МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Национальный исследовательский
Нижегородский государственный университет им. Н.И. Лобачевского»
(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра теоретической, компьютерной и экспериментальной механики**
Направление подготовки 01.04.02 Прикладная математика и информатика

Направленность (профиль) программы: Математическое моделирование
физико-механических процессов

# ОТЧЕТ

по учебной практике
по получению первичных профессиональных умений и навыков
«Современные компьютерные технологии»

**Выполнила:** студентка группы 3821М1ПМфм
Селезнева Ольга Александровна

**Руководитель:**
Доцент кафедры ТКиЭМ, к.т.н.
Савихин Олег Геннадьевич

Нижний Новгород
2022

# Оглавление

# 1. Лабораторная работа №1

## 1.1. Постановка задачи

Разработать приложения, в каждом из которых демонстрируется работа с отдельным визуальным компонентом Windows, либо одно приложение, в котором используются (определяются основные обработчики событий) все визуальные компоненты Windows.

## 1.2.Теоретическая часть

В Windows можно выделить следующие группы визуальных элементов:

1. Контейнеры в Windows Forms: «GroupBox», «Panel», «FlowLayoutPanel», «TableLayoutPanel», «TabControl», «SplitContainer». Данные компоненты отвечают за размеры элементов и их позиционирование в контейнере.
2. Элементы управления: кнопка «Button», метка «Label».
3. Элементы текстового ввода/вывода: текстовое поле «TextBox», элемент «MaskedTextBox».
4. Элементы переключения/выбора: «Radiobutton» и «CheckBox.ListBox», элемент «ComboBox».
5. Элементы привязки данных: «ListBox» и «ComboBox». А также элементы: «CheckedListBox», «NumericUpDown», «DomainUpDown», «ImageList», «ListView», «TreeView», «TrackBar», «Timer», «ProgressBar», «DateTimePicker», «MonthCalendar», «PictureBox».
6. Окна сообщений и диалогов: «MessageBox,OpenFileDialog», «SaveFileDialog», «FontDialog» и «ColorDialog».
7. Меню и панели инструментов. Панель инструментов: «ToolStrip». Создание меню: «MenuStrip». Строка состояния: «StatusStrip». Контекстное меню: «ContextMenuStrip».

## 1.3.Программная реализация

В программе представлены следующие основные функции:

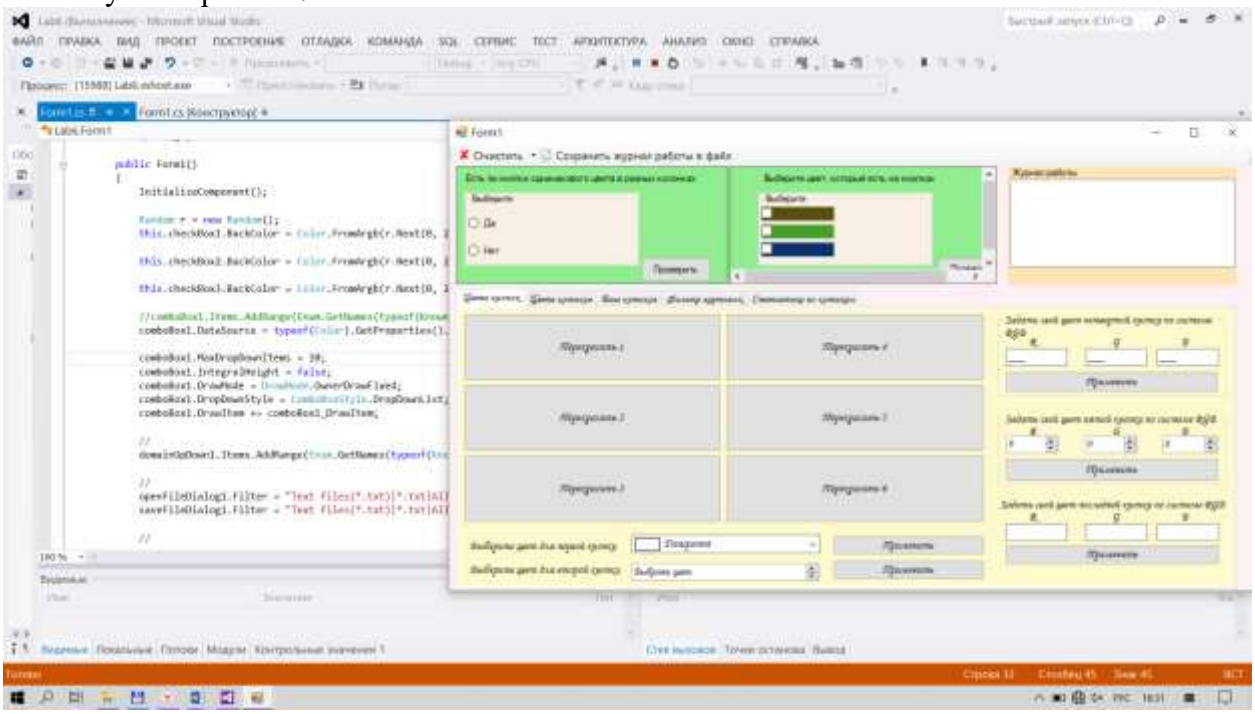
1. Инициализация основных компонент и загрузка формы:

```
public Form1()
{
   InitializeComponent();
```

```
        Random r = new Random();
        this.checkBox1.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

        this.checkBox2.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

        this.checkBox3.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

        //comboBox1.Items.AddRange(Enum.GetNames(typeof(KnownColor)));
        comboBox1.DataSource = typeof(Color).GetProperties().Where(x => x.PropertyType ==
typeof(Color)).Select(x => x.GetValue(null)).ToList();

        comboBox1.MaxDropDownItems = 10;
        comboBox1.IntegralHeight = false;
        comboBox1.DrawMode = DrawMode.OwnerDrawFixed;
        comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
        comboBox1.DrawItem += comboBox1_DrawItem;

        //
        domainUpDown1.Items.AddRange(Enum.GetNames(typeof(KnownColor)));

        //
        openFileDialog1.Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";
        saveFileDialog1.Filter = "Text files(*.txt)|*.txt|All files(*.*)|*.*";

        //

        listView1.SmallImageList = imageList2;

        //
        trackBar1.Scroll += trackBar1_Scroll;

        //
        dataGridView1.Rows.Clear();
        dataGridView1.Columns.Clear();
        //
        dataGridView1.RowCount = 2;
        dataGridView1.ColumnCount = 4;
        dataGridView1.Rows[0].Cells[0].Value = "Меньше 1";
        dataGridView1.Rows[0].Cells[1].Value = "1-7";
        dataGridView1.Rows[0].Cells[2].Value = "7-15";
        dataGridView1.Rows[0].Cells[3].Value = "От 15 лет";


        for (int i = 0; i < 4; i++)
        {
           dataGridView1.Rows[1].Cells[i].Value = 0;
        }

        //
```

```
        dataGridView2.Rows.Clear();
        dataGridView2.Columns.Clear();
        //
        dataGridView2.RowCount = 2;
        dataGridView2.ColumnCount = 7;
        dataGridView2.Rows[0].Cells[0].Value = "Мейн-кун";
        dataGridView2.Rows[0].Cells[1].Value = "Шотландская вислоухая";
        dataGridView2.Rows[0].Cells[2].Value = "Бенгальская кошка";
        dataGridView2.Rows[0].Cells[3].Value = "Сиамская кошка";
        dataGridView2.Rows[0].Cells[4].Value = "Русская голубая";
        dataGridView2.Rows[0].Cells[5].Value = "Сибирская кошка";
        dataGridView2.Rows[0].Cells[6].Value = "Другой вариант";

        for (int i = 0; i < 7; i++)
        {
           dataGridView2.Rows[1].Cells[i].Value = 0;
        }
    }
```

Результат работы:



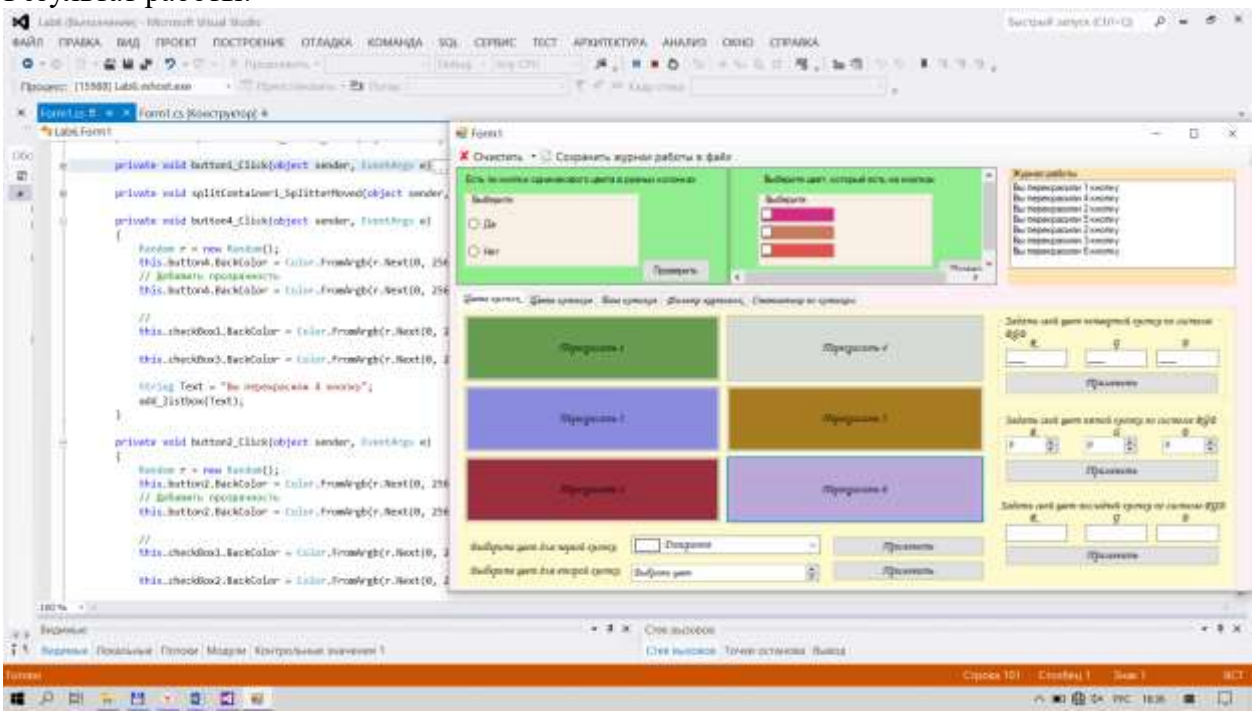
2. Методы, позволяющие перекрасить в рандомные цвета кнопки 1-6 на панели «Цвета кнопок».

```
 private void button1_Click(object sender, EventArgs e)
    {
        Random r = new Random();
        this.button1.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));
        // Добавить прозрачность
```

```
            this.button1.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0, 256),
r.Next(0, 256));

            //
            this.checkBox1.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

            this.checkBox2.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

            this.checkBox3.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

            String Text = "Вы перекрасили 1 кнопку";
            add_listbox(Text);
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Random r = new Random();
            this.button4.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));
            // Добавить прозрачность
            this.button4.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0, 256),
r.Next(0, 256));

            //
            this.checkBox1.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

            this.checkBox3.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

            String Text = "Вы перекрасили 4 кнопку";
            add_listbox(Text);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Random r = new Random();
            this.button2.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));
            // Добавить прозрачность
            this.button2.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0, 256),
r.Next(0, 256));

            //
            this.checkBox1.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

            this.checkBox2.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));
```

```
            String Text = "Вы перекрасили 2 кнопку";
            add_listbox(Text);
        }

        private void button5_Click(object sender, EventArgs e)
        {
            Random r = new Random();
            this.button5.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));
            // Добавить прозрачность
            this.button5.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0, 256),
r.Next(0, 256));

            checkBox2.BackColor = button5.BackColor;

            String Text = "Вы перекрасили 5 кнопку";
            add_listbox(Text);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Random r = new Random();
            this.button3.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));
            // Добавить прозрачность
            this.button3.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0, 256),
r.Next(0, 256));

            //this.checkBox1.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

            this.checkBox2.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));

            this.checkBox3.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));
            String Text = "Вы перекрасили 3 кнопку";
            add_listbox(Text);
        }

        private void button6_Click(object sender, EventArgs e)
        {
            Random r = new Random();
            this.button6.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0,
256));
            // Добавить прозрачность
            this.button6.BackColor = Color.FromArgb(r.Next(0, 256), r.Next(0, 256), r.Next(0, 256),
r.Next(0, 256));
            String Text = "Вы перекрасили 6 кнопку";
            add_listbox(Text);
        }
```

Результат работы:

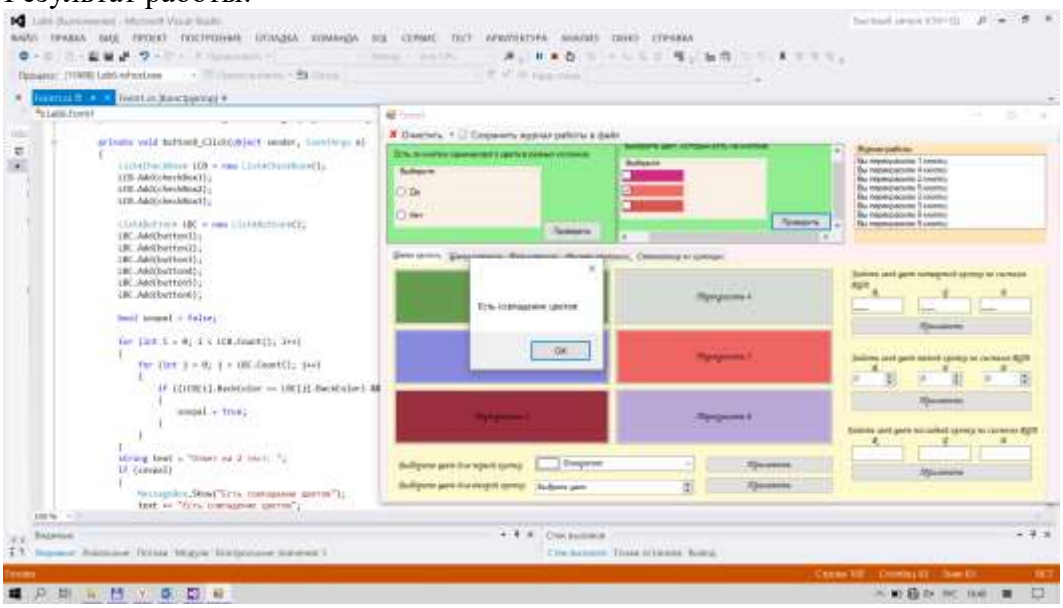


3. Метод, демонстрирующий работу элементов «checkBox». Элементы находятся в «splitContainer». В качестве задания пользователю предлагается найти одинаковые цвета на кнопке и в элементе checkbox, если они есть.

```
private void button8_Click(object sender, EventArgs e)
{
   List<CheckBox> LCB = new List<CheckBox>();
   LCB.Add(checkBox1);
   LCB.Add(checkBox2);
   LCB.Add(checkBox3);

   List<Button> LBC = new List<Button>();
   LBC.Add(button1);
   LBC.Add(button2);
   LBC.Add(button3);
   LBC.Add(button4);
   LBC.Add(button5);
   LBC.Add(button6);

   bool sovpal = false;

   for (int i = 0; i < LCB.Count(); i++)
   {
      for (int j = 0; j < LBC.Count(); j++)
      {
         if ((LCB[i].BackColor == LBC[j].BackColor) && (LCB[i].Checked))
         {
            sovpal = true;
```

```
            }
        }
    }
    string text = "Ответ на 2 тест: ";
    if (sovpal)
    {
        MessageBox.Show("Есть совпадение цветов");
        text += "Есть совпадение цветов";
    }
    else
    {
        MessageBox.Show("Неверно");
        text += "Неверно";
    }

    add_listbox(text);
}
```

Результат работы:

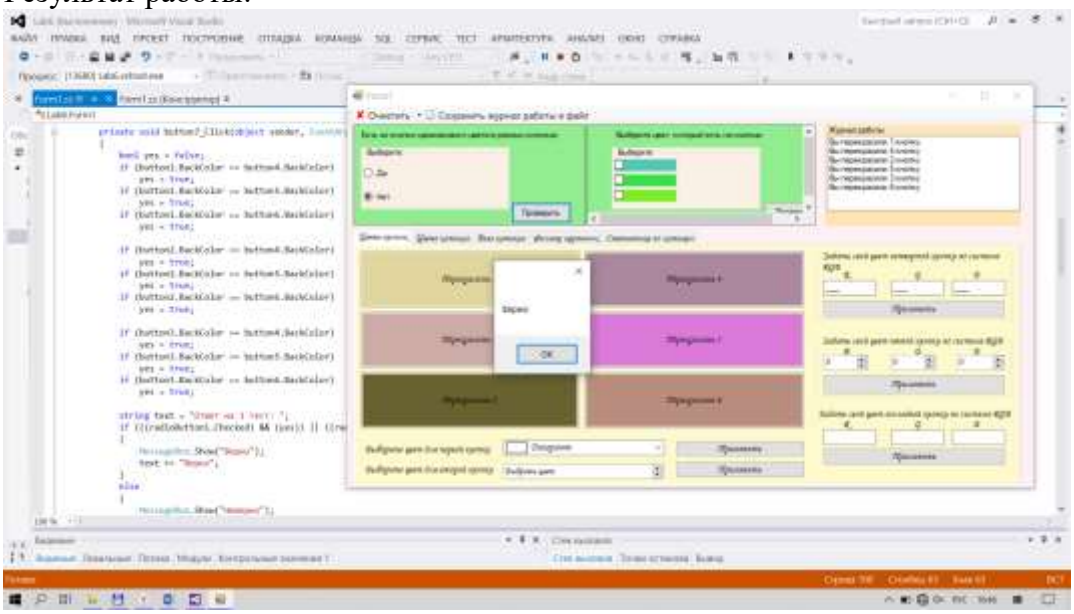


4. Метод, показывающий работу «radioButton». Элементы находятся в «splitContainer».

```
private void button7_Click(object sender, EventArgs e)
{
    bool yes = false;
    if (button1.BackColor == button4.BackColor)
        yes = true;
    if (button1.BackColor == button5.BackColor)
        yes = true;
    if (button1.BackColor == button6.BackColor)
        yes = true;

    if (button2.BackColor == button4.BackColor)
        yes = true;
    if (button2.BackColor == button5.BackColor)
        yes = true;
```

```
        if (button2.BackColor == button6.BackColor)
            yes = true;

        if (button3.BackColor == button4.BackColor)
            yes = true;
        if (button3.BackColor == button5.BackColor)
            yes = true;
        if (button3.BackColor == button6.BackColor)
            yes = true;

        string text = "Ответ на 1 тест: ";
        if (((radioButton1.Checked) && (yes)) || ((radioButton2.Checked) && (!yes)))
        {
            MessageBox.Show("Верно");
            text += "Верно";
        }
        else
        {
            MessageBox.Show("Неверно");
            text += "Неверно";
        }

        add_listbox(text);
    }
```

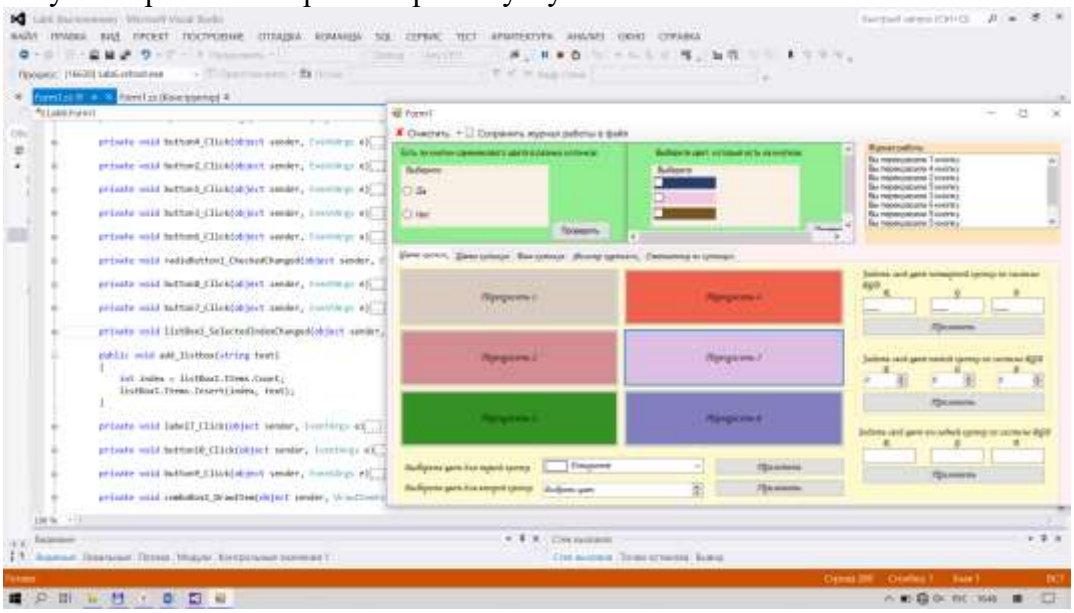
Результат работы:



5. Метод, добавляющий текст в «ListBox». На форме представлено в виде журнала работы.

```
public void add_listbox(string text)
    {
        int index = listBox1.Items.Count;
        listBox1.Items.Insert(index, text);
    }
```

Результат работы в верхнем правом углу:



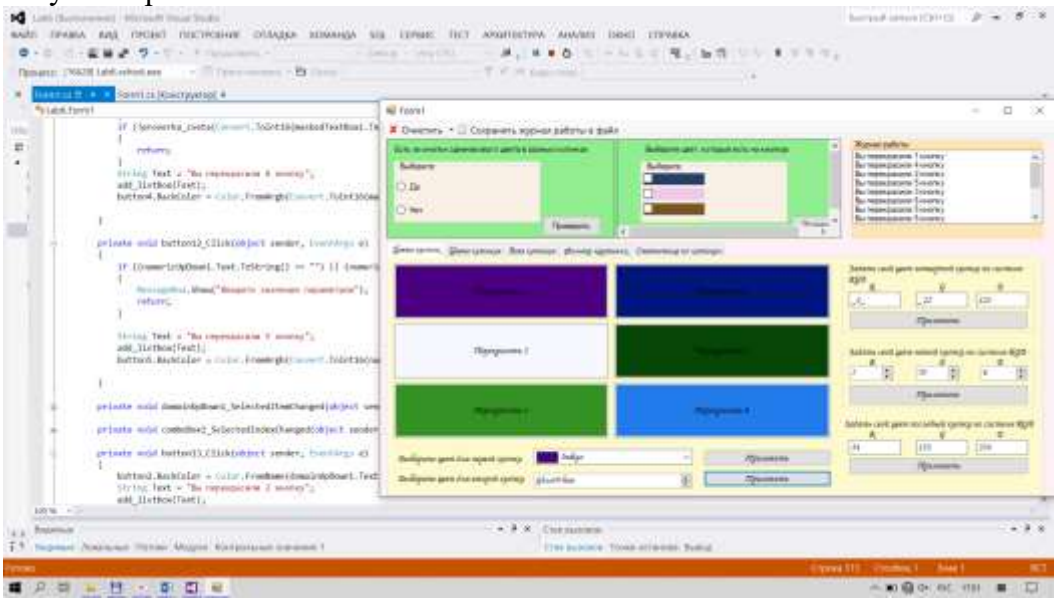
6. **Методы. Позволяющие перекрасить кнопки 1,2,4,5,6 с помощью элементов «textBox», «comboBox», «maskedTextBox», «numericUpDown» и «domainUpDown» соответственно**.

```
private void button10_Click(object sender, EventArgs e)
        {
            if ((textBox1.Text.ToString() == "") || (textBox2.Text.ToString() == "") ||
(textBox3.Text.ToString() == ""))
            {
                MessageBox.Show("Введите значение параметров");
                return;
            }
            if (!proverka_cveta(Convert.ToInt16(textBox1.Text), Convert.ToInt16(textBox2.Text),
Convert.ToInt16(textBox3.Text)))
            {
                return;
            }
            String Text = "Вы перекрасили 6 кнопку";
            add_listbox(Text);
            button6.BackColor = Color.FromArgb(Convert.ToInt16(textBox1.Text),
Convert.ToInt16(textBox2.Text), Convert.ToInt16(textBox3.Text));
        }

        private void button9_Click(object sender, EventArgs e)
        {
            button1.BackColor = Color.FromName(comboBox1.Text);
            String Text = "Вы перекрасили 1 кнопку";
            add_listbox(Text);
        }

        private void comboBox1_DrawItem(object sender, DrawItemEventArgs e)
        {
            e.DrawBackground();
            if (e.Index >= 0)
```

```
        {
            var txt = comboBox1.GetItemText(comboBox1.Items[e.Index]);
            var color = (Color)comboBox1.Items[e.Index];
            var r1 = new Rectangle(e.Bounds.Left + 1, e.Bounds.Top + 1,
                2 * (e.Bounds.Height - 2), e.Bounds.Height - 2);
            var r2 = Rectangle.FromLTRB(r1.Right + 2, e.Bounds.Top,
                e.Bounds.Right, e.Bounds.Bottom);
            using (var b = new SolidBrush(color))
                e.Graphics.FillRectangle(b, r1);
            e.Graphics.DrawRectangle(Pens.Black, r1);
            TextRenderer.DrawText(e.Graphics, txt, comboBox1.Font, r2,
                comboBox1.ForeColor, TextFormatFlags.Left | TextFormatFlags.VerticalCenter);
        }
    }

    public bool proverka_cveta(int r, int g, int b)
    {
        if ((r < 0) || (r > 255))
        {
            MessageBox.Show("Параметр r введен неверно. Проверьте, диапозон от 0 до
255");
            return false;
        }
        if ((g < 0) || (g > 255))
        {
            MessageBox.Show("Параметр g введен неверно. Проверьте, диапозон от 0 до
255");
            return false;
        }
        if ((b < 0) || (b > 255))
        {
            MessageBox.Show("Параметр b введен неверно. Проверьте, диапозон от 0 до
255");
            return false;
        }

        return true;
    }

    private void button11_Click(object sender, EventArgs e)
    {
        if ((maskedTextBox1.Text.ToString() == "") || (maskedTextBox2.Text.ToString() == "")
|| (maskedTextBox3.Text.ToString() == ""))
        {
            MessageBox.Show("Введите значение параметров");
            return;
        }
        if (!proverka_cveta(Convert.ToInt16(maskedTextBox1.Text),
Convert.ToInt16(maskedTextBox2.Text), Convert.ToInt16(maskedTextBox3.Text)))
        {
            return;
        }
```

```
            String Text = "Вы перекрасили 4 кнопку";
            add_listbox(Text);
            button4.BackColor = Color.FromArgb(Convert.ToInt16(maskedTextBox1.Text),
Convert.ToInt16(maskedTextBox2.Text), Convert.ToInt16(maskedTextBox3.Text));

        }

        private void button12_Click(object sender, EventArgs e)
        {
            if ((numericUpDown1.Text.ToString() == "") || (numericUpDown2.Text.ToString() ==
"") || (numericUpDown3.Text.ToString() == ""))
            {
                MessageBox.Show("Введите значение параметров");
                return;
            }

            String Text = "Вы перекрасили 5 кнопку";
            add_listbox(Text);
            button5.BackColor = Color.FromArgb(Convert.ToInt16(numericUpDown1.Text),
Convert.ToInt16(numericUpDown2.Text), Convert.ToInt16(numericUpDown3.Text));

        }

        private void button13_Click(object sender, EventArgs e)
        {
            button2.BackColor = Color.FromName(domainUpDown1.Text);
            String Text = "Вы перекрасили 2 кнопку";
            add_listbox(Text);
        }
```

Результат работы:



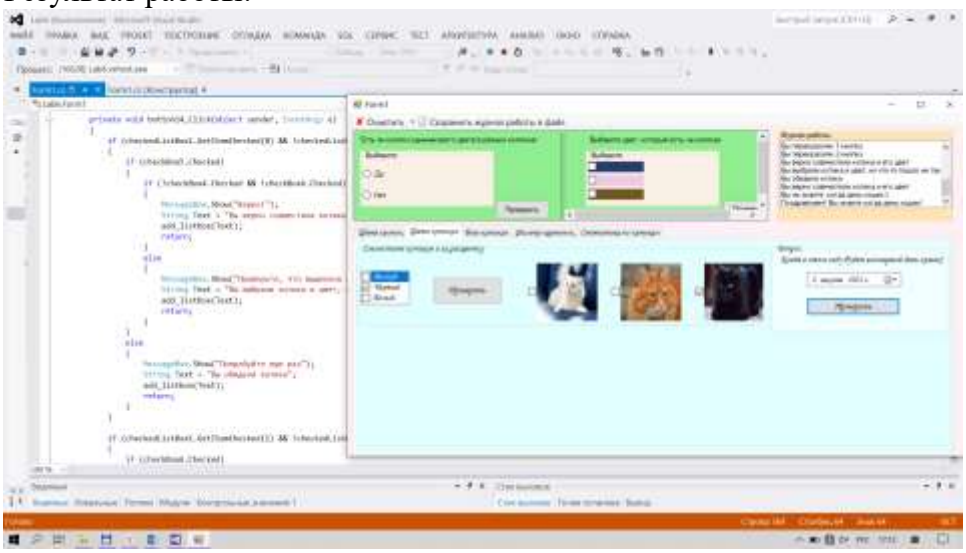
7. **Вкладка «Цвет котика». Предлагает вариант использования элементов «checkedListBox», «checkBox», «MessageBox», «dateTimePicker». Варианты отработки тестов можно увидеть в журнале работы**.

```
        private void button14_Click(object sender, EventArgs e)
        {
            if (checkedListBox1.GetItemChecked(0) && !checkedListBox1.GetItemChecked(1) &&
!checkedListBox1.GetItemChecked(2))
            {
                if (checkBox5.Checked)
                {
                    if (!checkBox4.Checked && !checkBox6.Checked)
                    {
                        MessageBox.Show("Верно!");
                        String Text = "Вы верно совместили котика и его цвет";
                        add_listbox(Text);
                        return;
                    }
                    else
                    {
                        MessageBox.Show("Проверьте, что выделена лишь одна запись и одна
картинка и повторите попытку");
                        String Text = "Вы выбрали котика и цвет, но что-то пошло не так";
                        add_listbox(Text);
                        return;
                    }
                }
                else
                {
                    MessageBox.Show("Попробуйте еще раз");
                    String Text = "Вы обидели котика";
                    add_listbox(Text);
                    return;
                }
            }
            if (checkedListBox1.GetItemChecked(1) && !checkedListBox1.GetItemChecked(0) &&
!checkedListBox1.GetItemChecked(2))
            {
                if (checkBox6.Checked)
                {
                    if (!checkBox4.Checked && !checkBox5.Checked)
                    {
                        MessageBox.Show("Верно!");
                        String Text = "Вы верно совместили котика и его цвет";
                        add_listbox(Text);
                        return;
                    }
                    else
                    {
                        MessageBox.Show("Проверьте, что выделена лишь одна запись и одна
картинка и повторите попытку");
                        String Text = "Вы выбрали котика и цвет, но что-то пошло не так";
                        add_listbox(Text);
                        return;
                    }
                }
```

```
            else
            {
                MessageBox.Show("Попробуйте еще раз");
                String Text = "Вы обидели котика";
                add_listbox(Text);
                return;
            }
        }
        if ( checkedListBox1.GetItemChecked(2) && !checkedListBox1.GetItemChecked(1)
&& !checkedListBox1.GetItemChecked(0))
        {
            if (checkBox4.Checked)
            {
                if (!checkBox5.Checked && !checkBox6.Checked)
                {
                    MessageBox.Show("Верно!");
                    String Text = "Вы верно совместили котика и его цвет";
                    add_listbox(Text);
                    return;
                }
                else
                {
                    MessageBox.Show("Проверьте, что выделена лишь одна запись и одна
картинка и повторите попытку");
                    String Text = "Вы выбрали котика и цвет, но что-то пошло не так";
                    add_listbox(Text);
                    return;
                }
            }
            else
            {
                MessageBox.Show("Попробуйте еще раз");
                String Text = "Вы обидели котика";
                add_listbox(Text);
                return;
            }
        }
        MessageBox.Show("Проверьте, что выделена лишь одна запись и одна картинка и
повторите попытку");
    }
    private void button15_Click(object sender, EventArgs e)
    {
        string text = "День кошек выбран ";
        string otvet = "8 августа 2022 г.";
        if (dateTimePicker1.Text == otvet)
        {
            MessageBox.Show("верно!");
            text += "верно!";
            String Text = "Поздравляем! Вы знаете когда день кошек!";
            add_listbox(Text);
        }
        else
```

```
        {
            MessageBox.Show("неверно");
            text += "неверно";
            String Text = "Вы не знаете, когда день кошек:(";
            add_listbox(Text);
        }

        //add_listbox(text);
    }
```

Результат работы:

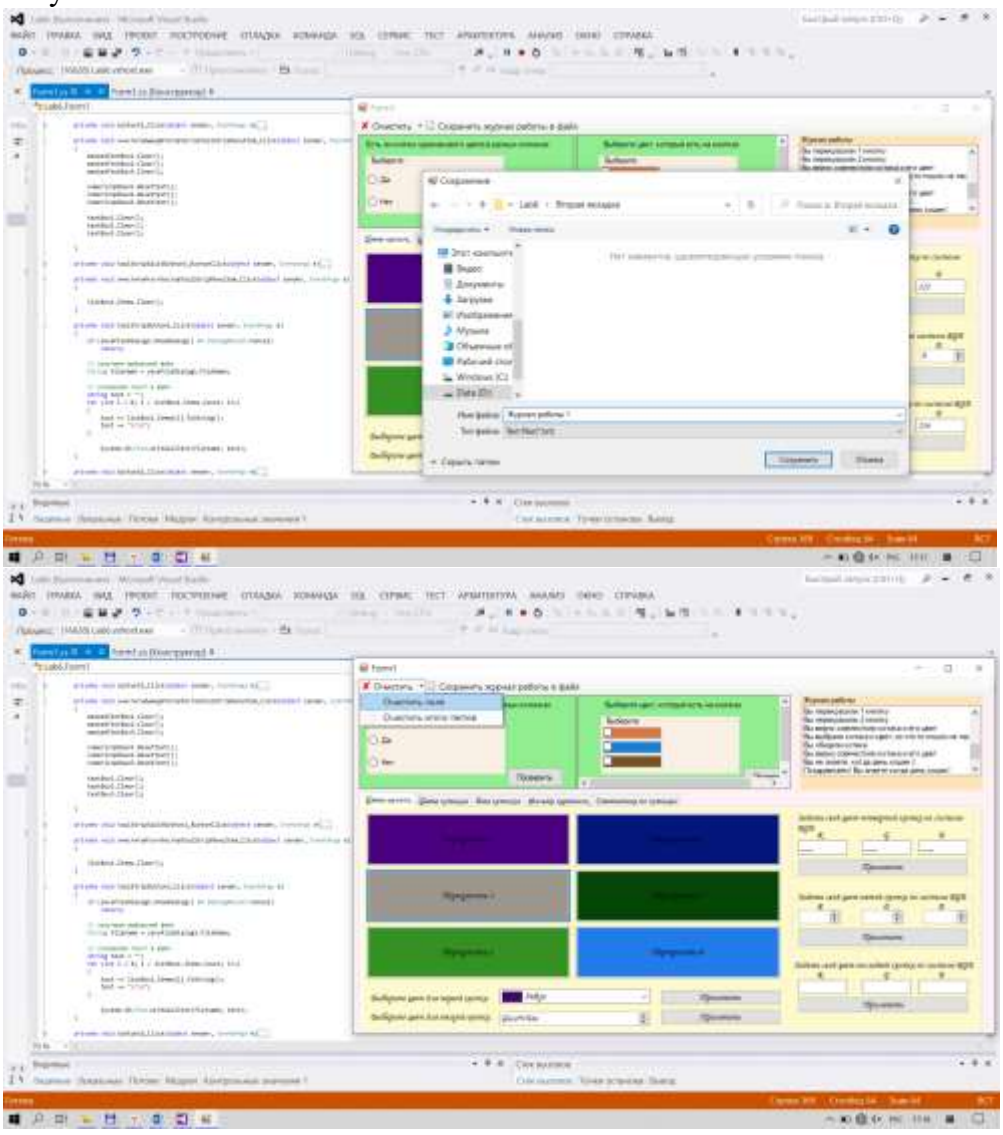


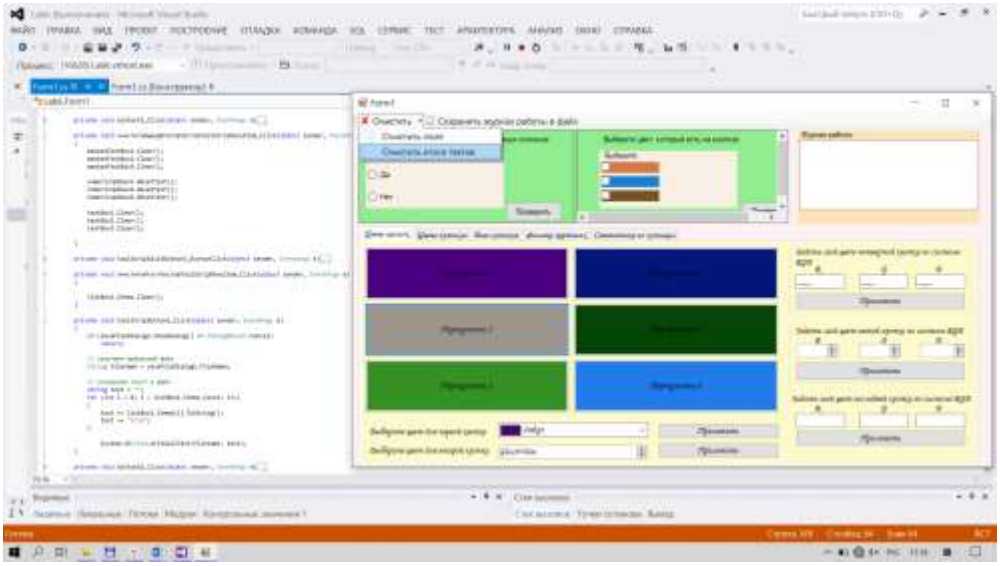
8. Вариант работы с визуальными элементами maskedTextBox, numericUpDown, textbox, listBox, saveFileDialog.

```
    private void очиститьВыводИтоговТестовToolStripMenuItem_Click(object sender, EventArgs e)
    {
        maskedTextBox1.Clear();
        maskedTextBox2.Clear();
        maskedTextBox3.Clear();

        numericUpDown1.ResetText();
        numericUpDown2.ResetText();
        numericUpDown3.ResetText();

        textBox1.Clear();
        textBox2.Clear();
        textBox3.Clear();

    }

    private void очиститьИтогиТестовToolStripMenuItem_Click(object sender, EventArgs e)
    {

        listBox1.Items.Clear();
```

```
    }

    private void toolStripButton1_Click(object sender, EventArgs e)
    {
        if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)
            return;

        // получаем выбранный файл
        String filename = saveFileDialog1.FileName;

        // сохраняем текст в файл
        string text = "";
        for (int i = 0; i < listBox1.Items.Count; i++)
        {
            text += listBox1.Items[i].ToString();
            text += "\r\n";
        }

            System.IO.File.WriteAllText(filename, text);
    }
```

Результаты:



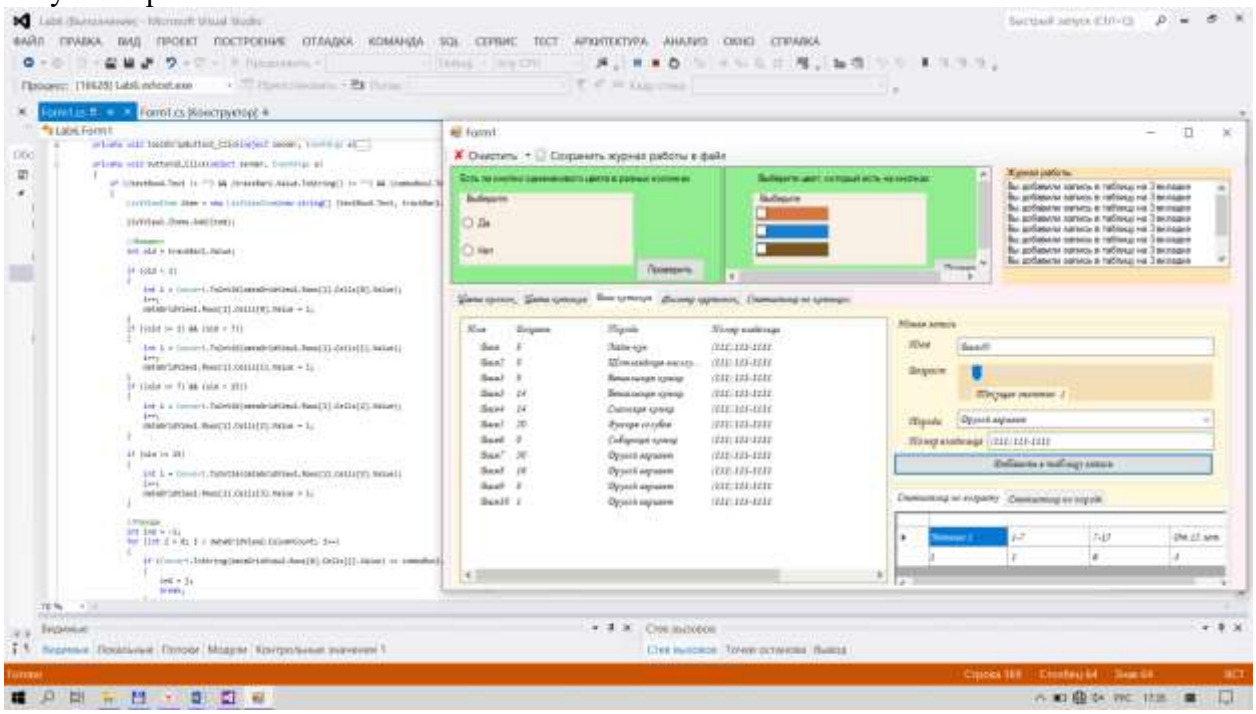


9. Вкладка «База котиков». Демонстрирует способ использования элементов: textbox, trackBar, maskedTextBox, comboBox, ListViewItem, dataGridView.

```
private void button16_Click(object sender, EventArgs e)
{
    if ((textBox4.Text != "") && (trackBar1.Value.ToString() != "") &&
(comboBox2.Text.ToString() != "") && (maskedTextBox4.Text.ToString() != ""))
    {
        ListViewItem item = new ListViewItem(new string[] {textBox4.Text,
trackBar1.Value.ToString(), comboBox2.GetItemText(comboBox2.SelectedItem),
maskedTextBox4.Text.ToString()});

        listView1.Items.Add(item);

        //Возраст
        int old = trackBar1.Value;

        if (old < 1)
        {
            int i = Convert.ToInt16(dataGridView1.Rows[1].Cells[0].Value);
            i++;
            dataGridView1.Rows[1].Cells[0].Value = i;
        }
        if ((old >= 1) && (old < 7))
        {
            int i = Convert.ToInt16(dataGridView1.Rows[1].Cells[1].Value);
            i++;
            dataGridView1.Rows[1].Cells[1].Value = i;
        }
        if ((old >= 7) && (old < 15))
        {
            int i = Convert.ToInt16(dataGridView1.Rows[1].Cells[2].Value);
            i++;
```

```
            dataGridView1.Rows[1].Cells[2].Value = i;
        }

        if (old >= 15)
        {
            int i = Convert.ToInt16(dataGridView1.Rows[1].Cells[3].Value);
            i++;
            dataGridView1.Rows[1].Cells[3].Value = i;
        }

        //Порода
        int ind = -1;
        for (int j = 0; j < dataGridView2.ColumnCount; j++)
        {
            if (Convert.ToString(dataGridView2.Rows[0].Cells[j].Value) ==
comboBox2.GetItemText(comboBox2.SelectedItem))
            {
                ind = j;
                break;
            }
        }
        int k = Convert.ToInt16(dataGridView2.Rows[1].Cells[ind].Value);
        k++;
        dataGridView2.Rows[1].Cells[ind].Value = k;


        String Text = "Вы добавили запись в таблицу на 3 вкладке";
        add_listbox(Text);
      }
    }
```

Результат работы:



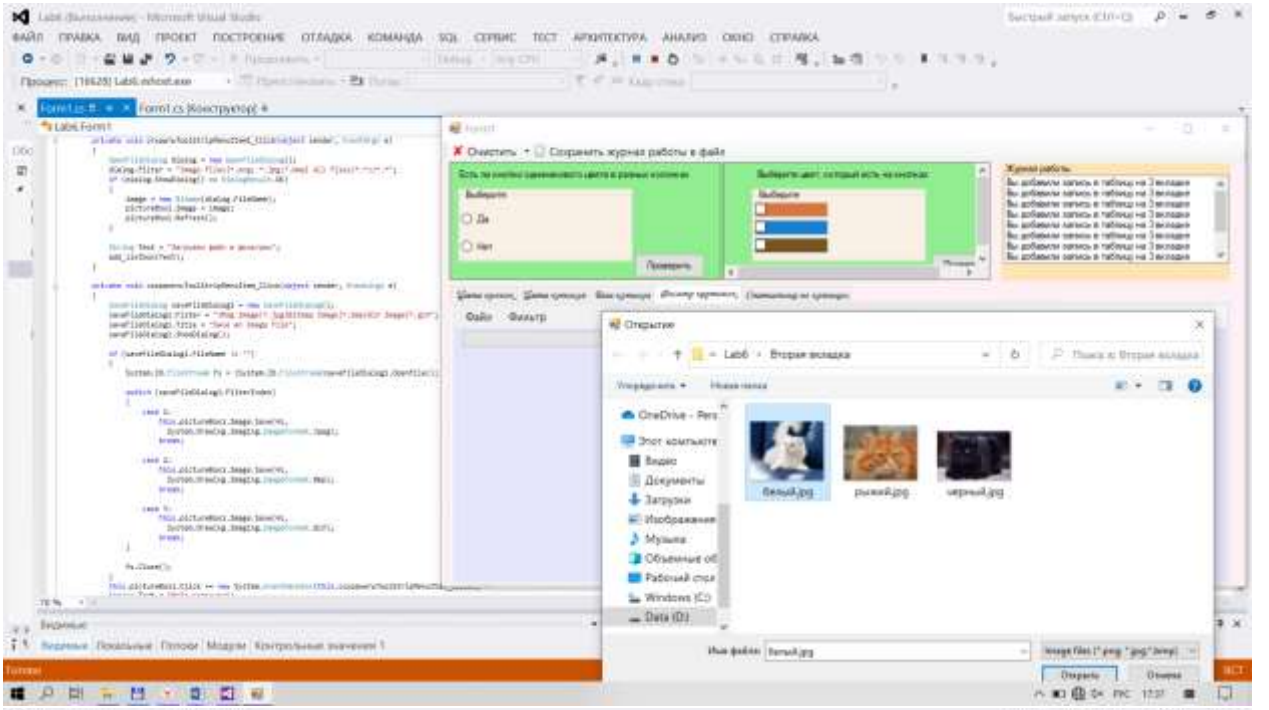
10. Вкладка «Фильтр картинок». Демонстрирует вариант работы таких элементов формы, как OpenFileDialog, SaveFileDialog, backgroundWorker, progressBar, pictureBox, label, colorDialog.

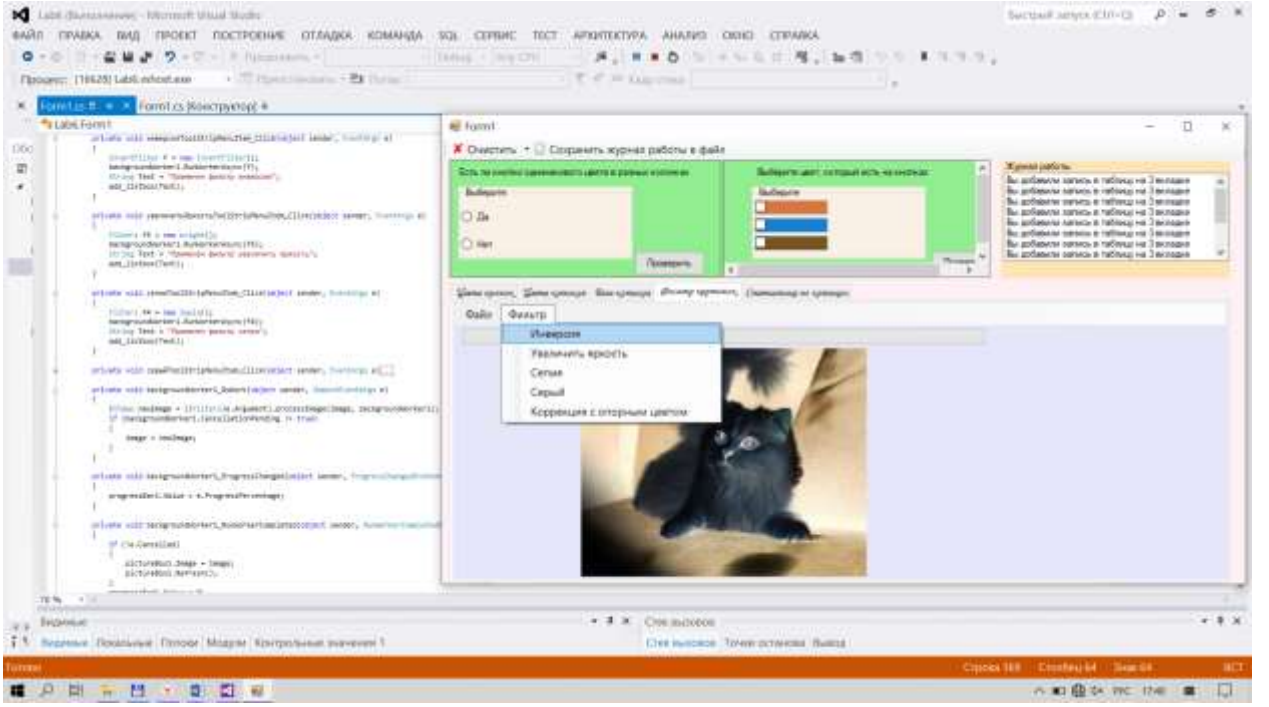
```
private void открытьToolStripMenuItem1_Click(object sender, EventArgs e)
{
   OpenFileDialog dialog = new OpenFileDialog();
   dialog.Filter = "Image files|*.png; *.jpg;*.bmp| All files(*.*)|*.*";
   if (dialog.ShowDialog() == DialogResult.OK)
   {
      image = new Bitmap(dialog.FileName);
      pictureBox1.Image = image;
      pictureBox1.Refresh();
   }

   String Text = "Загружен файл в фильтрах";
   add_listbox(Text);
}

private void сохранитьToolStripMenuItem_Click(object sender, EventArgs e)
{
   SaveFileDialog saveFileDialog1 = new SaveFileDialog();
   saveFileDialog1.Filter = "JPeg Image|*.jpg|Bitmap Image|*.bmp|Gif Image|*.gif";
   saveFileDialog1.Title = "Save an Image File";
   saveFileDialog1.ShowDialog();

   if (saveFileDialog1.FileName != "")
   {
      System.IO.FileStream fs = (System.IO.FileStream)saveFileDialog1.OpenFile();

      switch (saveFileDialog1.FilterIndex)
      {
         case 1:
            this.pictureBox1.Image.Save(fs,
              System.Drawing.Imaging.ImageFormat.Jpeg);
            break;

         case 2:
            this.pictureBox1.Image.Save(fs,
              System.Drawing.Imaging.ImageFormat.Bmp);
            break;

         case 3:
            this.pictureBox1.Image.Save(fs,
              System.Drawing.Imaging.ImageFormat.Gif);
            break;
      }

      fs.Close();
   }
```

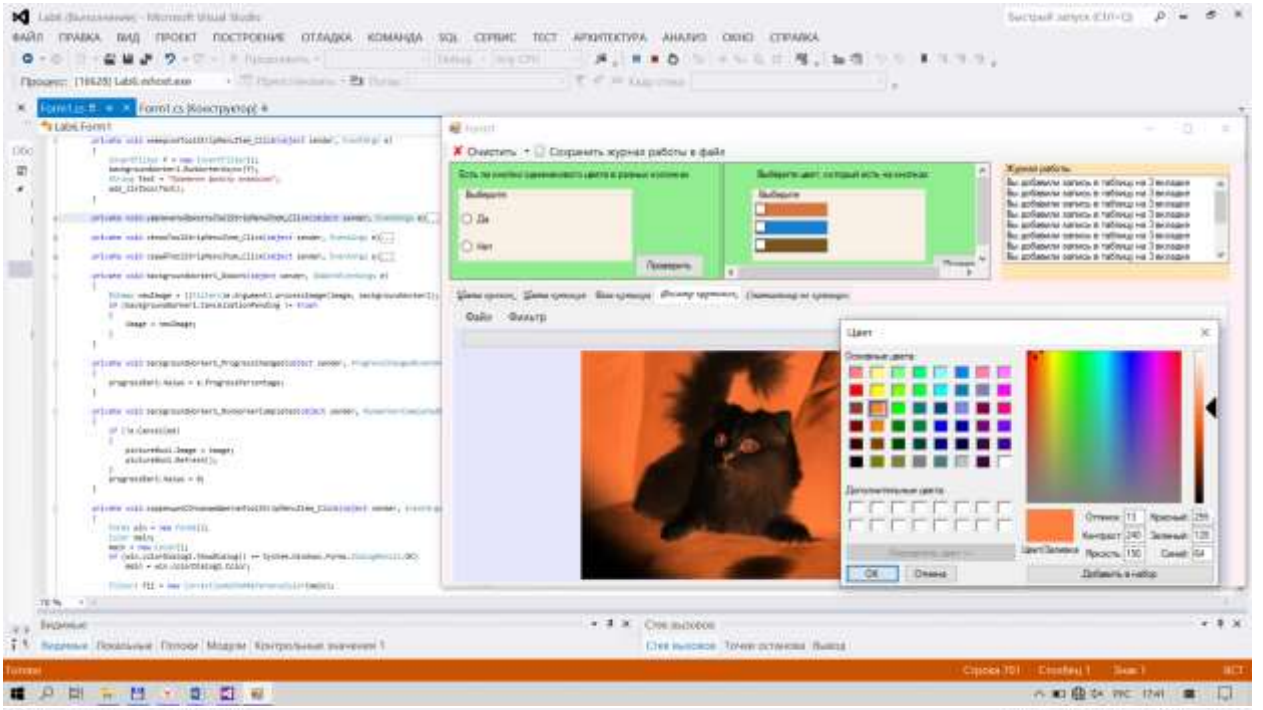
```
            this.pictureBox1.Click += new
System.EventHandler(this.сохранитьToolStripMenuItem_Click);
            String Text = "Файл сохранен";
            add_listbox(Text);
        }

        private void инверсияToolStripMenuItem_Click(object sender, EventArgs e)
        {
            InvertFilter f = new InvertFilter();
            backgroundWorker1.RunWorkerAsync(f);
            String Text = "Применен фильтр инверсия";
            add_listbox(Text);
        }

        private void увеличитьЯркостьToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Filters f5 = new bright();
            backgroundWorker1.RunWorkerAsync(f5);
            String Text = "Применен фильтр увеличить яркость";
            add_listbox(Text);
        }

        private void сепияToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Filters f4 = new Sepia();
            backgroundWorker1.RunWorkerAsync(f4);
            String Text = "Применен фильтр сепия";
            add_listbox(Text);
        }

        private void серыйToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Filters f3 = new GrayScale();
            backgroundWorker1.RunWorkerAsync(f3);
            String Text = "Применен серый фильтр";
            add_listbox(Text);
        }

        private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs e)
        {
            Bitmap newImage = ((Filters)e.Argument).processImage(image, backgroundWorker1);
            if (backgroundWorker1.CancellationPending != true)
            {
                image = newImage;
            }
        }

        private void backgroundWorker1_ProgressChanged(object sender,
ProgressChangedEventArgs e)
        {
            progressBar1.Value = e.ProgressPercentage;
        }
```

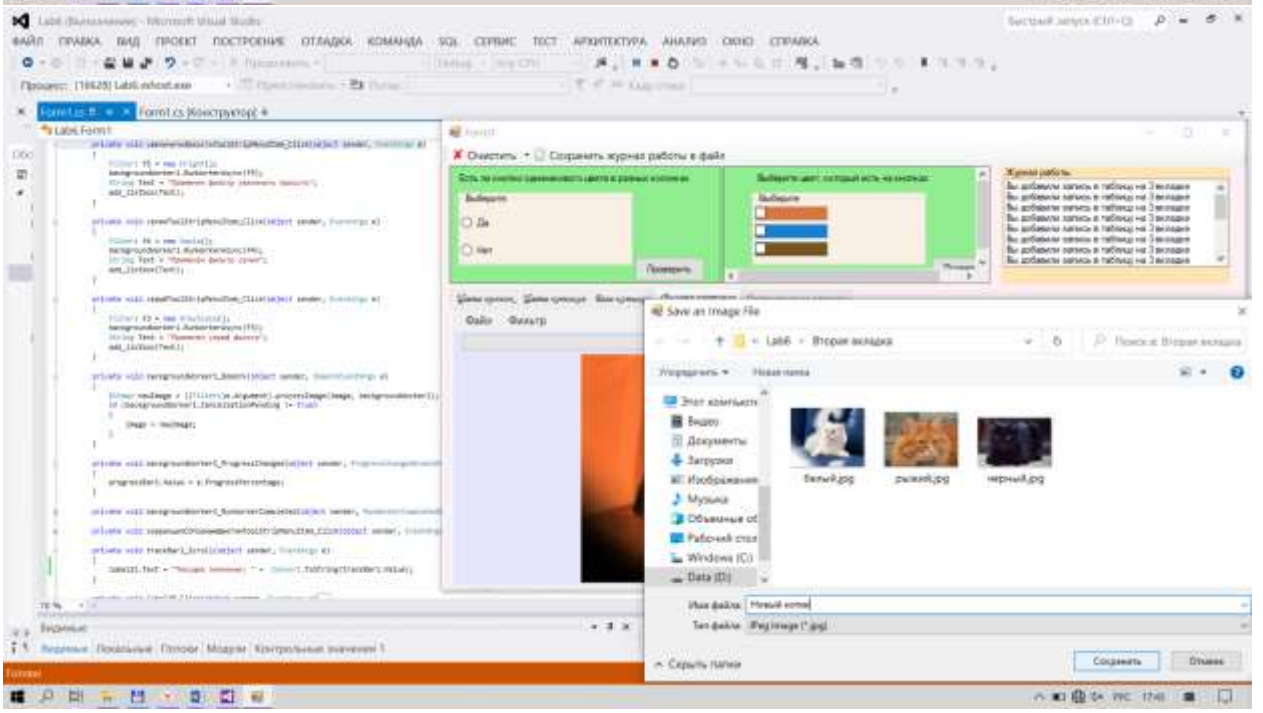
```
        private void backgroundWorker1_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
            if (!e.Cancelled)
            {
                pictureBox1.Image = image;
                pictureBox1.Refresh();
            }
            progressBar1.Value = 0;
        }

        private void коррекцияСОпорнымЦветомToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Form1 win = new Form1();
            Color main;
            main = new Color();
            if (win.colorDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK)
                main = win.colorDialog1.Color;

            Filters f11 = new CorrectionWithAReferenceColor(main);

            backgroundWorker1.RunWorkerAsync(f11);
            String Text = "Применена коррекция изображения с опорным цветом";
            add_listbox(Text);
        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {
            label21.Text = "Текущее значение: " +  Convert.ToString(trackBar1.Value);
        }
```

Результаты работы:

1. 

2. 

3. 

4. 

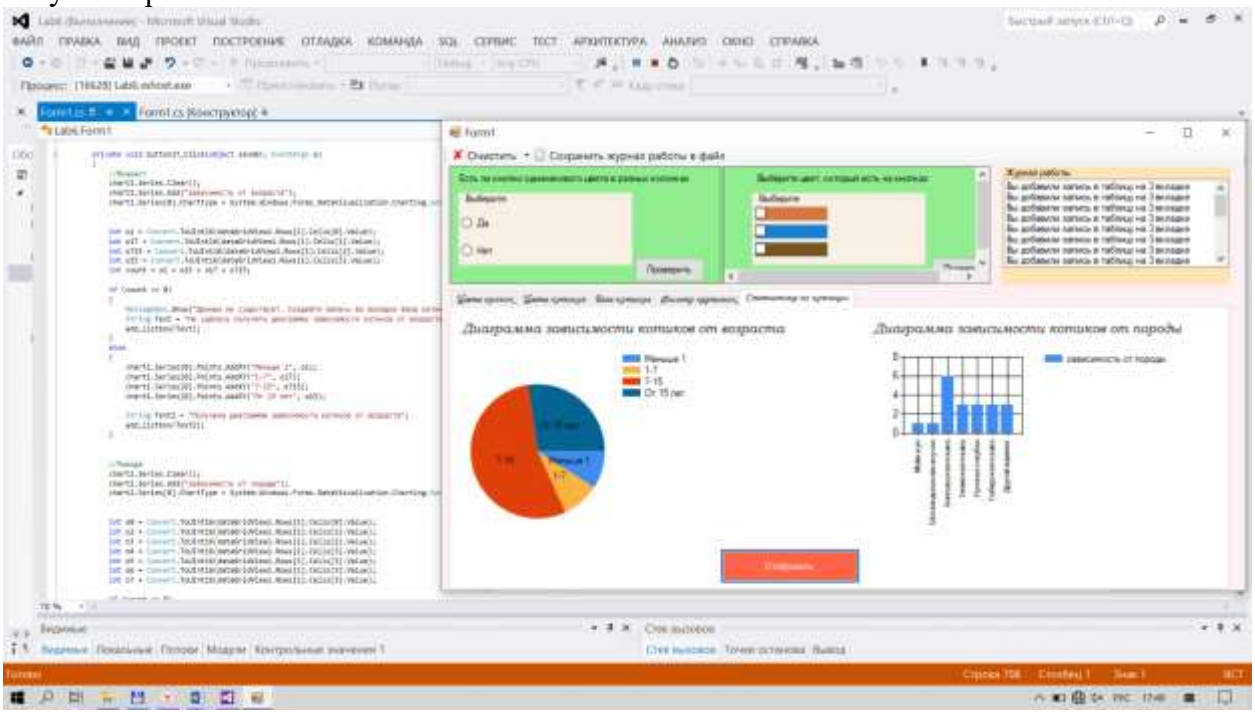
11. Вариант работы графического элемента chart во вкладке «Статистика по котикам».

```
private void button17_Click(object sender, EventArgs e)
{
    //Возраст
    chart1.Series.Clear();
    chart1.Series.Add("зависимость от возраста");
    chart1.Series[0].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;
```

```
        int o1 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[0].Value);
        int o17 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[1].Value);
        int o715 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[2].Value);
        int o15 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[3].Value);
        int count = o1 + o15 + o17 + o715;

        if (count <= 0)
        {
            MessageBox.Show("Данных не существует. Создайте запись во вкладке База
котиков ");
            String Text = "Не удалось получить диаграмму зависимости котиков от возраста.
База котиков пуста";
            add_listbox(Text);
        }
        else
        {
            chart1.Series[0].Points.AddXY("Меньше 1", o1);
            chart1.Series[0].Points.AddXY("1-7", o17);
            chart1.Series[0].Points.AddXY("7-15", o715);
            chart1.Series[0].Points.AddXY("От 15 лет", o15);

            String Text2 = "Получена диаграмма зависимости котиков от возраста";
            add_listbox(Text2);
        }


        //Порода
        chart2.Series.Clear();
        chart2.Series.Add("зависимость от породы");
        chart2.Series[0].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;


        int o0 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[0].Value);
        int o2 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[1].Value);
        int o3 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[2].Value);
        int o4 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[3].Value);
        int o5 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[3].Value);
        int o6 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[3].Value);
        int o7 = Convert.ToUInt16(dataGridView1.Rows[1].Cells[3].Value);

        if (count <= 0)
        {
            MessageBox.Show("Данных не существует. Создайте запись во вкладке База
котиков ");
            String Text = "Не удалось получить диаграмму зависимости котиков от породы.
База котиков пуста";
            add_listbox(Text);
        }
        else
        {
            chart2.Series[0].Points.AddXY("Мейн-кун", o0);
```

```
            chart2.Series[0].Points.AddXY("Шотландская вислоухая", o2);
            chart2.Series[0].Points.AddXY("Бенгальская кошка", o3);
            chart2.Series[0].Points.AddXY("Сиамская кошка", o4);
            chart2.Series[0].Points.AddXY("Русская голубая ", o5);
            chart2.Series[0].Points.AddXY("Сибирская кошка ", o6);
            chart2.Series[0].Points.AddXY("Другой вариант", o7);

            String Text2 = "Получена диаграмма зависимости котиков от породы";
            add_listbox(Text2);
        }
    }
}
```

Результат работы:



# 2. Лабораторная работа №2

## 2.1.Постановка задачи

Применяя примитивы синхронизации (Semaphore, SemaphoreSlim, Mutex, Monitor, Barrier, ReaderWriterLock) и структуры данных для конкурентного использования (BlockingCollection, ConcurrentBag, ConcurrentQueue, ConcurrentStack) разработать программу для решения следующей задачи синхронизации: «Задача о обедающих философах».

Провести исследования, как влияет на скорость выполнения программы использование различных объектов синхронизации.

## 2.2.Теоретическая часть

Задача об обедающих философах — классический пример, используемый в информатике для иллюстрации проблем синхронизации при разработке параллельных алгоритмов и техник решения этих проблем.

Задача была сформулирована в 1965 году Эдсгером Дейкстрой как экзаменационное упражнение для студентов. В качестве примера был взят конкурирующий доступ к ленточному накопителю. Вскоре задача была сформулирована Энтони Хоаром в том виде, в каком она известна сегодня.

Постановка задачи:

Пять безмолвных философов сидят вокруг круглого стола, перед каждым философом стоит тарелка спагетти. Вилки лежат на столе между каждой парой ближайших философов.

Каждый философ может либо есть, либо размышлять. Приём пищи не ограничен количеством оставшихся спагетти — подразумевается бесконечный запас. Тем не менее, философ может есть только тогда, когда держит две вилки — взятую справа и слева (альтернативная формулировка проблемы подразумевает миски с рисом и палочки для еды вместо тарелок со спагетти и вилок).

Каждый философ может взять ближайшую вилку (если она доступна) или положить — если он уже держит её. Взятие каждой вилки и возвращение её на стол являются раздельными действиями, которые должны выполняться одно за другим.

Вопрос задачи заключается в том, чтобы разработать модель поведения (параллельный алгоритм), при котором ни один из философов не будет голодать, то есть будет вечно чередовать приём пищи и размышления.

Решение:

Пример ниже показывает решение, где вилки не представляются явно. Философы могут есть, если ни один из их соседей не ест.

В отсутствие блокировок, связанных с вилками, философы должны обеспечивать то, что начало принятия пищи не основывается на старой информации о состоянии соседей. Например: Если философ Б видит, что А не ест в данный момент времени, а потом поворачивается и смотрит на В, А мог начать есть, пока философ Б смотрит на В. Используя одну взаимоисключающую блокировку (Мьютекс), можно избежать этой проблемы. Эта блокировка не связана с вилками, но она связана с решением процедур, которые могут изменить состояние философов. Это обеспечивается монитором.

Алгоритм монитора реализует схему «проверить, взять и положить» и совместно использует взаимоисключающую блокировку. Заметьте, что философы, желающие есть, не будут иметь вилок.

Если монитор разрешает философу, желающему есть, действовать, то философ снова завладевает первой вилкой, прежде чем взять уже свободную вторую.

По окончании текущего приёма пищи философ оповещает монитор о том, что обе вилки свободны.

Стоит заметить, что этот алгоритм монитора не решает проблемы голодания. Например, философ Б может бесконечно ждать своей очереди, если у философов А и В периоды приёма пищи всё время пересекаются. Чтобы гарантировать также, что ни один философ не будет голодать, можно отслеживать, сколько раз голодный философ не ел, когда его соседи положили вилки на стол. Если количество раз превысит некий предел, такой философ перейдёт в состояние Голодания и алгоритм монитора форсирует процедуру завладения вилками, выполняя условие недопущения голодания ни одного из соседей.

Философ, не имеющий возможности взять вилки из-за того, что его сосед голодает, находится в режиме полезного ожидания окончания приёма пищи соседом его соседа. Эта дополнительная зависимость снижает параллелизм. Увеличение значения порога перехода в состояние Голодание уменьшает этот эффект.

В программе будет представлено два решения с помощью монитора и семафора, чтобы оценить скорость выполнения программы при разных объектах синхронизации. Поскольку задача является по сути бесконечной, то для определенности будем считать ее выполненной, когда каждый философ поест хотя бы раз. Среднее время такого решения будем вычислять по медиане из нескольких серий экспериментов.

## 2.3.Программная реализация

```
class Fork
   {
      public Semaphore sem = new Semaphore(1, 1);
   }

class Philosopher
   {
      public int Id; //порядковый номер философа
      public Fork LeftFork;
      public Fork RightFork;
      public bool CurrentLeftFork = false;
      public bool CurrentRightFork = false;
      public Condition Status; // текущее состояние философа
      Random rand = new Random(); // рандомная задержка между состояниями философов
      public bool StopEat = false; // выход из цикла

      public enum Condition { Rest, Eat, Think } // состояния философа
      public Philosopher() { }
      public Philosopher(int id, Fork Left_Fork, Fork Right_Fork)
      {
         Id = id;
         LeftFork = Left_Fork;
         RightFork = Right_Fork;
         Status = Condition.Rest;
      }

      /*----------------------------------------*/
      //Через монитор

      public void Run()
      {
         //Запускаем попытку есть
         Task.Run(() => TryingToEat());
      }
      //Попытка поесть
      public void TryingToEat()
      {
         do
         {
            if (Status == Condition.Rest)
            {
               TakeLeftFork();

               if (CurrentLeftFork)
               {
                  TakeRightFork();

                  if (CurrentRightFork)
                  {
                     Eat();
```

```
                    PutForks();
                    Think();
                }
                else
                {
                    PutForks();
                    Rest();
                }
            }
            else
                Rest();
        }
        else
            Status = Condition.Rest;
    }
    while (!StopEat);

}

//Попытка получить левую вилку
public void TakeLeftFork()
{
    Monitor.TryEnter(LeftFork, ref CurrentLeftFork);
    //второй параметра - логическое значение, которое указывает, получено ли
владение над объектом из первого параметра
}
//Попытка получить правую вилку
public void TakeRightFork()
{
    Monitor.TryEnter(RightFork, ref CurrentRightFork);
    //второй параметра - логическое значение, которое указывает, получено ли
владение над объектом из первого параметра
}
//Положить левую вилку
public void PutLeftFork()
{
    if (CurrentLeftFork)
    {
        CurrentLeftFork = false;
        Monitor.Exit(LeftFork);
    }
}
//Положить правую вилку
public void PutRightFork()
{
    if (CurrentRightFork)
    {
        CurrentRightFork = false;
        Monitor.Exit(RightFork);
    }
}
//Положить обе вилки
```

```
public void PutForks()
{
   PutLeftFork();
   PutRightFork();
}
/*----------------------------------------*/


/*----------------------------------------*/
//Через семафор

public void RunS()
{
   //Запускаем попытку есть
   Task.Run(() => TryingToEat2());
}
//Попытка поесть
public void TryingToEat2()
{
   do
   {
      if (Status == Condition.Rest)
      {
         TakeLeftForkS();

         if (CurrentLeftFork)
         {
            TakeRightForkS();

            if (CurrentRightFork)
            {
               Eat();
               PutForksS();
               Think();
            }
            else
            {
               PutForksS();
               Rest();
            }
         }
         else
            Rest();
      }
      else
         Status = Condition.Rest;
   }
   while (!StopEat);

}

//Попытка получить левую вилку
```

```
public void TakeLeftForkS()
{
   if (LeftFork.sem.WaitOne())
      CurrentLeftFork = true;
}
//Попытка получить правую вилку
public void TakeRightForkS()
{
   if (RightFork.sem.WaitOne())
      CurrentRightFork = true;
}
//Положить левую вилку
public void PutLeftForkS()
{
   if (CurrentLeftFork)
   {
      CurrentLeftFork = false;
      LeftFork.sem.Release();
   }
}
//Положить правую вилку
public void PutRightForkS()
{
   if (CurrentRightFork)
   {
      CurrentRightFork = false;
      RightFork.sem.Release();
   }
}
//Положить обе вилки
public void PutForksS()
{
   PutLeftForkS();
   PutRightForkS();
}
/*----------------------------------------*/

// Изменение статуса философа
public void Eat()
{
   Status = Condition.Eat;
   waiting();
}

public void Think()
{
   Status = Condition.Think;
   waiting();
}

public void Rest()
{
```

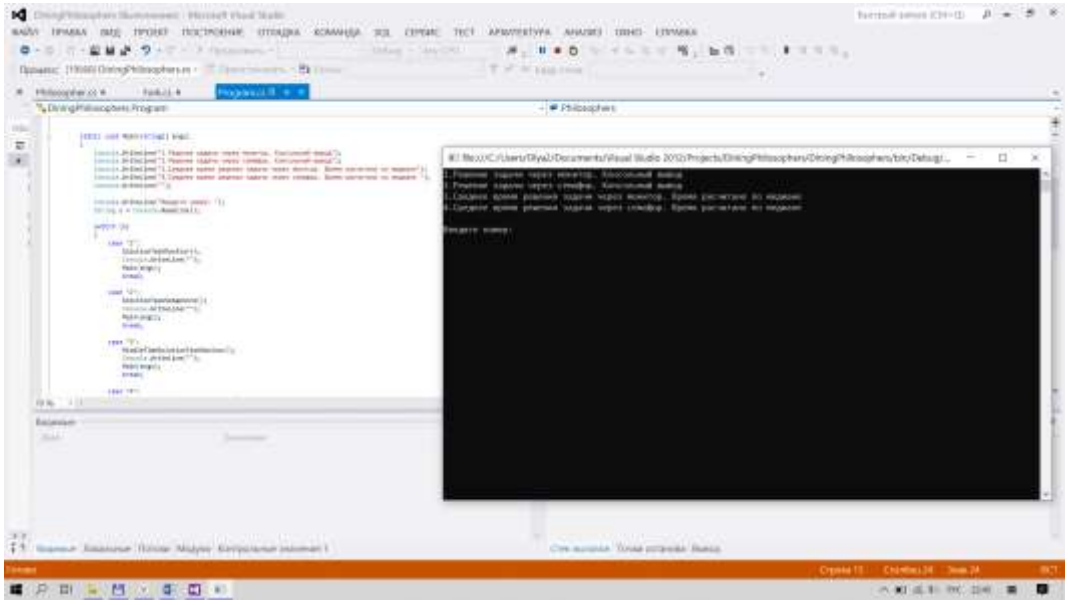
```
            Status = Condition.Rest;
            waiting();
        }

        //Получить текущий статус
        public String GetStatus()
        {
            if (Status == Condition.Eat)
                return " философ ест";
            if (Status == Condition.Rest)
                return " философ отдыхает";
            if (Status == Condition.Think)
                return " философ мыслит";
            else
                return "Статус не определен";
        }

        //Выжидание
        public void waiting()
        {
            //rand.Next(100)
            Thread.Sleep(100);
        }

        //
        public void Stop_Eat()
        {
            StopEat = true;
        }
    }

class Program
    {
        public static List<Fork> Forks;
        public static List<Philosopher> Philosophers;

        public static DateTime TimeToStop;
        public static Int16 Seconds; // продолжительность обеда

        public static HashSet<int> All = new HashSet<int>();
```

1. **Основной вид программы**

```
        static void Main(string[] args)
        {
            Console.WriteLine("1.Решение задачи через монитор. Консольный вывод");
            Console.WriteLine("2.Решение задачи через семафор. Консольный вывод");
            Console.WriteLine("3.Среднее время решения задачи через монитор. Время
расчитано по медиане");
            Console.WriteLine("4.Среднее время решения задачи через семафор. Время
расчитано по медиане ");
            Console.WriteLine("");
```

```
            Console.WriteLine("Введите номер: ");
            String s = Console.ReadLine();

            switch (s)
            {
                case "1":
                    SolutionTaskMonitor();
                    Console.WriteLine("");
                    Main(args);
                    break;

                case "2":
                    SolutionTaskSemaphore();
                    Console.WriteLine("");
                    Main(args);
                    break;

                case "3":
                    MiddleTimeSolutionTaskMonitor();
                    Console.WriteLine("");
                    Main(args);
                    break;

                case "4":
                    MiddleTimeSolutionTaskSemaphore();
                    Console.WriteLine("");
                    Main(args);
                    break;
                default:
                    Console.WriteLine("Нет такого варианта. Повторите попытку еще раз");
                    break;
            }
        }
```

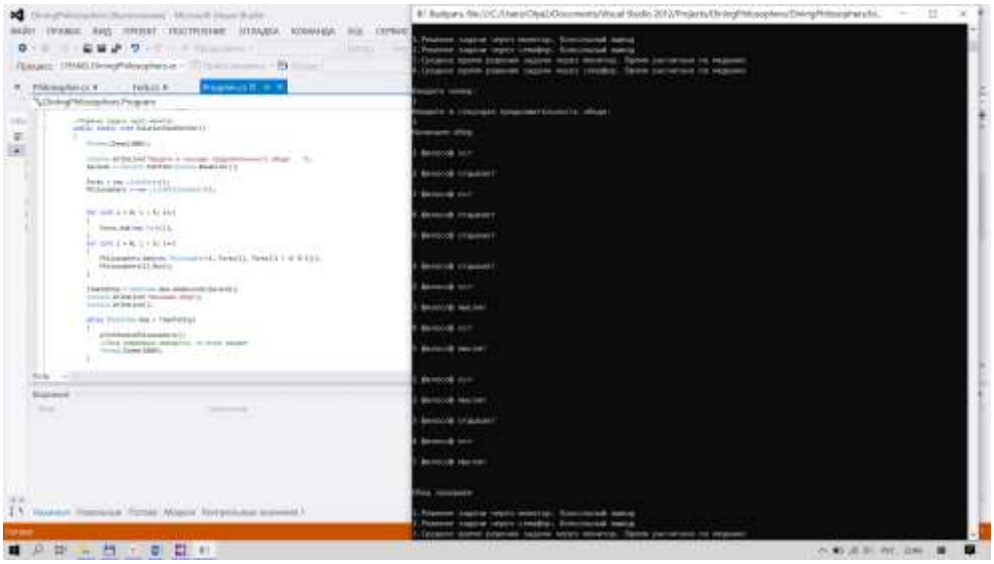
Результат работы:



**2. Решение задачи через монитор**

```
public static void SolutionTaskMonitor()
{
    Thread.Sleep(1000);

    Console.WriteLine("Введите в секундах продолжительность обеда:   ");
    Seconds = Convert.ToInt16(Console.ReadLine());

    Forks = new List<Fork>();
    Philosophers = new List<Philosopher>();


    for (int i = 0; i < 5; i++)
    {
        Forks.Add(new Fork());
    }
    for (int i = 0; i < 5; i++)
    {
        Philosophers.Add(new Philosopher(i, Forks[i], Forks[(i + 1) % 5]));
        Philosophers[i].Run();
    }

    TimeToStop = DateTime.Now.AddSeconds(Seconds);
    Console.WriteLine("Начинаем обед");
    Console.WriteLine();

    while (DateTime.Now < TimeToStop)
    {
        printStatusPhilosophers();
        //Пока информация выводится, то поток ожидает
        Thread.Sleep(1000);
    }


    //Завершаем обед
    for (int i = 0; i < 5; i++)
    {
        Philosophers[i].Stop_Eat();
    }

    Console.WriteLine("Обед завершен");
    //Console.ReadKey();
}
```

Результат работы:

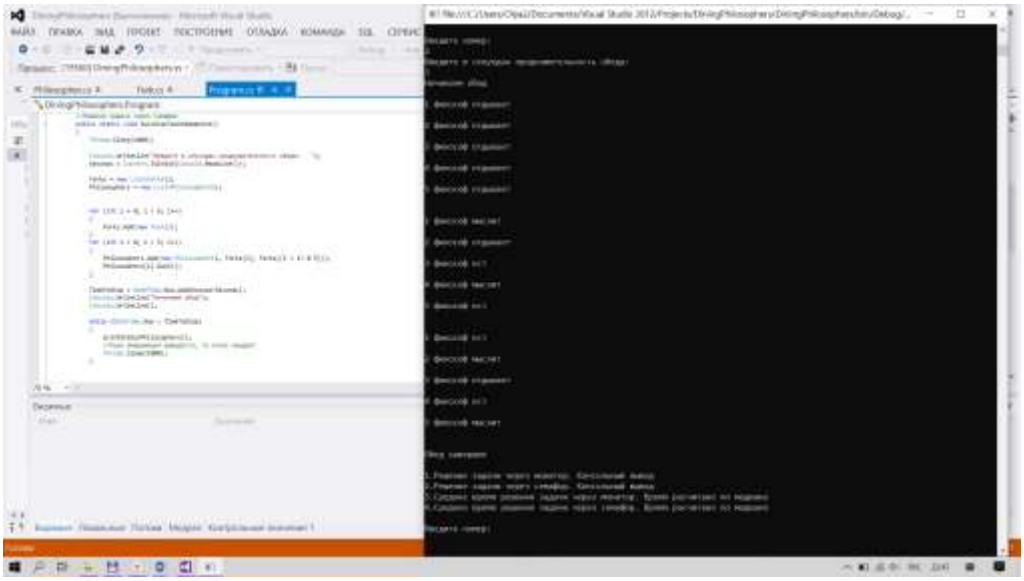


### 3. Решение задачи через Семафор

```
public static void SolutionTaskSemaphore()
{
    Thread.Sleep(1000);

    Console.WriteLine("Введите в секундах продолжительность обеда:   ");
    Seconds = Convert.ToInt16(Console.ReadLine());

    Forks = new List<Fork>();
    Philosophers = new List<Philosopher>();


    for (int i = 0; i < 5; i++)
    {
        Forks.Add(new Fork());
    }
    for (int i = 0; i < 5; i++)
    {
        Philosophers.Add(new Philosopher(i, Forks[i], Forks[(i + 1) % 5]));
        Philosophers[i].RunS();
    }

    TimeToStop = DateTime.Now.AddSeconds(Seconds);
    Console.WriteLine("Начинаем обед");
    Console.WriteLine();

    while (DateTime.Now < TimeToStop)
    {
        printStatusPhilosophers();
        //Пока информация выводится, то поток ожидает
        Thread.Sleep(1000);
    }


    //Завершаем обед
```

```
            for (int i = 0; i < 5; i++)
            {
                Philosophers[i].Stop_Eat();
            }

            Console.WriteLine("Обед завершен");
            //Console.ReadKey();
        }

        // Консольный вывод текущего состояния каждого философа
        public static void printStatusPhilosophers()
        {
            for (int i = 0; i < 5; i++)
            {
                Console.WriteLine((i + 1).ToString() + Philosophers[i].GetStatus());
                Console.WriteLine();
            }
            Console.WriteLine();
        }
```

Результат:



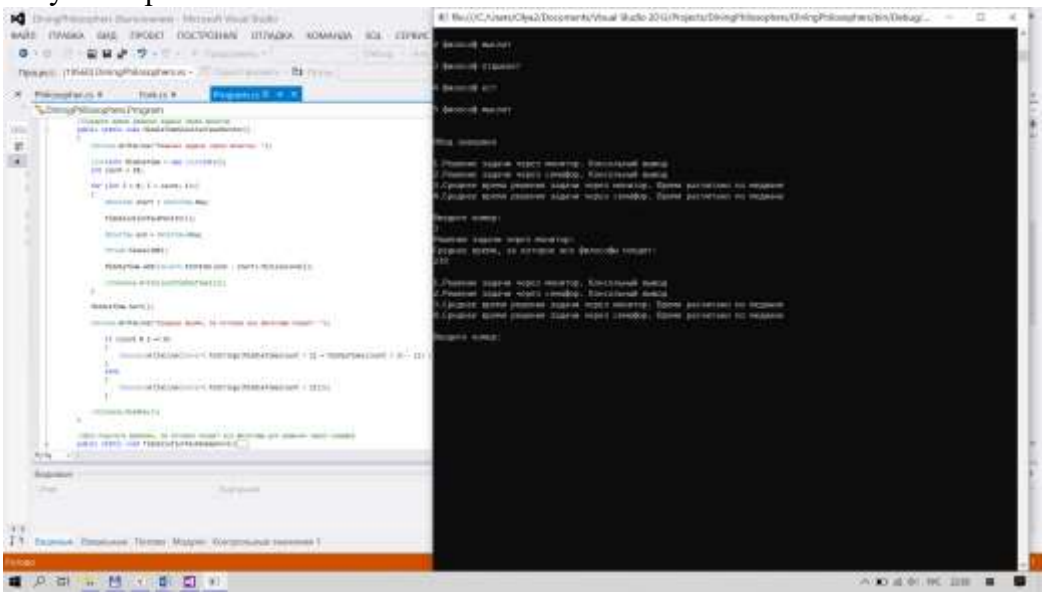
#### 4. Среднее время решения задачи через монитор

```
        //Для подсчета времени, за которое поедят все философы для решения через монитор
        public static void TimeSolutionTaskMonitor()
        {
            //DateTime start = DateTime.Now;

            Forks = new List<Fork>();
            Philosophers = new List<Philosopher>();
            All = new HashSet<int>();

            for (int i = 0; i < 5; i++)
            {
```

```
            Forks.Add(new Fork());
        }
        for (int i = 0; i < 5; i++)
        {
            Philosophers.Add(new Philosopher(i, Forks[i], Forks[(i + 1) % 5]));
            Philosophers[i].Run();
        }

        DateTime k1 = DateTime.Now;
        DateTime k2 = DateTime.Now;

        while ((All.Count != 5) || ((k2 - k1).Seconds > 5))
        {
            for (int i = 0; i < 5; i++)
            {
                if ((Philosophers[i].GetStatus() == " философ ест") || (Philosophers[i].GetStatus()
== " философ мыслит"))
                {
                    All.Add(i);
                }

            }
            k2 = DateTime.Now;

        }
        //Завершаем обед
        for (int i = 0; i < 5; i++)
        {
            Philosophers[i].Stop_Eat();
        }
        //DateTime end = DateTime.Now;

       // Console.WriteLine((end - start).Milliseconds);
        //Console.ReadKey();

        All.Clear();
    }

    //Среднее время решения задачи через монитор
     public static void MiddleTimeSolutionTaskMonitor()
     {
        Console.WriteLine("Решения задачи через монитор: ");

        List<int> MiddleTime = new List<int>();
        int count = 10;

        for (int i = 0; i < count; i++)
        {
            DateTime start = DateTime.Now;

            TimeSolutionTaskMonitor();
```

```
            DateTime end = DateTime.Now;

            Thread.Sleep(100);

            MiddleTime.Add(Convert.ToInt16((end - start).Milliseconds));

            //Console.WriteLine(MiddleTime[i]);
        }

        MiddleTime.Sort();

        Console.WriteLine("Среднее время, за которое все философы поедят: ");

            if (count % 2 == 0)
            {
                Console.WriteLine(Convert.ToString((MiddleTime[count / 2] + MiddleTime[(count
/ 2) - 1]) / 2));
            }
            else
            {
                Console.WriteLine(Convert.ToString((MiddleTime[count / 2])));
            }

        //Console.ReadKey();
    }
```

Результат работы:



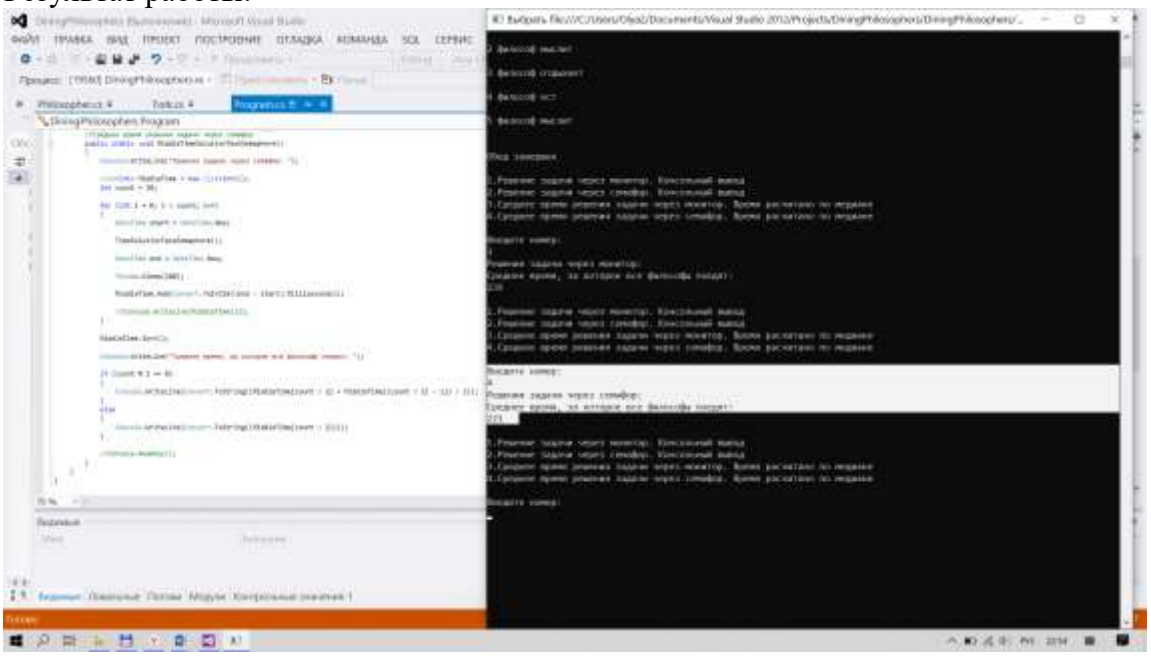
### 5. Среднее время решения задачи через семафор

```
//Для подсчета времени, за которое поедят все философы для решения через семафор
public static void TimeSolutionTaskSemaphore()
{
    //DateTime start = DateTime.Now;

    Forks = new List<Fork>();
    Philosophers = new List<Philosopher>();
```

```
            All = new HashSet<int>();

            for (int i = 0; i < 5; i++)
            {
                Forks.Add(new Fork());
            }
            for (int i = 0; i < 5; i++)
            {
                Philosophers.Add(new Philosopher(i, Forks[i], Forks[(i + 1) % 5]));
                Philosophers[i].RunS();
            }

            DateTime k1 = DateTime.Now;
            DateTime k2 = DateTime.Now;

            while ((All.Count != 5) || ((k2 - k1).Seconds > 5))
            {
                for (int i = 0; i < 5; i++)
                {
                    if ((Philosophers[i].GetStatus() == " философ ест") || (Philosophers[i].GetStatus()
== " философ мыслит"))
                    {
                        All.Add(i);
                    }

                }
                k2 = DateTime.Now;

            }
            //Завершаем обед
            for (int i = 0; i < 5; i++)
            {
                Philosophers[i].Stop_Eat();
            }
            //DateTime end = DateTime.Now;

            // Console.WriteLine((end - start).Milliseconds);
            //Console.ReadKey();

            All.Clear();
        }
        //Среднее время решения задачи через семафор
        public static void MiddleTimeSolutionTaskSemaphore()
        {
            Console.WriteLine("Решения задачи через семафор: ");

            List<int> MiddleTime = new List<int>();
            int count = 10;

            for (int i = 0; i < count; i++)
            {
                DateTime start = DateTime.Now;
```

```
            TimeSolutionTaskSemaphore();

            DateTime end = DateTime.Now;

            Thread.Sleep(100);

            MiddleTime.Add(Convert.ToInt16((end - start).Milliseconds));

            //Console.WriteLine(MiddleTime[i]);
        }

        MiddleTime.Sort();

        Console.WriteLine("Среднее время, за которое все философы поедят: ");

        if (count % 2 == 0)
        {
            Console.WriteLine(Convert.ToString((MiddleTime[count / 2] + MiddleTime[(count /
2) - 1]) / 2));
        }
        else
        {
            Console.WriteLine(Convert.ToString((MiddleTime[count / 2])));
        }

        //Console.ReadKey();
    }
}
```

Результат работы:



# 3. Лабораторная работа №3

## 3.1.Постановка задачи

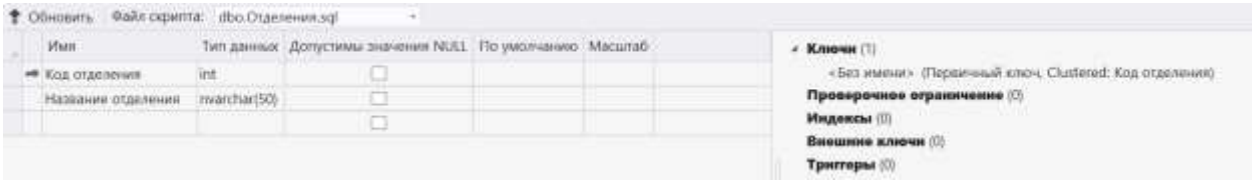
Разработать реляционную базу данных для выбранной предметной области: больница.

Интерфейс приложения для работы с базой данных спроектировать на основе шаблона Windows Forms Application. Для доступа к данным использовать технологию ADO.NET.

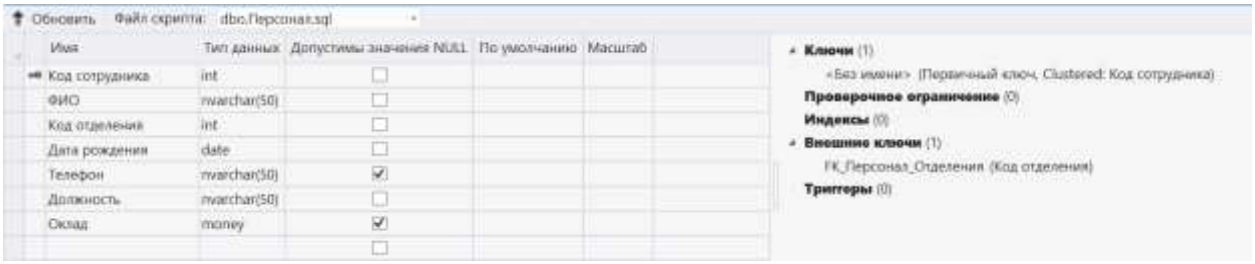
## 3.2.Теоретическая часть

База данных создана в SQL Server, созданы пять таблиц: отделения, персонал, пациенты, диагнозы, процедуры. А также связи между ними. Связи реализуются через такие ключи как «Код отделения», «Код диагноза», «Код процедуры», «Код сотрудника».

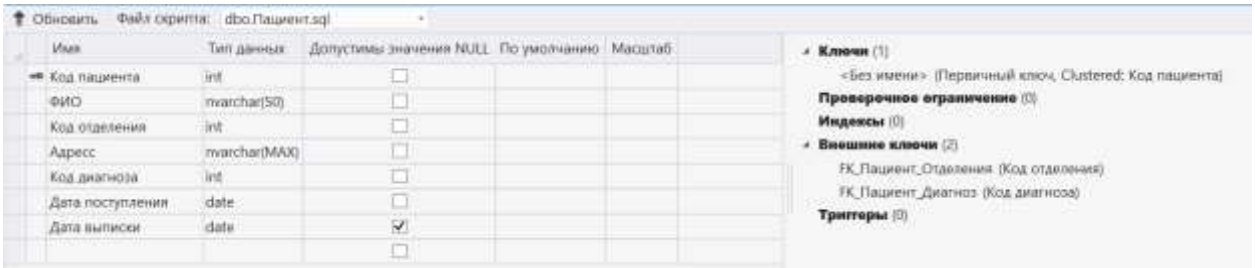
1. Отделения



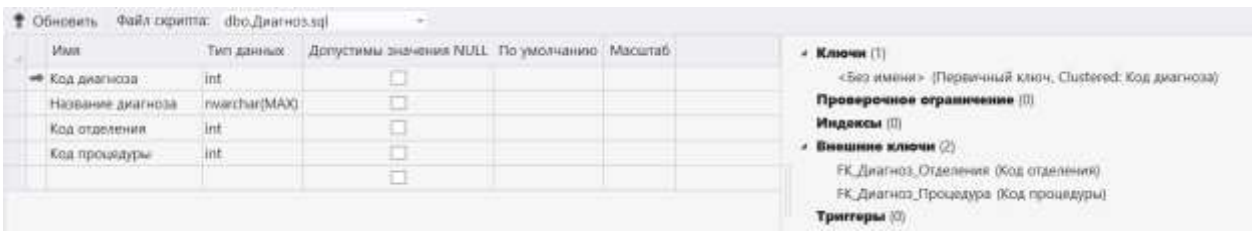

2. Персонал



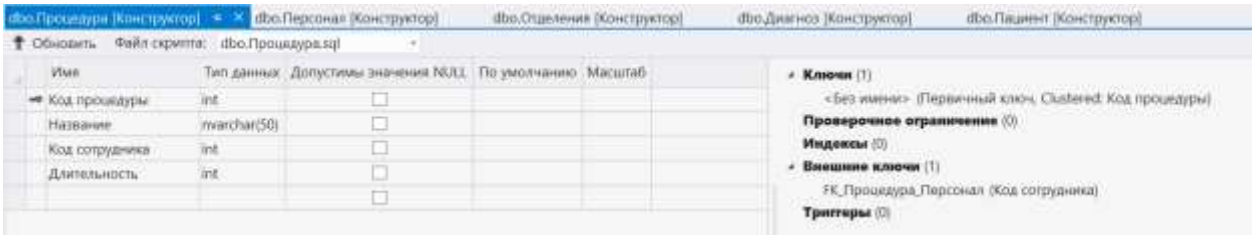

3. Пациенты




4. Диагнозы



5. Процедуры

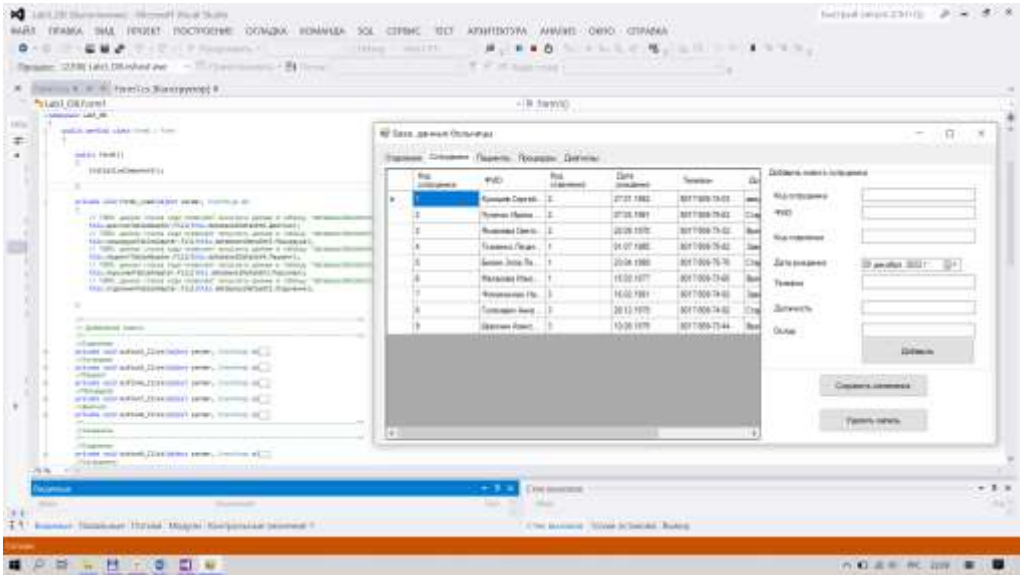



## 3.3.Программная реализация

### 1. Загрузка формы с подключенной базой данных

```
public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();

      }
      private void Form1_Load(object sender, EventArgs e)
      {
         // TODO: данная строка кода позволяет загрузить данные в таблицу "database1DataSet6.Диагноз". При необходимости она может быть перемещена или удалена.
         this.диагнозTableAdapter.Fill(this.database1DataSet6.Диагноз);
         // TODO: данная строка кода позволяет загрузить данные в таблицу "database1DataSet5.Процедура". При необходимости она может быть перемещена или удалена.
         this.процедураTableAdapter.Fill(this.database1DataSet5.Процедура);
         // TODO: данная строка кода позволяет загрузить данные в таблицу "database1DataSet4.Пациент". При необходимости она может быть перемещена или удалена.
         this.пациентTableAdapter.Fill(this.database1DataSet4.Пациент);
         // TODO: данная строка кода позволяет загрузить данные в таблицу "database1DataSet3.Персонал". При необходимости она может быть перемещена или удалена.
         this.персоналTableAdapter.Fill(this.database1DataSet3.Персонал);
         // TODO: данная строка кода позволяет загрузить данные в таблицу "database1DataSet2.Отделения". При необходимости она может быть перемещена или удалена.
         this.отделенияTableAdapter.Fill(this.database1DataSet2.Отделения);
      }
```

Результат:

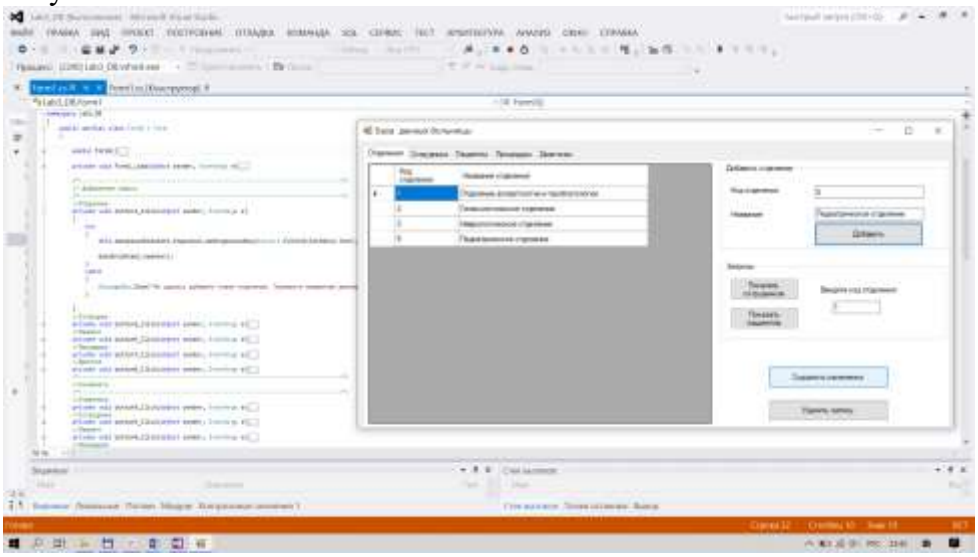


## 2. Добавить новую запись в таблицу

### 2.1. Отделение

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {

this.database1DataSet2.Отделения.AddОтделенияRow(Convert.ToInt32(textBox1.Text),
Convert.ToString(textBox2.Text));

        dataGridView1.Update();
    }
    catch
    {
        MessageBox.Show("Не удалось добавить новое отделение. Проверьте введенные
данные");
    }
}
```

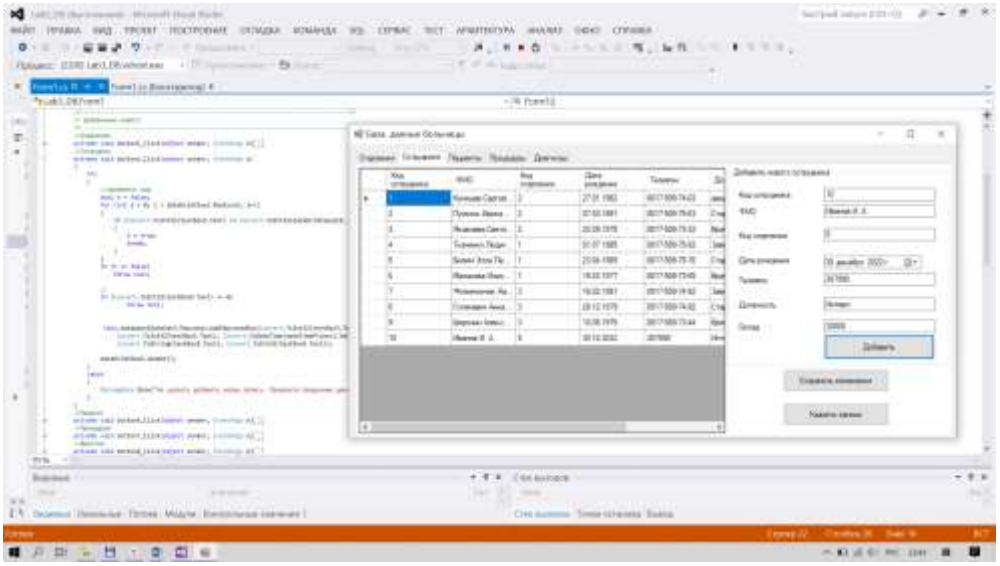
Результат:



### 2.2. Сотрудник

```
private void button2_Click(object sender, EventArgs e)
{
   try
   {
      //проверить код
      bool t = false;
      for (int i = 0; i < dataGridView1.RowCount; i++)
      {
         if (Convert.ToInt32(textBox5.Text) == Convert.ToInt32(dataGridView1[0,
i].Value))
         {
            t = true;
            break;
         }
      }
      if (t == false)
         throw null;

      //
      if (Convert.ToInt32(textBox9.Text) <= 0)
            throw null;
this.database1DataSet3.Персонал.AddПерсоналRow(Convert.ToInt32(textBox3.Text),
Convert.ToString(textBox4.Text),
            Convert.ToInt32(textBox5.Text),
Convert.ToDateTime(dateTimePicker1.Value.Date), Convert.ToString(textBox7.Text),
            Convert.ToString(textBox8.Text), Convert.ToInt32(textBox9.Text));

      dataGridView2.Update();
   }
   catch
   {
      MessageBox.Show("Не удалось добавить новую запись. Проверьте введенные
данные");
   }
}
```

Результат:

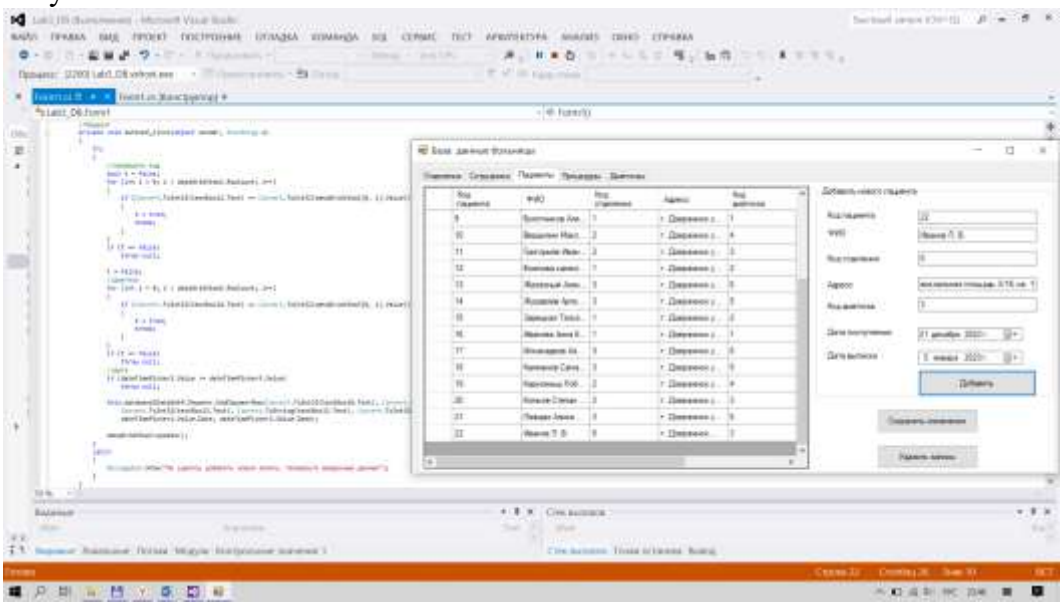


### 2.3. Пациент

```
private void button6_Click(object sender, EventArgs e)
{
   try
   {
      //проверить код
      bool t = false;
      for (int i = 0; i < dataGridView1.RowCount; i++)
      {
         if (Convert.ToInt32(textBox12.Text) == Convert.ToInt32(dataGridView1[0,
i].Value))
         {
            t = true;
            break;
         }
      }
      if (t == false)
         throw null;

      t = false;
      //диагноз
      for (int i = 0; i < dataGridView5.RowCount; i++)
      {
         if (Convert.ToInt32(textBox14.Text) == Convert.ToInt32(dataGridView5[0,
i].Value))
         {
            t = true;
            break;
         }
      }
      if (t == false)
         throw null;
      //дата
      if (dateTimePicker2.Value >= dateTimePicker3.Value)
         throw null;
```

```
            this.database1DataSet4.Пациент.AddПациентRow(Convert.ToInt32(textBox10.Text),
Convert.ToString(textBox11.Text),
                Convert.ToInt32(textBox12.Text), Convert.ToString(textBox13.Text),
Convert.ToInt32(textBox14.text),
                dateTimePicker2.Value.Date, dateTimePicker3.Value.Date);

            dataGridView3.Update();
        }
        catch
        {
            MessageBox.Show("Не удалось добавить новую запись. Проверьте введенные
данные");
        }
    }
```

Результат:

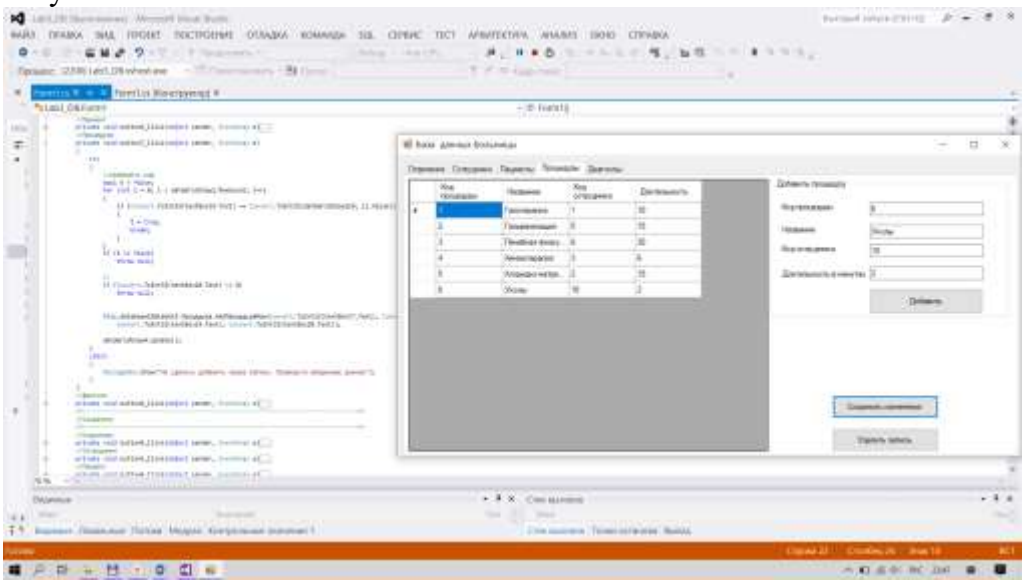


### 2.4. Процедура

```
private void button7_Click(object sender, EventArgs e)
{
    try
    {
        //проверить код
        bool t = false;
        for (int i = 0; i < dataGridView2.RowCount; i++)
        {
            if (Convert.ToInt32(textBox19.Text) == Convert.ToInt32(dataGridView2[0,
i].Value))
            {
                t = true;
                break;
            }
        }
        if (t == false)
            throw null;
```

```
            //
            if (Convert.ToInt32(textBox20.Text) <= 0)
                throw null;



this.database1DataSet5.Процедура.AddПроцедураRow(Convert.ToInt32(textBox17.Text),
Convert.ToString(textBox18.Text),
                Convert.ToInt32(textBox19.Text), Convert.ToInt32(textBox20.Text));

            dataGridView4.Update();
        }
        catch
        {
            MessageBox.Show("Не удалось добавить новую запись. Проверьте введенные
данные");
        }
    }
```

Результат:

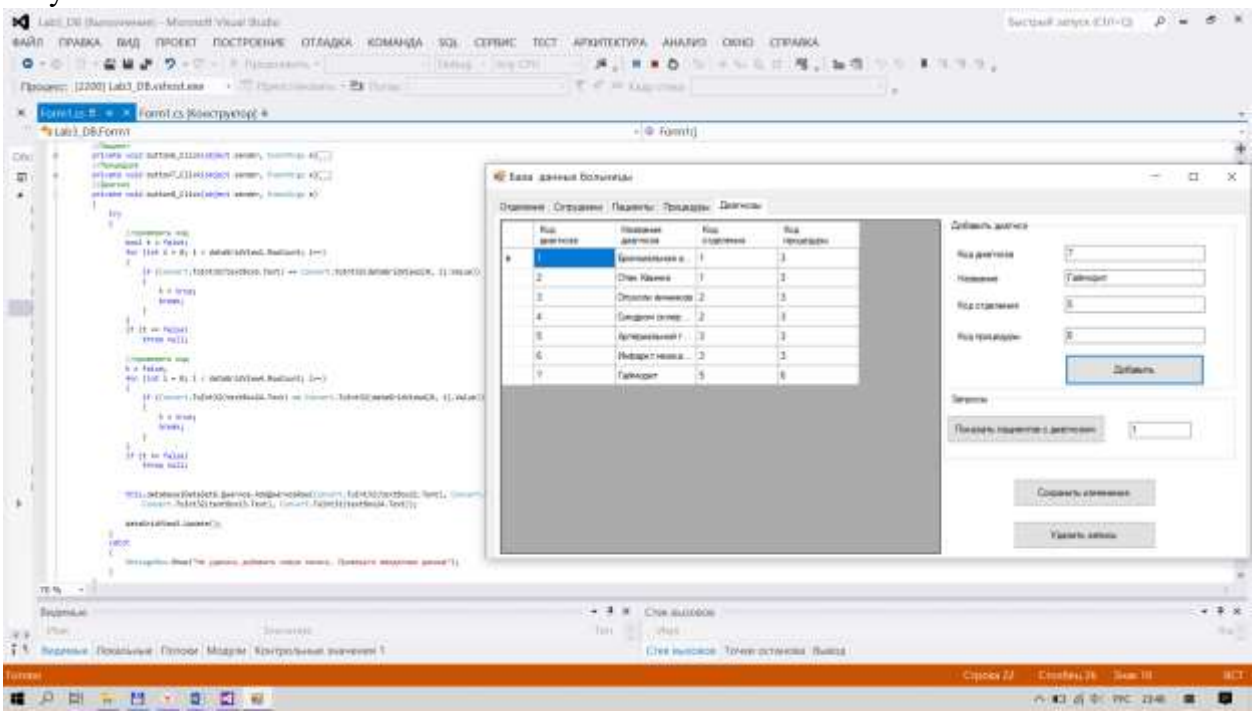


### 2.5. Диагноз

```
    private void button8_Click(object sender, EventArgs e)
    {
        try
        {
            //проверить код
            bool t = false;
            for (int i = 0; i < dataGridView1.RowCount; i++)
            {
                if (Convert.ToInt32(textBox5.Text) == Convert.ToInt32(dataGridView1[0,
i].Value))
                {
                    t = true;
                    break;
                }
```

```
            }
            if (t == false)
                throw null;

            //проверить код
            t = false;
            for (int i = 0; i < dataGridView4.RowCount; i++)
            {
                if (Convert.ToInt32(textBox24.Text) == Convert.ToInt32(dataGridView4[0,
i].Value))
                {
                    t = true;
                    break;
                }
            }
            if (t == false)
                throw null;


            this.database1DataSet6.Диагноз.AddДиагнозRow(Convert.ToInt32(textBox21.Text),
Convert.ToString(textBox22.Text),
                Convert.ToInt32(textBox23.Text), Convert.ToInt32(textBox24.Text));

            dataGridView5.Update();
        }
        catch
        {
            MessageBox.Show("Не удалось добавить новую запись. Проверьте введенные
данные");
        }
    }
```

Результат:

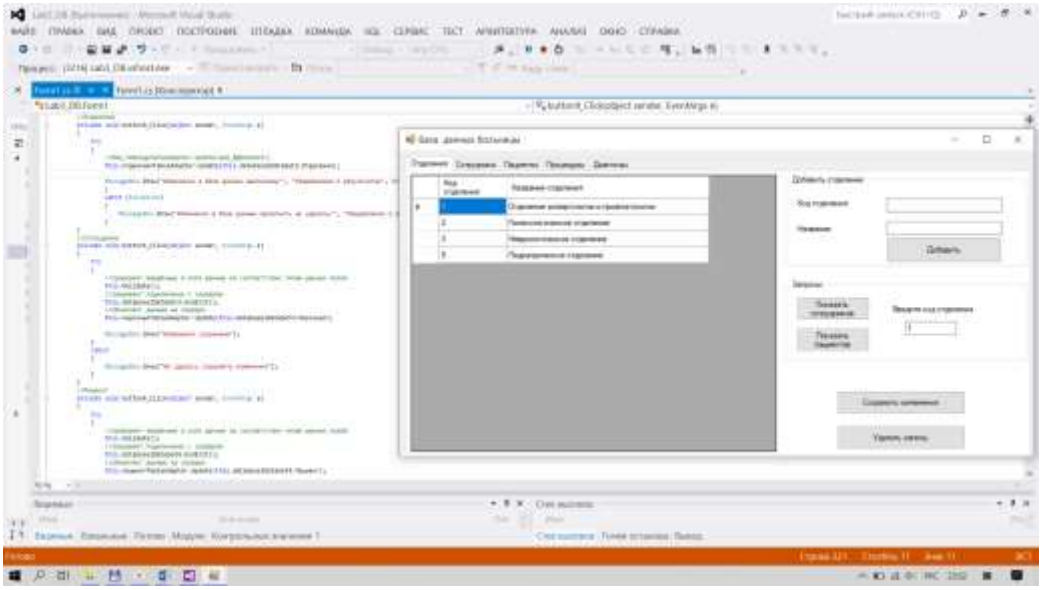


3. **Сохранить изменения в базу данных**
   **На каждой форме присутствует кнопка «Сохранить изменения». После закрытия и открытия приложения с базой данных, введенные данные сохраняются.**

```
//Отделения
private void button9_Click(object sender, EventArgs e)
{
   try
   {
      //Имя_ТаблицыTableAdapter.Update(имя_БДDataSet);
      this.отделенияTableAdapter.Update(this.database1DataSet2.Отделения);

      MessageBox.Show("Изменения в базе данных выполнены!", "Уведомление о
результатах", MessageBoxButtons.OK, MessageBoxIcon.Information);
      }
      catch (Exception)
      {
         MessageBox.Show("Изменения в базе данных выполнить не удалось!",
"Уведомление о результатах", MessageBoxButtons.OK, MessageBoxIcon.Error);
      }
}
//Сотрудники
private void button5_Click(object sender, EventArgs e)
{
   try
   {
      //проверяет введённые в поля данные на соответствие типам данных полей
      this.Validate();
      //закрывает подключение с сервером
      this.database1DataSet3.EndInit();
      //обновляет данные на сервере
      this.персоналTableAdapter.Update(this.database1DataSet3.Персонал);

      MessageBox.Show("Изменения сохранены");
   }
   catch
   {
      MessageBox.Show("Не удалось сохранить изменения");
   }
}
//Пациент
private void button4_Click(object sender, EventArgs e)
{
   try
   {
      //проверяет введённые в поля данные на соответствие типам данных полей
      this.Validate();
      //закрывает подключение с сервером
      this.database1DataSet4.EndInit();
      //обновляет данные на сервере
      this.пациентTableAdapter.Update(this.database1DataSet4.Пациент);
```

```
            MessageBox.Show("Изменения сохранены");
        }
        catch
        {
            MessageBox.Show("Не удалось сохранить изменения");
        }
    }
    //Процедура
    private void button13_Click(object sender, EventArgs e)
    {
        try
        {
            //проверяет введённые в поля данные на соответствие типам данных полей
            this.Validate();
            //закрывает подключение с сервером
            this.database1DataSet5.EndInit();
            //обновляет данные на сервере
            this.процедураTableAdapter.Update(this.database1DataSet5.Процедура);

            MessageBox.Show("Изменения сохранены");
        }
        catch
        {
            MessageBox.Show("Не удалось сохранить изменения");
        }
    }
    //Диагноз
    private void button15_Click(object sender, EventArgs e)
    {
        try
        {
            //проверяет введённые в поля данные на соответствие типам данных полей
            this.Validate();
            //закрывает подключение с сервером
            this.database1DataSet6.EndInit();
            //обновляет данные на сервере
            this.диагнозTableAdapter.Update(this.database1DataSet6.Диагноз);

            MessageBox.Show("Изменения сохранены");
        }
        catch
        {
            MessageBox.Show("Не удалось сохранить изменения");
        }
    }
```

Результат при новом открытие приложения:



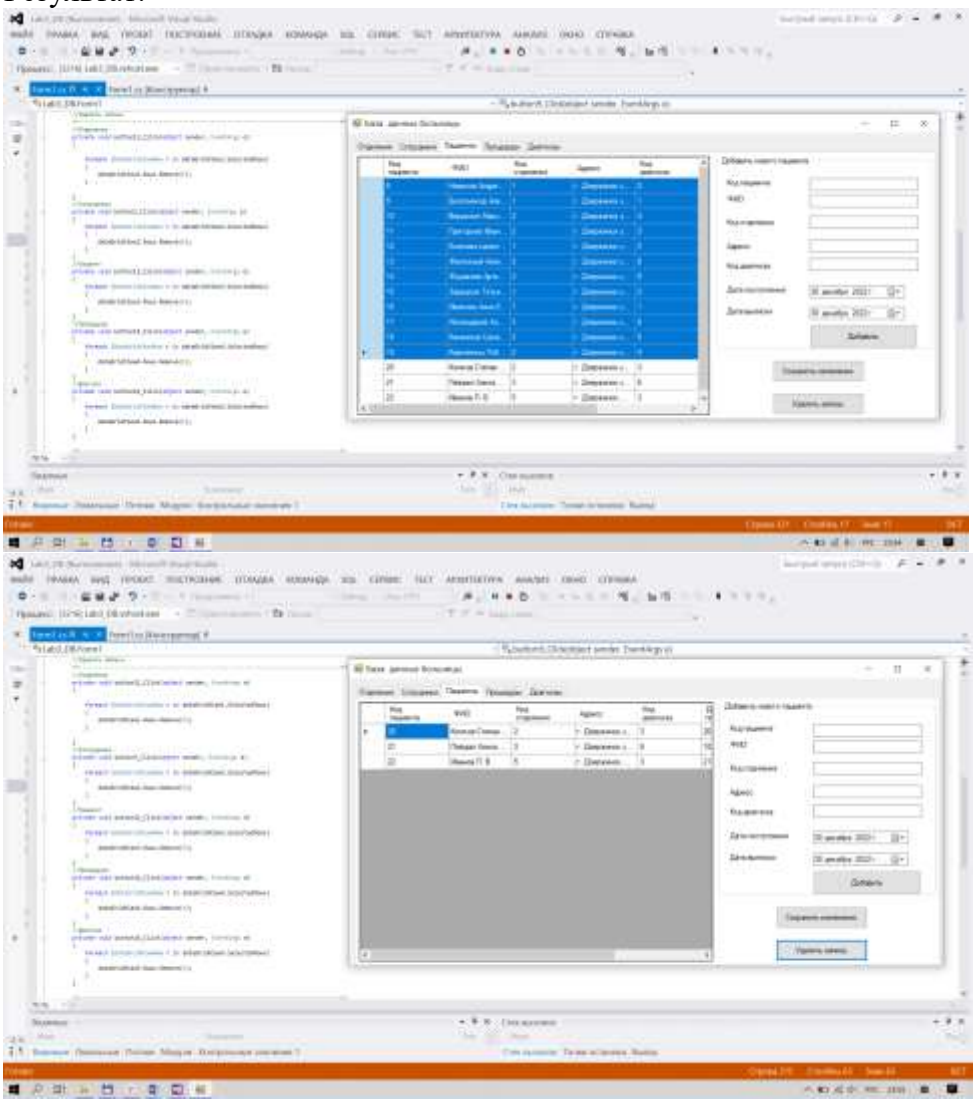
4. **Удаление**

   **Поскольку все функции реализованы одинаково, то результат будет продемонстрирован на таблице пациенты.**

```
//Отделения
private void button11_Click(object sender, EventArgs e)
{

   foreach (DataGridViewRow r in dataGridView1.SelectedRows)
   {
      dataGridView1.Rows.Remove(r);
   }

}
//Сотрудники
private void button3_Click(object sender, EventArgs e)
{
   foreach (DataGridViewRow r in dataGridView2.SelectedRows)
   {
      dataGridView2.Rows.Remove(r);
   }
}
//Пациент
private void button12_Click(object sender, EventArgs e)
{
   foreach (DataGridViewRow r in dataGridView3.SelectedRows)
   {
      dataGridView3.Rows.Remove(r);
   }
}
//Процедура
private void button14_Click(object sender, EventArgs e)
{
   foreach (DataGridViewRow r in dataGridView4.SelectedRows)
   {
```

```
            dataGridView4.Rows.Remove(r);
        }
    }
    //Диагноз
    private void button16_Click(object sender, EventArgs e)
    {
        foreach (DataGridViewRow r in dataGridView5.SelectedRows)
        {
            dataGridView5.Rows.Remove(r);
        }
    }
```

Результат:



**Данные изменения не будут сохранены целенаправленно.**

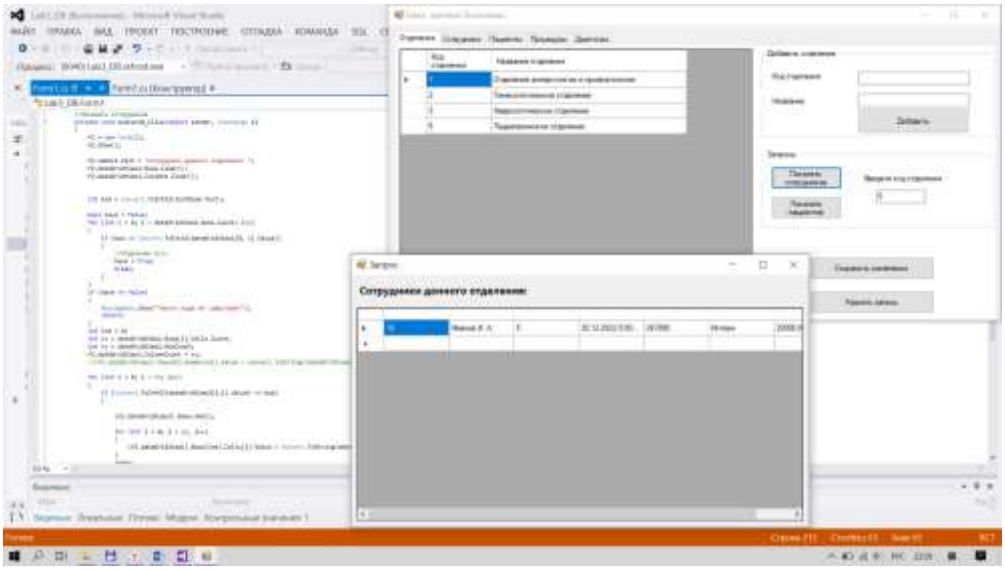
## 5. Запросы

### 5.1.Показать сотрудников отделения

```
private void button10_Click(object sender, EventArgs e)
{
    f2 = new Form2();
    f2.Show();
```

```
            f2.label1.Text = "Сотрудники данного отделения: ";
            f2.dataGridView1.Rows.Clear();
            f2.dataGridView1.Columns.Clear();


            int kod = Convert.ToInt32(textBox6.Text);

            bool have = false;
            for (int i = 0; i < dataGridView1.Rows.Count; i++)
            {
               if (kod == Convert.ToInt32(dataGridView1[0, i].Value))
               {
                  //Отделение есть
                  have = true;
                  break;
               }
            }
            if (have == false)
            {
               MessageBox.Show("Такого кода не существует");
               return;
            }
            int ind = 0;
            int cc = dataGridView2.Rows[1].Cells.Count;
            int rc = dataGridView2.RowCount;
            f2.dataGridView1.ColumnCount = cc;
            //(f2.dataGridView1).Rows[0].HeaderCell.Value =
Convert.ToString((dataGridView2).Rows[0].HeaderCell.Value);

            for (int i = 0; i < rc; i++)
            {
               if (Convert.ToInt32(dataGridView2[2,i].Value) == kod)
               {

                  (f2.dataGridView1).Rows.Add();

                  for (int j = 0; j < cc; j++)
                  {
                     (f2.dataGridView1).Rows[ind].Cells[j].Value =
Convert.ToString(dataGridView2.Rows[i].Cells[j].Value);
                  }
                  ind++;
               }
            }

            f2.Update();

         }
```

Результат:

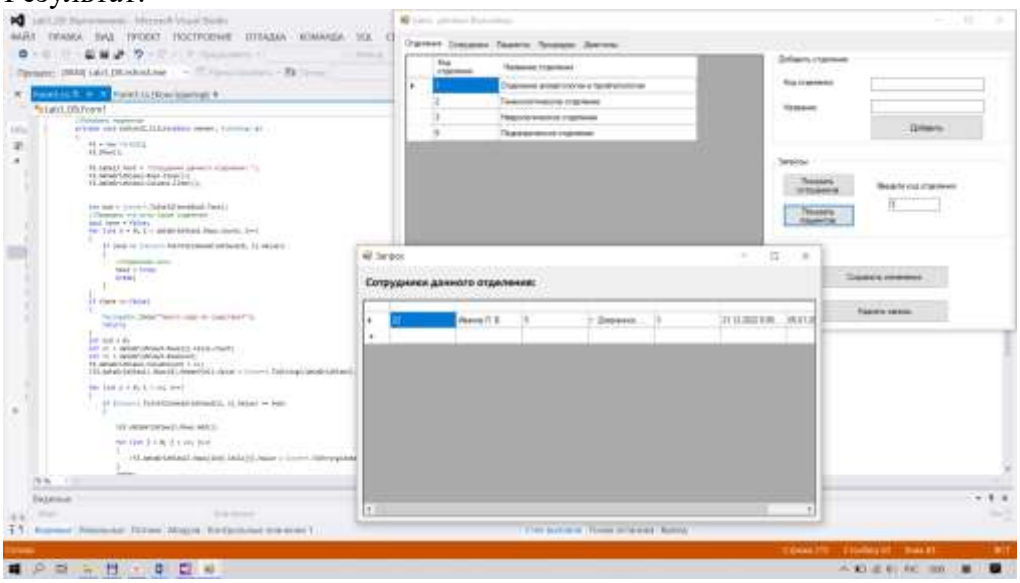


### 5.2. Показать пациентов отделения

```
private void button21_Click(object sender, EventArgs e)
{
    f2 = new Form2();
    f2.Show();

    f2.label1.Text = "Сотрудники данного отделения: ";
    f2.dataGridView1.Rows.Clear();
    f2.dataGridView1.Columns.Clear();


    int kod = Convert.ToInt32(textBox6.Text);
    //Проверка что есть такое отделение
    bool have = false;
    for (int i = 0; i < dataGridView1.Rows.Count; i++)
    {
        if (kod == Convert.ToInt32(dataGridView1[0, i].Value))
        {
            //Отделение есть
            have = true;
            break;
        }
    }
    if (have == false)
    {
        MessageBox.Show("Такого кода не существует");
        return;
    }
    int ind = 0;
    int cc = dataGridView3.Rows[1].Cells.Count;
    int rc = dataGridView3.RowCount;
    f2.dataGridView1.ColumnCount = cc;
    (f2.dataGridView1).Rows[0].HeaderCell.Value =
Convert.ToString((dataGridView3).Rows[0].HeaderCell.Value);
```

```
        for (int i = 0; i < rc; i++)
        {
            if (Convert.ToInt32(dataGridView3[2, i].Value) == kod)
            {

                (f2.dataGridView1).Rows.Add();

                for (int j = 0; j < cc; j++)
                {
                    (f2.dataGridView1).Rows[ind].Cells[j].Value =
Convert.ToString(dataGridView3.Rows[i].Cells[j].Value);
                }
                ind++;
            }
        }

        f2.Update();
    }
```

Результат:

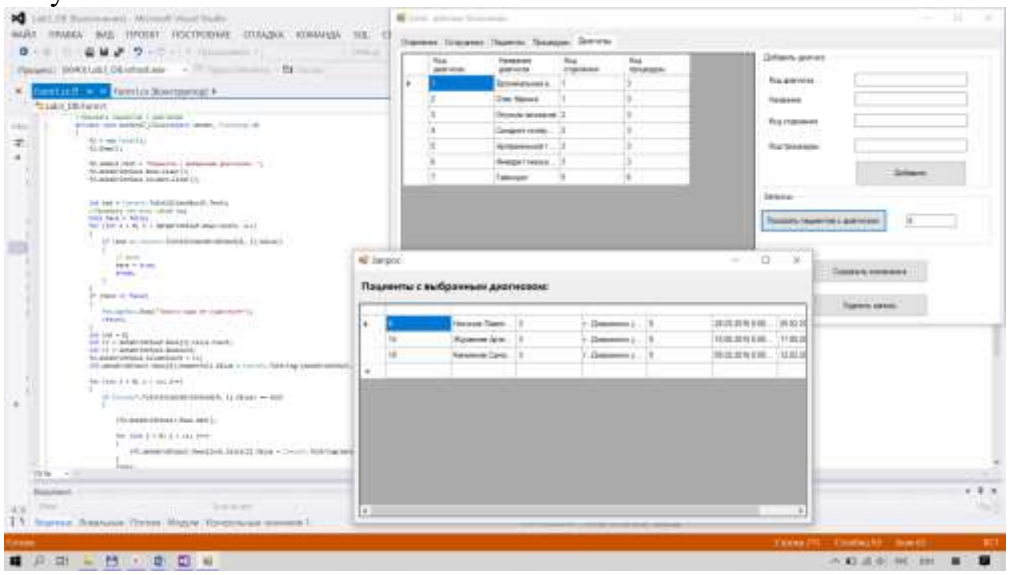


### 5.3. Показать пациентов с диагнозом. Диагноз определяется своим кодом

```
    private void button17_Click(object sender, EventArgs e)
    {
        f2 = new Form2();
        f2.Show();
        f2.label1.Text = "Пациенты с выбранным диагнозом: ";
        f2.dataGridView1.Rows.Clear();
        f2.dataGridView1.Columns.Clear();
        int kod = Convert.ToInt32(textBox15.Text);
        //Проверка что есть такой код
        bool have = false;
        for (int i = 0; i < dataGridView5.Rows.Count; i++)
        {
            if (kod == Convert.ToInt32(dataGridView5[0, i].Value))
```

```
            {
                // есть
                have = true;
                break;
            }
        }
        if (have == false)
        {
            MessageBox.Show("Такого кода не существует");
            return;
        }
        int ind = 0;
        int cc = dataGridView3.Rows[1].Cells.Count;
        int rc = dataGridView3.RowCount;
        f2.dataGridView1.ColumnCount = cc;
        (f2.dataGridView1).Rows[0].HeaderCell.Value =
Convert.ToString((dataGridView3).Rows[0].HeaderCell.Value);

        for (int i = 0; i < rc; i++)
        {
            if (Convert.ToInt32(dataGridView3[4, i].Value) == kod)
            {
                (f2.dataGridView1).Rows.Add();

                for (int j = 0; j < cc; j++)
                {
                    (f2.dataGridView1).Rows[ind].Cells[j].Value =
Convert.ToString(dataGridView3.Rows[i].Cells[j].Value);
                }
                ind++;
            }
        }
        f2.Update();
    }
}
```

Результат:



# 4. Лабораторная работа №4

## 4.1.Постановка задачи

Разработать графическое приложение Windows, которое:

1. Строит график функции;
2. Использует компонент Chart для построения диаграмм;
3. Использует средства Excel для построения диаграмм;
4. Позволяет манипулировать мышью с графическими примитивами;
5. Изменяет графические параметры изображения.

## 4.2.Теоретическая часть

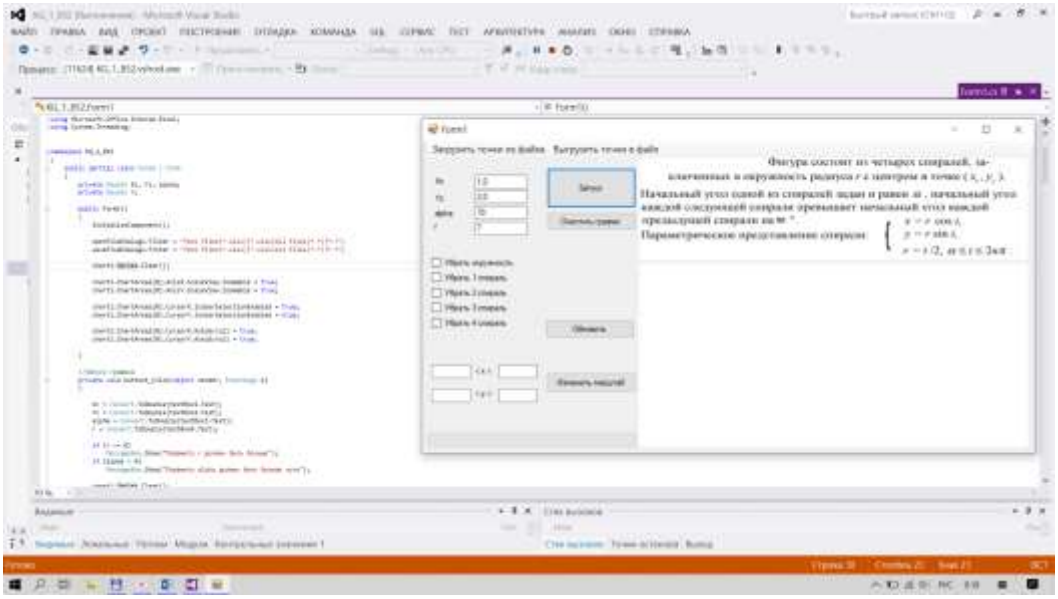
Для примера построения графика была выбрана фигура, состоящая из 4 спиралей, заключенных в окружность радиуса r, с центром в точке $(x_c, y_c)$. Кроме того, реализована возможность построения графика по точкам из файла .xlsx.

## 4.3.Программная реализация

### 1. Запуск программы

```
public partial class Form1 : Form
   {
      private Double Xc, Yc, alpha;
      private Double r;

      public Form1()
      {
         InitializeComponent();

         openFileDialog1.Filter = "Text files(*.xlsx)|*.xlsx|All files(*.*)|*.*";
         saveFileDialog1.Filter = "Text files(*.xlsx)|*.xlsx|All files(*.*)|*.*";

         chart1.Series.Clear();

         chart1.ChartAreas[0].AxisX.ScaleView.Zoomable = true;
         chart1.ChartAreas[0].AxisY.ScaleView.Zoomable = true;

         chart1.ChartAreas[0].CursorX.IsUserSelectionEnabled = true;
         chart1.ChartAreas[0].CursorY.IsUserSelectionEnabled = true;

         chart1.ChartAreas[0].CursorX.AutoScroll = true;
         chart1.ChartAreas[0].CursorY.AutoScroll = true;

      }
```

Результат:



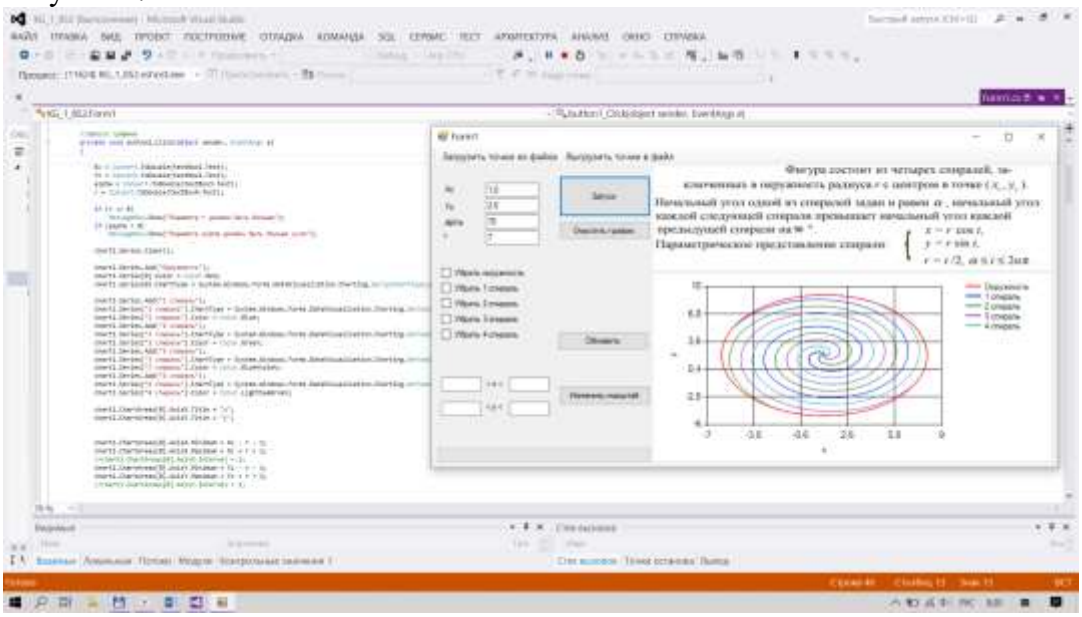
**2. Запуск графика**

```
private void button1_Click(object sender, EventArgs e)
{

    Xc = Convert.ToDouble(textBox1.Text);
    Yc = Convert.ToDouble(textBox2.Text);
    alpha = Convert.ToDouble(textBox3.Text);
    r = Convert.ToDouble(textBox4.Text);

    if (r <= 0)
       MessageBox.Show("Параметр r должен быть больше");
    if (alpha < 0)
       MessageBox.Show("Параметр alpha должен быть больше нуля");

    chart1.Series.Clear();

    chart1.Series.Add("Окружность");
    chart1.Series[0].Color = Color.Red;
    chart1.Series[0].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;

    chart1.Series.Add("1 спираль");
    chart1.Series["1 спираль"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
    chart1.Series["1 спираль"].Color = Color.Blue;
    chart1.Series.Add("2 спираль");
    chart1.Series["2 спираль"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
    chart1.Series["2 спираль"].Color = Color.Green;
    chart1.Series.Add("3 спираль");
    chart1.Series["3 спираль"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
    chart1.Series["3 спираль"].Color = Color.BlueViolet;
```

```
        chart1.Series.Add("4 спираль");
        chart1.Series["4 спираль"].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
        chart1.Series["4 спираль"].Color = Color.LightSeaGreen;

        chart1.ChartAreas[0].AxisX.Title = "x";
        chart1.ChartAreas[0].AxisY.Title = "y";


        chart1.ChartAreas[0].AxisX.Minimum = Xc - r - 1;
        chart1.ChartAreas[0].AxisX.Maximum = Xc + r + 1;
        //chart1.ChartAreas[0].AxisX.Interval = 1;
        chart1.ChartAreas[0].AxisY.Minimum = Yc - r - 1;
        chart1.ChartAreas[0].AxisY.Maximum = Yc + r + 1;
        //chart1.ChartAreas[0].AxisY.Interval = 1;



        Double a1 = alpha * Math.PI / 180;
        Double a2 = (alpha + 90) * Math.PI / 180;
        Double a3 = (alpha + 180) * Math.PI / 180;
        Double a4 = (alpha + 270) * Math.PI / 180;

        progressBar1.Minimum = 0;
        progressBar1.Maximum = Convert.ToInt32(2 * r * 100);
        progressBar1.Step = 1;

        for (double t = 0; t <= 2 * r ; t += 0.01)
        {
           chart1.Series["Окружность"].Points.AddXY(Xc + r * Math.Cos(t), Yc + r *
Math.Sin(t));

           Double x = t / 2 * Math.Cos(t);
           Double y = t / 2 * Math.Sin(t);

           Double x1 = Xc + x * Math.Cos(a1) - y * Math.Sin(a1);
           Double y1 = Yc + x * Math.Sin(a1) + y * Math.Cos(a1);
           chart1.Series["1 спираль"].Points.AddXY(x1, y1);

           Double x2 = Xc + x * Math.Cos(a2) - y * Math.Sin(a2);
           Double y2 = Yc + x * Math.Sin(a2) + y * Math.Cos(a2);
           chart1.Series["2 спираль"].Points.AddXY(x2, y2);


           Double x3 = Xc + x * Math.Cos(a3) - y * Math.Sin(a3);
           Double y3 = Yc + x * Math.Sin(a3) + y * Math.Cos(a3);
           chart1.Series["3 спираль"].Points.AddXY(x3, y3);

           Double x4 = Xc + x * Math.Cos(a4) - y * Math.Sin(a4);
           Double y4 = Yc + x * Math.Sin(a4) + y * Math.Cos(a4);
           chart1.Series["4 спираль"].Points.AddXY(x4, y4);
```

```
            progressBar1.PerformStep();
        }

        progressBar1.Value = 0;
    }
```

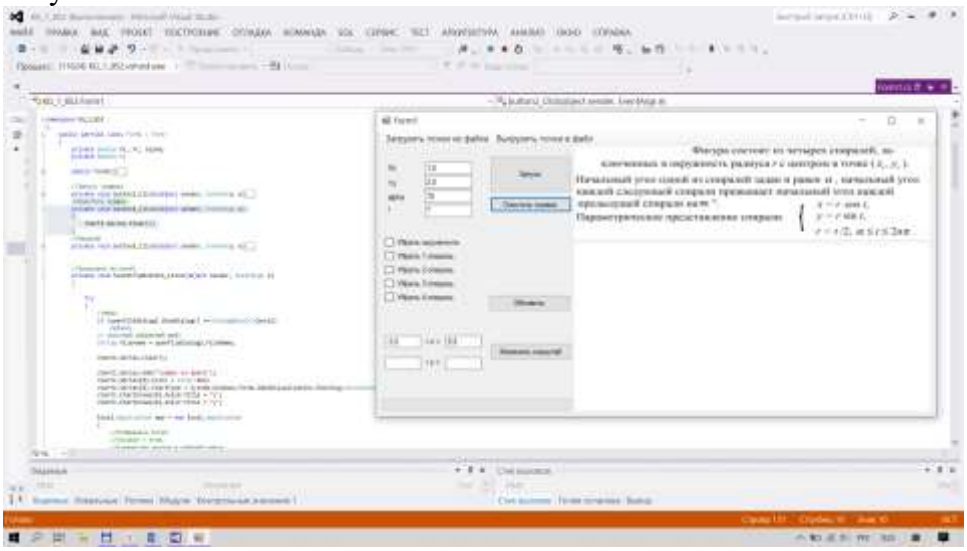
Результат:



### 3. Очистить график

```
    private void button2_Click(object sender, EventArgs e)
    {
        chart1.Series.Clear();
    }
```

Результат:

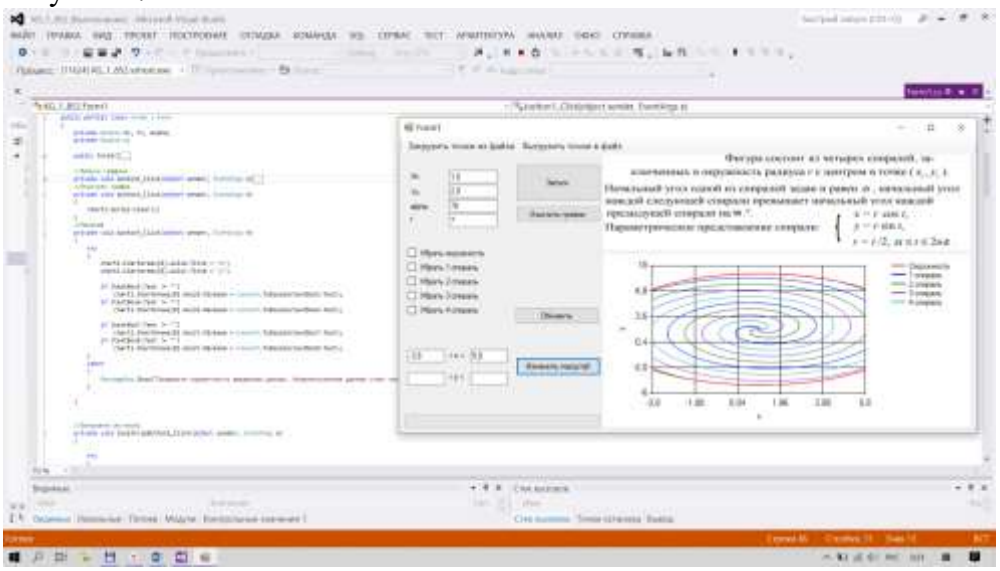


### 4. Масштаб

```
    private void button3_Click(object sender, EventArgs e)
    {
        try
        {
```

```
            chart1.ChartAreas[0].AxisX.Title = "x";
            chart1.ChartAreas[0].AxisY.Title = "y";

            if (textBox5.Text != "")
                chart1.ChartAreas[0].AxisX.Minimum = Convert.ToDouble(textBox5.Text);
            if (textBox6.Text != "")
                chart1.ChartAreas[0].AxisX.Maximum = Convert.ToDouble(textBox6.Text);

            if (textBox7.Text != "")
                chart1.ChartAreas[0].AxisY.Minimum = Convert.ToDouble(textBox7.Text);
            if (textBox8.Text != "")
                chart1.ChartAreas[0].AxisY.Maximum = Convert.ToDouble(textBox8.Text);
        }
        catch
        {
            MessageBox.Show("Проверьте корректность введенных данных.
Нецелочисленные данные стоит писать через ,");
        }

    }
```

Результат:



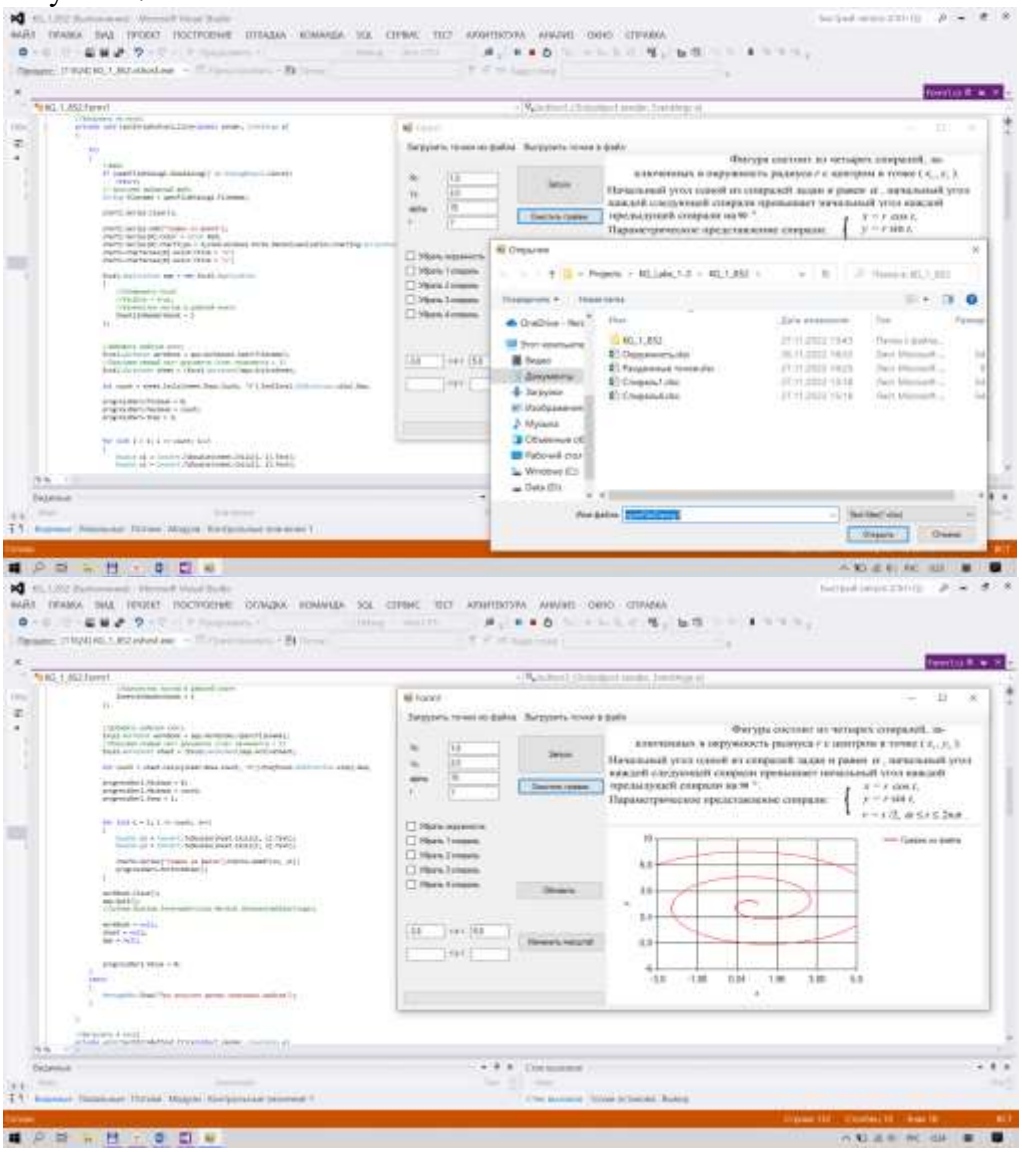
5. **Загрузить из excel**

```
private void toolStripButton1_Click(object sender, EventArgs e)
{

    try
    {
        //Файл
        if (openFileDialog1.ShowDialog() == DialogResult.Cancel)
            return;
        // получаем выбранный файл
        String filename = openFileDialog1.FileName;

        chart1.Series.Clear();
```

```
            chart1.Series.Add("График из файла");
            chart1.Series[0].Color = Color.Red;
            chart1.Series[0].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;
            chart1.ChartAreas[0].AxisX.Title = "x";
            chart1.ChartAreas[0].AxisY.Title = "y";

            Excel.Application app = new Excel.Application
            {
                //Отобразить Excel
                //Visible = true,
                //Количество листов в рабочей книге
                SheetsInNewWorkbook = 1
            };


            //Добавить рабочую книгу
            Excel.Workbook workBook = app.Workbooks.Open(filename);
            //Получаем первый лист документа (счет начинается с 1)
            Excel.Worksheet sheet = (Excel.Worksheet)app.ActiveSheet;

            int count = sheet.Cells[sheet.Rows.Count, "A"].End[Excel.XlDirection.xlUp].Row;

            progressBar1.Minimum = 0;
            progressBar1.Maximum = count;
            progressBar1.Step = 1;


            for (int i = 1; i <= count; i++)
            {
                Double x1 = Convert.ToDouble(sheet.Cells[i, 1].Text);
                Double y1 = Convert.ToDouble(sheet.Cells[i, 2].Text);

                chart1.Series["График из файла"].Points.AddXY(x1, y1);
                progressBar1.PerformStep();
            }

            workBook.Close();
            app.Quit();
            //System.Runtime.InteropServices.Marshal.ReleaseComObject(app);

            workBook = null;
            sheet = null;
            app = null;


            progressBar1.Value = 0;
        }
        catch
        {
            MessageBox.Show("При загрузке данных произошла ошибкаа");
        }
```

}

Результат:



### 6. Выгрузить в excel

```
private void toolStripButton2_Click(object sender, EventArgs e)
{
    if (chart1.Series.Count == 0)
    {
        MessageBox.Show("Нет точек графика");
        return;
    }
    if (chart1.Series.Count > 1)
    {
        MessageBox.Show("Выгрузить точки в файл невозможно, поскольку на диаграмме более одного графика. Повторите попытку позже");
        return;
    }

    try
```

```
{

   Excel.Application app = new Excel.Application
   {
      //Отобразить Excel
      //Visible = true,
      //Количество листов в рабочей книге
      SheetsInNewWorkbook = 1
   };
   //Добавить рабочую книгу
   Excel.Workbook workBook = app.Workbooks.Add(Type.Missing);
   //Отключить отображение окон с сообщениями
   app.DisplayAlerts = false;
   //Получаем первый лист документа (счет начинается с 1)
   Excel.Worksheet sheet = (Excel.Worksheet)app.Worksheets.get_Item(1);
   //Название листа (вкладки снизу)
   sheet.Name = "Точки";

   String NameGr = chart1.Series[0].Name;

   for (int i = 1; i < chart1.Series[NameGr].Points.Count; i ++)
   {
      /*
      sheet.Cells[i, 1] = chart1.Series["Окружность"].Points[i].XValue;
      sheet.Cells[i, 2] = chart1.Series["Окружность"].Points[i].YValues;
       */

      sheet.Cells[i, 1] = chart1.Series[NameGr].Points[i].XValue;
      sheet.Cells[i, 2] = chart1.Series[NameGr].Points[i].YValues;

   }
   if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)
      return;

   // получаем выбранный файл
   String filename = saveFileDialog1.FileName;

   app.Application.ActiveWorkbook.SaveAs(filename, Type.Missing, Type.Missing);

   workBook.Close();
   app.Quit();
   //System.Runtime.InteropServices.Marshal.ReleaseComObject(app);

   workBook = null;
   sheet = null;
   app = null;
   GC.Collect();
}
catch
{
   MessageBox.Show("При выгрузке данных произошла ошибка");
}
```

}

Результат: