**

по лабораторной работе № 3

Название: Генераторы псевдослучайных чисел

Дисциплина: Моделирование

Студент

ИУ7-72Б

(Группа)

\_\_\_\_\_  
(Подпись, дата)

Е.В. Брянская

(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

И.В. Рудаков

(И.О. Фамилия)

Москва, 2021

## 1 Задание

Программным способом сгенерировать последовательности по 1000 одноразрядных/двухразрядных/трёхразрядных целых чисел. А также найти готовые файлы со случайными числами.

Последовательности необходимо получить алгоритмическим и табличным способом.

Определить и вывести на экран количественный критерий оценки случайности этих последовательностей.

Также дать возможность пользователю ввести последовательность из 10 чисел с указанием количества разрядов. И для неё определить количественный критерий и вывести в соответствующем поле.

## 2 Теоретическая часть

Существует три основных способа получения последовательностей случайных чисел:

1. аппаратный;
2. табличный (файловый);
3. алгоритмический.

В рамках лабораторной работы рассмотрены алгоритмический и табличный способы.

### 2.1 Алгоритмический способ

Этот подход основан на базе специальных алгоритмов. К ним относятся:

- метод серединных произведений;
- метод перемешивания;
- линейный конгруэнтный метод.

Было принято решение взять последний для генерации последовательности псевдослучайных чисел.

В этом методе каждое следующее число рассчитывается на основе предыдущего по формуле (1).

$$R_{n+1} = (a \cdot R_n + b) \bmod N, n \geq 1 \quad (1)$$

где  $a$ ,  $b$  – коэффициенты,  $N$  – модуль.

Для качественного генератора требуется подобрать подходящие коэффициенты. Например, в таблице 2.1 приведены некоторые из них.

Таблица 2.1 — Примеры коэффициентов

<b>a</b>	<b>b</b>	<b>N</b>
106	1283	6075
430	2531	11979
84589	15989	217728
1103515245	12345	$2^{31}$
...	...	...

## 2.2 Табличный способ

В качестве источника случайных чисел используют специально заранее составленные таблицы, содержащие проверенные данные.

## 2.3 Критерий оценки

В рамках лабораторной работы в качестве статистической оценки используется критерий «хи-квадрат» ( $\chi^2$ -критерий).

Привлекая этот критерий можно оценить, удовлетворяет ли генератор требованию равномерного распределения или нет.

Используется статистика, представленная формулой (2).

$$V = \frac{1}{n} \sum_{s=\min}^{\max} \left( \frac{Y_s^2}{p_s} \right) - n \quad (2)$$

где  $n$  - длина последовательности,  $\min/\max$  - границы, в пределах которых находятся элементы последовательности,  $Y_s$  - число повторений числа  $s$ ,  $p = \frac{1}{\max - \min}$ .

Вычисляется квантиль хи-квадрат от вычисленного  $V$ . Если полученное значение будет меньше 0.1 или больше 0.9, то эти числа считаются недостаточно случайными. Если же нет, то последовательность принимается как случайная.

### 3 Результаты работы программы

Интерфейс предполагает вывод первых 10 элементов из 1000 каждой последовательности (за исключением той, которую введёт пользователь, так как в этом случае длина фиксирована и равна 10).

Результат работы программы приведен на рисунке 3.1. Можно заметить, что значения критерия в любом из столбцов находится в интервале от 0.1 до 0.9, что позволяет считать последовательности случайными.

Form1

Лабораторная работа №3

Алгоритмический			Табличный			Своя последовательность		
1 разряд	2 разряда	3 разряда	1 разряд	2 разряда	3 разряда	<input checked="" type="radio"/> 1 разряд	<input type="radio"/> 2 разряда	<input type="radio"/> 3 разряда
3	60	520	9	88	844			
0	19	582	3	79	323			
9	71	380	6	28	101			
4	71	861	3	55	553			
0	20	609	2	98	870			
7	52	760	9	13	535			
4	57	612	6	36	572			
2	53	886	4	50	633			
7	31	608	0	42	450			
8	88	437	9	10	591			

Количественный критерий оценки случайности последовательности

0.2	0.536	0.767	0.264	0.541	0.294			
-----	-------	-------	-------	-------	-------	--	--	--

Готово

Рисунок 3.1 — Пример 1

На рисунках 3.2-3.4 демонстрируются примеры пользовательских последовательностей.

Своя последовательность

☒ 1 разряд ☐ 2 разряда ☐ 3 разряда

7		
4		
6		
3		
0		
3		
2		
4		
5		
5		

Количественный критерий оценки случайности последовательности

0.26		
------	--	--

Рисунок 3.2 — Пример 2. Введенная пользователем последовательность из 10 чисел

Своя последовательность

☒ 1 разряд
 ☐ 2 разряда
 ☐ 3 разряда

2  
2  
2  
2  
2  
2  
2  
2  
2  
2

ательности

1

Рисунок 3.3 — Пример 3. Все числа одинаковые

Своя последовательность

☒ 1 разряд
 ☐ 2 разряда
 ☐ 3 разряда

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

ательности

0

Рисунок 3.4 — Пример 4. Подряд идущие числа

## 4 Код программы

На Листинге 1 представлены основные методы.

Листинг 1 — Основные методы

```

1 namespace calculations {
2   class MathCalc {
3     public static double Criteria(int[] arr, int elMin, int elMax) {
4       double y = 0, p = 1.0 / (elMax - elMin);
5
6       for (int i = elMin; i < elMax; i++)
7         y += Math.Pow(arr.Count(x => x == i), 2) / p;
8       y = y / (double)arr.Length - arr.Length;
9       return ChiSquared.CDF(elMax - elMin - 1, y);
10    }
11  }
12
13  class CongruentMethod {
14    private int a { get; }
15    private int b { get; }

```

```

16     private int N { get; }
17     private int r;
18
19     public CongruentMethod(int seed) {
20         a = 1103515245;
21         b = 12345;
22         r = seed;
23     }
24
25     public int next(int start, int end){
26         r = a * r + b;
27         return (int)((uint)r >> 16) % (end - start) + start;
28     }
29 }
30 }
31
32 namespace lab03 {
33     public partial class Form1 : Form {
34         ...
35         private void _fillAlg(int start, int end, string filename, string field, string fieldK){
36             int[] arr = new int[1000];
37             CongruentMethod alg = new CongruentMethod(rnd.Next());
38
39             for (int i = 0; i < arr.Length; i++)
40                 arr[i] = alg.next(start, end);
41             ...
42             fillCriteria(fieldK, MathCalc.Criteria(arr, start, end));
43         }
44
45         private void _useTable(int start, int end, string filename, string field, string fieldK)
46         {
47             int[] data = new int[1000], arr = new int[1000];
48             int num = rnd.Next(0, 1000), j = num;
49
50             _getArrFromFile(filename, ref data);
51
52             for (int i = 0; i < arr.Length; i++) {
53                 arr[i] = data[j];
54                 j = (j + 1) % data.Length;
55             }
56             fillCriteria(fieldK, MathCalc.Criteria(arr, start, end));
57         }
58
59         private void processUser(){
60             int[] arr = new int[10];
61             int flag, start, end;
62             string fieldK;
63
64             try{
65                 _getArrFromUser(out flag, ref arr);

```

```

66     catch (Exception e){
67         ...
68     }
69
70     switch (flag){
71         case 1:
72             start = 0;
73             end = 10;
74             fieldK = "criteriaO1";
75             break;
76         case 2:
77             start = 10;
78             end = 100;
79             fieldK = "criteriaO2";
80             break;
81         default:
82             start = 100;
83             end = 1000;
84             fieldK = "criteriaO3";
85             break;
86     }
87     fillCriteria(fieldK, MathCalc.Criteria(arr, start, end));
88 }
89 }
90 }

```