Казанский (Приволжский) Федеральный университет
Институт Вычислительной математики и информационных технологий

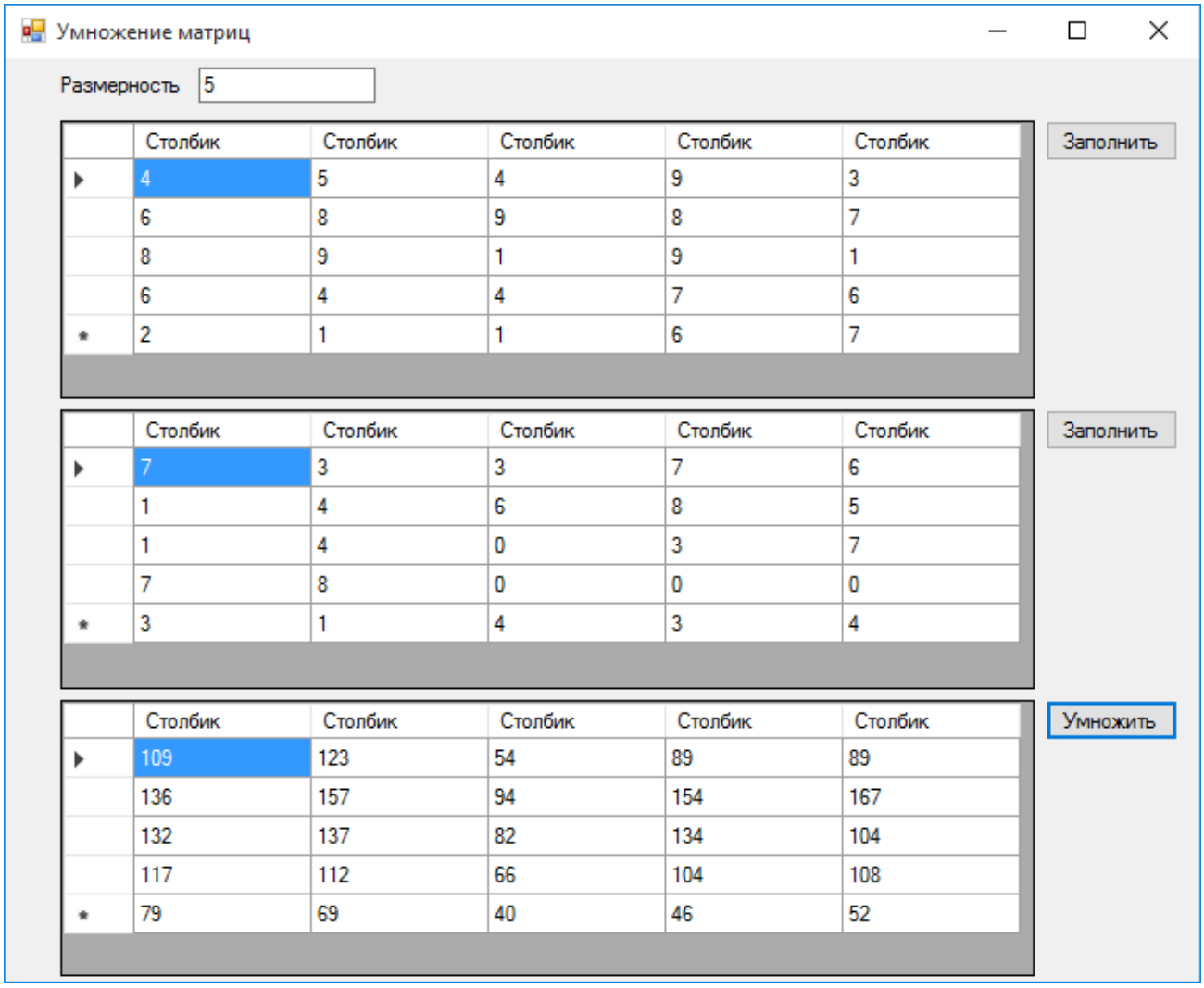
Отчёт по работам на практике ТиМП.

Студент 09-641 Десятов Александр

Казань – 2017

Перемножение матриц.

Скриншот:

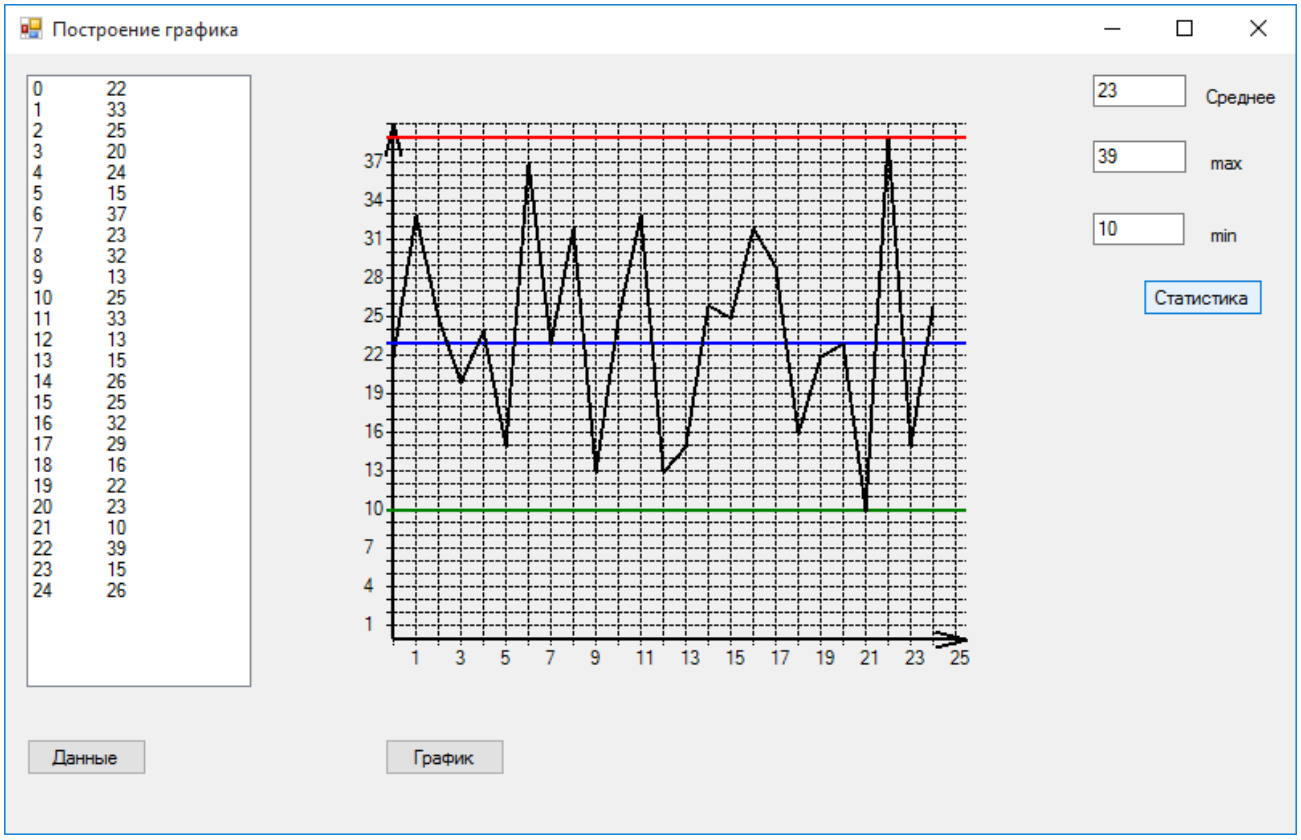



```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        int[,] A;
        int[,] B;
        int[,] C;
        int r;
        Random rnd = new Random();
        public Form1()
        {
            InitializeComponent();
        }
//заполняем первую матрицу
        private void button1_Click(object sender, EventArgs e)
        {
            dataGridView1.Columns.Clear();
            for (int i = 0; i < r; i++)
            {
                dataGridView1.Columns.Add("","Столбик");
                if(i!=0)
                    dataGridView1.Rows.Add();
            }
```

```
            A = new int[r, r];
            for (int i = 0; i < r; i++)
                for (int j = 0; j < r; j++)
                {
                    A[i, j] = rnd.Next(10);
                    dataGridView1.Rows[i].Cells[j].Value = Convert.ToString (A[i, j]);
                }
        }
//Заполняем вторую матрицу
        private void button2_Click(object sender, EventArgs e)
        {
            dataGridView2.Columns.Clear();
            for (int i = 0; i < r; i++)
            {
                dataGridView2.Columns.Add("", "Столбик");
                if (i != 0)
                    dataGridView2.Rows.Add();
            }
            B = new int[r, r];
            for (int i = 0; i < r; i++)
                for (int j = 0; j < r; j++)
                {
                    B[i, j] = rnd.Next(10);
                    dataGridView2.Rows[i].Cells[j].Value = Convert.ToString(B[i, j]);
                }
        }
//Умножаем матрицы
        private void button3_Click(object sender, EventArgs e)
        {
            dataGridView3.Columns.Clear();
            for (int i = 0; i < r; i++)
            {
                dataGridView3.Columns.Add("", "Столбик");
                if (i != 0)
                    dataGridView3.Rows.Add();
            }
            C = new int[r, r];
            for(int i = 0; i < r; i++)
                for (int j = 0; j < r; j++)
                {
                    C[i, j] = 0;
                    for (int q = 0; q < r; q++)
                    {
                        C[i, j] += A[i, q] * B[q, j];
                    }
                    dataGridView3.Rows[i].Cells[j].Value = Convert.ToString(C[i, j]);
                }
        }
//Размерность матриц
        private void textBox1_TextChanged(object sender, EventArgs e)
        {
            if(textBox1.Text != "")
                r = Int32.Parse(textBox1.Text);
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Построение графика

Скриншот:



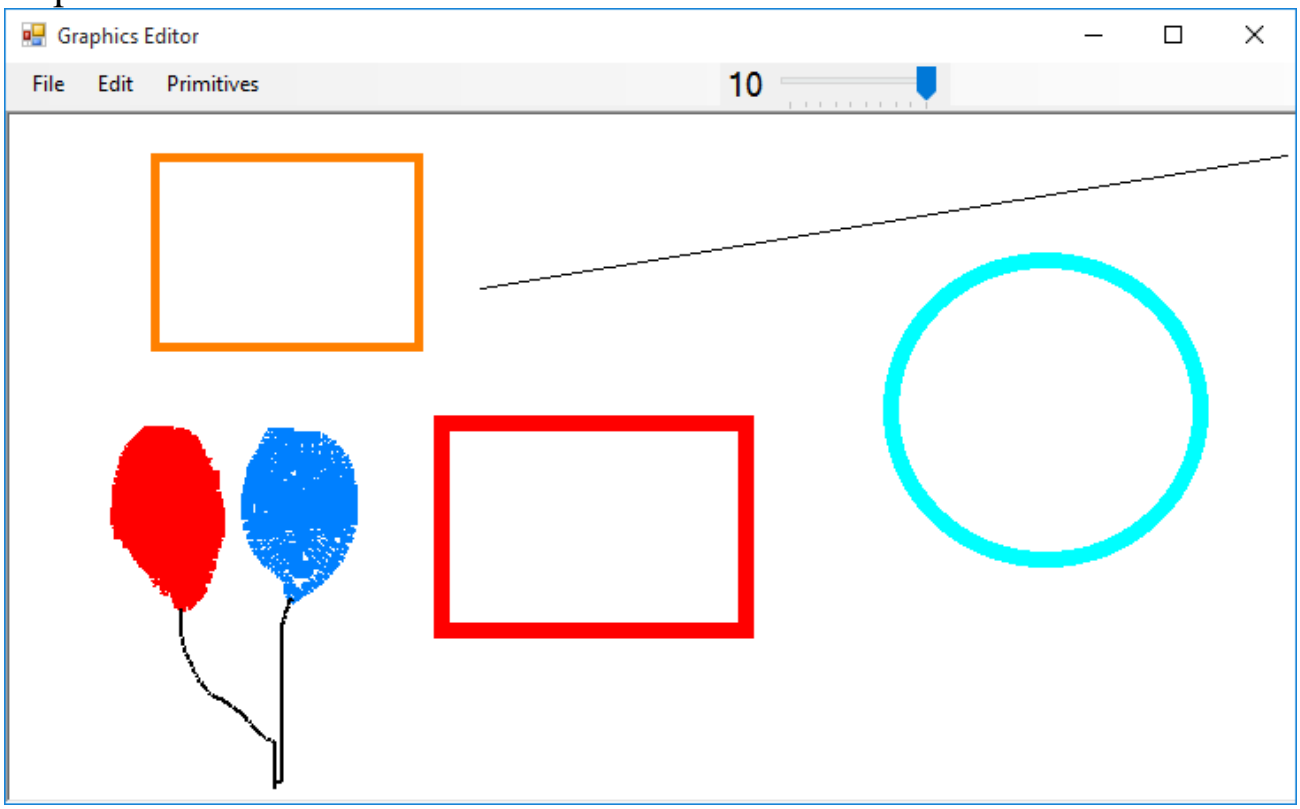

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace _4_октября_2017
{
    public partial class Form1 : Form
    {
        public Form1()
        {

            InitializeComponent();
        }
        int[] a = new int[25];//глобально, потому что нужно не только в листбокс
        int j = 0;
        int x0 = 30, y0 = 30, xk, yk; //0 0 слева вверху
        int dx, dy;
        Pen myPen = new Pen(Color.Black, 2);
        SolidBrush myBruch = new SolidBrush(Color.Black);
        private void Form1_Load(object sender, EventArgs e)
        {
            xk = pictureBox1.Width - 30;
            yk = pictureBox1.Height - 30;
            dx = (xk - x0) / 25;
            dy = (yk - y0) / 40;
        }
//Получаем рандомные значения и чертим график
        private void button1_Click(object sender, EventArgs e)
        {
            j = 0;
            listBox1.Items.Clear();
            Random rnd = new Random();
            for (int i = 0; i < 25; i++)
            {
                a[i] = rnd.Next(10, 40);
```

```
            }

            //Всё с кнопки График
            Graphics gr = pictureBox1.CreateGraphics();
            Color myColor = pictureBox1.BackColor;
            gr.Clear(myColor);
            gr.DrawLine(myPen, x0, y0, x0, yk);
            gr.DrawLine(myPen, x0, y0, x0 + 5, y0 + 20);
            gr.DrawLine(myPen, x0, y0, x0 - 5, y0 + 20);
            gr.DrawLine(myPen, x0, yk, xk, yk);
            gr.DrawLine(myPen, xk, yk, xk - 20, yk + 5);
            gr.DrawLine(myPen, xk, yk, xk - 20, yk - 5);
            for (int i = 0, k = 0, dx = (xk - x0) / 25; i < 26; i++, k++)
            {
                myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;
                myPen.Width = 1;
                gr.DrawLine(myPen, x0 + i * dx, y0, x0 + i * dx, yk + 4);
                if ((k + 1) % 2 == 0)
                {
                    gr.DrawString(k.ToString(), this.Font, myBruch, x0 + i * dx - 5, yk + 5);
                }
                myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid;
                myPen.Width = 2;
            }
            for (int i = 0, k = 0, dy = (yk - y0) / 40; i < 40; i++, k++)
            {
                myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;
                myPen.Width = 1;
                gr.DrawLine(myPen, x0 - 4, y0 + dy * i, xk, y0 + dy * i);
                if ((k + 1) % 3 == 0)
                {
                    gr.DrawString((39 - k).ToString(), this.Font, myBruch, x0 - 20, y0 + 1 + i *
dy);
                }
                myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid;
                myPen.Width = 2;
            }
            //

            timer1.Interval = 500;
            timer2.Enabled = true;
            timer2.Interval = 500;
        }
//Начертить график без обновления данных
        private void button2_Click(object sender, EventArgs e)
        {
            Graphics gr = pictureBox1.CreateGraphics();
            Color myColor = pictureBox1.BackColor;
            gr.Clear(myColor);
            gr.DrawLine(myPen, x0, y0, x0, yk);
            gr.DrawLine(myPen, x0, y0, x0 + 5, y0 + 20);
            gr.DrawLine(myPen, x0, y0, x0 - 5, y0 + 20);
            gr.DrawLine(myPen, x0, yk, xk, yk);
            gr.DrawLine(myPen, xk, yk, xk - 20, yk + 5);
            gr.DrawLine(myPen, xk, yk, xk - 20, yk - 5);
            for (int i = 0, k = 0, dx = (xk - x0) / 25; i < 26; i++, k++)
            {
                myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;
                myPen.Width = 1;
                gr.DrawLine(myPen, x0 + i * dx, y0, x0 + i * dx, yk + 4);
                if ((k + 1) % 2 == 0)
                {
                    gr.DrawString(k.ToString(), this.Font, myBruch, x0 + i * dx - 5, yk + 5);
                }
                myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid;
                myPen.Width = 2;
            }
            for (int i = 0, k = 0, dy = (yk - y0) / 40; i < 40; i++, k++)
```

```
            {
                myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash;
                myPen.Width = 1;
                gr.DrawLine(myPen, x0 - 4, y0 + dy * i, xk, y0 + dy * i);
                if ((k + 1) % 3 == 0)
                {
                    gr.DrawString((39 - k).ToString(), this.Font, myBruch, x0 - 20, y0 + i * dy);
                }
                myPen.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid;
                myPen.Width = 2;
            }
        }
//Найти среднее, максимальное и минимальное значения
        private void button3_Click(object sender, EventArgs e)
        {
            int sr, min = a[0], max = a[0], summ = 0;
            for (int i = 0; i < 25; i++)
            {
                summ += a[i];
                if (a[i] > max)
                {
                    max = a[i];
                }
                if (a[i] < min)
                {
                    min = a[i];
                }
            }
            sr = summ / 25;
            textBox1.Text = (sr).ToString();
            textBox2.Text = (max).ToString();
            textBox3.Text = (min).ToString();
            Graphics gr = pictureBox1.CreateGraphics();
            myPen.Color = Color.Green;
            gr.DrawLine(myPen, x0 - 4, yk - min * dy, xk, yk - min * dy);
            myPen.Color = Color.Blue;
            gr.DrawLine(myPen, x0 - 4, yk - sr * dy, xk, yk - sr * dy);
            myPen.Color = Color.Red;
            gr.DrawLine(myPen, x0 - 4, yk - max * dy, xk, yk - max * dy);
            myPen.Color = Color.Black;
        }
//Чтобы график строился постепенно, запускаем таймер
        private void timer1_Tick(object sender, EventArgs e)
        {
            timer1.Enabled = false;
            listBox1.Items.Add(j + "\t" + a[j]);
            if (j > 0)
            {
                Graphics gr = pictureBox1.CreateGraphics();
                gr.DrawLine(myPen, x0 + dx * (j - 1), yk - a[j - 1] * dy, x0 + dx * j, yk - a[j]
* dy);
            }
            j++;
            if (j == 25)
            {
                timer2.Enabled = false;
            }
        }
        private void timer2_Tick(object sender, EventArgs e)
        {
            timer1.Enabled = true;
        }
    }
}
```

Графический редактор

Скриншот:




```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;//для сохранения файла

namespace Графический_редактор_1._1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        Color myColor = Color.Black;
        int myWeight = 5;
        bool pPen = true, rRectangle = false,  eEllipse = false, sStraight_line =
false;//primitivs
        bool eEraser_white = false; // белый ластик
        bool m_down = false;
        int down_x, down_y;
        Bitmap btm;
        private void Form1_Load(object sender, EventArgs e)
        {
            //загружать белую картинку автоматически (фон)
            pictureBox1.ImageLocation = "D:\\Users\\Desyatov Alexander\\Документы\\Visual Studio
2012\\Projects\\Графический редактор 1.1\\1.jpg";
            trackBar1.Value = myWeight;
            label1.Text = myWeight.ToString();

        }
//Нажатие мыши
        private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
        {
//Для фигур, которые нужно растягивать
            if (rRectangle || eEllipse || sStraight_line)
            {
```

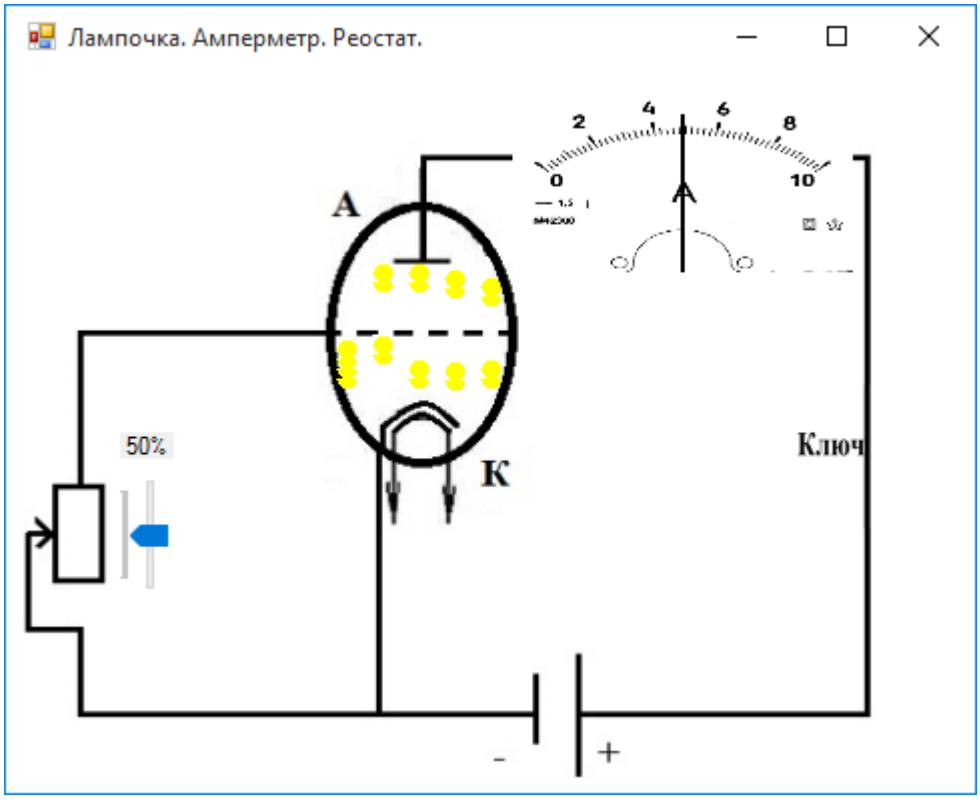
```
                btm = new Bitmap(pictureBox1.Image);
            }
            m_down = true;
            down_x = e.X;
            down_y = e.Y;
        }
//Движение мыши
        private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
        {
            if (m_down)
            {
                Bitmap btm1 = new Bitmap(1,1);
                Graphics gr;
                Pen myPen = new Pen(myColor, myWeight);
                if (eEraser_white || pPen)
                {
                    btm1 = new Bitmap(pictureBox1.Image);
                    gr = Graphics.FromImage(btm1);
                    if (eEraser_white)
                    {
                        Pen Eras = new Pen(Color.White, myWeight);
                        gr.DrawLine(Eras, down_x, down_y, e.X, e.Y);
                        Eras.Dispose();
                        //btm.GetPixel()
                    }
                    if (pPen)
                    {
                        gr.DrawLine(myPen, down_x, down_y, e.X, e.Y);
                    }
                    down_x = e.X;
                    down_y = e.Y;
                    gr.Dispose();
                }
                if (rRectangle || eEllipse || sStraight_line)
                {
                    btm1 = new Bitmap(btm);
                    gr = Graphics.FromImage(btm1);
                    int x = down_x > e.X ? e.X : down_x, y = down_y > e.Y ? e.Y : down_y;
                    if (rRectangle)
                    {
                        gr.DrawRectangle(myPen, x, y, Math.Abs(e.X - down_x), Math.Abs(e.Y -
down_y));
                    }
                    if (eEllipse)
                    {
                        gr.DrawEllipse(myPen, x, y, Math.Abs(e.X - down_x), Math.Abs(e.Y -
down_y));
                    }
                    if (sStraight_line)
                    {
                        gr.DrawLine(myPen, down_x, down_y, e.X, e.Y);
                    }
                    gr.Dispose();
                }
                pictureBox1.Image = btm1;
                myPen.Dispose();
            }
        }
//Отпускание мыши
        private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
        {
            m_down = false;
        }
//Открыть какой-либо рисунок
        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            OpenFileDialog diOp = new OpenFileDialog();
            diOp.Filter = "All files|*.*||*.jpg||*.jpeg||*.png||*.bmp";
```

```
            if(diOp.ShowDialog() == DialogResult.OK)
            {
                pictureBox1.Image = new Bitmap(diOp.OpenFile());
                pictureBox1.SizeMode = PictureBoxSizeMode.AutoSize;
            }
        }
//Сохранить рисунок
        private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SaveFileDialog diSa = new SaveFileDialog();
            diSa.Filter = "|*.jpg||*.jpeg||*.png||*.bmp|All files|*.*";
            diSa.Title = "Save as";
            diSa.OverwritePrompt = true;
            diSa.CheckPathExists = true;
            if (diSa.ShowDialog() == DialogResult.OK)
            {
                string file_name = diSa.FileName;
                Bitmap saved_image = new Bitmap(pictureBox1.Image, pictureBox1.Width,
pictureBox1.Height);
                FileStream fs = new FileStream(file_name, FileMode.Create, FileAccess.ReadWrite);
                saved_image.Save(fs, System.Drawing.Imaging.ImageFormat.Jpeg);
                fs.Close();
            }
        }
//Закрыть
        private void closeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }
//Выбрать цвет
        private void colorToolStripMenuItem_Click(object sender, EventArgs e)
        {
            colorDialog1 = new ColorDialog();
            if (colorDialog1.ShowDialog() == DialogResult.OK)
            {
                myColor = colorDialog1.Color;
            }
        }
//Определить ширину
        private void widthToolStripMenuItem_Click(object sender, EventArgs e)
        {
            label1.Visible = !label1.Visible;
            trackBar1.Visible = !trackBar1.Visible;
        }
//Изменение ширины
        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            label1.Text = trackBar1.Value.ToString();
            myWeight = trackBar1.Value;
        }
//Вызов ластика
        private void eraserToolStripMenuItem_Click(object sender, EventArgs e)
        {
            eEraser_white = true;
            pPen = false;
            rRectangle = false;
            eEllipse = false;
            sStraight_line = false;
        }
//Вызов прямоугольника
        private void rectangleToolStripMenuItem_Click(object sender, EventArgs e)
        {
            eEraser_white = false;
            pPen = false;
            rRectangle = true;
            eEllipse = false;
            sStraight_line = false;
        }
```

```
//Вызов эллипса
        private void ellipseToolStripMenuItem_Click(object sender, EventArgs e)
        {
            eEraser_white = false;
            pPen = false;
            rRectangle = false;
            eEllipse = true;
            sStraight_line = false;
        }
//Построить прямую линию
        private void lineToolStripMenuItem_Click(object sender, EventArgs e)
        {
            eEraser_white = false;
            pPen = false;
            rRectangle = false;
            eEllipse = false;
            sStraight_line = true;
        }
//Рисовать карандашом
        private void penToolStripMenuItem_Click(object sender, EventArgs e)
        {
            eEraser_white = false;
            pPen = true;
            rRectangle = false;
            eEllipse = false;
            sStraight_line = false;
        }
    }
}
```

Лампочка. Амперметр. Реостат.

Скриншот:



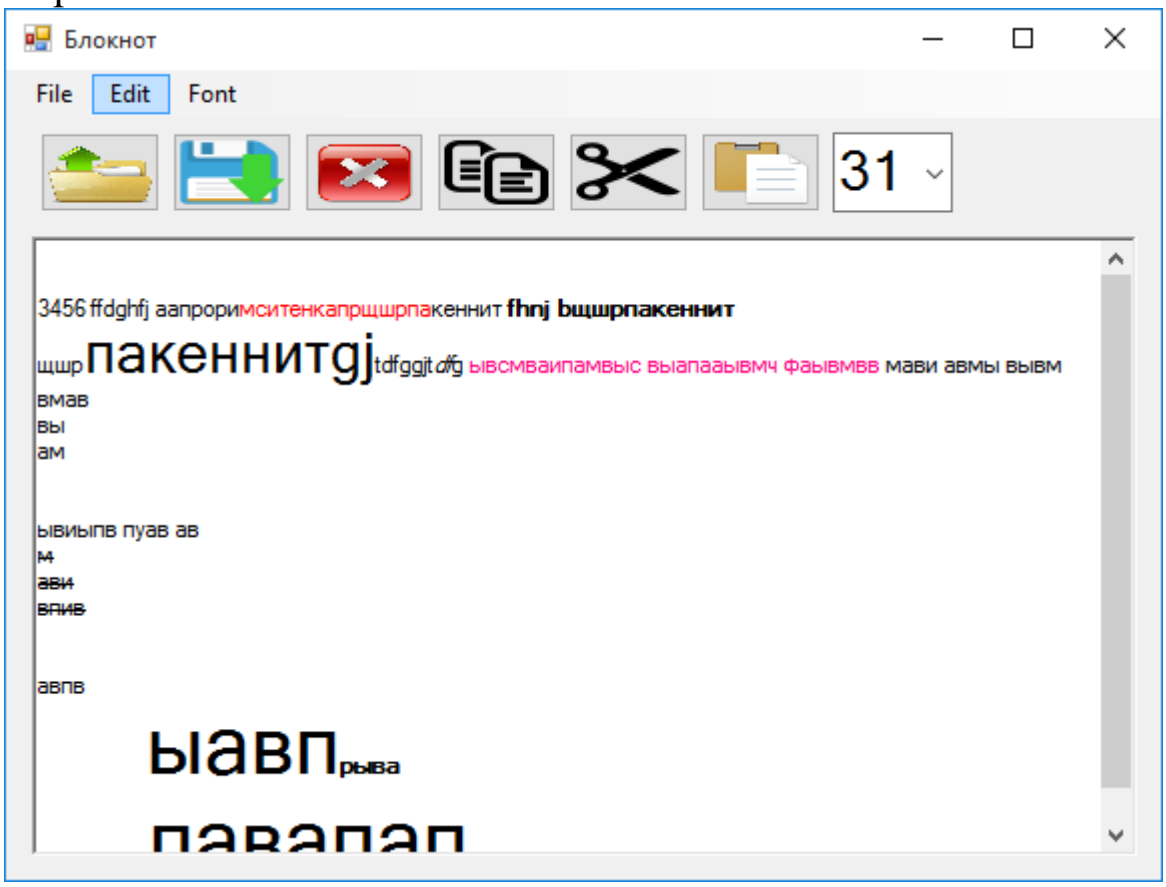

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace Лампочки
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        bool switch_on = false, vikl = true, vkl = true;
        PictureBox[] electrons = new PictureBox[10];
        private void Form1_Load(object sender, EventArgs e)
        {
            Strelka();//Показания амперметра
            Electron_Create();//создание электронов
            timer1.Start();//запуск таймера
            timer1.Interval = 50;
        }
//Замыкание ключа
        private void pictureBox2_Click(object sender, EventArgs e)
        {
            if (switch_on)
            {
                switch_on = false;
                pictureBox2.BackgroundImage = Image.FromFile("D:\\Users\\Desyatov
Alexander\\Документы\\Visual Studio 2012\\Projects\\Лампочки\\выкл.jpg");
                vikl = false;
            }
            else
            {
                switch_on = true;
                pictureBox2.BackgroundImage = Image.FromFile("D:\\Users\\Desyatov
Alexander\\Документы\\Visual Studio 2012\\Projects\\Лампочки\\вкл.jpg");
                vkl = false;
```

```
            }
            Strelka();
        }
//Регулирование реостата
        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            label1.Text = trackBar1.Value.ToString() + "%";
            if(switch_on)
                Strelka();
        }
//Изменение показаний амперметра
        private void Strelka()
        {
            Bitmap btm = new Bitmap(pictureBox1.Width,pictureBox1.Height);
            Graphics gr = Graphics.FromImage(btm);
            Pen myPen = new Pen(Color.Black, 2);
            int leight = 80, xk, yk;
            if (!switch_on)
            {
                xk = (int)(pictureBox1.Width / 8);
            }
            else
            {
                double k_x = 3 * pictureBox1.Width;
                k_x /= 400;
                xk = (int)(pictureBox1.Width / 8 + (100 - trackBar1.Value) * k_x);
            }
            yk = (int) (pictureBox1.Height - Math.Sqrt(leight * leight - Math.Pow(xk -
pictureBox1.Width / 2, 2)));
            gr.DrawLine(myPen, pictureBox1.Width/2, pictureBox1.Height, xk, yk);
            pictureBox1.Image = btm;
        }
//Создание электронов
        private void Electron_Create()
        {
            Random rnd = new Random();
            for (int i = 0; i < 10; i++)
            {
                PictureBox pic0 = new PictureBox();
                pic0.Size = new Size(10, 10);
                pic0.BackColor = Color.Transparent;
                Bitmap btm = new Bitmap(pic0.Width, pic0.Height);
                Graphics g = Graphics.FromImage(btm);
                Brush myBrush = new SolidBrush(Color.Yellow);
                g.FillEllipse(myBrush,0,0,10,10);
                pic0.Image = btm;
                electrons[i] = pic0;
                Controls.Add(electrons[i]);
                electrons[i].BringToFront();
                electrons[i].Visible = false;
                Electron_Pozition(i, rnd);
            }
        }
//Действия по таймеру
        private void timer1_Tick(object sender, EventArgs e)
        {
            if (switch_on)
            {
                if (!vkl)// чтоб видимость включить один раз
                {
                    vkl = true;
                    for (int i = 0; i < 10; i++)
                    {
                        electrons[i].Visible = true;
                    }
                }
                Random rnd = new Random();
                int n = trackBar1.Value/10;
```

```
                for (int i = 0; i < 10; i++)
                {
                    electrons[i].Location = new Point(electrons[i].Location.X,
electrons[i].Location.Y - 5);
                    if (electrons[i].Location.Y < (pictureBox3.Height / 2 +
pictureBox3.Location.Y) && i < n)
                    {
                        Electron_Pozition(i, rnd);
                    }
                    if (electrons[i].Location.Y < pictureBox3.Location.Y)
                    {
                        Electron_Pozition(i, rnd);
                    }

                }
            }
            else
            {
                if (!vikl)
                {
                    vikl = true;
                    for (int i = 0; i < 10; i++)
                    {
                        electrons[i].Visible = false;
                    }
                }
            }
        }
//Новое положение электрона перед началом движения
        private void Electron_Pozition(int i, Random rnd)
        {
             electrons[i].Location = new Point(pictureBox3.Location.X + (i % 5) * 18,
pictureBox3.Location.Y + pictureBox3.Height - rnd.Next(10, pictureBox3.Height / 2));
        }
    }
}
```

Блокнот (Текстовый редактор)

Скриншот:



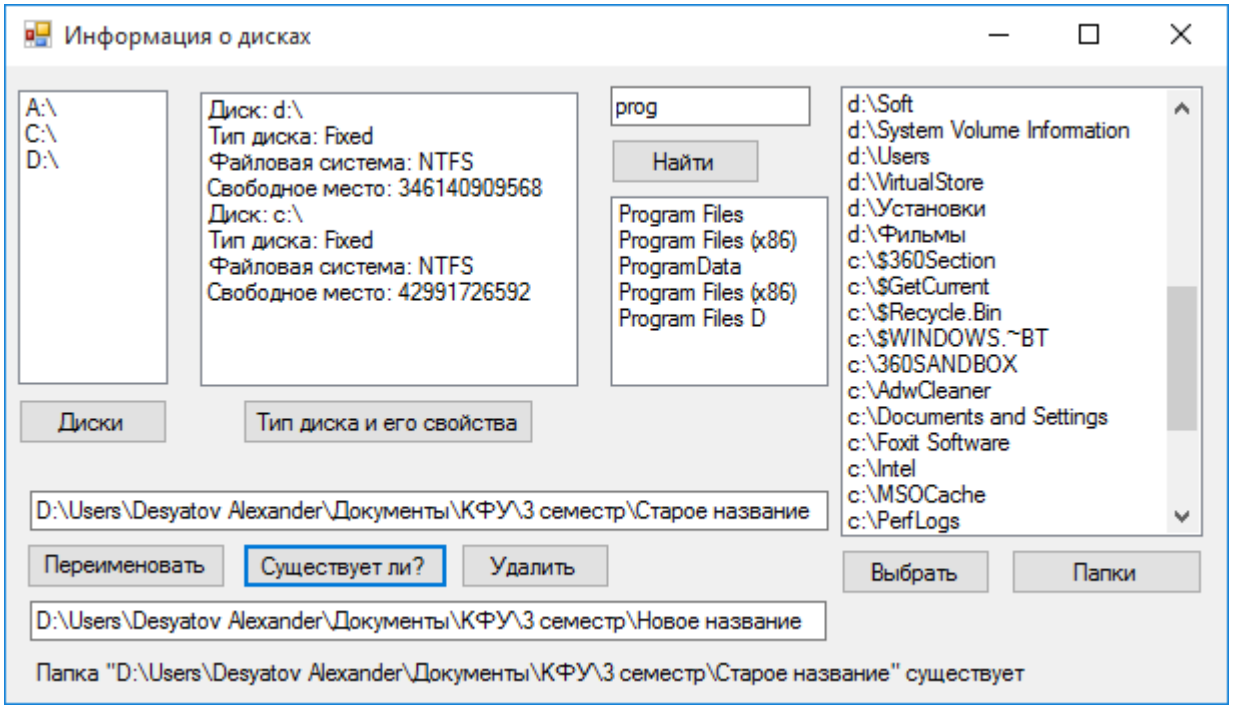

```
using System;
using System.Drawing;
using System.Windows.Forms;
using System.IO;//

namespace Блокнот
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        string BuferText = "";//копировать вставить
        private void Form1_Load(object sender, EventArgs e)
        {

        }

//Открытие какого-либо текстового файла
        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            openFileDialog1.Filter = "|*.txt|All files|*.*";
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                string file_name = openFileDialog1.FileName;
                FileStream file_stream = File.Open(file_name, FileMode.Open, FileAccess.Read);
                if (file_stream != null)
                {
                    StreamReader stream_reader = new StreamReader(file_stream);
                    richTextBox1.Text = stream_reader.ReadToEnd();
                    file_stream.Close();
                }
            }
        }
```

```
//Сохранить текстовый файл
        private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
        {
            saveFileDialog1.Filter = "|*.txt|All files|*.*";
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                string file_name = saveFileDialog1.FileName;
                FileStream file_stream = File.Open(file_name, FileMode.Create, FileAccess.Write);
                if (file_stream != null)
                {
                    StreamWriter stream_writer = new StreamWriter(file_stream);
                    stream_writer.Write(richTextBox1.Text);
                    stream_writer.Flush();
                    file_stream.Close();
                }
            }
        }
//Закрыть текстовый файл
        private void closeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }
//Скопировать текст
        private void copyToolStripMenuItem_Click(object sender, EventArgs e)
        {
            BuferText = richTextBox1.SelectedText;
        }
//Вставить текст
        private void insertToolStripMenuItem_Click(object sender, EventArgs e)
        {
            richTextBox1.SelectedText = BuferText;
        }
//Вырезать текст
        private void cutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            BuferText = richTextBox1.SelectedText;
            richTextBox1.SelectedText = "";
        }
//Поменять цвет текста
        private void colorToolStripMenuItem_Click(object sender, EventArgs e)
        {
            if(colorDialog1.ShowDialog() == DialogResult.OK)
            {
                richTextBox1.SelectionColor = colorDialog1.Color;
            }
        }
//Сделать текст жирным
        private void boldToolStripMenuItem_Click(object sender, EventArgs e)
        {
            richTextBox1.SelectionFont = new Font(richTextBox1.Font.FontFamily, this.Font.Size,
FontStyle.Bold);
        }
//Сделать текст курсивом
        private void italicToolStripMenuItem_Click(object sender, EventArgs e)
        {
            richTextBox1.SelectionFont = new Font(richTextBox1.Font.FontFamily, this.Font.Size,
FontStyle.Italic);
        }
//Подчеркнуть текст
        private void underlineToolStripMenuItem_Click(object sender, EventArgs e)
        {
            richTextBox1.SelectionFont = new Font(richTextBox1.Font.FontFamily, this.Font.Size,
FontStyle.Underline);
        }
//Зачеркнуть текст
        private void strikeoutToolStripMenuItem_Click(object sender, EventArgs e)
        {
```

```
            richTextBox1.SelectionFont = new Font(richTextBox1.Font.FontFamily, this.Font.Size,
FontStyle.Strikeout);
        }
//Изменить размер текста
        private void sizeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            comboBox1.Visible = !comboBox1.Visible;
        }
//Выбрать размер текста
        private void comboBox1_SelectedValueChanged(object sender, EventArgs e)
        {
            richTextBox1.SelectionFont = new Font(richTextBox1.Font.FontFamily,
Convert.ToInt32(comboBox1.Text));
        }
    }
}
```

Информация о дисках

Скриншот:



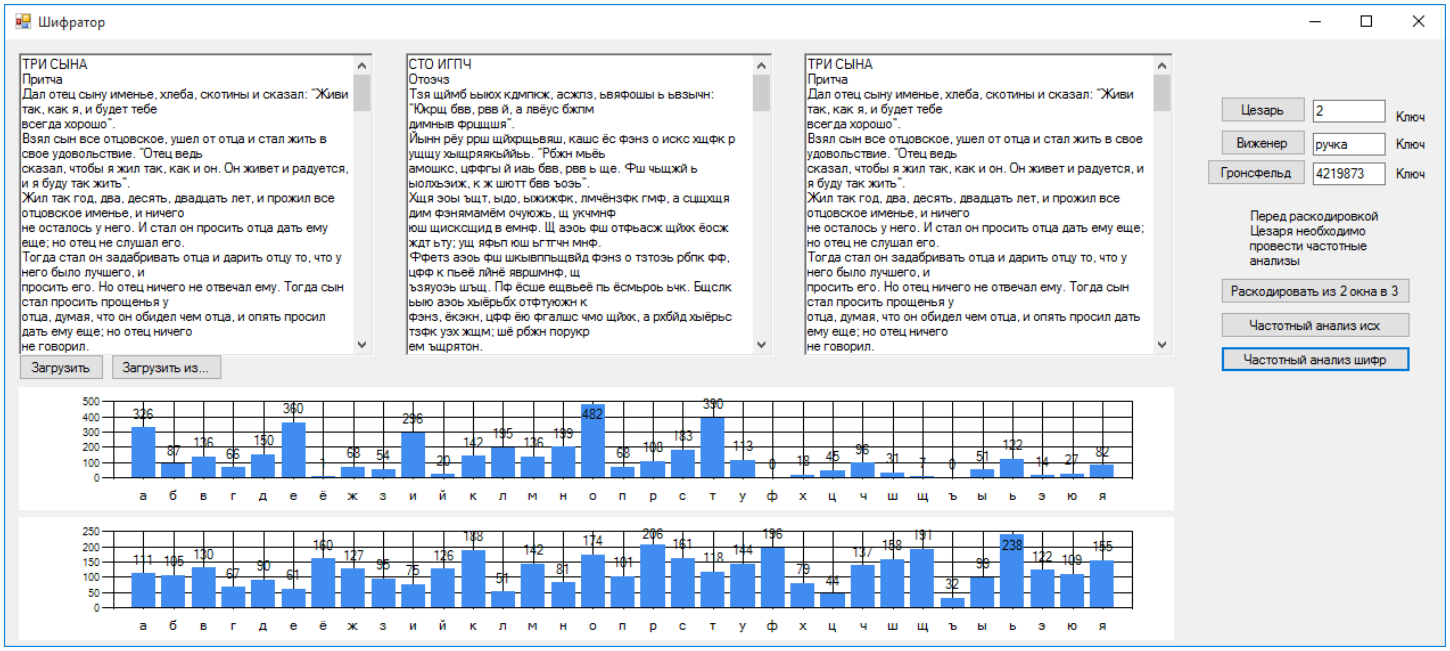

```
using System;
using System.Windows.Forms;
using System.IO;

namespace Список_дисков
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        //Список дисков
        private void button1_Click(object sender, EventArgs e)
        {
            listBox1.Items.Clear();
            string[] astrLogicalDrives = System.IO.Directory.GetLogicalDrives();
            foreach (string disk in astrLogicalDrives)
                listBox1.Items.Add(disk);
        }
        //Список папок
        private void button2_Click(object sender, EventArgs e)
        {
            listBox2.Items.Clear();
            string[] astrFolders = System.IO.Directory.GetDirectories(@"d:\");
            foreach (string folder in astrFolders)
                listBox2.Items.Add(folder);
            string[] astrFolders2 = System.IO.Directory.GetDirectories(@"c:\");
            foreach (string folder2 in astrFolders2)
                listBox2.Items.Add(folder2);
        }

        //Тип дисков
        private void button4_Click(object sender, EventArgs e)
        {
            listBox4.Items.Clear();
            System.IO.DriveInfo drv = new System.IO.DriveInfo(@"d:\");
            listBox4.Items.Add("Диск: " + drv.Name);
            if (drv.IsReady)
```

```
            {
                listBox4.Items.Add("Тип диска: " + drv.DriveType.ToString());
                listBox4.Items.Add("Файловая система: " + drv.DriveFormat.ToString());
                listBox4.Items.Add("Свободное место: " + drv.AvailableFreeSpace.ToString());
            }
            System.IO.DriveInfo drv2 = new System.IO.DriveInfo(@"c:\");
            listBox4.Items.Add("Диск: " + drv2.Name);
            if (drv2.IsReady)
            {
                listBox4.Items.Add("Тип диска: " + drv2.DriveType.ToString());
                listBox4.Items.Add("Файловая система: " + drv2.DriveFormat.ToString());
                listBox4.Items.Add("Свободное место: " + drv2.AvailableFreeSpace.ToString());
            }
        }
        //Найти папку по маске
        private void button5_Click(object sender, EventArgs e)
        {
            listBox3.Items.Clear();
            System.IO.DirectoryInfo di = new System.IO.DirectoryInfo(@"c:\");
            System.IO.DirectoryInfo[] folders = di.GetDirectories("*" + textBox1.Text + "*");
            foreach (System.IO.DirectoryInfo maskdirs in folders)
            {
                listBox3.Items.Add(maskdirs);
            }
            System.IO.DirectoryInfo di2 = new System.IO.DirectoryInfo(@"d:\");
            System.IO.DirectoryInfo[] folders2 = di2.GetDirectories("*" + textBox1.Text + "*");
            foreach (System.IO.DirectoryInfo maskdirs2 in folders2)
            {
                listBox3.Items.Add(maskdirs2);
            }
        }

        private void button3_Click_1(object sender, EventArgs e)
        {
            string old = textBox2.Text;
            string new_name = textBox3.Text;
            Directory.Move(old,new_name);
        }
        //Удаление папки
        private void button6_Click(object sender, EventArgs e)
        {
            try
            {
                string folder = textBox2.Text;
                System.IO.Directory.Delete(folder);
                MessageBox.Show("Папка удалена");
            }
            catch (Exception)
            {
                Directory.Delete("D:\\Users\\Desyatov Alexander\\Документы\\КФУ\\3 семестр\\Новая
папка", recursive: true);
                //удаляем внутренности папки
                MessageBox.Show("Папка удалена!");
            }
            finally { };
        }
        //Выбрать папку
        private void button7_Click(object sender, EventArgs e)
        {
            FolderBrowserDialog fbd = new FolderBrowserDialog();
            //верхний уровень
            fbd.RootFolder = Environment.SpecialFolder.MyComputer;
            //Заголовки в диалоговом окне
            fbd.Description = "Выберерите папку";
            fbd.ShowNewFolderButton = true;
            if (fbd.ShowDialog() == DialogResult.OK)
                this.Text = fbd.SelectedPath;
        }
```

```
        //Проверка на существование папки
        private void button8_Click(object sender, EventArgs e)
        {
            if (System.IO.Directory.Exists(textBox2.Text))
                label1.Text = "Папка \"" + textBox2.Text + "\" существует";
            else
                label1.Text = "Папка не существует";
        }
    }
}
```

Шифратор

Скриншот:




```
using System;
using System.Windows.Forms;
using System.IO;

namespace Шифр1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        bool cez = false, viz = false, gron = false;
        string alfavit_RU = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";//33
        char[] alf_RU; // {'A','B'};
        string[] alfavit_RU_str;
        string alfavit_ru = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";
        char[] alf_ru;
        string[] alfavit_ru_str;
        string alfavit_EN = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";//26
        char[] alf_EN;
        string[] alfavit_EN_str;
        string alfavit_en = "abcdefghijklmnopqrstuvwxyz";
        char[] alf_en;
        string[] alfavit_en_str;
        string alfavit_num = "0123456789";//10
        char[] alf_num;
        string[] alfavit_num_str;
        string alfavit_znaki = ".,!?@\"'#№%^:*()-_+={[]}/|\\><";//28
        char[] alf_znaki;
        string[] alfavit_znaki_str;
        int al_RU_all, al_ru_all, al_EN_all, al_en_all, al_num_all, al_znaki_all;
        int[] kol_vhozd_RU;
        int[] kol_vhozd_ru;
        int[] kol_vhozd_RU_sh;
        int[] kol_vhozd_ru_sh;

        //load
        private void Form1_Load(object sender, EventArgs e)
        {
            al_RU_all = alfavit_RU.Length;
            alf_RU = alfavit_RU.ToCharArray(0, al_RU_all);
```

```
            alfavit_RU_str = new string[al_RU_all];
            for (int i = 0; i < al_RU_all; i++)//для анализа
            {
                alfavit_RU_str[i] = alf_RU[i].ToString();
            }
            al_ru_all = alfavit_ru.Length;
            alf_ru = alfavit_ru.ToCharArray(0, al_ru_all);
            alfavit_ru_str = new string[al_ru_all];
            for (int i = 0; i < al_ru_all; i++)
            {
                alfavit_ru_str[i] = alf_ru[i].ToString();
            }
            al_EN_all = alfavit_EN.Length;
            alf_EN = alfavit_EN.ToCharArray(0, al_EN_all);
            alfavit_EN_str = new string[al_EN_all];
            for (int i = 0; i < al_EN_all; i++)
            {
                alfavit_EN_str[i] = alf_EN[i].ToString();
            }
            al_en_all = alfavit_en.Length;
            alf_en = alfavit_en.ToCharArray(0, al_en_all);
            alfavit_en_str = new string[al_en_all];
            for (int i = 0; i < al_en_all; i++)
            {
                alfavit_en_str[i] = alf_en[i].ToString();
            }
            al_num_all = alfavit_num.Length;
            alf_num = alfavit_num.ToCharArray(0, al_num_all);
            alfavit_num_str = new string[al_num_all];
            for (int i = 0; i < al_num_all; i++)
            {
                alfavit_num_str[i] = alf_num[i].ToString();
            }
            al_znaki_all = alfavit_znaki.Length;
            alf_znaki = alfavit_znaki.ToCharArray(0, al_znaki_all);
            alfavit_znaki_str = new string[al_znaki_all];
            for (int i = 0; i < al_znaki_all; i++)
            {
                alfavit_znaki_str[i] = alf_znaki[i].ToString();
            }
            kol_vhozd_RU = new int[al_RU_all];
            kol_vhozd_ru = new int[al_ru_all];
            kol_vhozd_RU_sh = new int[al_RU_all];
            kol_vhozd_ru_sh = new int[al_ru_all];
        }
        //загрузить из...
        private void button6_Click(object sender, EventArgs e)
        {
            openFileDialog1.Filter = "|*.txt|All files|*.*";
            if (openFileDialog1.ShowDialog() == DialogResult.OK)
            {
                string file_name = openFileDialog1.FileName;
                FileStream file_stream = File.Open(file_name, FileMode.Open, FileAccess.Read);
                if (file_stream != null)
                {
                    StreamReader stream_reader = new StreamReader(file_stream);
                    richTextBox1.Text = stream_reader.ReadToEnd();
                    file_stream.Close();
                }
            }
        }
        //загрузить
        private void button1_Click(object sender, EventArgs e)
        {
            FileStream file_stream = File.Open("D:\\Users\\Desyatov Alexander\\Документы\\Visual
Studio 2012\\Projects\\Шифр1\\Рассказы\\3 сына.txt", FileMode.Open, FileAccess.Read);
            if (file_stream != null)
            {
```

```
            StreamReader stream_reader = new StreamReader(file_stream);
            richTextBox1.Text = stream_reader.ReadToEnd();
            file_stream.Close();
        }
    }
    //Цезарь
    private void button2_Click(object sender, EventArgs e)
    {
        cez = true;
        viz = false;
        gron = false;
        richTextBox2.Text = "";
        int key_c = Convert.ToInt32(textBox1.Text);
        int length_t = richTextBox1.Text.Length;
        char[] text = richTextBox1.Text.ToCharArray(0, length_t);
        char[] shifr = new char[length_t];
        for (int i = 0; i < length_t; i++)
        {
            bool nashel = false;
            for (int j = 0; !nashel && j < al_RU_all; j++)
            {
                if (text[i] == alf_RU[j])
                {
                    int nom = 0;
                    if (j + key_c >= al_RU_all)
                        nom = (j + key_c) % al_RU_all;
                    if (j + key_c < al_RU_all && j + key_c >= 0)
                        nom = j + key_c;
                    if (j + key_c < 0)
                    {
                        nom = j + key_c;
                        while (nom < 0)
                            nom += al_RU_all;
                    }
                    shifr[i] = alf_RU[nom];
                    nashel = true;
                }
            }
            for (int j = 0; !nashel && j < al_ru_all; j++)
            {
                if (text[i] == alf_ru[j])
                {
                    int nom = 0;
                    if (j + key_c >= al_ru_all)
                        nom = (j + key_c) % al_ru_all;
                    if (j + key_c < al_RU_all && j + key_c >= 0)
                        nom = j + key_c;
                    if (j + key_c < 0)
                    {
                        nom = j + key_c;
                        while (nom < 0)
                            nom += al_ru_all;
                    }
                    shifr[i] = alf_ru[nom];
                    nashel = true;
                }
            }
            for (int j = 0; !nashel && j < al_EN_all; j++)
            {
                if (text[i] == alf_EN[j])
                {
                    int nom = 0;
                    if (j + key_c >= al_EN_all)
                        nom = (j + key_c) % al_EN_all;
                    if (j + key_c < al_EN_all && j + key_c >= 0)
                        nom = j + key_c;
                    if (j + key_c < 0)
                    {
```

```
            nom = j + key_c;
            while (nom < 0)
                nom += al_EN_all;
        }
        shifr[i] = alf_EN[nom];
        nashel = true;
    }
}
for (int j = 0; !nashel && j < al_en_all; j++)
{
    if (text[i] == alf_en[j])
    {
        int nom = 0;
        if (j + key_c >= al_en_all)
            nom = (j + key_c) % al_en_all;
        if (j + key_c < al_en_all && j + key_c >= 0)
            nom = j + key_c;
        if (j + key_c < 0)
        {
            nom = j + key_c;
            while (nom < 0)
                nom += al_en_all;
        }
        shifr[i] = alf_en[nom];
        nashel = true;
    }
}
for (int j = 0; !nashel && j < al_num_all; j++)
{
    if (text[i] == alf_num[j])
    {
        int nom = 0;
        if (j + key_c >= al_num_all)
            nom = (j + key_c) % al_num_all;
        if (j + key_c < al_num_all && j + key_c >= 0)
            nom = j + key_c;
        if (j + key_c < 0)
        {
            nom = j + key_c;
            while (nom < 0)
                nom += al_num_all;
        }
        shifr[i] = alf_num[nom];
        nashel = true;
    }
}
for (int j = 0; !nashel && j < al_znaki_all; j++)
{
    if (text[i] == alf_znaki[j])
    {
        int nom = 0;
        if (j + key_c >= al_znaki_all)
            nom = (j + key_c) % al_znaki_all;
        if (j + key_c < al_znaki_all && j + key_c >= 0)
            nom = j + key_c;
        if (j + key_c < 0)
        {
            nom = j + key_c;
            while (nom < 0)
                nom += al_znaki_all;
        }
        shifr[i] = alf_znaki[nom];
        nashel = true;
    }
}
if (!nashel)
{
    shifr[i] = text[i];
```

```
                nashel = true;
            }
        }
        for (int i = 0; i < length_t; i++)
        {
            richTextBox2.Text += shifr[i].ToString();
        }
    }
    //Виженер
    private void button3_Click(object sender, EventArgs e)
    {
        cez = false;
        viz = true;
        gron = false;
        richTextBox2.Text = "";
        //для русских текстов
        string stroka_RU_string = "ОПРСТУФХЦАБВГДЕЁЖЗИЙКЛМНЧШЩЪЫЬЭЮЯ";//33 = al_RU_all
        string stroka_ru_string = "опрстуфхцабвгдеёжзийклмнчшщъыьэюя";//33 = al_ru_all
        char[] stroka_RU_char = stroka_RU_string.ToCharArray(0, al_RU_all);
        char[] stroka_ru_char = stroka_ru_string.ToCharArray(0, al_ru_all);
        char[,] tabl_RU_char = new char[al_RU_all, al_RU_all];
        char[,] tabl_ru_char = new char[al_ru_all, al_ru_all];
        for (int i = 0; i < al_RU_all; i++)//заполняем таблицу
        {
            for (int j = 0; j < al_RU_all; j++)
            {
                tabl_RU_char[i, j] = stroka_RU_char[(i + j) % al_RU_all];
            }
        }
        for (int i = 0; i < al_ru_all; i++)//заполняем таблицу
        {
            for (int j = 0; j < al_ru_all; j++)
            {
                tabl_ru_char[i, j] = stroka_ru_char[(i + j) % al_ru_all];
            }
        }
        int length_key_v = textBox2.Text.Length;
        char[] key_v = textBox2.Text.ToCharArray(0, length_key_v);
        int length_t = richTextBox1.Text.Length;
        char[] text = richTextBox1.Text.ToCharArray(0, length_t);
        char[] shifr = new char[length_t];
        for (int q = 0; q < length_t; q++)
        {//Для больших и маленьких русских букв
            bool nashel = false;
            int i = 0, j = 0;
            int num_ch_key = q % length_key_v;
            for (; j < al_RU_all && j < al_ru_all; j++)
            {//буквы ключа
                if (key_v[num_ch_key] == alf_RU[j] || key_v[num_ch_key] == alf_ru[j])
                {
                    break;
                }
            }
            for (; !nashel && i < al_RU_all && i < al_ru_all; i++)
            {//буквы алфавита
                if (text[q] == alf_RU[i])
                {
                    shifr[q] = tabl_RU_char[i, j];
                    nashel = true;
                    break;
                }
                if (text[q] == alf_ru[i])
                {
                    shifr[q] = tabl_ru_char[i, j];
                    nashel = true;
                    break;
                }
            }
```

```
            if (!nashel)
            {
                shifr[q] = text[q];
                nashel = true;
            }
        }
        for (int i = 0; i < length_t; i++)
        {
            richTextBox2.Text += shifr[i].ToString();
        }
    }
    //Гронсфельд
    private void button4_Click(object sender, EventArgs e)
    {
        cez = false;
        viz = false;
        gron = true;
        richTextBox2.Text = "";
        int length_key_g = textBox3.Text.Length;
        char[] key_g_char = textBox3.Text.ToCharArray(0, length_key_g);
        int[] key_g = new int[length_key_g];
        for (int i = 0; i < length_key_g; i++)
        {
            key_g[i] = (int)Char.GetNumericValue(key_g_char[i]);
        }
        label4.Text = ((int)Char.GetNumericValue(key_g_char[0])).ToString();
        int length_t = richTextBox1.Text.Length;
        char[] text = richTextBox1.Text.ToCharArray(0, length_t);
        char[] shifr = new char[length_t];
        for (int i = 0; i < length_t; i++)
        {
            int num_ch_key = i % length_key_g;
            bool nashel = false;
            for (int j = 0; !nashel && j < al_RU_all; j++)
            {
                if (text[i] == alf_RU[j])
                {
                    int nom = 0;
                    if (j + key_g[num_ch_key] >= al_RU_all)
                        nom = (j + key_g[num_ch_key]) % al_RU_all;
                    if (j + key_g[num_ch_key] < al_RU_all && j + key_g[num_ch_key] >= 0)
                        nom = j + key_g[num_ch_key];
                    if (j + key_g[num_ch_key] < 0)
                    {
                        nom = j + key_g[num_ch_key];
                        while (nom < 0)
                            nom += al_RU_all;
                    }
                    shifr[i] = alf_RU[nom];
                    nashel = true;
                }
            }
            for (int j = 0; !nashel && j < al_ru_all; j++)
            {
                if (text[i] == alf_ru[j])
                {
                    int nom = 0;
                    if (j + key_g[num_ch_key] >= al_ru_all)
                        nom = (j + key_g[num_ch_key]) % al_ru_all;
                    if (j + key_g[num_ch_key] < al_RU_all && j + key_g[num_ch_key] >= 0)
                        nom = j + key_g[num_ch_key];
                    if (j + key_g[num_ch_key] < 0)
                    {
                        nom = j + key_g[num_ch_key];
                        while (nom < 0)
                            nom += al_ru_all;
                    }
                    shifr[i] = alf_ru[nom];
```

```
            nashel = true;
        }
    }
    for (int j = 0; !nashel && j < al_EN_all; j++)
    {
        if (text[i] == alf_EN[j])
        {
            int nom = 0;
            if (j + key_g[num_ch_key] >= al_EN_all)
                nom = (j + key_g[num_ch_key]) % al_EN_all;
            if (j + key_g[num_ch_key] < al_EN_all && j + key_g[num_ch_key] >= 0)
                nom = j + key_g[num_ch_key];
            if (j + key_g[num_ch_key] < 0)
            {
                nom = j + key_g[num_ch_key];
                while (nom < 0)
                    nom += al_EN_all;
            }
            shifr[i] = alf_EN[nom];
            nashel = true;
        }
    }
    for (int j = 0; !nashel && j < al_en_all; j++)
    {
        if (text[i] == alf_en[j])
        {
            int nom = 0;
            if (j + key_g[num_ch_key] >= al_en_all)
                nom = (j + key_g[num_ch_key]) % al_en_all;
            if (j + key_g[num_ch_key] < al_en_all && j + key_g[num_ch_key] >= 0)
                nom = j + key_g[num_ch_key];
            if (j + key_g[num_ch_key] < 0)
            {
                nom = j + key_g[num_ch_key];
                while (nom < 0)
                    nom += al_en_all;
            }
            shifr[i] = alf_en[nom];
            nashel = true;
        }
    }
    for (int j = 0; !nashel && j < al_num_all; j++)
    {
        if (text[i] == alf_num[j])
        {
            int nom = 0;
            if (j + key_g[num_ch_key] >= al_num_all)
                nom = (j + key_g[num_ch_key]) % al_num_all;
            if (j + key_g[num_ch_key] < al_num_all && j + key_g[num_ch_key] >= 0)
                nom = j + key_g[num_ch_key];
            if (j + key_g[num_ch_key] < 0)
            {
                nom = j + key_g[num_ch_key];
                while (nom < 0)
                    nom += al_num_all;
            }
            shifr[i] = alf_num[nom];
            nashel = true;
        }
    }
    for (int j = 0; !nashel && j < al_znaki_all; j++)
    {
        if (text[i] == alf_znaki[j])
        {
            int nom = 0;
            if (j + key_g[num_ch_key] >= al_znaki_all)
                nom = (j + key_g[num_ch_key]) % al_znaki_all;
            if (j + key_g[num_ch_key] < al_znaki_all && j + key_g[num_ch_key] >= 0)
```

```
                        nom = j + key_g[num_ch_key];
                    if (j + key_g[num_ch_key] < 0)
                    {
                        nom = j + key_g[num_ch_key];
                        while (nom < 0)
                            nom += al_znaki_all;
                    }
                    shifr[i] = alf_znaki[nom];
                    nashel = true;
                }
            }
            if (!nashel)
            {
                shifr[i] = text[i];
                nashel = true;
            }
        }
        for (int i = 0; i < length_t; i++)
        {
            richTextBox2.Text += shifr[i].ToString();
        }
    }
    //Частотный анализ исх
    private void button7_Click(object sender, EventArgs e)
    {

        int length_t = richTextBox1.Text.Length;
        char[] text = richTextBox1.Text.ToCharArray(0, length_t);
        for (int i = 0; i < length_t; i++)
        {
            bool nashel = false;
            if (text[i] == ' ' || text[i] == '\n')
            {
                nashel = true;
            }
            for (int j = 0; !nashel && j < al_RU_all; j++)
            {
                if (text[i] == alf_RU[j])
                {
                    kol_vhozd_RU[j]++;
                    nashel = true;
                }
            }
            for (int j = 0; !nashel && j < al_RU_all; j++)
            {
                if (text[i] == alf_ru[j])
                {
                    kol_vhozd_ru[j]++;
                    nashel = true;
                }
            }
        }
        chart1.Series[0].Points.Clear();
        for (int i = 0; i < al_ru_all; i++)
        {
            chart1.Series[0].Points.AddXY(alfavit_ru_str[i], kol_vhozd_ru[i] +
kol_vhozd_RU[i]);
        }
    }
    //Частотный анализ шифр
    private void button8_Click(object sender, EventArgs e)
    {


        int length_t = richTextBox2.Text.Length;
        char[] text = richTextBox2.Text.ToCharArray(0, length_t);
        for (int i = 0; i < length_t; i++)
        {
```

```
                bool nashel = false;
                if (text[i] == ' ' || text[i] == '\n')
                {
                    nashel = true;
                }
                for (int j = 0; !nashel && j < al_RU_all; j++)
                {
                    if (text[i] == alf_RU[j])
                    {
                        kol_vhozd_RU_sh[j]++;
                        nashel = true;
                    }
                }
                for (int j = 0; !nashel && j < al_RU_all; j++)
                {
                    if (text[i] == alf_ru[j])
                    {
                        kol_vhozd_ru_sh[j]++;
                        nashel = true;
                    }
                }
            }
            chart2.Series[0].Points.Clear();
            for (int i = 0; i < al_ru_all; i++)
            {
                chart2.Series[0].Points.AddXY(alfavit_ru_str[i], kol_vhozd_ru_sh[i] +
kol_vhozd_RU_sh[i]);
            }
        }
        //Расшифровать
        private void button5_Click(object sender, EventArgs e)
        {
            if(cez)//Цезарь
            {
                richTextBox3.Text = "";
//Ключ находим с помощью частотного анализа, а не из текстбокса
                int num_ish = 0, num_sh = 0;
                for(int i = 0, max_ish = 0, max_sh = 0; i < al_ru_all && i < al_RU_all; i++){
                    if (max_ish < kol_vhozd_ru[i] + kol_vhozd_RU[i])
                    {
                        max_ish = kol_vhozd_ru[i] + kol_vhozd_RU[i];
                        num_ish = i;
                    }
                    if (max_sh < kol_vhozd_ru_sh[i] + kol_vhozd_RU_sh[i])
                    {
                        max_sh = kol_vhozd_ru_sh[i] + kol_vhozd_RU_sh[i];
                        num_sh = i;
                    }
                }
                int key_c = num_sh - num_ish;
                int length_t = richTextBox2.Text.Length;
                char[] shifr = richTextBox2.Text.ToCharArray(0, length_t);
                char[] deshifr = new char[length_t];
                for (int i = 0; i < length_t; i++)
                {
                    bool nashel = false;
                    for (int j = 0; !nashel && j < al_RU_all; j++)
                    {
                        if (shifr[i] == alf_RU[j])
                        {
                            int nom = 0;
                            if (j - key_c >= al_RU_all)
                                nom = (j - key_c) % al_RU_all;
                            if (j - key_c < al_RU_all && j - key_c >= 0)
                                nom = j - key_c;
                            if (j - key_c < 0)
                            {
                                nom = j - key_c;
```

```
            while (nom < 0)
                nom += al_RU_all;
        }
        deshifr[i] = alf_RU[nom];
        nashel = true;
    }
}
for (int j = 0; !nashel && j < al_ru_all; j++)
{
    if (shifr[i] == alf_ru[j])
    {
        int nom = 0;
        if (j - key_c >= al_ru_all)
            nom = (j - key_c) % al_ru_all;
        if (j - key_c < al_RU_all && j - key_c >= 0)
            nom = j - key_c;
        if (j - key_c < 0)
        {
            nom = j - key_c;
            while (nom < 0)
                nom += al_ru_all;
        }
        deshifr[i] = alf_ru[nom];
        nashel = true;
    }
}
for (int j = 0; !nashel && j < al_EN_all; j++)
{
    if (shifr[i] == alf_EN[j])
    {
        int nom = 0;
        if (j - key_c >= al_EN_all)
            nom = (j - key_c) % al_EN_all;
        if (j - key_c < al_EN_all && j - key_c >= 0)
            nom = j - key_c;
        if (j - key_c < 0)
        {
            nom = j - key_c;
            while (nom < 0)
                nom += al_EN_all;
        }
        deshifr[i] = alf_EN[nom];
        nashel = true;
    }
}
for (int j = 0; !nashel && j < al_en_all; j++)
{
    if (shifr[i] == alf_en[j])
    {
        int nom = 0;
        if (j - key_c >= al_en_all)
            nom = (j - key_c) % al_en_all;
        if (j - key_c < al_en_all && j - key_c >= 0)
            nom = j - key_c;
        if (j - key_c < 0)
        {
            nom = j - key_c;
            while (nom < 0)
                nom += al_en_all;
        }
        deshifr[i] = alf_en[nom];
        nashel = true;
    }
}
for (int j = 0; !nashel && j < al_num_all; j++)
{
    if (shifr[i] == alf_num[j])
    {
```

```
                int nom = 0;
                if (j - key_c >= al_num_all)
                    nom = (j - key_c) % al_num_all;
                if (j - key_c < al_num_all && j - key_c >= 0)
                    nom = j - key_c;
                if (j - key_c < 0)
                {
                    nom = j - key_c;
                    while (nom < 0)
                        nom += al_num_all;
                }
                deshifr[i] = alf_num[nom];
                nashel = true;
            }
        }
        for (int j = 0; !nashel && j < al_znaki_all; j++)
        {
            if (shifr[i] == alf_znaki[j])
            {
                int nom = 0;
                if (j - key_c >= al_znaki_all)
                    nom = (j - key_c) % al_znaki_all;
                if (j - key_c < al_znaki_all && j - key_c >= 0)
                    nom = j - key_c;
                if (j - key_c < 0)
                {
                    nom = j - key_c;
                    while (nom < 0)
                        nom += al_znaki_all;
                }
                deshifr[i] = alf_znaki[nom];
                nashel = true;
            }
        }
        if (!nashel)
        {
            deshifr[i] = shifr[i];
            nashel = true;
        }
    }
    for (int i = 0; i < length_t; i++)
    {
        richTextBox3.Text += deshifr[i].ToString();
    }
}
if(viz)//Виженер
{
    richTextBox3.Text = "";
    string stroka_RU_string = "ОПРСТУФХЦАБВГДЕЁЖЗИЙКЛМНЧШЩЪЫЬЭЮЯ";//33 = al_RU_all
    string stroka_ru_string = "опрстуфхцабвгдеёжзийклмнчшщъыьэюя";//33 = al_ru_all
    char[] stroka_RU_char = stroka_RU_string.ToCharArray(0, al_RU_all);
    char[] stroka_ru_char = stroka_ru_string.ToCharArray(0, al_ru_all);
    char[,] tabl_RU_char = new char[al_RU_all, al_RU_all];
    char[,] tabl_ru_char = new char[al_ru_all, al_ru_all];
    for (int i = 0; i < al_RU_all; i++)//заполняем таблицу
    {
        for (int j = 0; j < al_RU_all; j++)
        {
            tabl_RU_char[i, j] = stroka_RU_char[(i + j) % al_RU_all];
        }
    }
    for (int i = 0; i < al_ru_all; i++)//заполняем таблицу
    {
        for (int j = 0; j < al_ru_all; j++)
        {
            tabl_ru_char[i, j] = stroka_ru_char[(i + j) % al_ru_all];
        }
    }
```

```
    int length_key_v = textBox2.Text.Length;
    char[] key_v = textBox2.Text.ToCharArray(0, length_key_v);
    int length_t = richTextBox2.Text.Length;
    char[] shifr = richTextBox2.Text.ToCharArray(0, length_t);
    char[] deshifr = new char[length_t];
    for (int q = 0; q < length_t; q++)
    {//Для больших и маленьких русских букв
        bool nashel = false;
        int i = 0, j = 0;
        int num_ch_key = q % length_key_v;
        for (; j < al_RU_all && j < al_ru_all; j++)
        {//буквы ключа
            if (key_v[num_ch_key] == alf_RU[j] || key_v[num_ch_key] == alf_ru[j])
            {
                break;
            }
        }
        for (; !nashel && i < al_RU_all && i < al_ru_all; i++)
        {//буквы алфавита
            if (shifr[q] == tabl_RU_char[i, j])
            {
                deshifr[q] = alf_RU[i];
                nashel = true;
                break;
            }
            if (shifr[q] == tabl_ru_char[i, j])
            {
                deshifr[q] = alf_ru[i];
                nashel = true;
                break;
            }
        }
        if (!nashel)
        {
            deshifr[q] = shifr[q];
            nashel = true;
        }
    }
    for (int i = 0; i < length_t; i++)
    {
        richTextBox3.Text += deshifr[i].ToString();
    }
}
if(gron)//Гронсфельд
{
    richTextBox3.Text = "";
    int length_key_g = textBox3.Text.Length;
    char[] key_g_char = textBox3.Text.ToCharArray(0, length_key_g);
    int[] key_g = new int[length_key_g];
    for (int i = 0; i < length_key_g; i++)
    {
        key_g[i] = (int)Char.GetNumericValue(key_g_char[i]);
    }
    label4.Text = ((int)Char.GetNumericValue(key_g_char[0])).ToString();
    int length_t = richTextBox2.Text.Length;
    char[] shifr = richTextBox2.Text.ToCharArray(0, length_t);
    char[] deshifr = new char[length_t];
    for (int i = 0; i < length_t; i++)
    {
        int num_ch_key = i % length_key_g;
        bool nashel = false;
        for (int j = 0; !nashel && j < al_RU_all; j++)
        {
            if (shifr[i] == alf_RU[j])
            {
                int nom = 0;
                if (j - key_g[num_ch_key] >= al_RU_all)
                    nom = (j - key_g[num_ch_key]) % al_RU_all;
```

```
            if (j - key_g[num_ch_key] < al_RU_all && j - key_g[num_ch_key] >= 0)
                nom = j - key_g[num_ch_key];
            if (j - key_g[num_ch_key] < 0)
            {
                nom = j - key_g[num_ch_key];
                while (nom < 0)
                    nom += al_RU_all;
            }
            deshifr[i] = alf_RU[nom];
            nashel = true;
        }
    }
    for (int j = 0; !nashel && j < al_ru_all; j++)
    {
        if (shifr[i] == alf_ru[j])
        {
            int nom = 0;
            if (j - key_g[num_ch_key] >= al_ru_all)
                nom = (j - key_g[num_ch_key]) % al_ru_all;
            if (j - key_g[num_ch_key] < al_RU_all && j - key_g[num_ch_key] >= 0)
                nom = j - key_g[num_ch_key];
            if (j - key_g[num_ch_key] < 0)
            {
                nom = j - key_g[num_ch_key];
                while (nom < 0)
                    nom += al_ru_all;
            }
            deshifr[i] = alf_ru[nom];
            nashel = true;
        }
    }
    for (int j = 0; !nashel && j < al_EN_all; j++)
    {
        if (shifr[i] == alf_EN[j])
        {
            int nom = 0;
            if (j - key_g[num_ch_key] >= al_EN_all)
                nom = (j - key_g[num_ch_key]) % al_EN_all;
            if (j - key_g[num_ch_key] < al_EN_all && j - key_g[num_ch_key] >= 0)
                nom = j - key_g[num_ch_key];
            if (j - key_g[num_ch_key] < 0)
            {
                nom = j - key_g[num_ch_key];
                while (nom < 0)
                    nom += al_EN_all;
            }
            deshifr[i] = alf_EN[nom];
            nashel = true;
        }
    }
    for (int j = 0; !nashel && j < al_en_all; j++)
    {
        if (shifr[i] == alf_en[j])
        {
            int nom = 0;
            if (j - key_g[num_ch_key] >= al_en_all)
                nom = (j - key_g[num_ch_key]) % al_en_all;
            if (j - key_g[num_ch_key] < al_en_all && j - key_g[num_ch_key] >= 0)
                nom = j - key_g[num_ch_key];
            if (j - key_g[num_ch_key] < 0)
            {
                nom = j - key_g[num_ch_key];
                while (nom < 0)
                    nom += al_en_all;
            }
            deshifr[i] = alf_en[nom];
            nashel = true;
        }
```

```
                    }
                    for (int j = 0; !nashel && j < al_num_all; j++)
                    {
                        if (shifr[i] == alf_num[j])
                        {
                            int nom = 0;
                            if (j - key_g[num_ch_key] >= al_num_all)
                                nom = (j - key_g[num_ch_key]) % al_num_all;
                            if (j - key_g[num_ch_key] < al_num_all && j - key_g[num_ch_key] >= 0)
                                nom = j - key_g[num_ch_key];
                            if (j - key_g[num_ch_key] < 0)
                            {
                                nom = j - key_g[num_ch_key];
                                while (nom < 0)
                                    nom += al_num_all;
                            }
                            deshifr[i] = alf_num[nom];
                            nashel = true;
                        }
                    }
                    for (int j = 0; !nashel && j < al_znaki_all; j++)
                    {
                        if (shifr[i] == alf_znaki[j])
                        {
                            int nom = 0;
                            if (j - key_g[num_ch_key] >= al_znaki_all)
                                nom = (j - key_g[num_ch_key]) % al_znaki_all;
                            if (j - key_g[num_ch_key] < al_znaki_all && j - key_g[num_ch_key] >=
0)
                                nom = j - key_g[num_ch_key];
                            if (j - key_g[num_ch_key] < 0)
                            {
                                nom = j - key_g[num_ch_key];
                                while (nom < 0)
                                    nom += al_znaki_all;
                            }
                            deshifr[i] = alf_znaki[nom];
                            nashel = true;
                        }
                    }
                    if (!nashel)
                    {
                        deshifr[i] = shifr[i];
                        nashel = true;
                    }
                }
                for (int i = 0; i < length_t; i++)
                {
                    richTextBox3.Text += deshifr[i].ToString();
                }
            }
            if (!cez && !viz && !gron)
            {
                richTextBox3.Text = "<<Сначала нужно что-то зашифровать>>";
            }
        }
    }
}
```