Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

**ОТЧЕТ**

по лабораторной работе №3

по курсу «Защита информации в сети Internet»

на тему «Генераторы псевдослучайных последовательностей»

Выполнили  
студенты группы 18ВВ1:

Ручкин М.А.

Коновалова Д.А.

Дубинин А.В.

Приняли:

к.т.н., доцент Дубравин А.В.

к.т.н., доцент Карамышева Н.С.

2020

**Цель работы:** изучить методы генерации случайных чисел, разработать генератор случайных чисел, используя один из методов.

**Задания:** Разработать генератор случайных чисел, используя метод в соответствии с номером варианта. Исследовать качество генератора, путем оценки распределения генерируемых чисел. Для этого необходимо разделить весь диапазон генерируемых чисел на 10 равных интервалов и подсчитать количество чисел, попадающих в каждый интервал.

В качестве начального значения следует выбирать текущее время в формате Unix time (можно получить, используя функцию time\_t time(time\_t\* timer)).

Для линейного конгруэнтного метода значение m всегда должно быть 231 -1.

*Таблица: Источники энтропии*

|  |  |
| --- | --- |
| № Варианта | Метод генерации случайных чисел |
| 8 | Линейный конгруэнтный метод, a = 48271, c = 9 |

Листинг:

#include <iostream>

#include <time.h>

#include <locale.h>

#define TABLE\_SIZE 8

using namespace std;

double congruential(int &); // прототип функции генерации псевдослучайных чисел

bool nullTest(double\* a, int size);

void interval\_count(double\* mass, int position, int\* counter);

int main()

{

setlocale(LC\_ALL, "Rus");

const int number\_numbers = 100; // количество псевдослучайных чисел

double\* mass\_rand = new double[TABLE\_SIZE];

for (int i = 0; i < TABLE\_SIZE; i++)

mass\_rand[i] = 0;

int\* mass\_count = new int[10];

for (int i = 0; i < 10; i++)

mass\_count[i] = 0;

time\_t timec = time(0);

int x0 = timec; // начальное значение

cout << "\n";

int k = 0;

int sch = 1;

for (int i = 0; i < number\_numbers; i++)

{

k = rand() % TABLE\_SIZE;

while (k == TABLE\_SIZE)

k = rand() % TABLE\_SIZE;

if (mass\_rand[k] == 0)

mass\_rand[k] = congruential(x0);

else

{

cout << "["<< sch << "] " << mass\_rand[k] << "\n";

sch++;

interval\_count(mass\_rand, k, mass\_count);

mass\_rand[k] = congruential(x0);

}

}

while (!nullTest(mass\_rand, TABLE\_SIZE))

{

k = rand() % TABLE\_SIZE;

while (k == TABLE\_SIZE)

k = rand() % TABLE\_SIZE;

if (mass\_rand[k] != 0)

{

cout << "[" << sch << "] " << mass\_rand[k] << "\n";

sch++;

interval\_count(mass\_rand, k, mass\_count);

mass\_rand[k] = 0;

}

}

cout << "\n Распределение \n";

float n = 0;

for (int i = 0; i < 10; i++)

{

cout << "[" << n << " - " << n + 0.1 << "]" << " -> " << mass\_count[i] << "\n";

n = n + 0.1;

}

delete mass\_rand;

delete mass\_count;

cin.peek();

return 0;

}

bool nullTest(double\* a, int size) // функция проверки таблицы на не пустые ячейки

{

int count = 0;

for (int i = 0; i < size; i++)

{

if (a[i] != 0)

count++;

}

if (count == 0)

return true;

else

return false;

}

void interval\_count(double\* mass, int position, int\* counter) // функция подсчета частоты в интервале

{

if ((0 < mass[position]) && (mass[position] < 0.1))

counter[0]++;

else if ((0.1 < mass[position]) && (mass[position] < 0.2))

counter[1]++;

else if ((0.2 < mass[position]) && (mass[position] < 0.3))

counter[2]++;

else if ((0.3 < mass[position]) && (mass[position] < 0.4))

counter[3]++;

else if ((0.4 < mass[position]) && (mass[position] < 0.5))

counter[4]++;

else if ((0.5 < mass[position]) && (mass[position] < 0.6))

counter[5]++;

else if ((0.6 < mass[position]) && (mass[position] < 0.7))

counter[6]++;

else if ((0.7 < mass[position]) && (mass[position] < 0.8))

counter[7]++;

else if ((0.8 < mass[position]) && (mass[position] < 0.9))

counter[8]++;

else if ((0.9 < mass[position]) && (mass[position] < 1))

counter[9]++;

}

double congruential(int &x) // функция генерации псевдослучайных чисел

{

const unsigned int m = 2147483647, // m = 2^31 - 1 = 2.147.483.647

a = 48271, // множитель

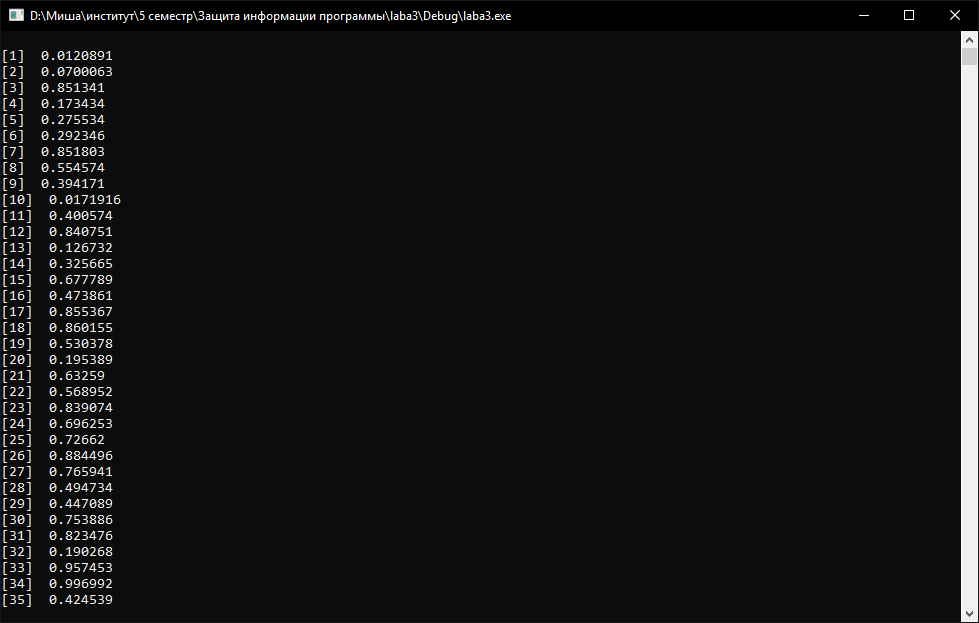
c = 8; // инкрементирующее значение

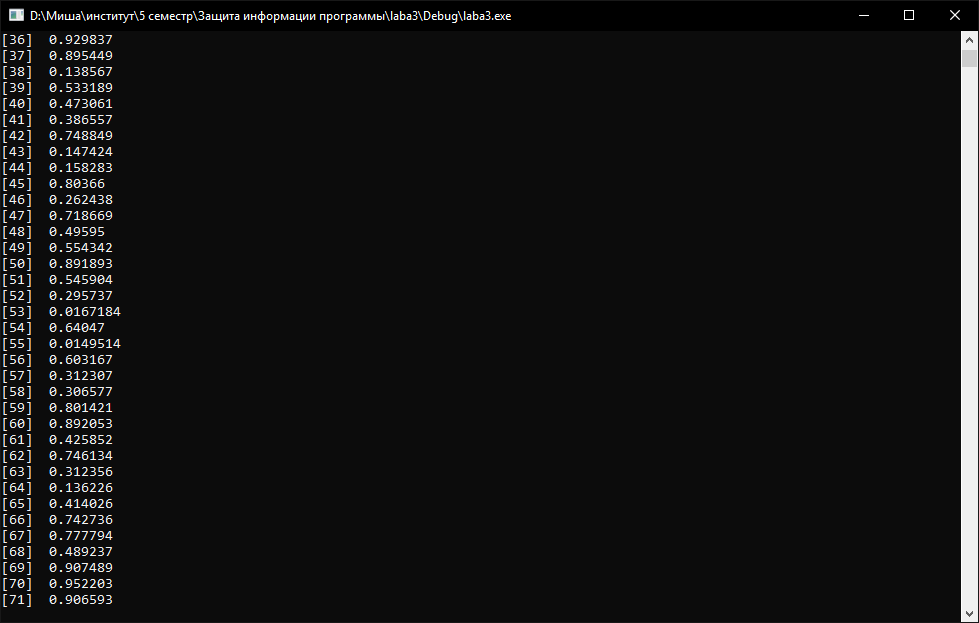
x = ((a \* x) + c) % m; // формула линейного конгруэнтного метода генерации псевдослучайных чисел

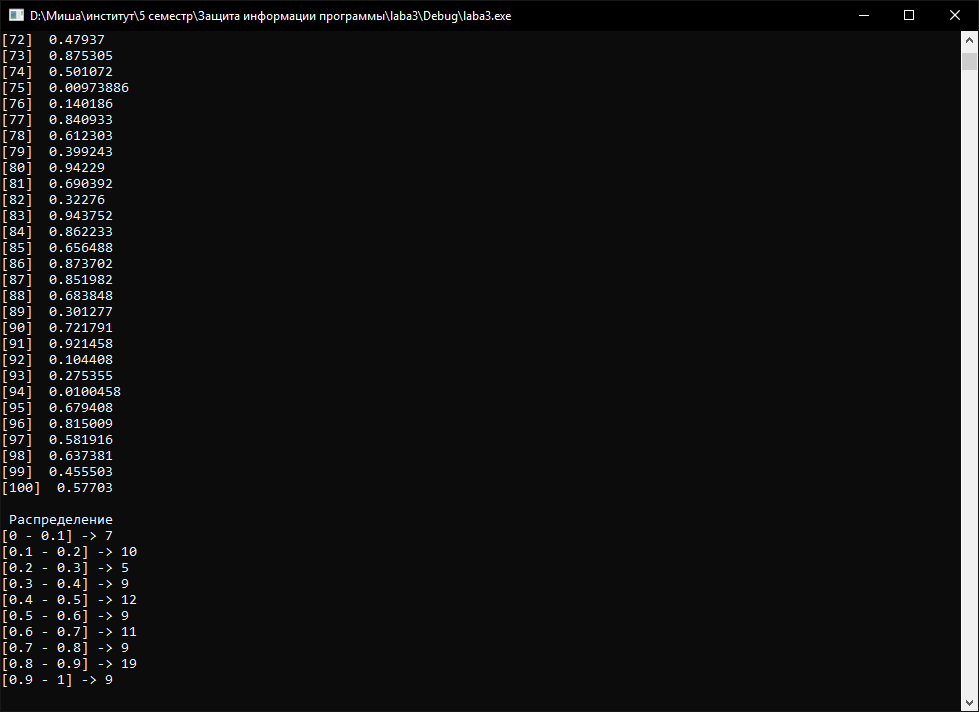
return (x / double(m));

}

Вывод программы:







Вывод: в ходе лабораторной работы разработали генератор случайных чисел, используя линейный конгруэнтный метод.