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**

ОТЧЕТ
по лабораторной работе №3

Название: Генерация случайных чисел

Дисциплина: Моделирование

Студент	ИУ7-74Б		Д.В.Жабин
	(Группа)	(Подпись,дата)	(И.О.Фамилия)
Преподаватель			И.В.Рудаков
		(Подпись,дата)	(И.О.Фамилия)

Москва, 2022

Задание

Необходимо сгенерировать последовательности из 1000 целых чисел (одноразрядных, двухразрядных, трёхразрядных) алгоритмическим и табличным способом.

Определить и вывести на экран значения критерия оценки случайности этих последовательностей.

Также дать возможность пользователю ввести последовательность чисел, для неё также определить критерий случайности и вывести на экран.

Способы получения последовательностей случайных чисел

Выделяют 3 основных способа:

1. Аппаратный – случайные числа вырабатываются специальной электронной приставкой, то есть генератором случайных чисел. Как правило это практически любое внешнее устройство компьютера. Реализация этого способа не требует дополнительных вычислительных операций по выработке чисел, а необходима только операция обращения к этому внешнему устройству. Физическая основа - как правило шум на проводниковых и полупроводниковых приборах. Состоит из:
   - Источник шума
   - Ключевая схема
   - Формировщик импульсов
   - Пересчётная схема
2. Табличный – формируется таблица и записывается в память. Числа заранее проверены на случайность и последовательность можно многократно использовать. Недостаток – количество чисел ограничено.
3. Алгоритмический – однократная проверка, можно многократно воспроизводить последовательность, относительно малое место в оперативной памяти, не используются внешние устройства. Недостаток – запас чисел ограничен периодом, требуются затраты машинного времени.

Линейный конгруэнтный метод

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

$$\begin{gathered} \mathrm{m} > 0, \text{ модуль} \\ 0 \leq \mathrm{a} \leq \mathrm{m}, \text{ множитель} \\ 0 \leq \mathrm{c} \leq \mathrm{m}, \text{ приращение} \\ 0 \leq X_0 \leq \mathrm{m}, \text{ начальное число} \end{gathered}$$

Последовательность случайных чисел генерируется при помощи формулы:

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

Критерий случайности последовательности чисел

В данной лабораторной работе используется следующий критерий оценки случайности:

$$C = \sum_{i=1}^{N} -p_i * \log_N p_i \tag{2}$$

где N – длина последовательности чисел, p_i – отношение количества вхождений очередного числа x_i в последовательность к её длине N.

Таким образом, чем больше количество вхождений числа в последовательность (freq), тем ближе значение ΔC на очередном шаге к $\frac{freq}{N}$. В случае, когда $freq = N$ (последовательность состоит из одинаковых чисел), $\log_N p_i$ окажется равным 0, а следовательно, и значение критерия случайности C = 0.

Текст программы

В листинге 1 приведен код разработанной программы.

Листинг 1: Текст программы

```
from PyQt5 import QtWidgets, uic
from PyQt5.QtWidgets import QTableWidgetItem
from math import log
from itertools import islice
import sys

STEP = 10

class Window(QtWidgets.QMainWindow):
  def __init__(self):
    QtWidgets.QWidget.__init__(self)
    uic.loadUi("main.ui", self)
    self.fill_alg.clicked.connect(lambda: fillAlgNums(self))
    self.fill_table.clicked.connect(lambda: fillTableNums(self))
    self.manual_input.returnPressed.connect(lambda: calcManual(self))
    self.meas_alg_1.setReadOnly(True)
    self.meas_alg_2.setReadOnly(True)
    self.meas_alg_3.setReadOnly(True)
    self.meas_table_1.setReadOnly(True)
    self.meas_table_2.setReadOnly(True)
    self.meas_table_3.setReadOnly(True)
    self.meas_manual.setReadOnly(True)
    self.alg_table.setEditTriggers(QtWidgets.QAbstractItemView.
       NoEditTriggers)
    self.table_table.setEditTriggers(QtWidgets.QAbstractItemView.
       NoEditTriggers)
    self.line_num = 0

    for i in range(10):
      self.alg_table.insertRow(i)
    for i in range(10):
      self.table_table.insertRow(i)
```

```
def randomCriteria(seq):
  N = len(seq)
  if N == 0:
    return 0

  numsFreq = dict()
  for x in seq:
    if x not in numsFreq.keys():
      numsFreq.update({x: 1})
    else:
      numsFreq[x] += 1

  crit = 0
  for x in numsFreq.keys():
    p = numsFreq[x] / N
    crit -= p * log(p, N)
  return crit

class LinearCongruent:
  m = 2**32
  a = 1664525
  c = 1013904223
  _cur = 1

  def next(self):
    self._cur = (self.a * self._cur + self.c) % self.m
    return self._cur

def fillAlgNums(win):
  table = win.alg_table
  one_digit = [generator.next() % 9 + 1 for i in range(1000)]
  two_digits = [generator.next() % 90 + 10 for i in range(1000)]
  three_digits = [generator.next() % 900 + 100 for i in range(1000)]

  for i in range(10):
    item = QTableWidgetItem(str(one_digit[i * STEP]))
    table.setItem(i, 0, item)
  for i in range(10):
    item = QTableWidgetItem(str(two_digits[i * STEP]))
    table.setItem(i, 1, item)
  for i in range(10):
    item = QTableWidgetItem(str(three_digits[i * STEP]))
    table.setItem(i, 2, item)

  crit_one = randomCriteria(one_digit)
  crit_two = randomCriteria(two_digits)
  crit_three = randomCriteria(three_digits)
  win.meas_alg_1.setText('{:.4%}'.format(crit_one))
  win.meas_alg_2.setText('{:.4%}'.format(crit_two))
  win.meas_alg_3.setText('{:.4%}'.format(crit_three))

def fillTableNums(win):
  table = win.table_table
```

```
    numbers = set()
    with open('nums.txt') as file:
      lines = islice(file, win.line_num, None)
      for l in lines:
      numbers.update(set(l.split(" ")[1:-1]))
      win.line_num += 1
      if len(numbers) > 3000:
        break
      numbers.remove("")
      numbers = list(numbers)[:3000]

    one_digit = [int(i) % 9 + 1 for i in numbers[:1000]]
    two_digits = [int(i) % 90 + 10 for i in numbers[1000:2000]]
    three_digits = [int(i) % 900 + 100 for i in numbers[2000:3000]]

    for i in range(10):
      item = QTableWidgetItem(str(one_digit[i * STEP]))
      table.setItem(i, 0, item)
    for i in range(10):
      item = QTableWidgetItem(str(two_digits[i * STEP]))
      table.setItem(i, 1, item)
    for i in range(10):
      item = QTableWidgetItem(str(three_digits[i * STEP]))
      table.setItem(i, 2, item)

    crit_one = randomCriteria(one_digit)
    crit_two = randomCriteria(two_digits)
    crit_three = randomCriteria(three_digits)
    win.meas_table_1.setText(' {:.4%}'.format(crit_one))
    win.meas_table_2.setText(' {:.4%}'.format(crit_two))
    win.meas_table_3.setText(' {:.4%}'.format(crit_three))

  def calcManual(win):
    input = win.manual_input.text().split(" ")
    seq = []
    for x in input:
      try:
        int(x)
      except ValueError:
        continue
      else:
        seq.append(int(x))

    crit = randomCriteria(seq)
    win.meas_manual.setText(' {:.4%}'.format(crit))

  if __name__ == "__main__":
    generator = LinearCongruent()
    app = QtWidgets.QApplication(sys.argv)
    w = Window()
    w.show()
    sys.exit(app.exec_())
```

Примеры работы

На рисунках 1-3 приведены примеры работы программы с разным вводом пользователя.

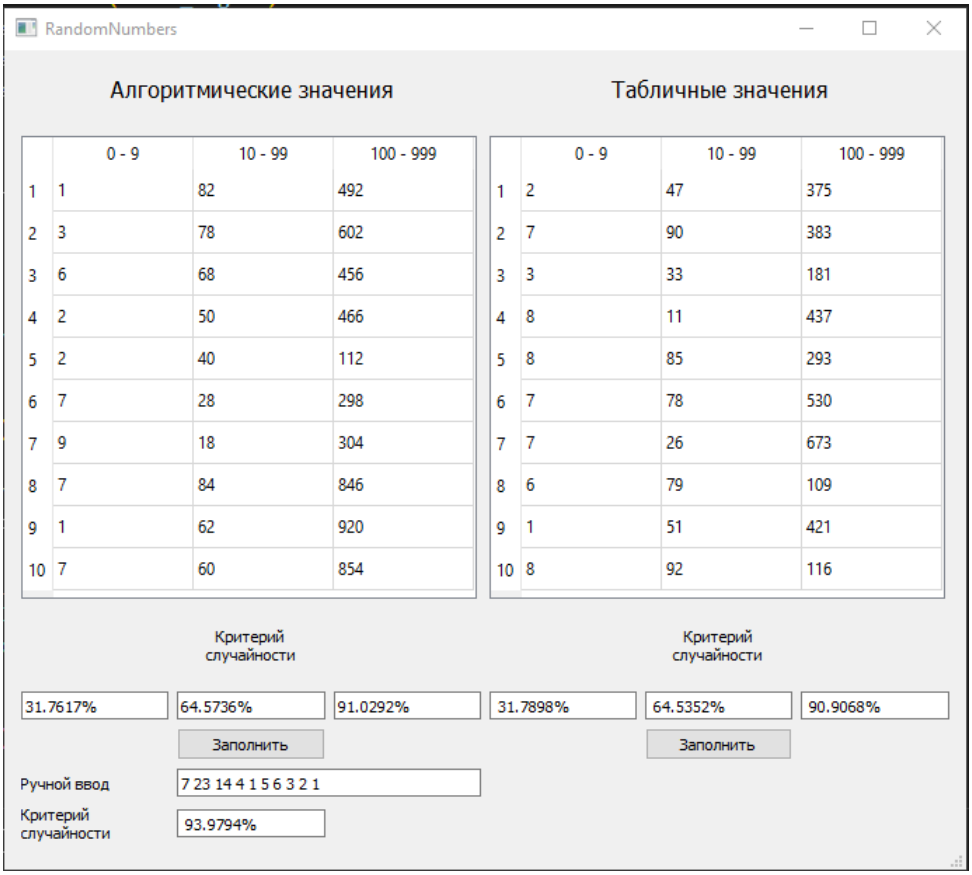

Рис. 1: Пример 1. Разные числа

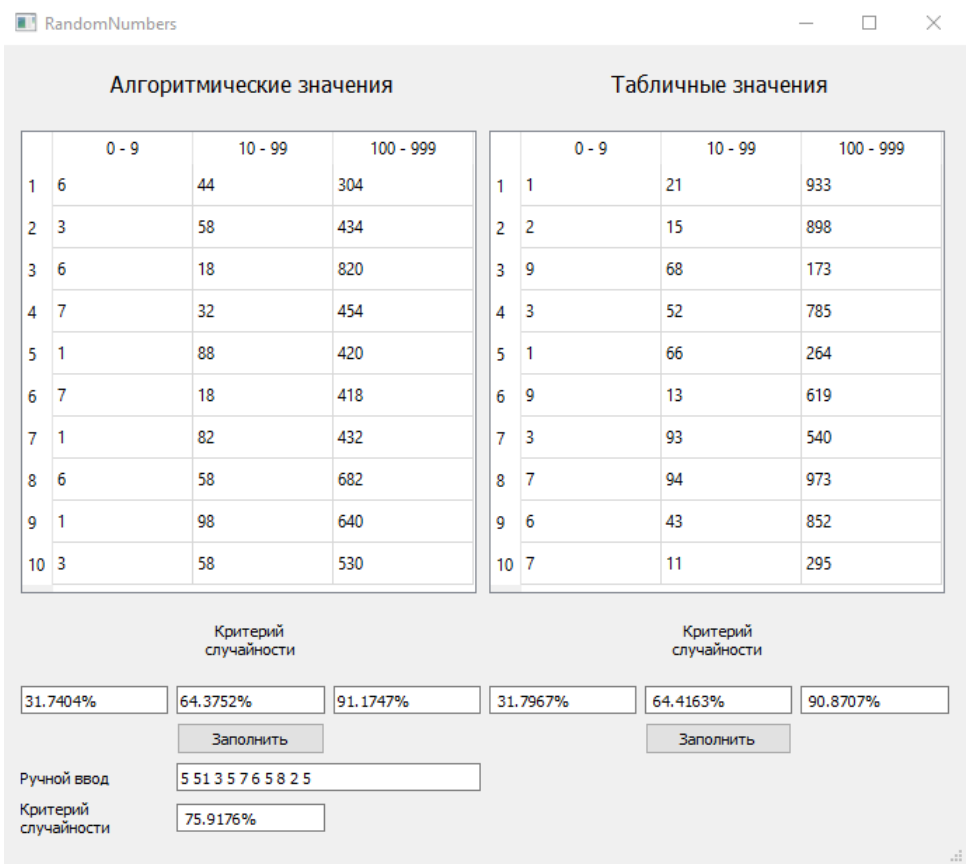

Рис. 2: Пример 2. Несколько повторяющихся чисел

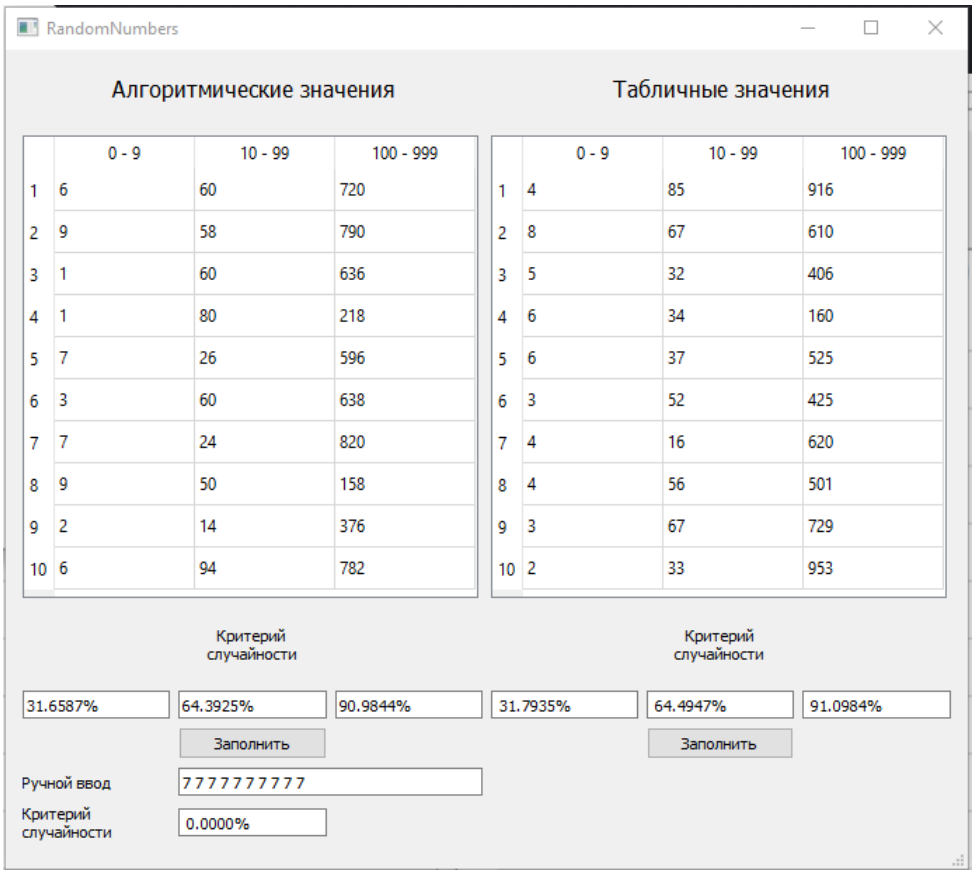

Рис. 3: Пример 3. Одинаковые числа