

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 дисциплине "Моделирование"

Тема <u>Генерация случайных чисел</u>
Студент <u>Мальшев И. А.</u>
Группа <u>ИУ7-71Б</u>
Оценка (баллы)
Праполаваталь : Рупаков И В

1 | Задание

Написать программу, которая генерирует псевдослучайную последовательность одноразрядных, двухразрядных и трёхразрядных целых чисел с использование табличного и алгоритмического способа.

Для каждой сгенерированной последовательности чисел вычислить собственный количественный критерий оценки случайности.

Предусмотреть ввод десяти чисел для проверки работы программы.

2 Решение

2.1 Теоретическая часть

Среди способов получения последовательности случайных чисел различают табличный и алгоритмический.

Табличный способ

Табличные генераторы случайных чисел в качестве источника случайных чисел используют специальным образом составленные таблицы, содержащие проверенные некоррелированные, то есть никак не зависящие друг от друга, цифры. Так как цифры в таблице не зависят друг от друга, то таблицу можно обходить разными способами, например, сверху вниз, или справа налево, или, скажем, можно выбирать цифры, находящиеся на четных позициях.

Достоинство данного метода в том, что он дает действительно случайные числа, так как таблица содержит проверенные некоррелированные цифры. Недостатки метода: для хранения большого количества цифр требуется много памяти; большие трудности порождения и проверки такого рода таблиц, повторы при использовании таблицы уже не гарантируют случайности числовой последовательности, а значит, и надежности результата.

Алгоритмический способ

Числа, генерируемые с помощью этого способа, всегда являются псевдослучайными (или квазислучайными), то есть каждое последующее сгенерированное число зависит от предыдущего:

$$r_{i+1} = f(r_i) \tag{2.1}$$

Последовательности, составленные из таких чисел, образуют петли, то есть обязательно существует цикл, повторяющийся бесконечное число раз. Повторяющиеся циклы называются **периодами**.

Достоинством данного способа является быстродействие: генераторы практически не требуют ресурсов памяти, компактны. Недостатки: числа нельзя в полной мере назвать случайными, поскольку между ними имеется зависимость, а также наличие периодов в последовательности квазислучайных чисел.

Выбранный алгоритмический способ

В качестве используемого метода генерации последовательности случайных чисел был выбран линейный конгруэнтный метод. Суть данного метода заключается в вычислении последовательности случайных чисел, полагая:

$$X_{n+1} = (aX_n + c) \mod m, \tag{2.2}$$

где X_{n+1} – это следующее число в последовательности, a – множитель, причём $0 \le a < m, c$ – приращение, причём $0 \le c \le m$, m – натуральное число, относительно которого вычисляется остаток от деления, причём $m \ge 2$.

При выборе значения m необходимо учитывать следующие условия:

- Данное число должно быть довольно большим, так как период не может иметь более m элементов;
- Данное значение должно быть таким, чтобы случайные значения вычислялись быстро.

Для улучшения статистических свойств числовой последовательности во многих генераторах используется только часть битов результата. В качестве констант в данной работе используются следующие значения:

- a = 1103515245
- \bullet = 12345
- $m = 2^{32}$

Критерий оценки случайности

В качестве критерия было предложено следующее: пусть имеется последовательность a_i . Берется разность между элементами последовательности $a_{i+1} - a_i$, которая образует последовательность элементов s_i . Если средняя вероятность появления элемента последовательности чисел s_i примерно равна $\frac{1}{n}$, где n – число элементов последовательности s_i , то такая последовательность является случайной.

2.2 Листинг

Далее представлен фрагмент программы, выполняющий поставленное задание.

```
internal class LCGenerator
2
  {
3
    long curElem = 1;
4
    long a, c, m;
5
    public LCGenerator(long a, long c, long m)
      this.a = a;
      this.c = c;
10
       this.m = m;
    }
11
12
13
    public long Seed
14
15
      get => curElem;
16
       set => curElem = value;
17
18
19
    public IEnumerable <int > GetRandomSequence(int count, int requiredDigits)
20
21
      List<int> res = new List<int>();
22
```

```
23
       int requiredDigitsDivider = (int)Math.Pow(10, requiredDigits);
24
       int minAppendValue = requiredDigitsDivider / 10 - 1;
25
       int addedElements = 0;
26
27
      for (int i = 0; i < count; i++)</pre>
28
29
         curElem = (curElem * a + c) % m;
30
         var elem = curElem % requiredDigitsDivider;
31
         if (elem >= minAppendValue)
32
           res.Add((int)elem);
33
         else
34
           addedElements --;
      }
35
36
37
      return res;
38
    }
39|}
40
41 internal static class Criteria
42 {
43
    public static double TestCriteria(IEnumerable<int> sequence)
44
45
      List<int> s = new List<int>();
46
      for (int i = 0; i < sequence.Count() - 1; i++)</pre>
47
48
49
         var elem = sequence.ElementAt(i + 1) - sequence.ElementAt(i);
50
         if (!s.Contains(elem))
51
52
           s.Add(elem);
      }
53
54
55
      int sCount = s.Count();
56
57
      int[] references = new int[sCount];
58
59
      for (int i = 0; i < sCount; i++)</pre>
60
         for (int j = 0; j < sequence.Count() - 1; <math>j++)
61
         {
62
           var elem = sequence.ElementAt(j + 1) - sequence.ElementAt(j);
63
64
           if (s[i] == elem)
65
             references[i] += 1;
66
         }
67
      double res = references.Select(x => x / (double)(sequence.Count() - 1)).Sum()
68
          / sCount; ;
69
      return res;
70
    }
71 }
```

2.3 Результаты работы

На рисунках 2.1-2.2 представлен пользовательский интерфейс программы в исходном состоянии.

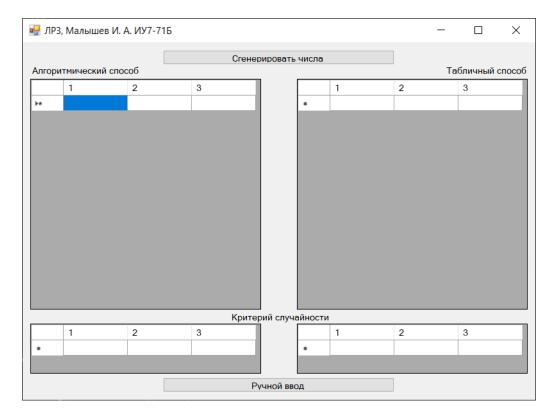


Рис. 2.1: Пользовательский интерфейс программы до генерации чисел.

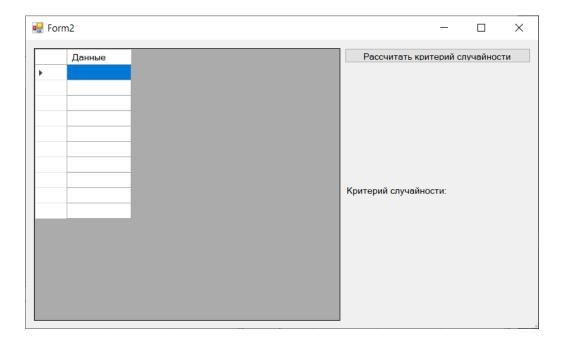


Рис. 2.2: Пользовательский интерфейс программы до ввода чисел.

2.3.1 Пример генерации чисел

На рисунках 2.3 представлен пример результатов работы программы с указанными данными.

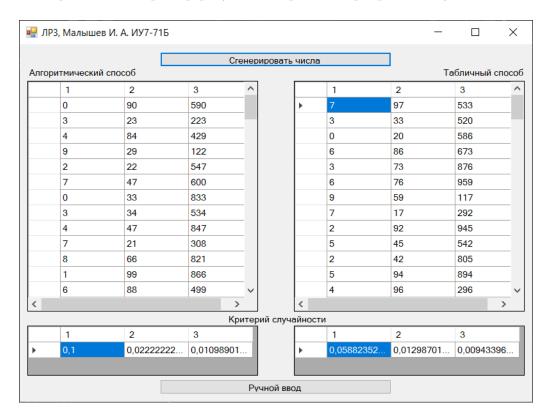


Рис. 2.3: Исходные данные и результат.

2.3.2 Примеры ручного ввода

На рисунках 2.4-2.6 представлен пример результатов работы программы с указанными данными.

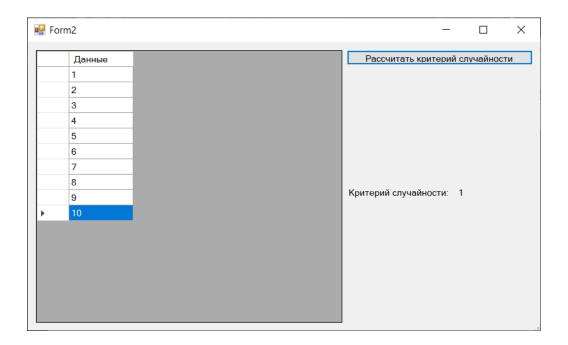


Рис. 2.4: Числа 1..10.

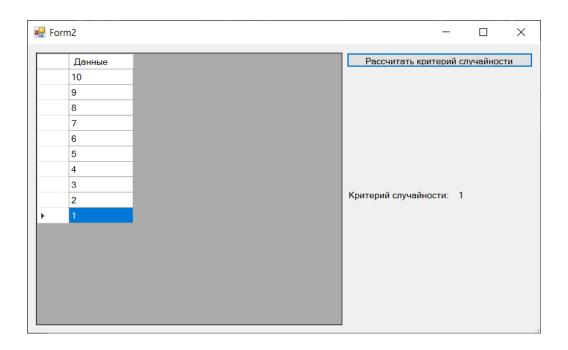


Рис. 2.5: Числа 1..10 в обратном порядке.

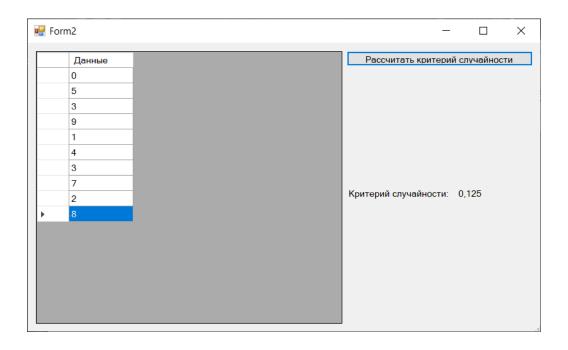


Рис. 2.6: Случайные данные.