МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационной безопасности

**Отчет**

по лабораторной работе №5

на тему: «**Исследование основных вероятностных характеристик последовательностей, вырабатываемых генераторами псевдослучайных чисел**»

по дисциплине «Информационная безопасность»

Выполнили: Кожухова О.А. Шифр: 170582

Шорин В.Д. Шифр: 171406

Институт приборостроения, автоматизации и информационных технологий

Направление: 09.03.04 «Программная инженерия»

Группа: 71-ПГ

Проверил: Еременко В.Т.

Отметка о зачете:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата «\_\_\_\_» \_\_\_\_\_\_ 2021г.

Орел, 2021 г.

**Задание**

1 Генерация последовательностей.

Используя программу предыдущей лабораторной работы, реализующую работу двух моделей генераторов, и выбранные начальные значения ячеек памяти регистра – b и накопителя – c, выработать и записать в память следующие последовательности. Длина последовательности при всех генерациях равна 1000 (N=1000).

1) С помощью ГПЧ1, используя первый вариант схемы генератора, при начальном заполнении регистра сдвига b=(01010101011) выработать последовательность

Г(1) = {gi(1)}

2) С помощью ГПЧ1, используя второй вариант схемы генератора, при том же начальном заполнении регистра выработать последовательность

Г(2) = {gi(2)}

3) С помощью ГПЧ2, используя первый вариант схемы генератора, при том же начальном заполнении регистра b начальном заполнении продвигающего накопителя c=(1100111) выработать последовательность

Г(3) = {gi(3)}

4) С помощью ГПЧ2, используя второй вариант схемы генератора, при том же начальном заполнении регистра b начальном заполнении продвигающего накопителя c выработать последовательность

Г(4) = {gi(4)}

2 Расчет величин

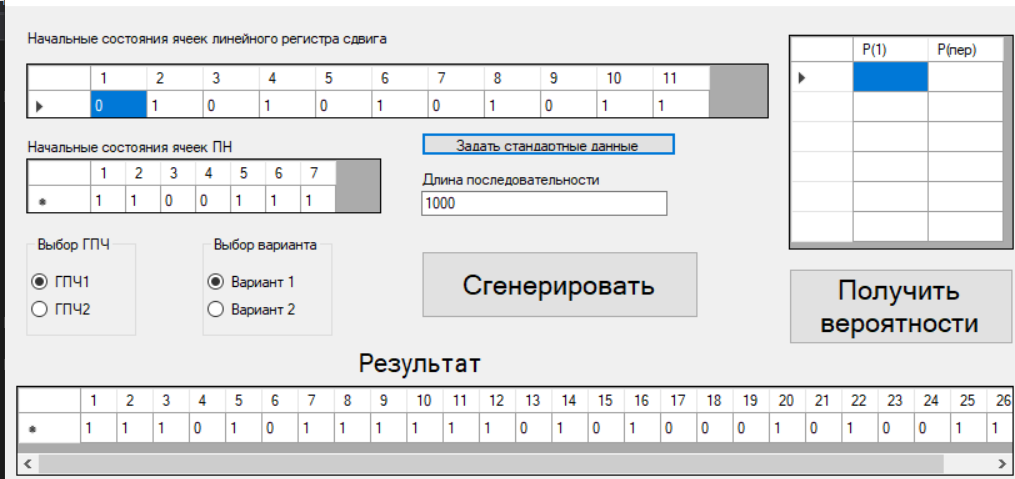
С помощью полученных последовательностей вычислить и записать в память следующие последовательности:

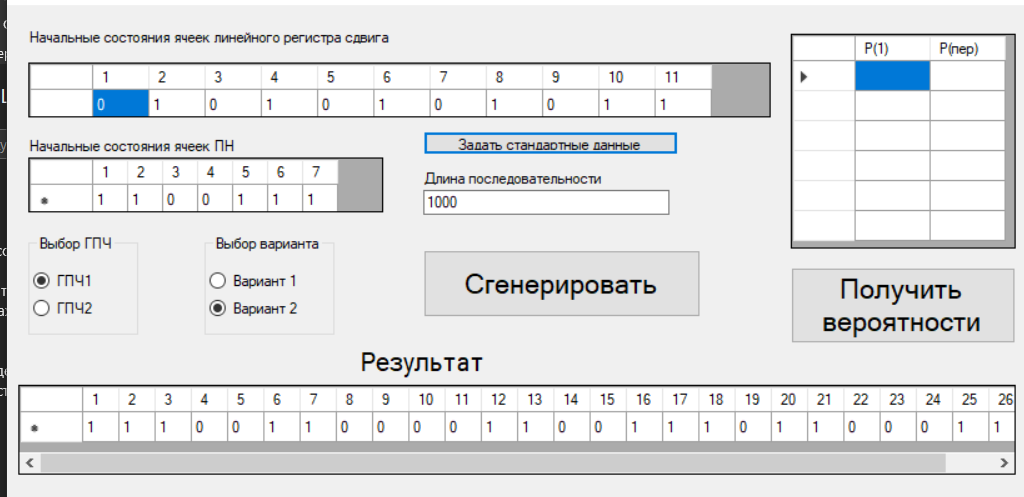
Г(1) Г(3) = {gi(1) gi(3)}

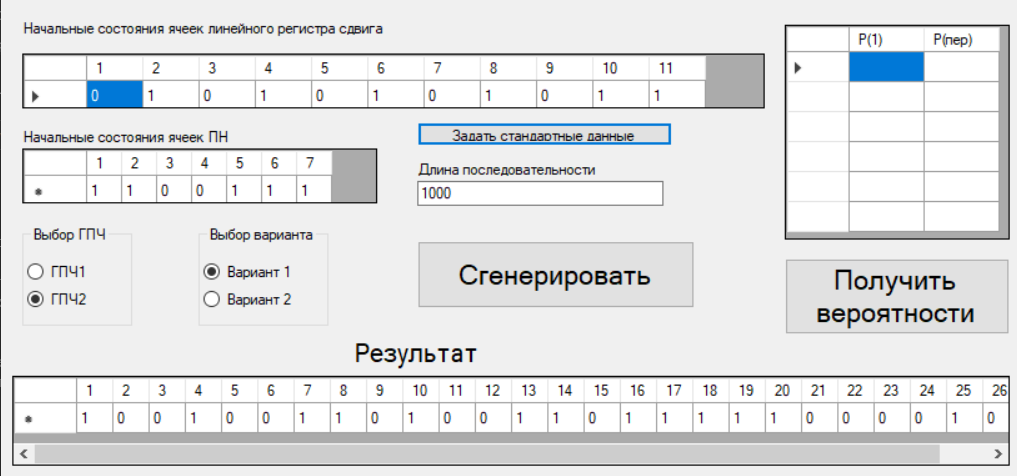
Г(2) Г(4) = {gi(2) gi(4)}

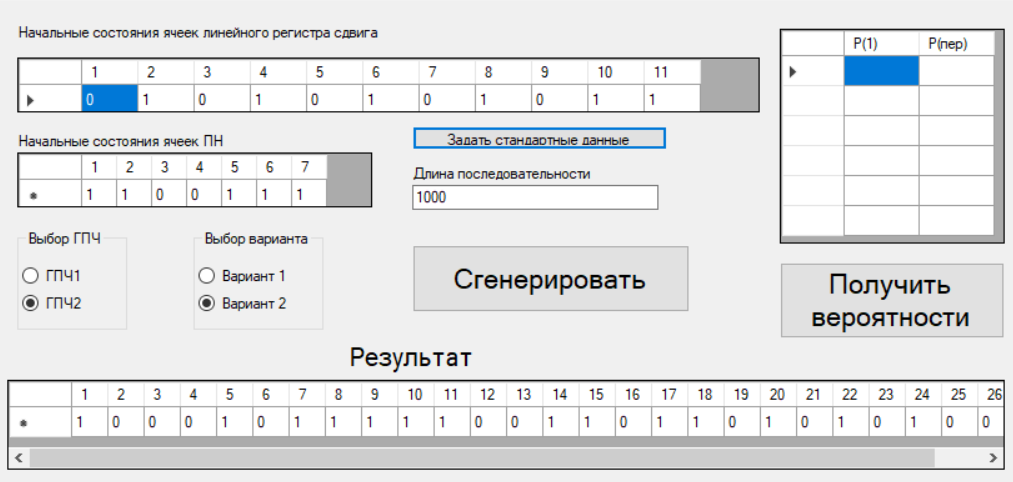
Вычислить величины P(1) и Pпер для полученных 6 последовательностей.

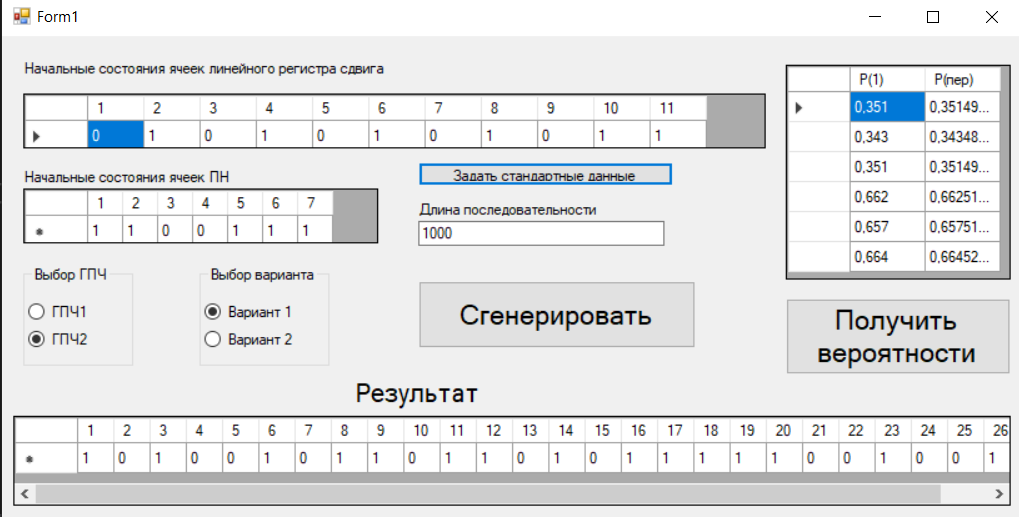
**Ход работы**











**Код**

**«Form.cs»**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace IS\_L\_4

{

public partial class Form1 : Form

{

private int[] defaultB11 = { 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1 };

private int[] defaultC7 = { 1, 1, 0, 0, 1, 1, 1 };

//private int[] defaultB11 = { 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0 };

//private int[] defaultC7 = { 1, 0, 1, 0, 1, 0, 1 };

private int[] k11\_v1 = { 2, 4, 6, 8, 11, 10, 1, 3, 5, 7, 9 };

private int[] k7\_v1 = { 3, 4, 5, 6, 7, 1, 2 };

private int[] k11\_v2 = { 1, 3, 5, 7, 9, 11, 10, 8, 4, 6, 2 };

private int[] k7\_v2 = { 3, 5, 7, 1, 4, 2, 6 };

private List<string> sequences;

private double[,] results;

public Form1()

{

InitializeComponent();

CreateTables();

sequences = new List<string>();

results = new double[6, 2];

}

private string GPC1()

{

string result = "";

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

result += dataGridView1[i, 0].Value;

}

result = AddElevenElement(result);

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

dataGridView1[i, 0].Value = result[i];

}

if (rbV1.Checked)

{

result = Switchboard(result, k11\_v1);

result = F1\_v1(result);

}

else if (rbV2.Checked)

{

result = Switchboard(result, k11\_v2);

result = F1\_v2(result);

}

return result;

}

private string GPC2()

{

string result = "";

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

result += dataGridView1[i, 0].Value;

}

result = AddElevenElement(result);

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

dataGridView1[i, 0].Value = result[i];

}

string s = "";

if (rbV1.Checked)

{

s = Summator(result);

result = "";

for (int i = 0; i < 7; i++)

{

result += dataGridView2[i, 0].Value;

}

result = PN(result, s);

for (int i = 0; i < 7; i++)

{

dataGridView2[i, 0].Value = result[i];

}

result = Switchboard(result, k7\_v1);

result = F2\_v1(result);

}

else if (rbV2.Checked)

{

s = Summator(result);

result = "";

for (int i = 0; i < 7; i++)

{

result += dataGridView2[i, 0].Value;

}

result = PN(result, s);

for (int i = 0; i < 7; i++)

{

dataGridView2[i, 0].Value = result[i];

}

result = Switchboard(result, k7\_v2);

result = F2\_v2(result);

}

return result;

}

private string PN(string s, string e)

{

string result = s.Remove(0, 1);

return result + e;

}

private string Summator(string s)

{

return (int.Parse(s[0].ToString()) ^ int.Parse(s[1].ToString()) ^ int.Parse(s[2].ToString()) ^ int.Parse(s[3].ToString()) ^

int.Parse(s[4].ToString()) ^ int.Parse(s[5].ToString()) ^ int.Parse(s[6].ToString()) ^ int.Parse(s[7].ToString()) ^

int.Parse(s[8].ToString()) ^ int.Parse(s[9].ToString()) ^ int.Parse(s[10].ToString())).ToString();

}

private void CreateTables()

{

for (int i = 0; i < 11; i++)

{

var col = new DataGridViewColumn();

col.HeaderText = (i + 1).ToString();

col.Width = dataGridView1.Width / 13;

col.CellTemplate = new DataGridViewTextBoxCell();

dataGridView1.Columns.Add(col);

if (i == 0)

{

dataGridView1.Rows.Add();

}

dataGridView1[i, 0].Value = '0';

}

for (int i = 0; i < 7; i++)

{

var col = new DataGridViewColumn();

col.HeaderText = (i + 1).ToString();

col.Width = dataGridView2.Width / 10;

col.CellTemplate = new DataGridViewTextBoxCell();

dataGridView2.Columns.Add(col);

dataGridView2[i, 0].Value = '0';

}

var column = new DataGridViewColumn();

column.HeaderText = "P(1)";

column.Width = dataGridView3.Width / 3;

column.CellTemplate = new DataGridViewTextBoxCell();

dataGridView3.Columns.Add(column);

column = new DataGridViewColumn();

column.HeaderText = "P(пер)";

column.Width = dataGridView3.Width / 3;

column.CellTemplate = new DataGridViewTextBoxCell();

dataGridView3.Columns.Add(column);

for (int i = 0; i < 6; i++)

{

dataGridView3.Rows.Add();

}

}

private string F1\_v1(string s)

{

return (int.Parse(s[0].ToString()) ^ int.Parse(s[1].ToString()) & int.Parse(s[2].ToString()) ^ int.Parse(s[3].ToString()) &

int.Parse(s[5].ToString()) & int.Parse(s[8].ToString()) ^ int.Parse(s[9].ToString()) | int.Parse(s[10].ToString())).ToString();

}

private string F2\_v1(string s)

{

return (int.Parse(s[0].ToString()) & int.Parse(s[6].ToString()) ^ int.Parse(s[2].ToString()) & int.Parse(s[3].ToString())).ToString();

}

private string F1\_v2(string s)

{

return (int.Parse(s[0].ToString()) | int.Parse(s[9].ToString()) ^ int.Parse(s[1].ToString()) | int.Parse(s[4].ToString()) &

int.Parse(s[6].ToString()) ^ int.Parse(s[10].ToString())).ToString();

}

private string F2\_v2(string s)

{

return (int.Parse(s[0].ToString()) & int.Parse(s[5].ToString()) ^ int.Parse(s[3].ToString()) & int.Parse(s[4].ToString())).ToString();

}

private string AddElevenElement(string s)

{

string res = s;

res = res.Remove(0, 1);

res += (int.Parse(res[0].ToString()) ^ int.Parse(res[8].ToString())).ToString();

return res;

}

private string Switchboard(string s, int[] positions)

{

string result = "";

for (int i = 0; i < s.Length; i++)

{

result += s[positions[i] - 1];

}

return result;

}

private void SetDefaultValues()

{

for (int i = 0; i < defaultB11.Length; i++)

{

dataGridView1[i, 0].Value = defaultB11[i];

}

for (int i = 0; i < defaultC7.Length; i++)

{

dataGridView2[i, 0].Value = defaultC7[i];

}

textBox1.Text = "1000";

}

private void dataGridView1\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

if (!dataGridView1[e.ColumnIndex, e.RowIndex].Value.Equals("1"))

{

dataGridView1[e.ColumnIndex, e.RowIndex].Value = '0';

}

}

private void dataGridView2\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

if (!dataGridView2[e.ColumnIndex, e.RowIndex].Value.Equals("1"))

{

dataGridView2[e.ColumnIndex, e.RowIndex].Value = '0';

}

}

private void button1\_Click(object sender, EventArgs e)

{

SetDefaultValues();

}

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

{

int n;

if (int.TryParse(textBox1.Text, out n))

{

string res = "";

if (rbGPC1.Checked)

{

res = GPC1();

}

else if (rbGPC2.Checked)

{

res = GPC2();

}

Random random = new Random();

while(res.Length != n)

{

res += random.Next(0, 2);

}

sequences.Add(res);

dgvResult.Columns.Clear();

for (int i = 0; i < 27; i++)

{

var col = new DataGridViewColumn();

col.Width = dgvResult.Width / 27;

col.CellTemplate = new DataGridViewTextBoxCell();

col.HeaderText = (i + 1).ToString();

dgvResult.Columns.Add(col);

dgvResult[i, 0].Value = res[i];

}

}

else

{

MessageBox.Show("Должно быть число");

}

}

private void button2\_Click(object sender, EventArgs e)

{

string s1 = "";

string s2 = "";

int n = 1000;

if (sequences.Count == 4)

{

for (int i = 0; i < 1000; i++)

{

s1 += int.Parse(sequences[0][i].ToString()) ^ int.Parse(sequences[1][i].ToString());

s2 += int.Parse(sequences[2][i].ToString()) ^ int.Parse(sequences[3][i].ToString());

}

sequences.Add(s1);

sequences.Add(s2);

for (int i = 0; i < 6; i++)

{

int sum = 0;

for (int j = 0; j < sequences[i].Length; j++)

{

sum += int.Parse(sequences[i][j].ToString());

}

double p1 = 0.001 \* sum;

results[i, 0] = p1;

double p2 = 1.0 / (n - 1) \* sum;

results[i, 1] = p2;

}

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 2; j++)

{

results[i, j] += (results[i, j] > 0.5) ? 0.073 : -0.073;

dataGridView3[j, i].Value = results[i, j];

}

}

}

else

{

MessageBox.Show("6 последовательностей должно быть");

}

}

}

}