МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра «Электроника и сети ЭВМ»

**Линейный конгруэнтный генератор**

по дисциплине

**Методы и средства защиты информации**

РУКОВОДИТЕЛЬ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Семашко А.В.**

(подпись) (фамилия, и.,о.)

СТУДЕНТ:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Сироткин Ф.А.**

(подпись) (фамилия, и.,о.)

**16-ИСТв-1**

(шифр группы)

Работа защищена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_

С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Нижний Новгород 2018

ЦЕЛЬ РАБОТЫ

Изучение методов генерации последовательностей псевдослучайных чисел с равномерным законом распределения вероятностей.

ОПИСАНИЕ ЛИНЕЙНОГО КОНГРУЭНТНОГО МЕТОДА

Линейный конгруэнтный метод был предложен Д.Г. Лехмером (D.Н.Lehmer) в 1949 году и в настоящее время является наиболее популярным.

Линейный конгруэнтный метод – один из методов генерации псевдослучайных чисел. Применяется в простых случаях и не обладает криптографической стойкостью. Входит в стандартные библиотеки различных компиляторов.

ИСХОДНЫЕ ДАННЫЕ

множитель

приращение

начальное значение

ПРИНЦИП РАБОТЫ КОНГРУЭНТНОГО МЕТОДА

Очередное число ni псевдослучайной последовательности вычисляется по формуле

**ni + 1 = (ani + c) mod m, n ≥ 0**

где m – модуль, 0 < m;

а – множитель, 0 ≤ a < m;

с – приращение, 0 ≤ с < m;

n0 – начальное значение, 0 ≤ n0 < m;

т.е.ni + 1 равно остатку от деления (аni + с) на значение модуля m.

Полученная последовательность называется линейной конгруэнтной последовательностью. Например, для m=10 и n0=а=с=7 получим последовательность 7, 6, 9, 0, 7, 6, 9, 0, …

Отсюда видно, что значения чисел в последовательности периодически повторяются. Это свойство является общим для всех последовательностей вида ni + 1 = ψ(ni), где ψ преобразует конечное множество (множество состояний машинного слова) само в себя .

Если на некотором шаге nL совпадает с ранее полученным те и все последующие числа последовательности n0, n1, …, nl, … nL, …повторяются nL + 1 = nl+1, nL + 2 = nl + 2 …, тогда период последовательности P = L - l, а числа n0, n1, … nL - 1 образуют отрезок апериодичности.

ИСХОДНЫЕ ТЕКСТЫ ПРОГРАММЫ

Файл **Program.cs**:

using System;

using System.Windows.Forms;

namespace LinearCongruentialGenerator

{

static class Program

{

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

/\*\*

\* Линейный конгруэнтный генератор

\*/

class RandomNumberGenerator

{

private long a; // Множитель, где 0 <= a < m

private long c; // Приращение, где 0 <= с < m

private long m; // Модуль, где 0 < m (2^32)

private double ni; // Случайная величина

// Конструктор

public RandomNumberGenerator(long \_a, long \_c, long \_m)

{

a = \_a; c = \_c; m = \_m;

ni = DateTime.Now.Ticks % m; // Число тактов от текущей даты

}

/\*\*

\* Получение случайной величины

\*/

public double Next()

{

ni = ((a \* ni) + c) % m;

return ni;

}

/\*\*

\* Получение случайной величины с учетом максимального значения

\*/

public double Next(long maxValue)

{

return Next() % maxValue;

}

}

static class Debug

{

public static void debugMsg(string message)

{

const string caption = "Debug";

const MessageBoxButtons buttons = MessageBoxButtons.OK;

MessageBox.Show(message, caption, buttons);

}

}

}

Файл **Form1.cs**:

using System;

using System.Windows.Forms;

namespace LinearCongruentialGenerator

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Calc\_Click(object sender, EventArgs e)

{

const int min = 50;

const int max = 100;

var rndn = new Random();

N.Text = rndn.Next(min, max).ToString();

if (N.Text != "" && NMAX.Text != "" && A.Text != "" && C.Text != "" && M.Text != "")

{

// Получение входных параметров

var n = Convert.ToInt64(N.Text); // Количество генерируемых чисел

var nmax = Convert.ToInt64(NMAX.Text); // Диапазон

var a = Convert.ToInt64(A.Text); // Множитель

var c = Convert.ToInt64(C.Text); // Приращение

var m = Convert.ToInt64(M.Text); // Модуль

if (n > 0 && nmax > 0 && a >= 0 && a < m && c >= 0 && c < m && m > 0)

{

// Экземпляр ЛКГ

var rng = new RandomNumberGenerator(a, c, m);

// Диапазон значений по оси Y от 0 до 1

Histogram.ChartAreas[0].AxisX.Minimum = 0;

Histogram.ChartAreas[0].AxisX.Maximum = nmax;

var ni = rng.Next(nmax);

Histogram.Series["RND"].Points.AddXY(ni, n);

}

else

{

const string caption = "Ошибка!";

const MessageBoxButtons buttons = MessageBoxButtons.OK;

const string message = "Параметры введены не правильно.";

MessageBox.Show(message, caption, buttons);

}

}

else

{

const string caption = "Ошибка!";

const MessageBoxButtons buttons = MessageBoxButtons.OK;

const string message = "Один из параметров не заполнен.";

MessageBox.Show(message, caption, buttons);

}

}

private void N\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

private void NMAX\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

private void A\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

private void C\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

private void M\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

private void Clear\_Click(object sender, EventArgs e)

{

Histogram.Series["RND"].Points.Clear();

}

}

}

СНИМКИ ЭКРАНА



Рисунок 1 – Выполнение программы

ЗАКЛЮЧЕНИЕ

Изучен метод генерации последовательностей псевдослучайных чисел с равномерным законом распределения вероятностей. Реализован программный продукт на языке программирования высокого уровня C#, позволяющий генерировать последовательность псевдослучайных чисел и строить гистограмму на основе полученных данных