МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра САПР

ОТЧЕТ

По учебной практике(технологической(проектно-технологической) практике)

Тема: «Создание приложений Windows Forms в MVS C#»

Студент гр. 9309 _____ Юшин Е.В.

Руководитель _____ Калмычков В.А.

Санкт-Петербург
2021

ЗАДАНИЕ
на курсовую работу

Студент Юшин Е.В.

Группа 9309

Тема работы: разработка графических приложений в Windows Forms на языке C#.

Содержание пояснительной записки: «Оглавление», «Цель работы», «Задание 1», «Задание 2», «Задание 3», «Выводы», «Ссылки на источники».

Предполагаемый объем пояснительной записки:

Не менее 40 страниц.

Дата выдачи задания: 01.07.2021

Дата сдачи реферата: 14.07.2021

Дата защиты реферата: 14.07.2021

Студент гр. 9309 Юшин Е.В.

Преподаватель Калмычков В.А.

Санкт-Петербург
2021

АННОТАЦИЯ К ПОЯСНИТЕЛЬНОЙ ЗАПИСКЕ

Написанные программы реализованы с применением графических примитивов в Windows Forms. В вводном задании мы знакомимся с базовыми возможностями Windows Forms. В первом индивидуальном задании мы производим движение заданной фигуры по заданной траектории. Во втором индивидуальном задании мы производим прорисовку фрактала и вывод дерева, обеспечив поэтапный вывод.

SUMMARY

The written programs are implemented using graphical primitives in Windows Forms. In this introductory assignment, we explore the basic capabilities of Windows Forms. In the first individual task, we make the movement of a given figure along a given trajectory. In the second individual task, we draw the fractal and draw the tree, providing a phased output.

Оглавление

1. ЦЕЛЬ РАБОТЫ

Целью учебной практики является изучение и освоение базовых понятий, методов и приемов использования инструментальных средств и технологий программирования при решении практических задач с выбором различных структур данных и организацией программного графического интерфейса пользователя, закрепление и приобретение новых знаний и практических навыков программирования.

Ознакомительное задание

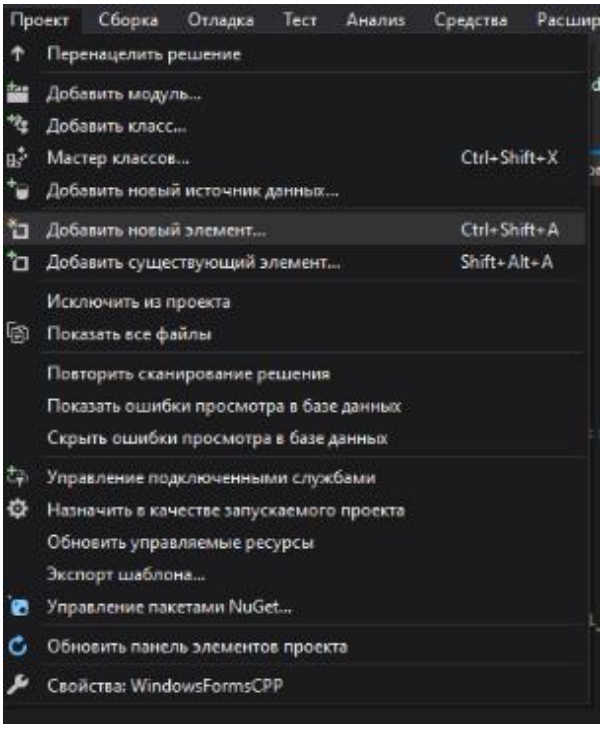
Цель работы

Приобрести первоначальные навыки и опыт создания программ с графическим интерфейсом пользователя (UI) на C++.

Сценарии ознакомительных практических занятий

Занятие № 1. Элементы button, textBox и label

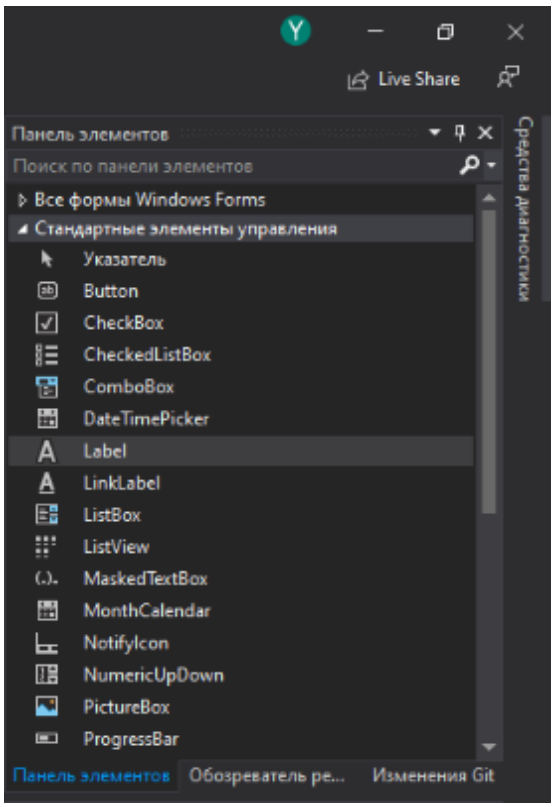
Для начала создадим новый пустой проект CLR, после нажимаем проект, добавить новый элемент, затем Форма Windows Forms .



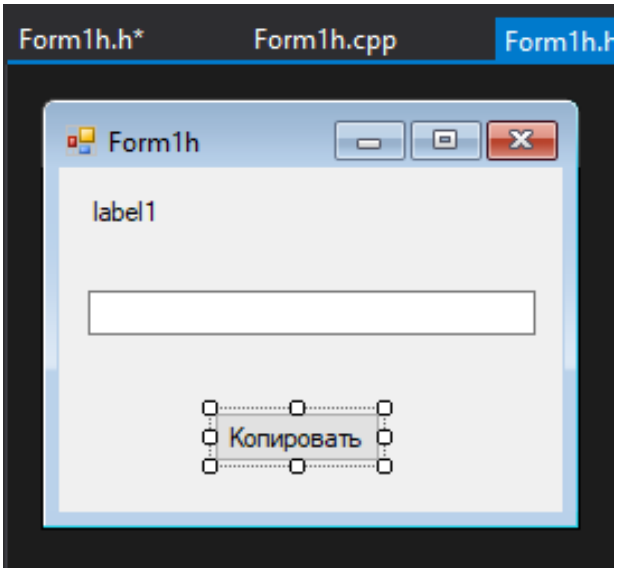
Переходим в Form1h.cpp и вставляем следующий код:

```
##include "Form1h.h"
#include <Windows.h>

using namespace TestWindowsForm;

int WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int) {
    Application::EnableVisualStyles();
    Application::SetCompatibleTextRenderingDefault(false);
    Application::Run(gcnew Form1);
    return 0;
}
```

Далее переходим в конструктор, нажимаем на Стандартные элементы управления и добавляем TextBox, Button, label, поочерёдно перетаскивая их в наше окно.



Немного редактируем нашу форму, нажав на Button1, открываем свойства Text и вписываем туда Копировать, после чего меняем размеры формы и TextBox.



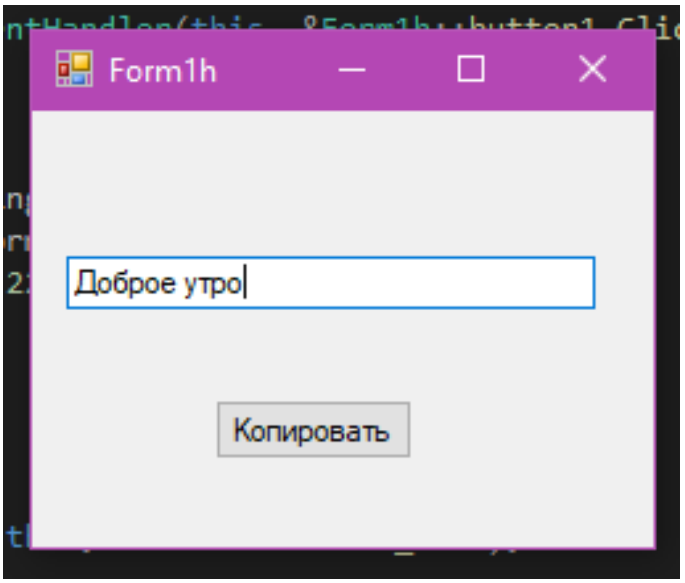
Кликаем дважды по нашей форме, появится окно Form1h.h с кодом. В событие Form1h_Load вписываем label1->Text = "", из-за чего при запуске программы текст в элементе label1 станет равным "".

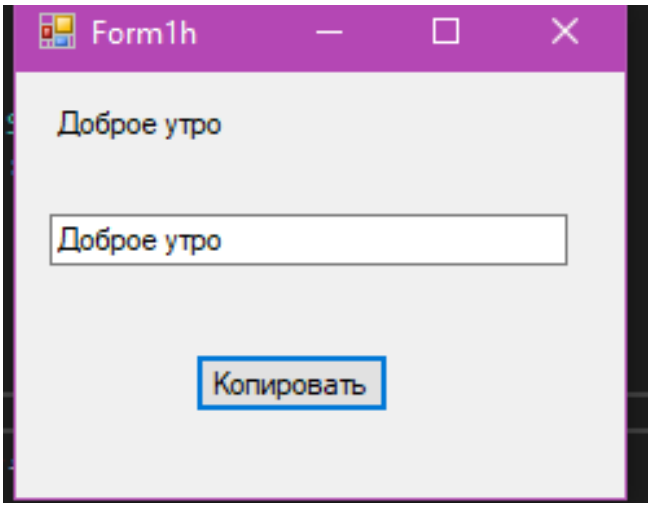
После дважды кликнув на кнопку Копировать откроем событие button1_Click и впишем туда label1->Text = textBox1->Text; Теперь при загрузке формы, кликнув на кнопку произойдет описанное действие – текст, написанный в textBox1 скопируется в label1. Теперь имеем следующий код:

```
#pragma endregion
```

```
private: System::Void Form1h_Load (System::Object^ sender, System::EventArgs^ e) {
label1->Text = "";
}
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
label1->Text = textBox1->Text;

}
```

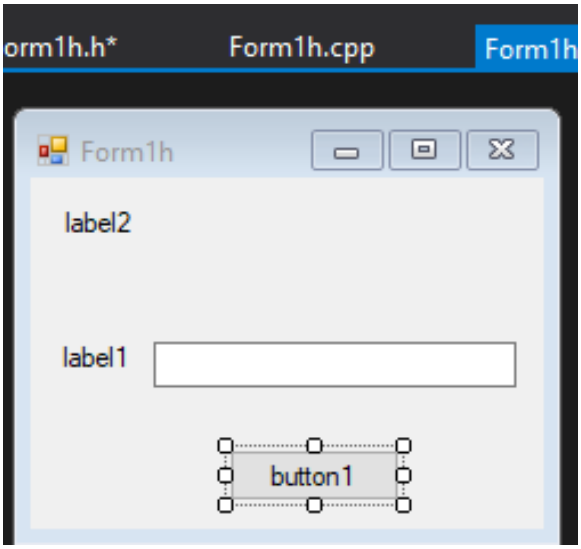
Запустим программу:





Занятие № 2. Элемент MessageBox, Подсказка ToolTip

Создание простой программы, в которой при нажатии на кнопку будет “выскакивать” небольшое окошко, сообщающее нам о том, какой текст был записан в текстовом поле. Добавим на форму button и textBox, а так же два label. Далее кликнем два раза сначала по форме, а затем по кнопке.



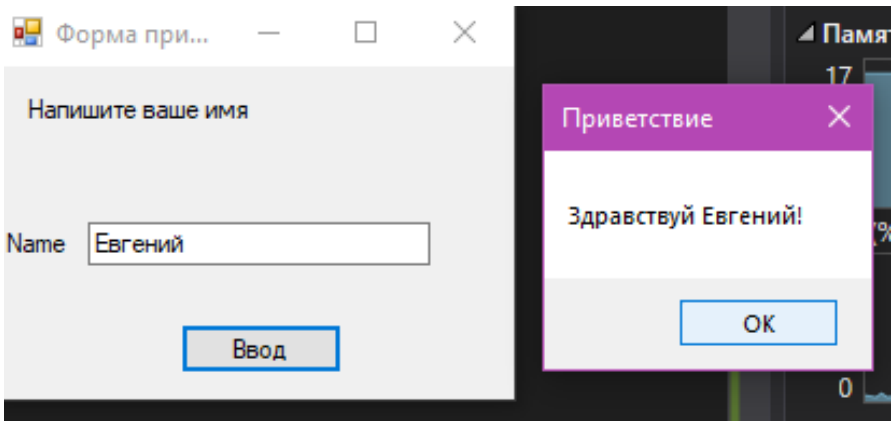

```
#pragma endregion
 private: System::Void Form1h_Load (System::Object^ sender, System::EventArgs^ e) {

 }
 private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {

 }
};
}
```

Сначала рассмотрим MessageBox, который будет приветствовать пользователя после ввода имени.

Дополним код:

```
private: System::Void Form1h_Load (System::Object^ sender,
System::EventArgs^ e) {this->Text = "Форма приветствия";
label1->Text = "Name: ";
label2->Text = "Напишите ваше имя.";button1-
>Text = "Ввод";
}
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
MessageBox::Show("Здравствуй " + textBox1->Text + "!", "Приветствие");

}
```

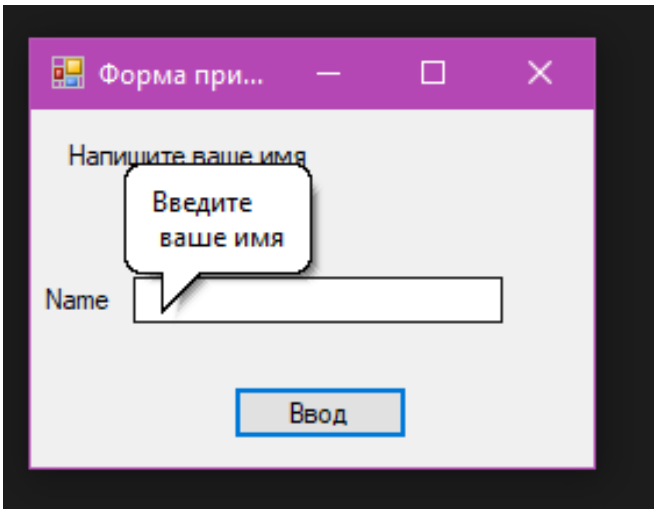
Теперь после ввода имени будет выводиться окно с приветствием.



Теперь рассмотрим элемент ToolTip. Он выводит текстовое поле, информирующее пользователя о чём-то(в нашем случае просит ввести имя) . Получится следующий код:

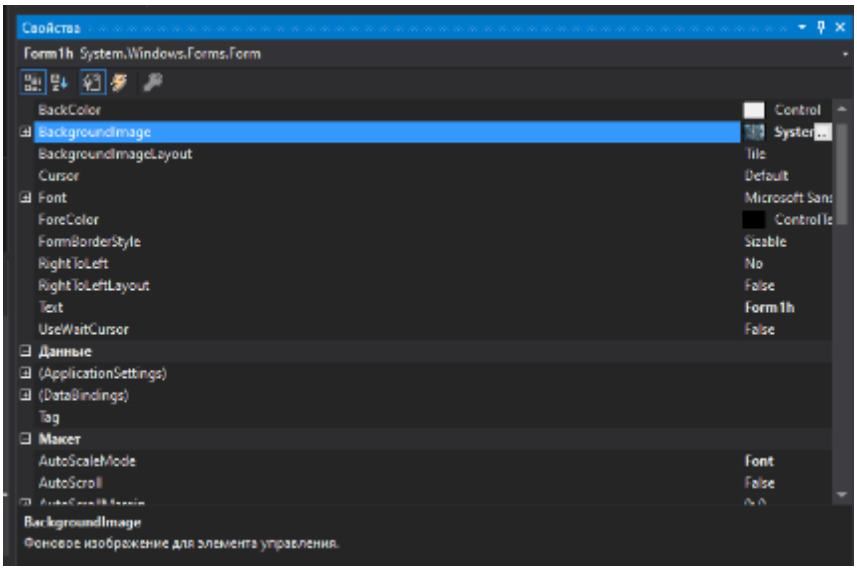
```
#pragma endregion
private: System::Void Form1h_Load (System::Object^ sender, System::EventArgs^ e) {
this->Text = "Форма приветствия";
label1->Text = "Name: ";
label2->Text = "Напишите ваше имя.";button1-
>Text = "Ввод";
toolTip1->SetToolTip(textBox1, "Введите\nваше имя");toolTip1-
>IsBalloon = true;
}
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
MessageBox::Show("Здравствуй " + textBox1->Text + "!", "Приветствие");
}
```

Теперь при наведении курсора на textBox1 будет выскакивать такое сообщение:

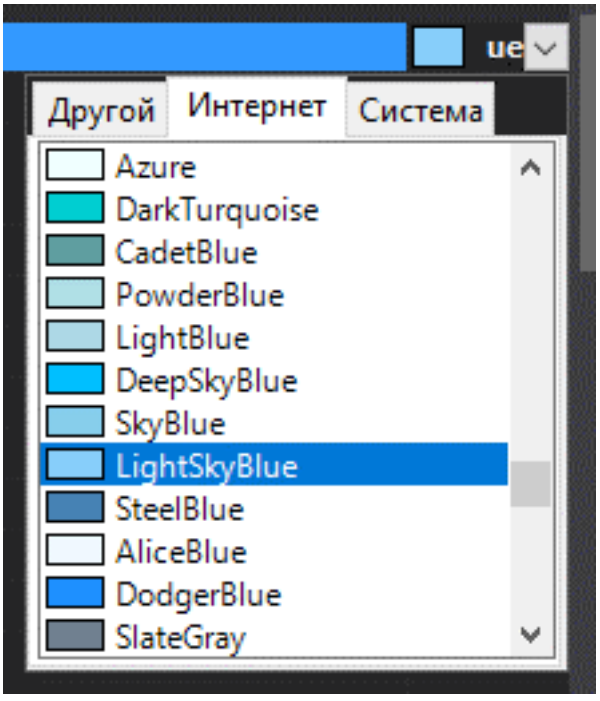


Занятие № 3. Изменение шрифта текста и цвета формы и элементов

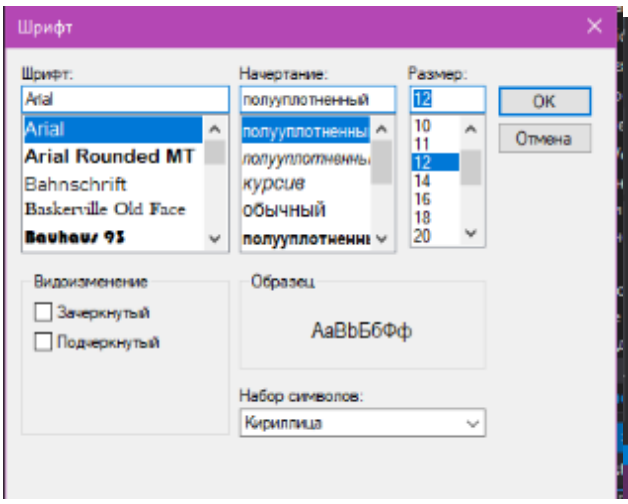
Нажмем на форму и найдем в ее свойствах BackGroundImage. После этого выберем Локальный ресурс, после чего нажмем на кнопку Импорт. Откроется проводник –в нём выбираем нужное изображение.

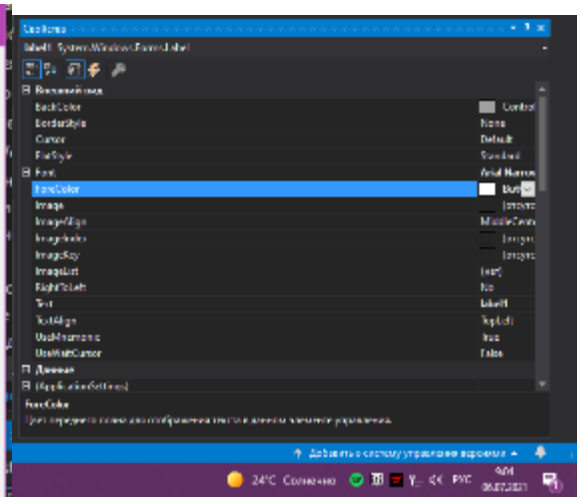


Далее нажимаем на элемент button, выбираем в его свойствах BackColor и ставим нужный цвет.



Теперь нужно изменить шрифт элемента label. Для этого нажмем на него и выберем свойство Font, выберем нужный размер и стиль шрифта.





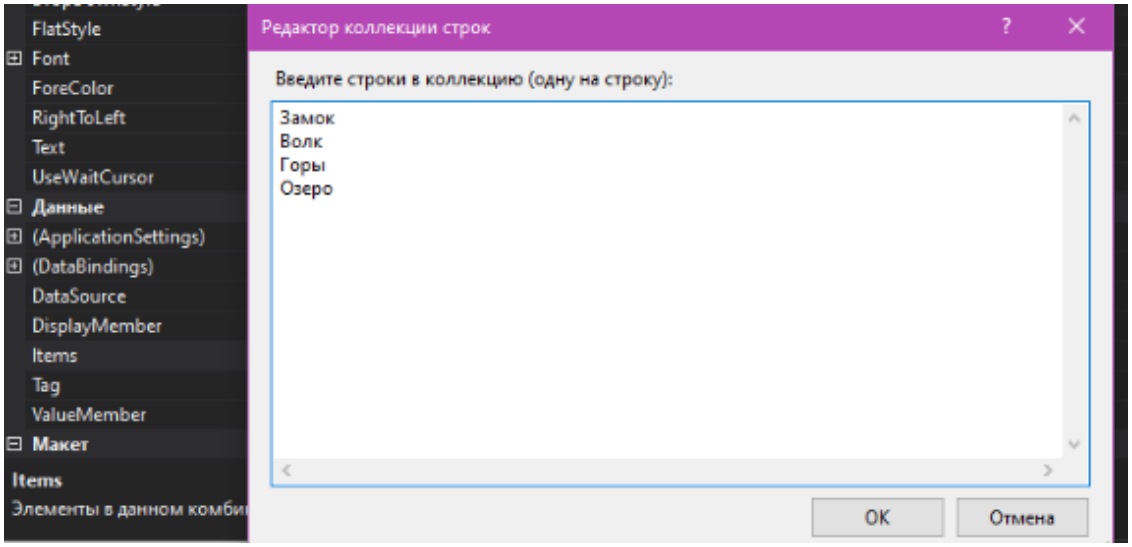
Вернёмся к коду:

```
#pragma endregion
private: System::Void Form1h_Load(System::Object^ sender, System::EventArgs^ e) {
this->Text = "Доска объявлений";
label1->Text = "";
}


private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{
label1->Text = textBox1->Text;

}
```

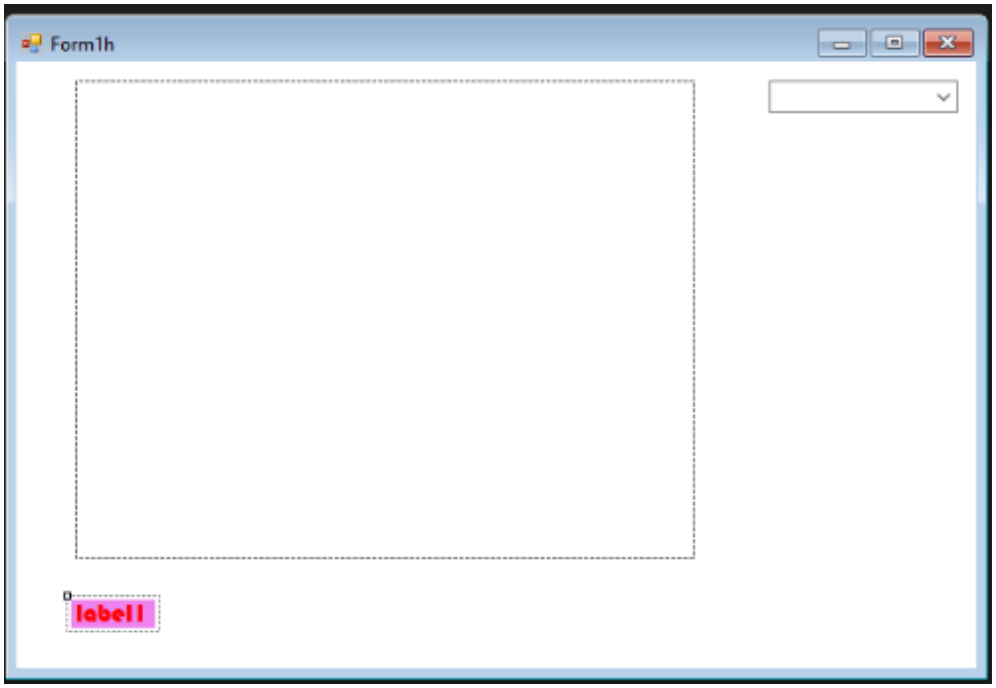
Теперь рассмотрим загрузку изображения в PictureBox при помощи ComboBox. Ранее мы узнали, как загружать изображение в коде.Теперь познакомимся с функционированием такого элемента, как comboBox.

Имеется 4 картинки. На форме будет comboBox, в котором находится некоторый список из четырех слов. При выборе одного из слов в списке должна появляться картинка, а в label ее название.

Для того что бы занести в comboBox некоторый список, нужно найти в панели свойств - свойство Items и написать через enter слова:



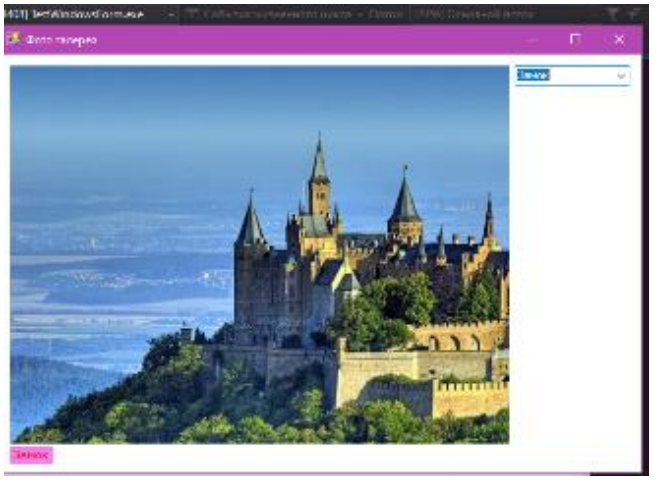
Также перенесем на форму lable и PictureBox. Получится форма:

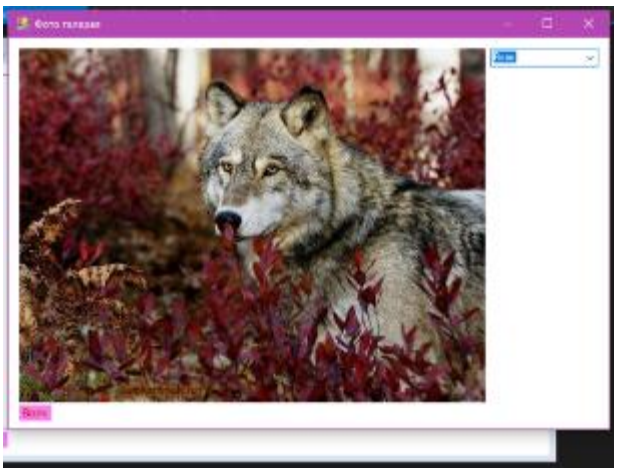


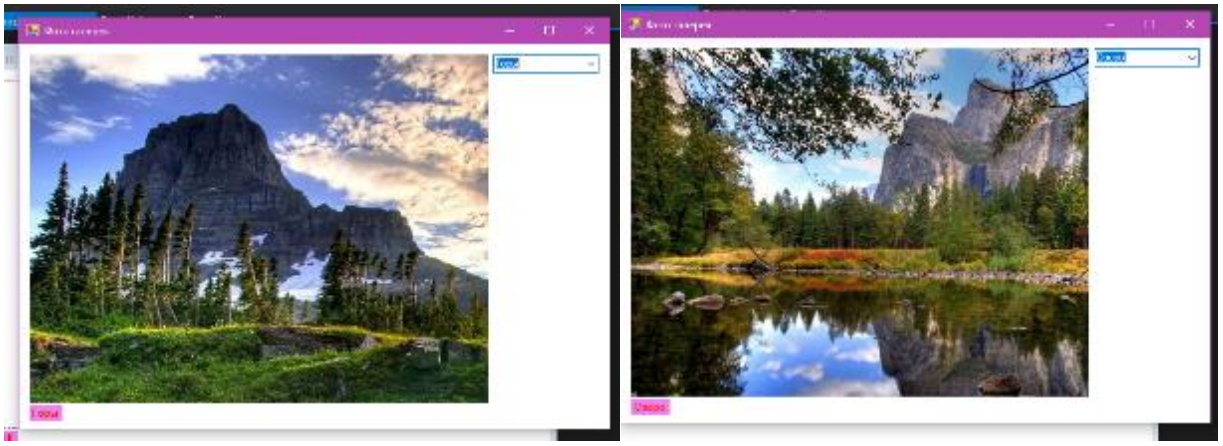
Перейдем к коду:

```
#pragma endregion
private: System::Void Form1h_Load (System::Object^ sender, System::EventArgs^ e) {this-
>Text = "Фото галерея";
label1->Text = ""; comboBox1->Text = "Список";}
private: System::Void comboBox1_SelectedIndexChanged(System::Object^ sender,
System::EventArgs^ e) {
switch (comboBox1->SelectedIndex){
case 0: pictureBox1->Image = Image::FromFile("d:\\Castle.jpg"); label1->Text = "Замок";
break;
        case 1: pictureBox1->Image = Image::FromFile("d:\\Wolf.jpg"); label1->Text =
"Волк"; break;
        case 2: pictureBox1->Image = Image::FromFile("d:\\Mountain.jpg"); label1->Text =
"Горы"; break;
        case 3: pictureBox1->Image = Image::FromFile("d:\\Lake.jfif"); label1->Text =
"Озеро"; break;
        }
}
};
```

В результате имеем:

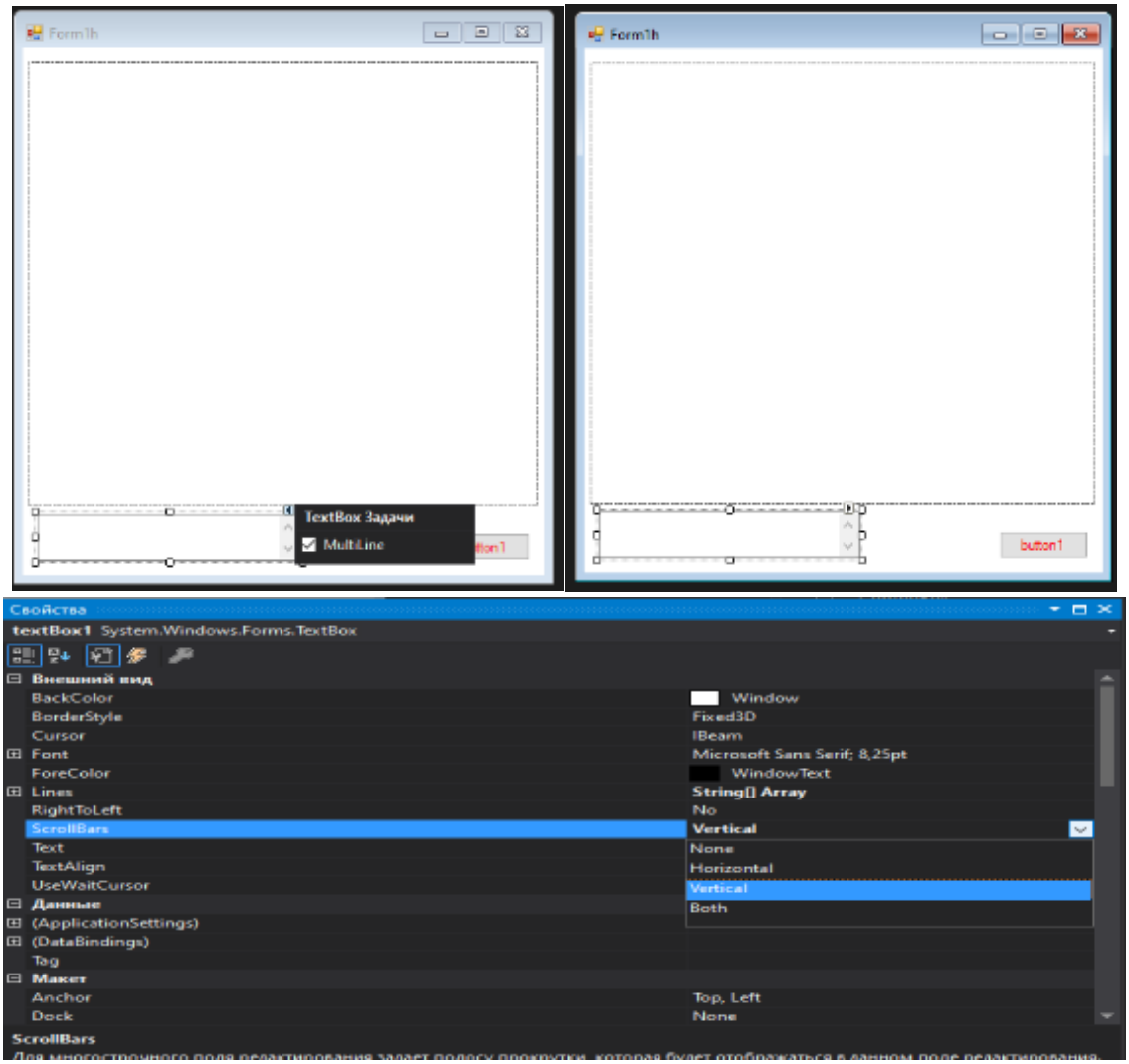






Рисование текста на PictureBox

Создадим новый проект Windows Forms и перетащим на форму три элемента: textBox, PictureBox и button. У textBox нужно включить режим Multeline и свойство ScrolBars.

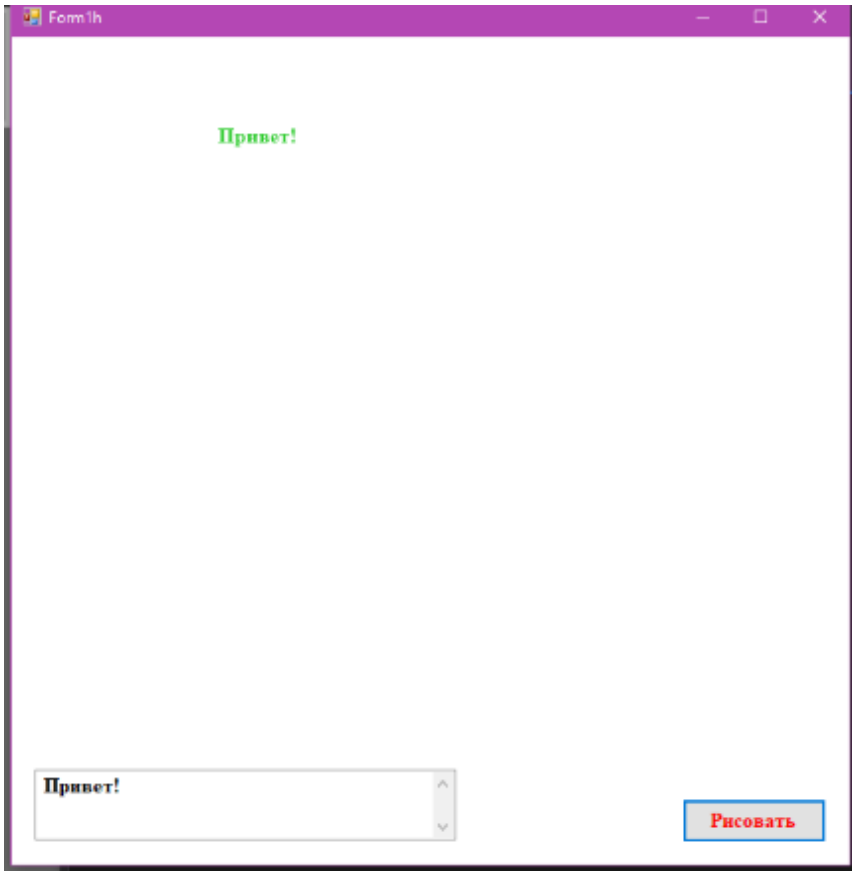


В коде программы задаётся размер, шрифт и цвет отображаемого текста.

```
#pragma endregion
private: System::Void Form1h_Load(System::Object^ sender, System::EventArgs^ e)
{
	Font = gcnew System::Drawing::Font("Times New Roman", 12, FontStyle::Bold);
	button1->Text = "Рисовать";
}
	private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
	{
		String^ Text = String::Format("{0}", textBox1->Text);
		Brush^ Кисть = gcnew SolidBrush(Color::LimeGreen);
		Graphics^ G = pictureBox1->CreateGraphics();
```

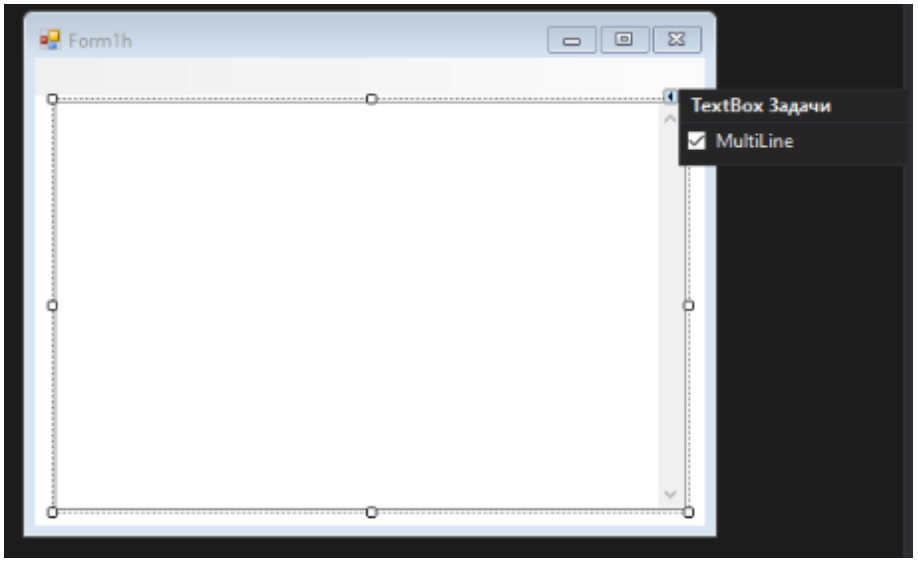
```
            G->TextRenderingHint = System::Drawing::Text::TextRenderingHint::AntiAlias;
            G->DrawString(Text, Font, Кисть, 150, 50); // Координаты размещения текста
        }
};
    }
```

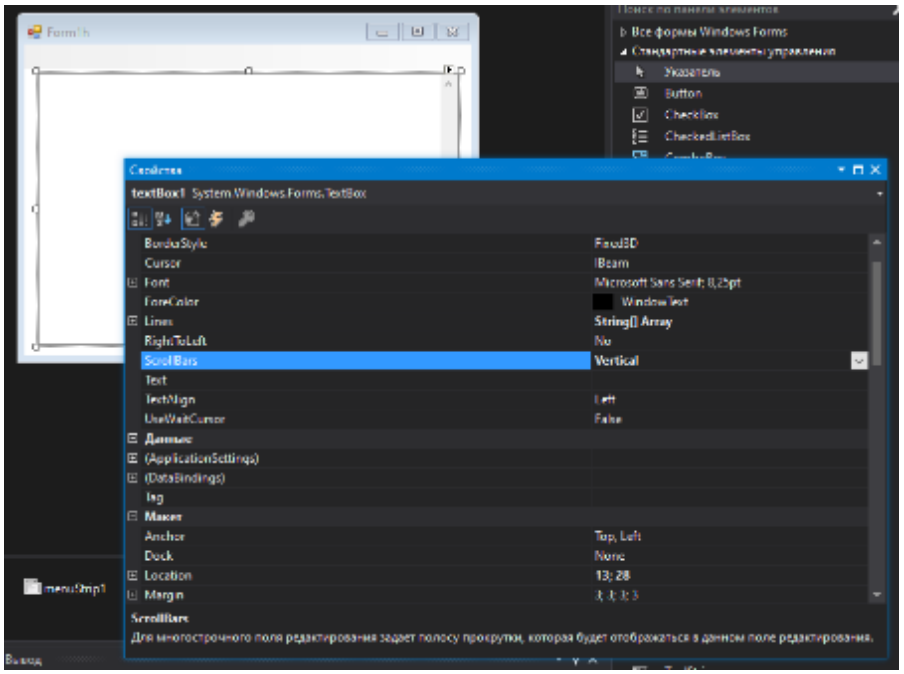
После вставки кода получится следующая картинка:

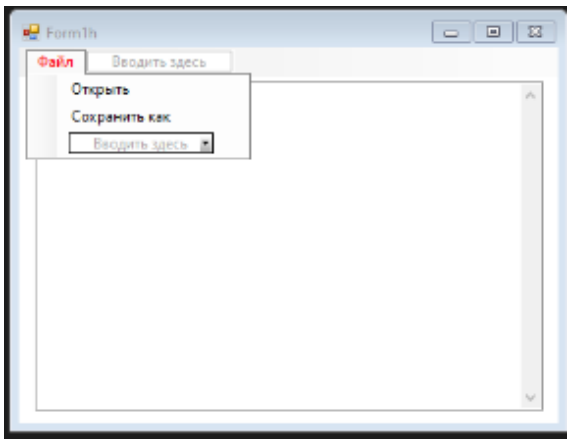


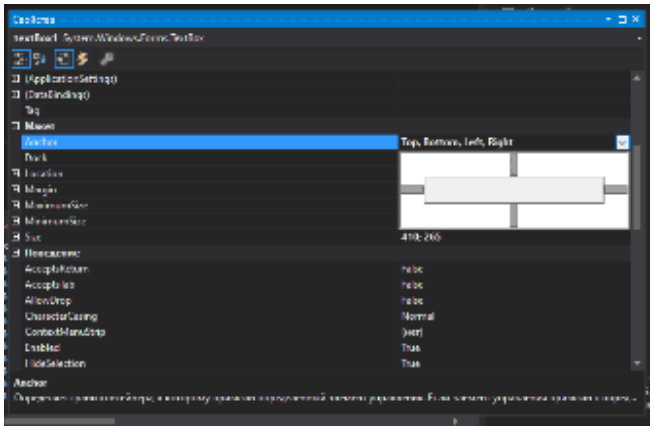
Занятие № 4. Элемент MenuStrip и свойство Anchor, Открытие и запись текстового файла

Перенесем на форму элемент "MenuStrip" и элемент "textBox". У элемента “textBox” включим "Multeline" и свойство "ScrollBars"->"Vertical".









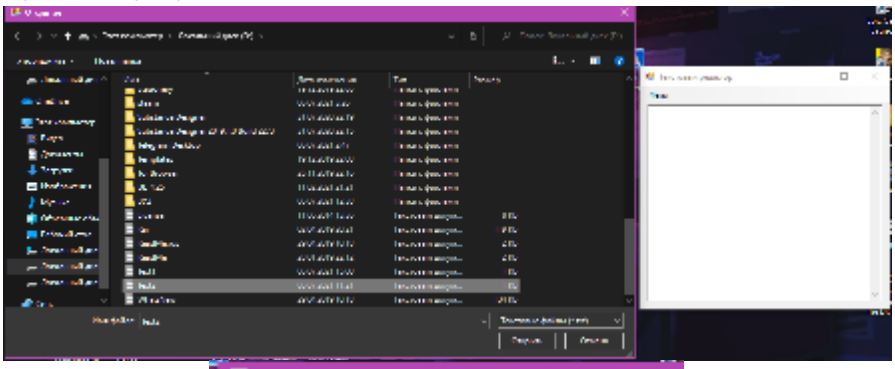
Код программы:

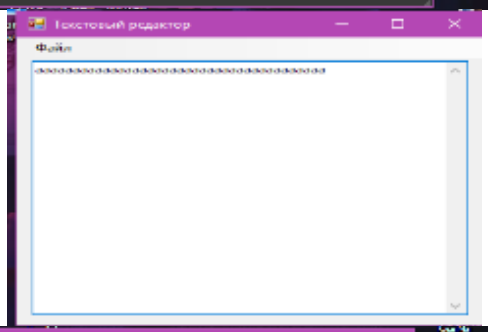
```
#pragma endregion
	private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e)
	{
		this->Text = "Текстовый редактор";
		openFileDialog1->FileName = "D:Text2.txt";
		openFileDialog1->Filter = "Текстовые файлы (*.txt)|*.txt|All files (*.*)|*.*";
		saveFileDialog1->Filter = "Текстовые файлы (*.txt)|*.txt|All files (*.*)|*.*";
	}
	private: System::Void открытьToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e)
	{
		openFileDialog1->ShowDialog();
		if (openFileDialog1->FileName == nullptr) return;
		try {
			auto MyReader = gcnew IO::StreamReader(openFileDialog1->FileName,
System::Text::Encoding::GetEncoding(1251));
			textBox1->Text = MyReader->ReadToEnd(); MyReader->Close();
		}
		catch (IO::FileNotFoundException^ Ситуация) {
			MessageBox::Show(Ситуация->Message + "\nФайл не найден", "Ошибка",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
		}
		catch (Exception^ Ситуация) {
			MessageBox::Show(Ситуация->Message, "Ошибка", MessageBoxButtons::OK,
MessageBoxIcon::Exclamation);
		}
	}
	private: System::Void сохранитьToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) { saveFileDialog1->FileName = openFileDialog1->FileName; if
(saveFileDialog1->ShowDialog() == Windows::Forms::DialogResult::OK) Save(); } void Save() {
		try { // Создание экземпляра StreamWriter для записи в файл:
			auto MyWriter = gcnew IO::StreamWriter(saveFileDialog1->FileName,
false, System::Text::Encoding::GetEncoding(1251));
			MyWriter->Write(textBox1->Text);
			MyWriter->Close();
			textBox1->Modified = false;
		}
		catch (Exception^ Ситуация)
```

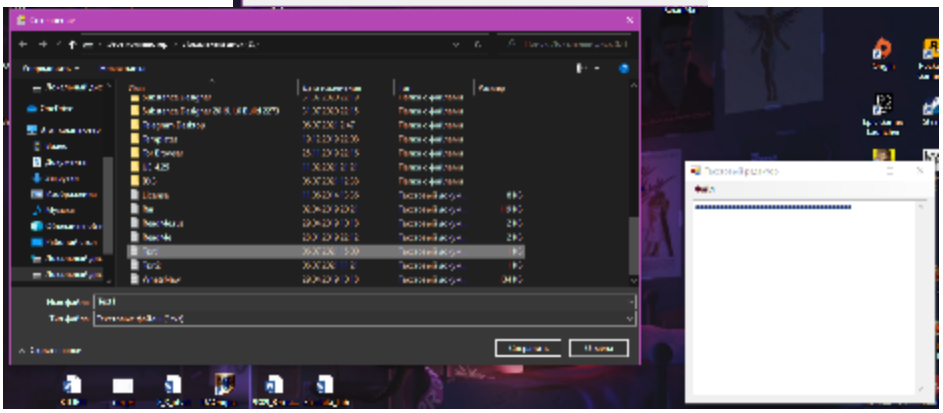
```
		{
			MessageBox::Show(Ситуация->Message, "Ошибка", MessageBoxButtons::OK,
MessageBoxIcon::Exclamation);
		}
	}
	private: System::Void выходToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e)
	{
	this->Close();
	}
private: System::Void Form1_FormClosing(System::Object ^ sender,
System::Windows::Forms::FormClosingEventArgs ^ e) {
	if (textBox1->Modified == false) return;
	auto MeBox = MessageBox::Show("Текст был изменен. \nСохранить изменения?", "Простой
редактор", MessageBoxButtons::YesNoCancel, MessageBoxIcon::Exclamation);
	if (MeBox == Windows::Forms::DialogResult::No) return;
	if (MeBox == Windows::Forms::DialogResult::Cancel) e->Cancel = true;
	if (MeBox == Windows::Forms::DialogResult::Yes) {
		if (saveFileDialog1->ShowDialog() == Windows::Forms::DialogResult::OK)
		{ Save(); return; }
		else e->Cancel = true;
	}
}

};
```

}

Результат работы программы:

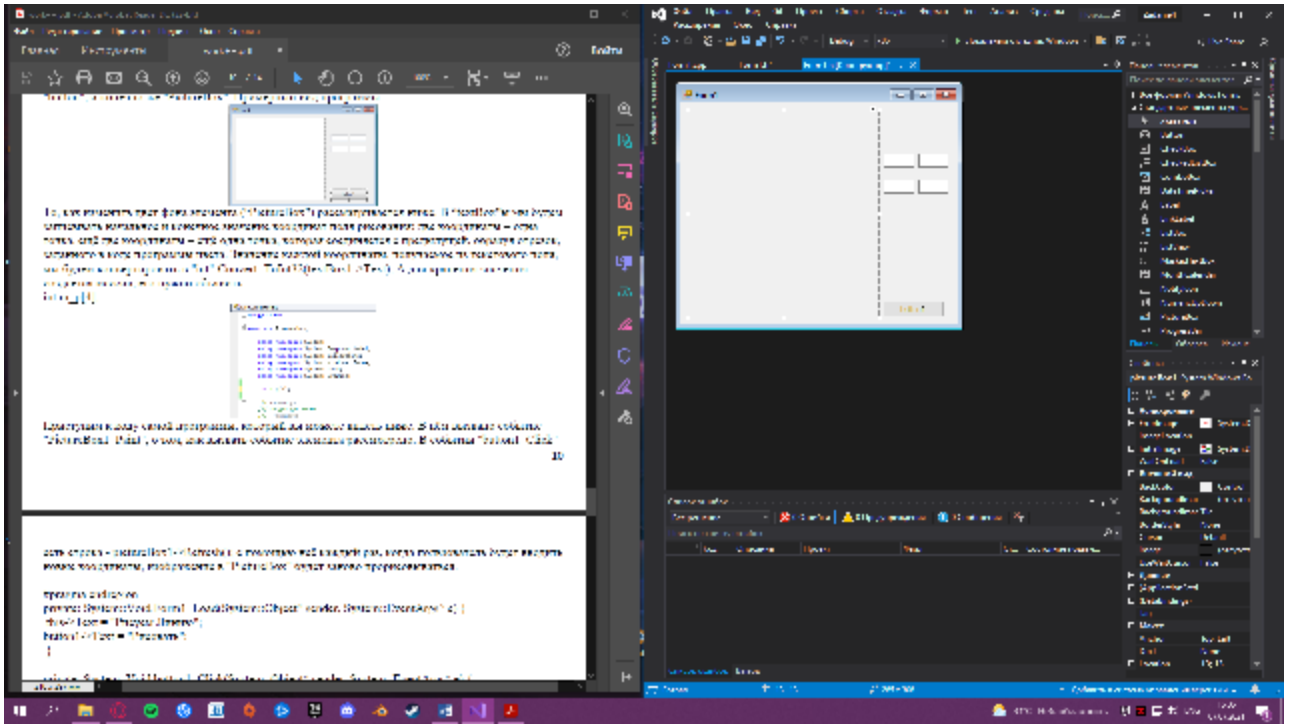






Рисование линий, треугольника, эллипса и окружности в PictureBox

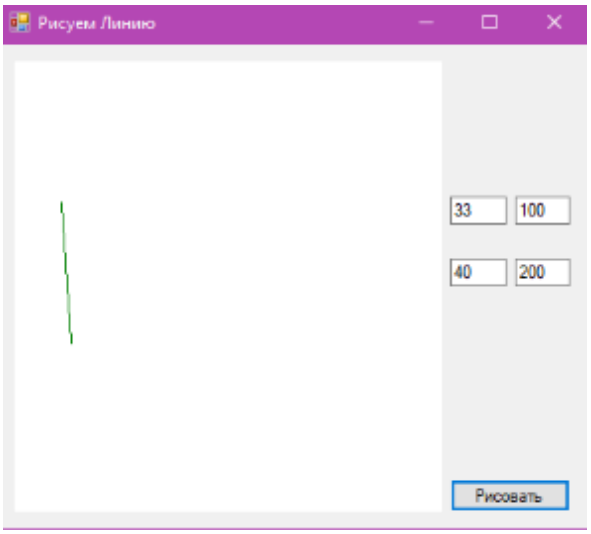
Для этого нам понадобятся: 4 "textBox", "button" и "PictureBox".



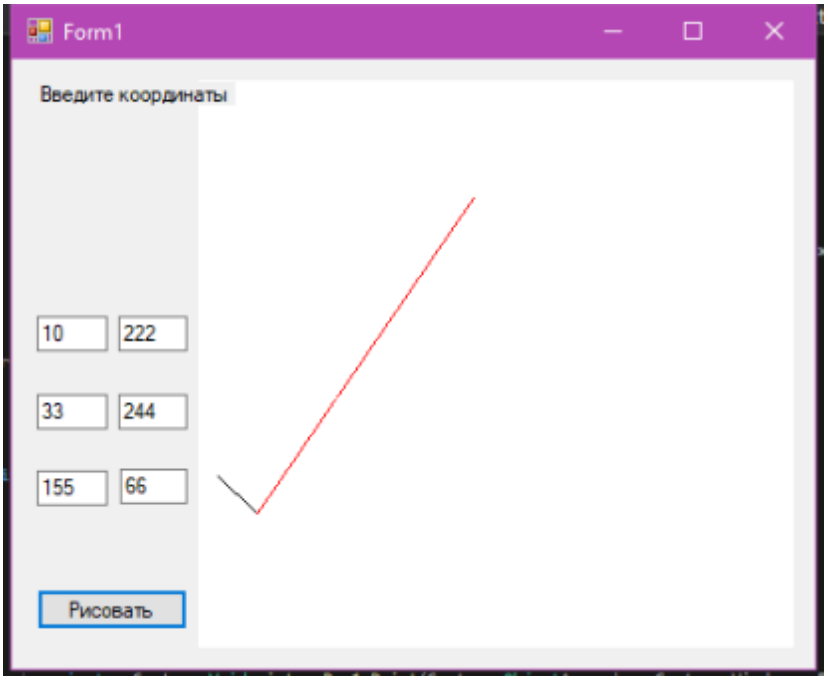
В “textBox”ы мы будем записывать начальное и конечное значение координат поля рисования: две координаты – одна точка, еще две координаты

– ещѐ одна точка, которая соединяется с предыдущей, образуя отрезок, заданного в коде программы цвета. Значение каждой координаты, получаемое из текстового поля, мы будем конвертировать в "int"- Convert::ToInt32(textBox1->Text); А для хранения значения создается массив.

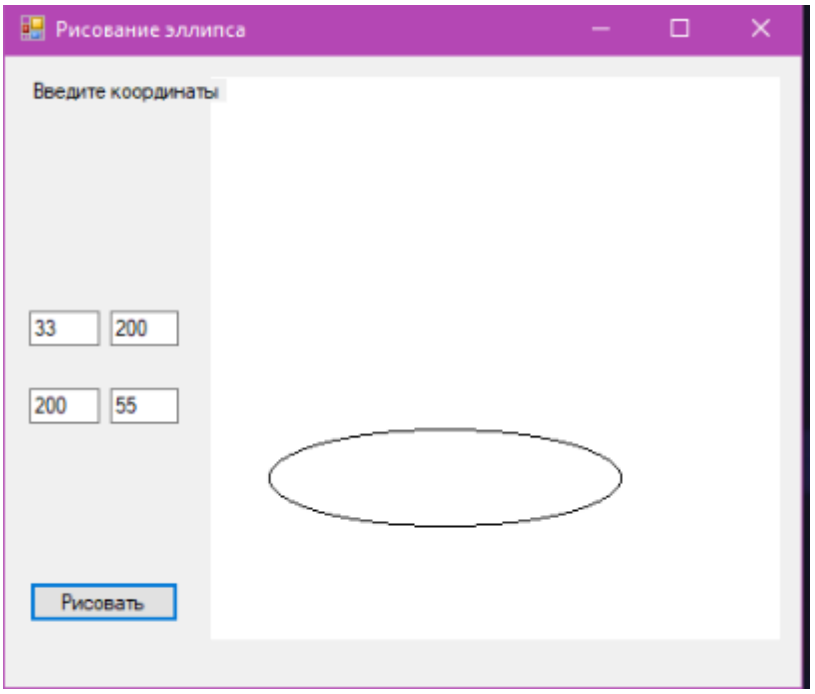
Получается такой результат:



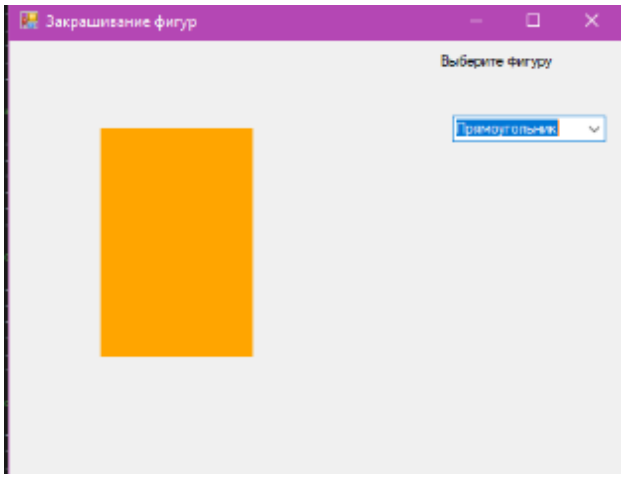
Нарисуем треугольник:

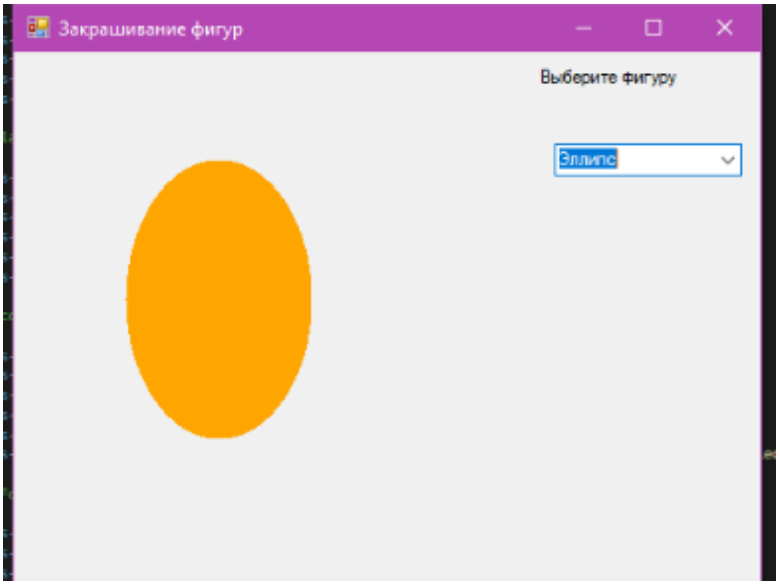


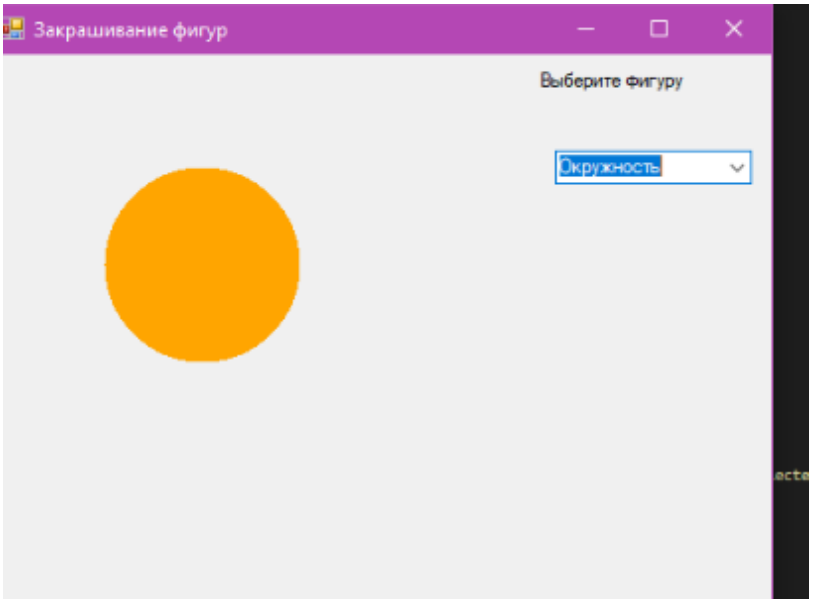
Эллипс:



Закрасим фигуры:



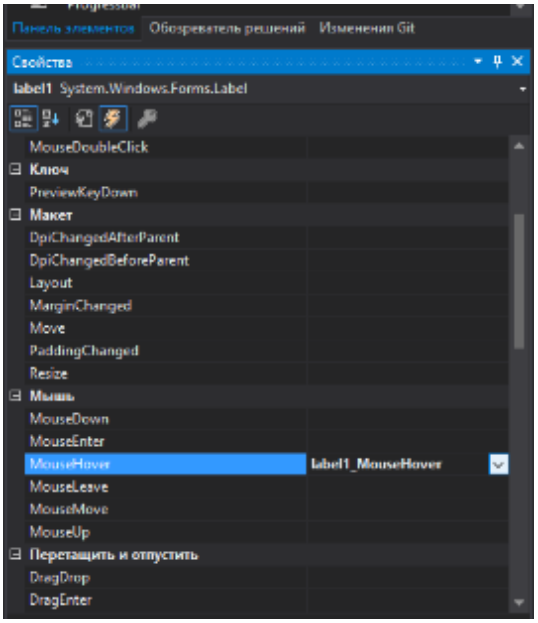

Закрашивание фигур
Выберите фигуру
Эллипс


Закрашивание фигур
Выберите фигуру
Окружность

Событие MouseHover

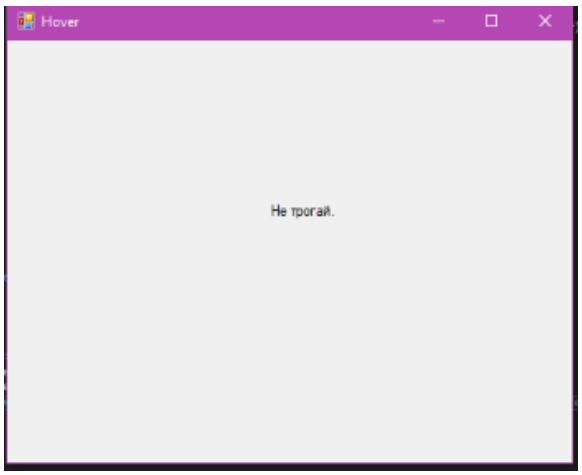
При наведении курсора мыши на надпись "не трогай" после чего изменяется цвет и цвет текст, а также выскакивает “MessageBox”, с какой-нибудь надписью.

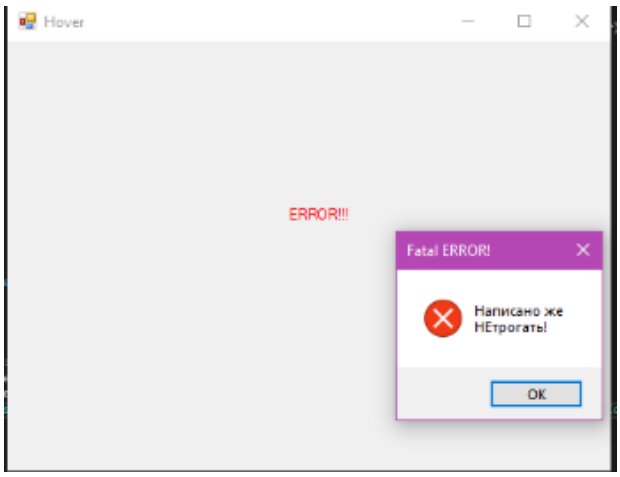
Выберем событие MouseHover в окне событий у label1



Открываем его код и вводим туда:

```
#pragma endregion
 private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e) {
             Form1::Text = "Hover";
             label1->TextAlign = ContentAlignment::MiddleCenter;
             label1->Text = "Не трогай!"; }

private: System::Void label1_MouseHover(System::Object^ sender, System::EventArgs^ e) {
       label1->TextAlign = ContentAlignment::MiddleCenter; label1->Text = "ERROR!!!";
       label1->ForeColor = Color::Red;
       MessageBox::Show("Написано же\nНЕ трогать!", "Fatal ERROR!",
MessageBoxButtons::OK, MessageBoxIcon::Error);
}
```




Hover
ERROR!!!
Fatal ERROR!
Написано же
НЕтрогать!
OK

Формирование траектории для движения простого геометрического объекта

Пусть гипоциклоида задается параметрическим уравнением:

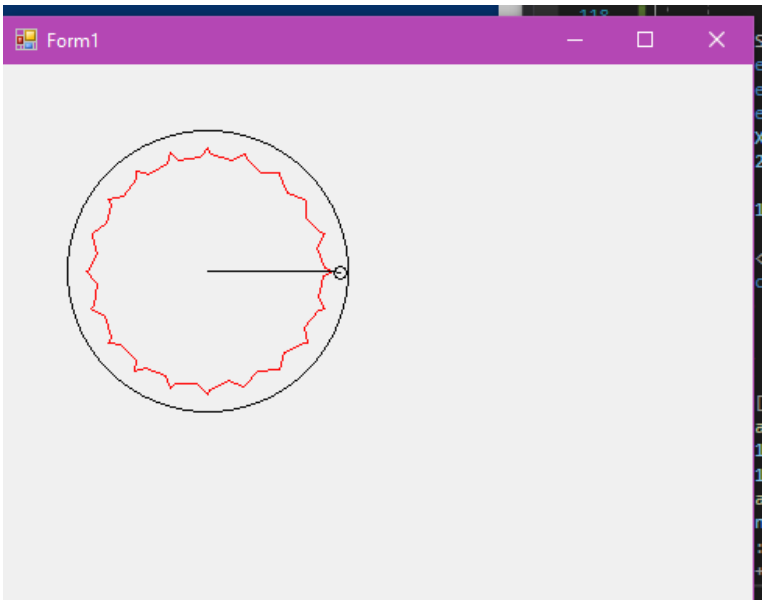
$$\begin{cases} x = r(k-1)\left(\cos t + \frac{\cos((k-1)t)}{k-1}\right) \\ y = r(k-1)\left(\sin t - \frac{\sin((k-1)t)}{k-1}\right) \end{cases}$$

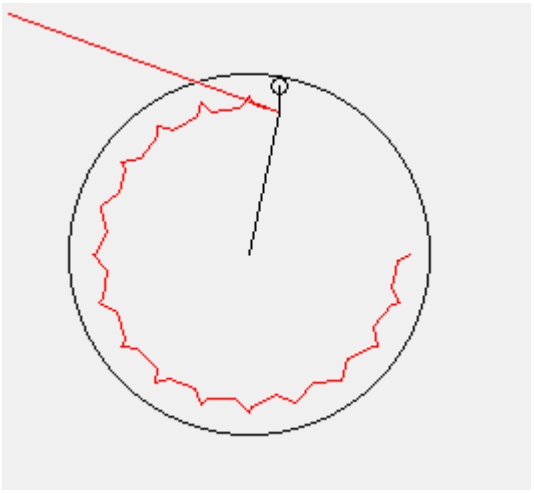
Используем библиотеки `<Windows.h>` для остановки по времени и `<cmath>` для расознания компилятором sin и cos.

Код программы:

```
#pragma endregion
 private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e) {
 }

       private: void Paint_Circle(int cX, int cY, int centX, int centY, int radius, int
x, int y)
       {
              Graphics^ Графика = pictureBox1->CreateGraphics();
              Графика->DrawEllipse(Pens::Black, centX + cX - radius, cY - radius - centY,
radius * 2, radius * 2);
              Графика->DrawLine(Pens::Black, centX + cX, cY - centY, cX + x, cY + y);
              // прорисовка радиуса большей окружности
       }
       private: void Paint_Graphic(int cX, int cY, int r2, int x, int y, array<Point>^ p)
       {
              Graphics^ Графика = pictureBox1->CreateGraphics();
              Графика->Clear(BackColor);
              Paint_Circle(cX, cY, 0, 0, r2, x, y);
              Графика->DrawLines(Pens::Red, p); // траектория
       }


private: System::Void pictureBox1_Click(System::Object^ sender, System::EventArgs^ e) {
       double InitT = 0, LastT = 6.3; // оборот в 360 градусов (6,28 радиан)
       double Step = 0.1, angle = InitT;
       double x, y, x1, y1;
       int cX = 120, cY = 120; // центр большой окружности
       int R2 = 90; // радиус большой окружности
       int k = 20; // число областей на траектории
       int R1 = int(R2 / k); // радиус меньшей (движущейся) окружности
       int i = 0; // количество точек прорисовки
       array<Point>^ p;
       p = gcnew array<Point>(64); // точки для прорисовки (LastT/Step)
       while (angle <= LastT)
       {
              x = R1 * (k - 1) * (cos(angle) + cos((k - 1) * angle) / (k - 1));
              y = R1 * (k - 1) * (sin(angle) - sin((k - 1) * angle) / (k - 1));
              p[i] = Drawing::Point(cX + int(x), cY + int(y)); // расчет очередной точки
траектории
              Paint_Graphic(cX, cY, R2, x, y, p);
              x1 = (R2 - R1) * sin(angle + 1.57);
              y1 = (R2 - R1) * cos(angle + 1.57);
              Paint_Circle(cX, cY, int(x1), int(y1), R1, x, y);
              angle += Step;
              ::Sleep(90); //время приостановки прорисовки
              i++;
       }

     }
```

Результат работы программы:





Вывод

В ходе выполнения данной работы были приобретены базовые навыки работы с Windows Forms на языке C++, изучены такие элементы формы: button, textbox, label, MessageBox, ToolTip, PictureBox, ComboBox. Также изучены способы вызова событий, редактирование свойств элементов формы,приобретены базовые навыки графической работы с элементом PictureBox.

Индивидуальное задание 1

Цель работы

Научиться работать с графическими примитивами в Windows Forms MVS на языке программирования C#.

Формулировка задания

По ранее реализованному образцу действий по формированию движения по траектории необходимо реализовать

Приложение, которое обеспечивает движение заданного объекта по заданной траектории.

Элементы управления и настройки, меню и все подходящее под параметрическую настройку должны обеспечить:

- максимальное задание всех мыслимых свойств используемых в задании объектов, в частности:

- выбор цветов (объекта и траектории)

- выбор толщины линий (объекта и траектории)

- выбор вида линий (сплошная, пунктирная, штриховая и т.д.)

- пульсирование размера объекта (от min к max) с разной скоростью

- выбор скорости движения объекта по траектории (например, числовое задание или из выпадающего списка)

- в отдельных заданиях выбор стиля и цвета заливки

- определение скорости движения объекта по траектории

- число повторов прохождения объекта по траектории

- шаг перемещения объекта по траектории

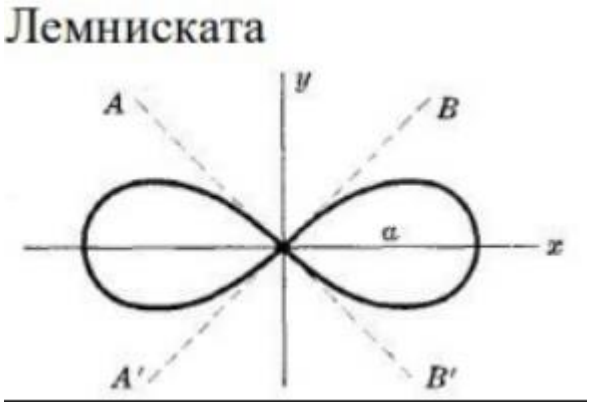
- шаг изменение угла объекта при его вращении вокруг центра (вершины, ключевой точки и т.п.)

- направление движение объекта (по/против часовой стрелки или вперед/назад и т.п.)

- и все остальное приходящее в голову по характеристикам объекта или движения.

Математическая постановка

Необходимо графически реализовать движение шестиугольника по траектории «Лемниската»:



Для этого были написаны формулы для точек данной траектории на языке c#:

Для отрезка: -pi/4;pi/4:

```
double r = Math.Sqrt(R * R * Math.Cos(2 * angle));
x = r * Math.Cos(angle);
y = r * Math.Sin(angle);
```

в которых angle является углом в радианах, а R – радиус лепестка.

Для отрезка: 3*pi/4;5*pi/4:

```
double r = Math.Sqrt(R * R * Math.Abs(Math.Cos(2 * angle)));
x = r * Math.Cos(angle);
y = r * Math.Sin(angle);
```

Используемые элементы интерфейса

В интерфейсы программы были использованы такие элементы, как:

1) Label – из свойств изменялось только название

2) ComboBox – из свойств заменялась коллекция, которую содержит данный элемент
3) PictureBox – изменялось свойство, отвечающее за содержащуюся внутри картинку, и её расположение внутри элемента.
4) TextBox – в свойствах изменялся только текст элемента
5) ToolTip – в свойствах изменялись значения, отвечающие за частоту появления подсказки, а так же вид подсказки
6) Button – в свойствах изменялось название элемента

Графические примитивы:

В программе были использованы следующие графические примитивы:

1) Pen – использовалось для изменения цвета контуров
2) SolidBrush – использовалось для изменения цвета заливки фигуры
3) DrawLines – использовалось для прорисовки линий между точками траектории
4) DrawPolygon – использовалось для прорисовки фигуры (шестиугольника)
5) FillPolygon – использовалось для заливки фигуры

Текст программы

```
using System;
using System.Collections.Generic;
using System.Threading;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Practice
{
    public partial class Form1 : Form
    {
        Pen penTr = new Pen(DefaultBackColor);
        Pen penB = new Pen(DefaultBackColor);
        SolidBrush brushB = new SolidBrush(DefaultBackColor);
        PointF[] box = new PointF[4];
        int pulse = 20, pulse_min = 20, pulse_max = 20;
        int multiplier = 1;
        public Form1()
        {
            InitializeComponent();
            penTr.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid;
            penB.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid;
```

```
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            this.Text = "Рисование фигуры";
            button1.Text = "Запуск";
            toolTip1.SetToolTip(textBox3, "min");
            toolTip2.SetToolTip(textBox4, "max");
            toolTip3.SetToolTip(textBox8, "x");
            toolTip4.SetToolTip(textBox9, "y");
            label11.Text = "Движение по \nчасовой стрелке.";
        }
        private void Paint_Graphic(float cX, float cY, float x, float y, PointF[] p)
        {
            Graphics Графика = pictureBox1.CreateGraphics();
            Графика.Clear(BackColor);
            Графика.DrawLines(penTr, p); // траектория
            double angle = -Math.PI * 0.5;
            PointF[] points = new PointF[6];
            for (int i = 0; i < 6; i++)
            {
                points[i].X = (float)(cX + x + Math.Round(Math.Cos(angle + Math.PI * 2.0
* i / 6) * pulse));
                points[i].Y = (float)(cY + y + Math.Round(Math.Sin(angle + Math.PI * 2.0
* i / 6) * pulse));
            }
            Графика.DrawPolygon(penB, points);
            Графика.FillPolygon(brushB, points);
        }
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                double InitT = -Math.PI/4;
                double Step = 0.1, angle = InitT;
                double x, y;
                int R = 100;
                bool indicator = false;
                int cX = Convert.ToInt32(textBox8.Text), cY =
Convert.ToInt32(textBox9.Text); // центр большой окружности
                int i = 0; // количество точек прорисовки
                int speed = 50;
                int repeater = 1;
                int stepper = 0;
                PointF[] p = new PointF[64];// точки для прорисовки (LastT/Step)
                penTr.Width = Convert.ToSingle(textBox1.Text);
                penB.Width = Convert.ToSingle(textBox2.Text);
                pulse_min = Convert.ToInt32(textBox3.Text);
                pulse_max = Convert.ToInt32(textBox4.Text);
                if (pulse_max < pulse_min)
                {
                    MessageBox.Show("max < min!", "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    goto END;
                }
                pulse = pulse_min;
                speed = Convert.ToInt32(textBox5.Text);
                if (speed >= 0 && speed <= 100)
                {
                    speed = 100 - speed;
                }
                else if (speed < 0)
                {
                    MessageBox.Show("speed cannot be under 0!", "Ошибка!",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    goto END;
```

```
                }
                else
                {
                    speed = 0;
                }
                repeater = Convert.ToInt32(textBox6.Text);
                stepper = Convert.ToInt32(textBox7.Text);
                R = Convert.ToInt32(textBox10.Text);
                if (stepper == 0)
                {
                    MessageBox.Show("step cannot be 0!", "Ошибка!", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    goto END;
                }
                for (int ii = 0; ii < repeater; ii++)
                {
                    i = 0;
                    angle = InitT;
                    while (angle <= Math.PI/4)
                    {
                        if (pulse < pulse_max && indicator == false)
                        {
                            pulse++;
                        }
                        else if (pulse == pulse_max)
                        {
                            indicator = true;
                            pulse--;
                        }
                        else if (pulse == pulse_min)
                        {
                            indicator = false;
                        }
                        else if (pulse > pulse_min)
                        {
                            pulse--;
                        }
                        double r = Math.Sqrt(R * R * Math.Cos(2 * angle));
                        x = r * Math.Cos(angle);
                        y = r * Math.Sin(angle);
                        y = y * multiplier;
                        p[i].X = Convert.ToSingle(cX + x);
                        p[i].Y = Convert.ToSingle(cY + y);
                        for (int j = i; j < 64; j++)
                        {
                            p[j] = p[i];
                        }
                        if (i % stepper == 0 || i % 63 == 0)
                        {
                            Paint_Graphic(cX, cY, Convert.ToSingle(x),
Convert.ToSingle(y), p);
                        }
                        angle = angle + Step;
                        angle = Math.Round(angle, 1);
                        Thread.Sleep(speed); //время приостановки прорисовки
                        i++;
                    }
                    angle = 3 * Math.PI / 4;
                    while (angle <= 5*Math.PI / 4)
                    {
                        if (pulse < pulse_max && indicator == false)
                        {
                            pulse++;
                        }
                        else if (pulse == pulse_max)
```

```
                        {
                            indicator = true;
                            pulse--;
                        }
                        else if (pulse == pulse_min)
                        {
                            indicator = false;
                        }
                        else if (pulse > pulse_min)
                        {
                            pulse--;
                        }
                       double r = Math.Sqrt(R * R * Math.Abs(Math.Cos(2 * angle)));
                        x = r * Math.Cos(angle);
                        y = r * Math.Sin(angle);
                        y = y * (-multiplier);
                        p[i].X = Convert.ToSingle(cX + x);
                        p[i].Y = Convert.ToSingle(cY + y);
                        for (int j = i; j < 64; j++)
                        {
                            p[j] = p[i];
                        }
                        if (i % stepper == 0 || i % 63 == 0)
                        {
                            Paint_Graphic(cX, cY, Convert.ToSingle(x),
Convert.ToSingle(y), p);
                        }
                        angle = angle + Step;
                        angle = Math.Round(angle, 1);
                        Thread.Sleep(speed); //время приостановки прорисовки
                        i++;
                    }
                    p[i].X = cX;
                    p[i].Y = cY;
                    Paint_Graphic(cX, cY, 0, 0, p);
                }
            END:
                pulse_max = pulse_min;
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
            }
        }
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            switch (comboBox1.SelectedIndex)
            {
                case 0:
                    penTr.Color = Color.Black; break;
                case 1:
                    penTr.Color = Color.Green; break;
                case 2:
                    penTr.Color = Color.Blue; break;
                case 3:
                    penTr.Color = Color.DeepPink; break;
                case 4:
                    penTr.Color = Color.Orange; break;
                case 5:
                    penTr.Color = Color.Red; break;
            }
        }
        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
```

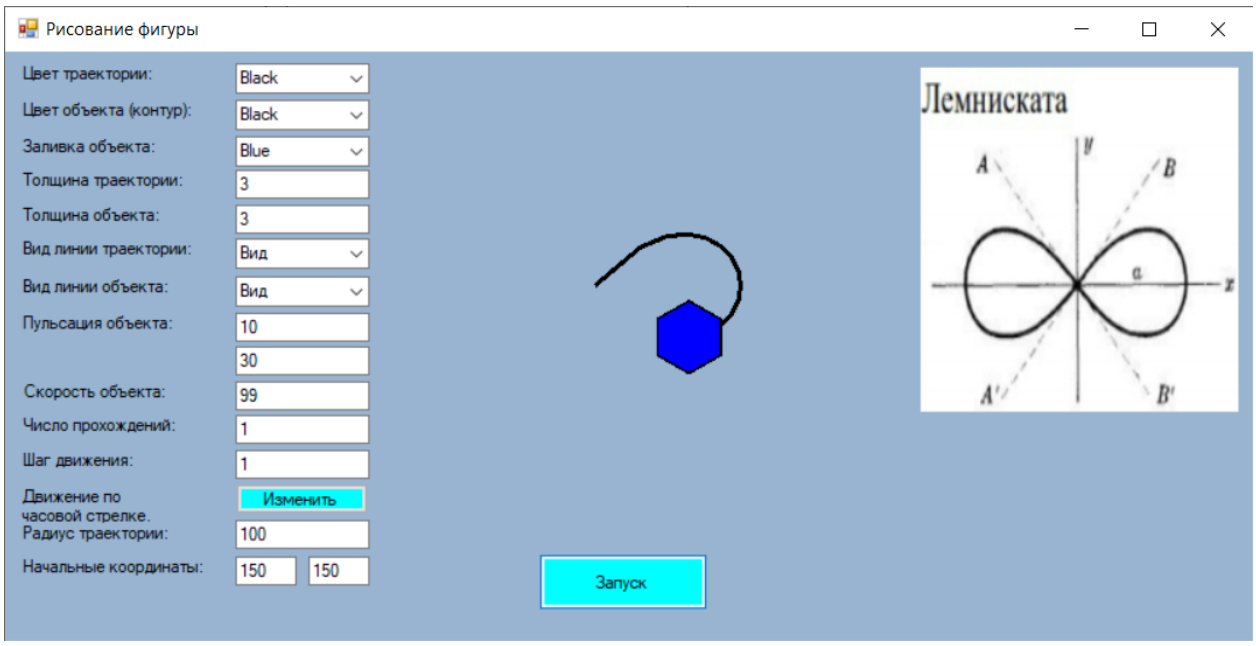
```
            switch (comboBox2.SelectedIndex)
            {
                case 0:
                    penB.Color = Color.Black; break;
                case 1:
                    penB.Color = Color.Green; break;
                case 2:
                    penB.Color = Color.Blue; break;
                case 3:
                    penB.Color = Color.DeepPink; break;
                case 4:
                    penB.Color = Color.Orange; break;
                case 5:
                    penB.Color = Color.Red; break;
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (multiplier == 1)
            {
                label11.Text = "Движение против \nчасовой стрелки.";
                multiplier = -1;
            }
            else
            {
                label11.Text = "Движение по \nчасовой стрелке.";
                multiplier = 1;
            }
        }
        private void comboBox5_SelectedIndexChanged(object sender, EventArgs e)
        {
            switch (comboBox5.SelectedIndex)
            {
                case 0:
                    brushB.Color = Color.Black; break;
                case 1:
                    brushB.Color = Color.Green; break;
                case 2:
                    brushB.Color = Color.Blue; break;
                case 3:
                    brushB.Color = Color.DeepPink; break;
                case 4:
                    brushB.Color = Color.Orange; break;
                case 5:
                    brushB.Color = Color.Red; break;
            }
        }
        private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
        {
            switch (comboBox3.SelectedIndex)
            {
                case 0:
                    penTr.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid; break;
                case 1:
                    penTr.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash; break;
                case 2:
                    penTr.DashStyle = System.Drawing.Drawing2D.DashStyle.Dot; break;
                case 3:
                    penTr.DashStyle = System.Drawing.Drawing2D.DashStyle.DashDot; break;
                case 4:
                    penTr.DashStyle = System.Drawing.Drawing2D.DashStyle.DashDotDot;
break;
            }
        }
```

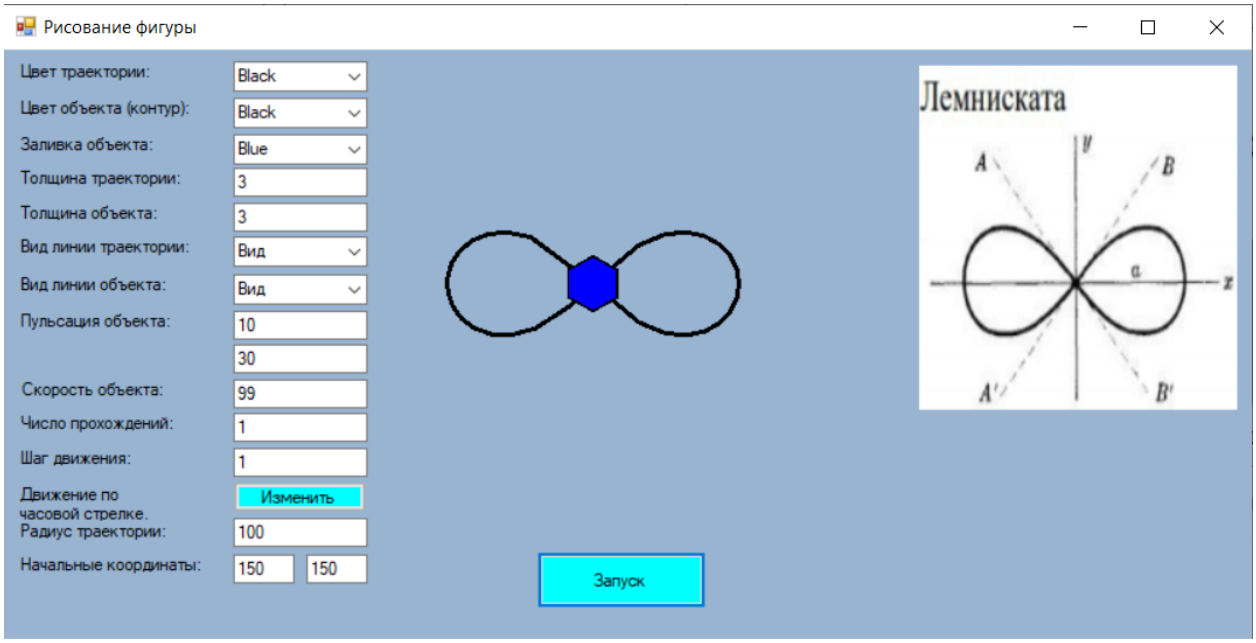
```
        private void comboBox4_SelectedIndexChanged(object sender, EventArgs e)
        {
            switch (comboBox4.SelectedIndex)
            {
                case 0:
                    penB.DashStyle = System.Drawing.Drawing2D.DashStyle.Solid; break;
                case 1:
                    penB.DashStyle = System.Drawing.Drawing2D.DashStyle.Dash; break;
                case 2:
                    penB.DashStyle = System.Drawing.Drawing2D.DashStyle.Dot; break;
                case 3:
                    penB.DashStyle = System.Drawing.Drawing2D.DashStyle.DashDot; break;
                case 4:
                    penB.DashStyle = System.Drawing.Drawing2D.DashStyle.DashDotDot;
break;
            }
        }
    }
}
```

Примеры работы

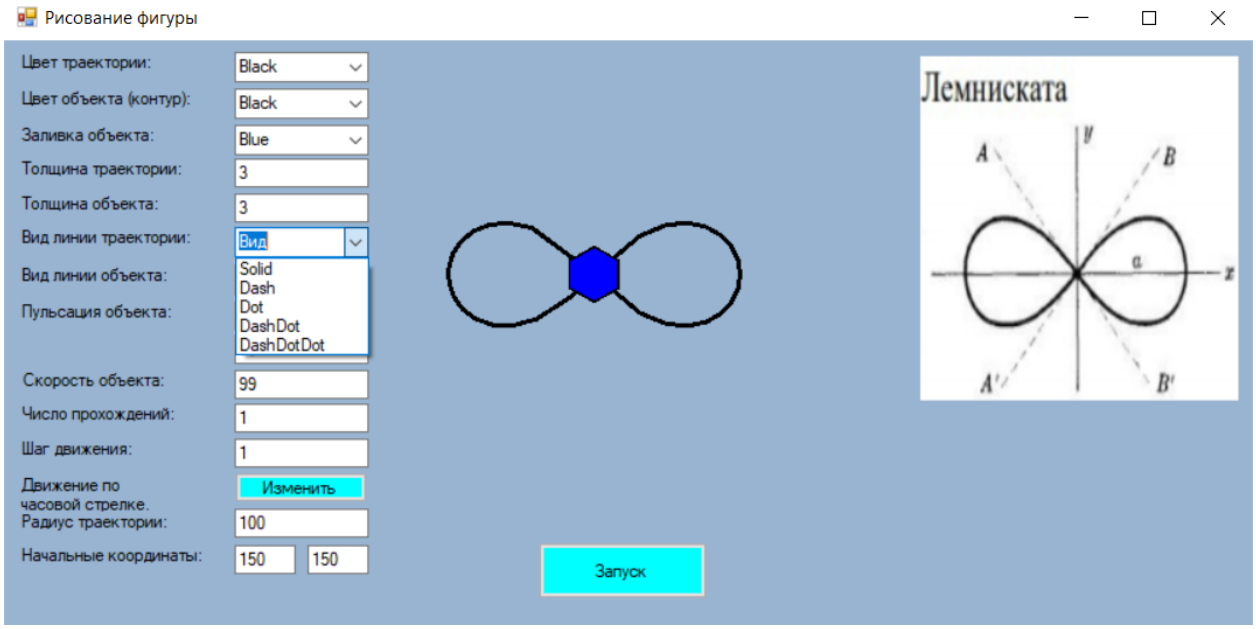
Программа на этапе запуска:

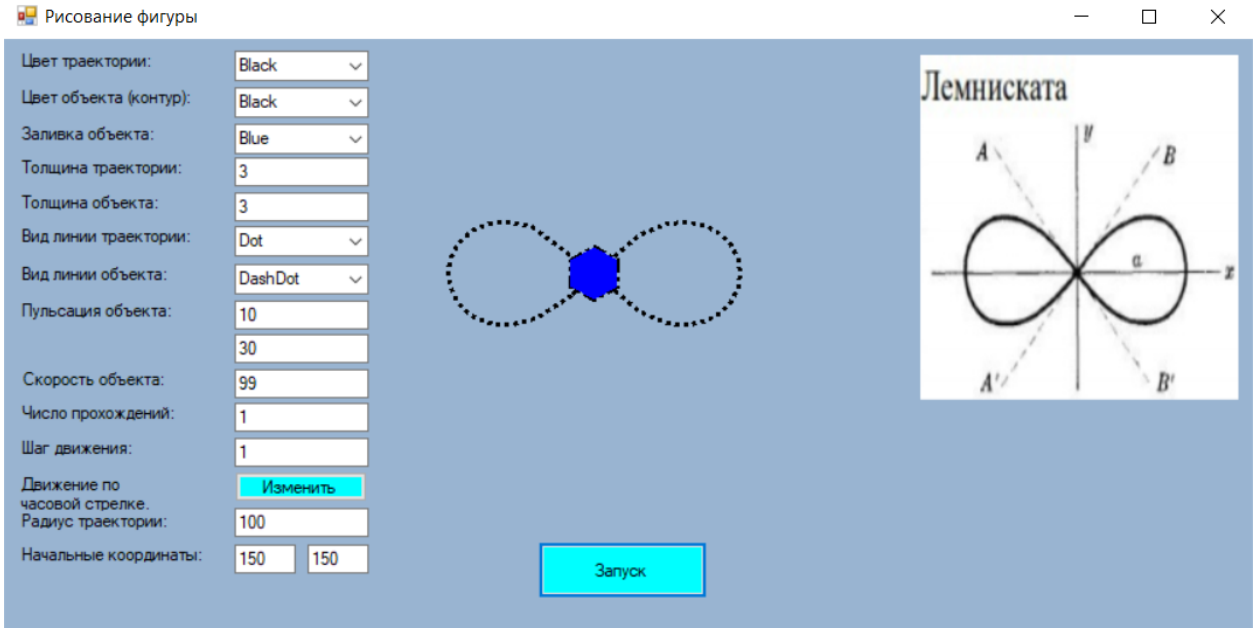
Проверка правильности прорисовки фигуры и траектории:



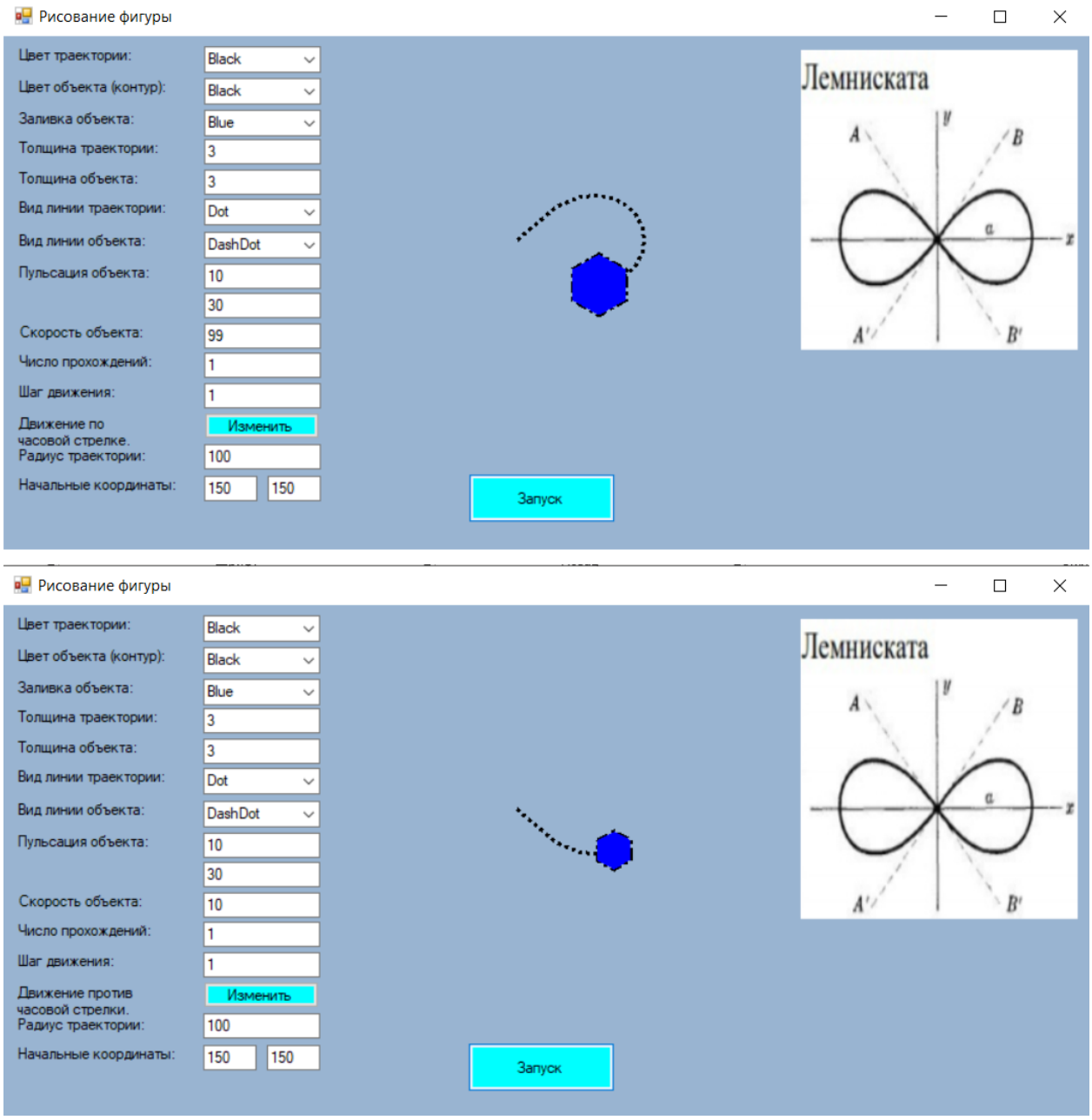


Проверка правильности изменения контуров:





Проверка изменения траектории движения:

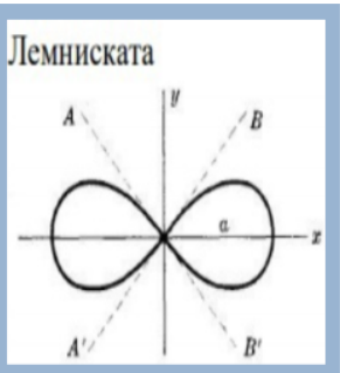


Проверка пульсации фигуры:

Рисование фигуры

Цвет траектории: Black
Цвет объекта (контур): Black
Заливка объекта: Blue
Толщина траектории: 3
Толщина объекта: 3
Вид линии траектории: Dot
Вид линии объекта: DashDot
Пульсация объекта: 10
30
Скорость объекта: 10
Число прохождений: 100
Шаг движения: 1
Движение против часовой стрелки. Изменить
Радиус траектории: 100
Начальные координаты: 150 150

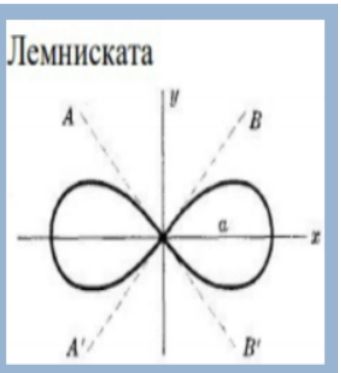
Запуск



Рисование фигуры

Цвет траектории: Black
Цвет объекта (контур): Black
Заливка объекта: Blue
Толщина траектории: 3
Толщина объекта: 3
Вид линии траектории: Dot
Вид линии объекта: DashDot
Пульсация объекта: 10
30
Скорость объекта: 10
Число прохождений: 100
Шаг движения: 1
Движение против часовой стрелки. Изменить
Радиус траектории: 100
Начальные координаты: 150 150

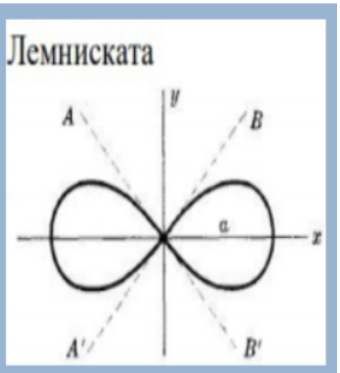
Запуск



Рисование фигуры

Цвет траектории: Black
Цвет объекта (контур): Black
Заливка объекта: Blue
Толщина траектории: 3
Толщина объекта: 3
Вид линии траектории: Dot
Вид линии объекта: DashDot
Пульсация объекта: 10
100
Скорость объекта: 99
Число прохождений: 3
Шаг движения: 1
Движение по часовой стрелке. Изменить
Радиус траектории: 100
Начальные координаты: 150 150

Запуск



Выводы

В ходе выполнения данной практической работы мы освоили навыки графического интерфейса в Windows Forms C#, а именно научились реализовывать рисование по точкам заданной траектории, и движение геометрической фигуры по этой траектории.

Индивидуальное задание 2

Цель работы

Научиться графически отображать фрактал заданного дерева на Windows Forms, C#.

Формулировка задания

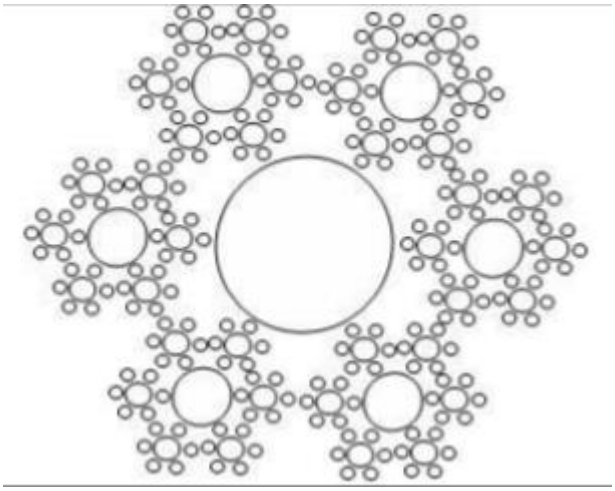
Формирование графического изображения на основе математического описания фрактала используя дерево, у которого для каждого узла число потомков согласуется с правилом выполнения очередного шага при построении фрактала.

Каждый уровень дерева на изображении должен иметь свой цвет и на фрактале для его элементов, появляющихся на определенном шаге этого уровня дерева, должен использоваться тот же цвет. Дерево и фрактал должны быть разноцветными.

Создать программу прорисовки фрактала «Круговая снежинка».

Математическая постановка

Необходимо графически реализовать фрактал «Круговая снежинка», она формируется размещением 6 окружностей (на основе правильного шестиугольника) снаружи базовой окружности:



Для этого был использован рекурсивный метод заполнения потомков дерева, каждый элемент которого имеет по 6 потомков.

Реализация построения фрактала на основе дерева:

Дерево реализуется следующим образом:

Дерево состоит из 4х уровней, вызывается метод fill_tree, в котором запускаются 6 конструкторов Node для формирования у данного родителя 6 дочерних узлов. Координаты этих узлов вычисляются с помощью переменных up, down и shift, которые отвечают за отклонение от центра родителя узла на shift значений по X, и на up & down значений по Y. Результат вычислений записывается в узел в переменную mid, принимающей значение X, Y.

Таким образом, методом рекурсии до 4 уровня все узлы дерева заполняются координатами и необходимыми для построения фрактала данными.

Данные для фрактала заполняются следующим образом:

В конструкторе каждого дочернего узла высчитывается его угол в правильном шестиугольнике родительского узла, полученные координаты записываются в переменную coordinates. Затем, во время построения фрактала графически, с помощью обхода в ширину, итератор проходит по

всем веткам дерева и прорисовывает каждый круг снежинки, исходя из переменных R и coordinates.

Используемые элементы интерфейса

В интерфейсы программы были использованы такие элементы, как:

7) PictureBox – изменялось свойство, отвечающее за содержащуюся внутри картинку, и её расположение внутри элемента.
8) Button – используется для запуска программы

Графические примитивы

В программе были использованы следующие графические примитивы:

6) Pen – использовалось для изменения цвета дерева на выводе, а также для изменения цвета фрактала
7) DrawLine – использовалось для прорисовки линий между точками дерева
8) DrawEllipse – использовалось для прорисовки элементов дерева и фрактала снежинки

Текст программы

Form1.cs:

```
public partial class Form1 : Form
    {
        static double angle = -Math.PI * 0.5;
        public Form1()
        {
            InitializeComponent();
        }
        public class Node
        {
            public Point mid;
            public PointF coordinates;
            public int r, R;
            public Color color;
            public int step;
            public Node child1;
            public Node child2;
            public Node child3;
            public Node child4;
            public Node child5;
            public Node child6;
            public Node(int Radius)
            {
                mid = new Point(250, 300);
                coordinates = new PointF(73, 100);
                r = 10;
                R = Radius / 4;
                color = Color.Navy;
```

```
                child1 = null;
                child2 = null;
                child3 = null;
                child4 = null;
                child5 = null;
                child6 = null;
            }
            public Node(int Radius, PointF midder, int i)
            {
                mid = new Point(250, 300);
                coordinates.X = (float)(midder.X + Math.Round(Math.Cos(angle + Math.PI *
2.0 * i / 6) * Radius));
                coordinates.Y = (float)(midder.Y + Math.Round(Math.Sin(angle + Math.PI *
2.0 * i / 6) * Radius));
                r = 10;
                R = Radius / 4;
                color = Color.Navy;
                child1 = null;
                child2 = null;
                child3 = null;
                child4 = null;
                child5 = null;
                child6 = null;
            }
        }
        public class bfs_Iterator
        {
            public bfs_Iterator(Node start)
            {
                queue = new Queue<Node>();
                queue.Enqueue(start);
            }
            public bool has_next()
            {
                if (queue.Count == 0) return false;
                else return true;
            }
            public Node next()
            {
                Node cur;
                cur = queue.Dequeue();
                if (cur.child1 != null)
                {
                    queue.Enqueue(cur.child1);
                }
                if (cur.child2 != null)
                {
                    queue.Enqueue(cur.child2);
                }
                if (cur.child3 != null)
                {
                    queue.Enqueue(cur.child3);
                }
                if (cur.child4 != null)
                {
                    queue.Enqueue(cur.child4);
                }
                if (cur.child5 != null)
                {
                    queue.Enqueue(cur.child5);
                }
                if (cur.child6 != null)
                {
                    queue.Enqueue(cur.child6);
                }
```

```
                return cur;
            }
            private Queue<Node> queue;
        }
        public class Tree
        {
            public Node root;
            private void fill_tree(Node cur, Size up, Size down, int i)
            {
                if (i > 0)
                {
                    Size shift = new Size((int)Math.Pow(2, i - 1) * 10, 0);
                    cur.child1 = new Node(cur.R, cur.coordinates, 0);
                    cur.child2 = new Node(cur.R, cur.coordinates, 1);
                    cur.child3 = new Node(cur.R, cur.coordinates, 2);
                    cur.child4 = new Node(cur.R, cur.coordinates, 3);
                    cur.child5 = new Node(cur.R, cur.coordinates, 4);
                    cur.child6 = new Node(cur.R, cur.coordinates, 5);
                    cur.child1.r = cur.child2.r = cur.child3.r = cur.child4.r =
cur.child5.r = cur.child6.r = cur.r - 1;
                    cur.child1.mid = cur.mid + up - shift - shift - shift;
                    cur.child2.mid = cur.mid + up - shift - shift;
                    cur.child3.mid = cur.mid + up - shift;
                    cur.child4.mid = cur.mid + down + shift;
                    cur.child5.mid = cur.mid + down + shift + shift;
                    cur.child6.mid = cur.mid + down + shift + shift + shift;
                    fill_tree(cur.child1, up, down, i - 1);
                    fill_tree(cur.child2, up, down, i - 1);
                    fill_tree(cur.child3, up, down, i - 1);
                    fill_tree(cur.child4, up, down, i - 1);
                    fill_tree(cur.child5, up, down, i - 1);
                    fill_tree(cur.child6, up, down, i - 1);
                }
            }
            public bfs_Iterator create_bfs_iterator()
            {
                return new bfs_Iterator(root);
            }
            public Tree()
            {
                Size up = new Size(0, 40);
                Size down = new Size(0, 40);
                root = new Node(256);
                fill_tree(root, up, down, 3);
            }
        }
        public Tree tree;
        void FractalColors(int n, Node cur)
        {
            if ((n <= 4) && (cur != null))
            {
                cur.step = n;
                switch (n)
                {
                    case 1:
                        cur.color = Color.Pink;
                        break;
                    case 2:
                        cur.color = Color.Silver;
                        break;
                    case 3:
                        cur.color = Color.Gold;
                        break;
                    case 4:
                        cur.color = Color.Blue;
```

```
                    break;
                }
                FractalColors(n + 1, cur.child1);
                FractalColors(n + 1, cur.child2);
                FractalColors(n + 1, cur.child3);
                FractalColors(n + 1, cur.child4);
                FractalColors(n + 1, cur.child5);
                FractalColors(n + 1, cur.child6);
            }
        }
        private void DrawTree()
        {
            bfs_Iterator iterator = tree.create_bfs_iterator();
            Node cur;
            Pen pen = new Pen(Color.Black, 2);
            Graphics graphics = pictureBox1.CreateGraphics();
            while (iterator.has_next())
            {
                cur = iterator.next();
                pen.Color = cur.color;
                graphics.DrawEllipse(pen, cur.mid.X - cur.r / 2, cur.mid.Y - cur.r / 2,
cur.r, cur.r);
                if ((cur.child1 != null) && (cur.child2 != null) && (cur.child3 != null)
&& (cur.child4 != null))
                {
                    graphics.DrawLine(pen, cur.mid, cur.child1.mid);
                    graphics.DrawLine(pen, cur.mid, cur.child2.mid);
                    graphics.DrawLine(pen, cur.mid, cur.child3.mid);
                    graphics.DrawLine(pen, cur.mid, cur.child4.mid);
                    graphics.DrawLine(pen, cur.mid, cur.child5.mid);
                    graphics.DrawLine(pen, cur.mid, cur.child6.mid);
                }
            }
        }

        private void Form1_Load_1(object sender, EventArgs e)
        {
            button1.Text = "Start";
            tree = new Tree();
            int n = 1;
            FractalColors(n, tree.root);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            DrawTree();
            bfs_Iterator iterator = tree.create_bfs_iterator();
            Node cur;
            Graphics graphics = pictureBox1.CreateGraphics();
            cur = iterator.next();
            for (int jj = 2; jj< 6; jj++)
            {
                while (iterator.has_next() && cur.step < jj)
                {
                    Paint_Circle(cur, jj);
                    cur = iterator.next();
                }
                iterator = tree.create_bfs_iterator();
                while (iterator.has_next())
                {
                    cur = iterator.next();
                    cur.coordinates.X = cur.coordinates.X + 170;
                }
                iterator = tree.create_bfs_iterator();
                cur = iterator.next();
```

```
            }
        }
        public void Paint_Circle(Node cur, int stepper)
        {
            if (cur.step < stepper)
            {
                Pen pen = new Pen(Color.Black, 2);
                Graphics graphics = pictureBox1.CreateGraphics();
                pen.Color = cur.color;
                graphics.DrawEllipse(pen, cur.coordinates.X - cur.R / 2,
cur.coordinates.Y - cur.R / 2, cur.R, cur.R);
            }
        }
    }
```

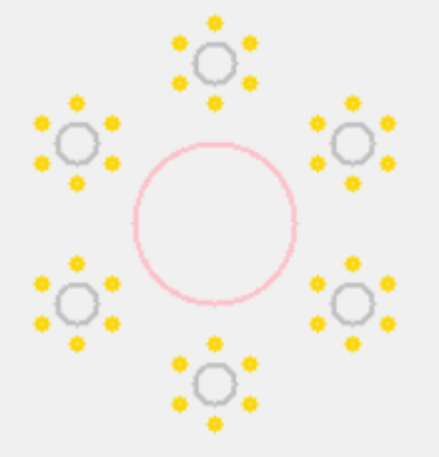
Примеры работы

В результате работы программы на экране прорисовываются все этапы построения фрактала в ряд, а также само дерево:

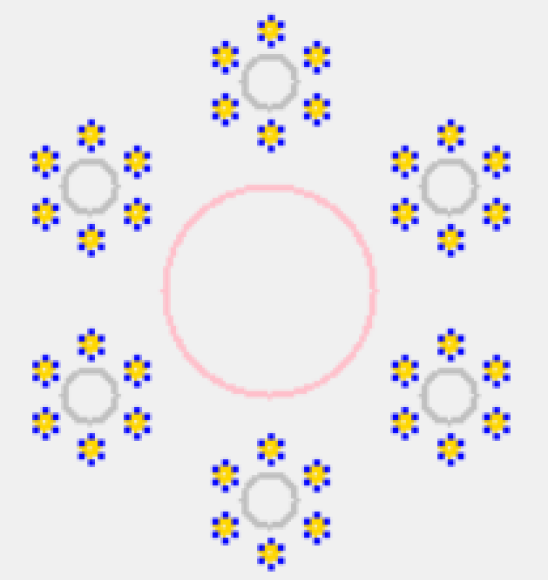
1 этап:

2 этап:

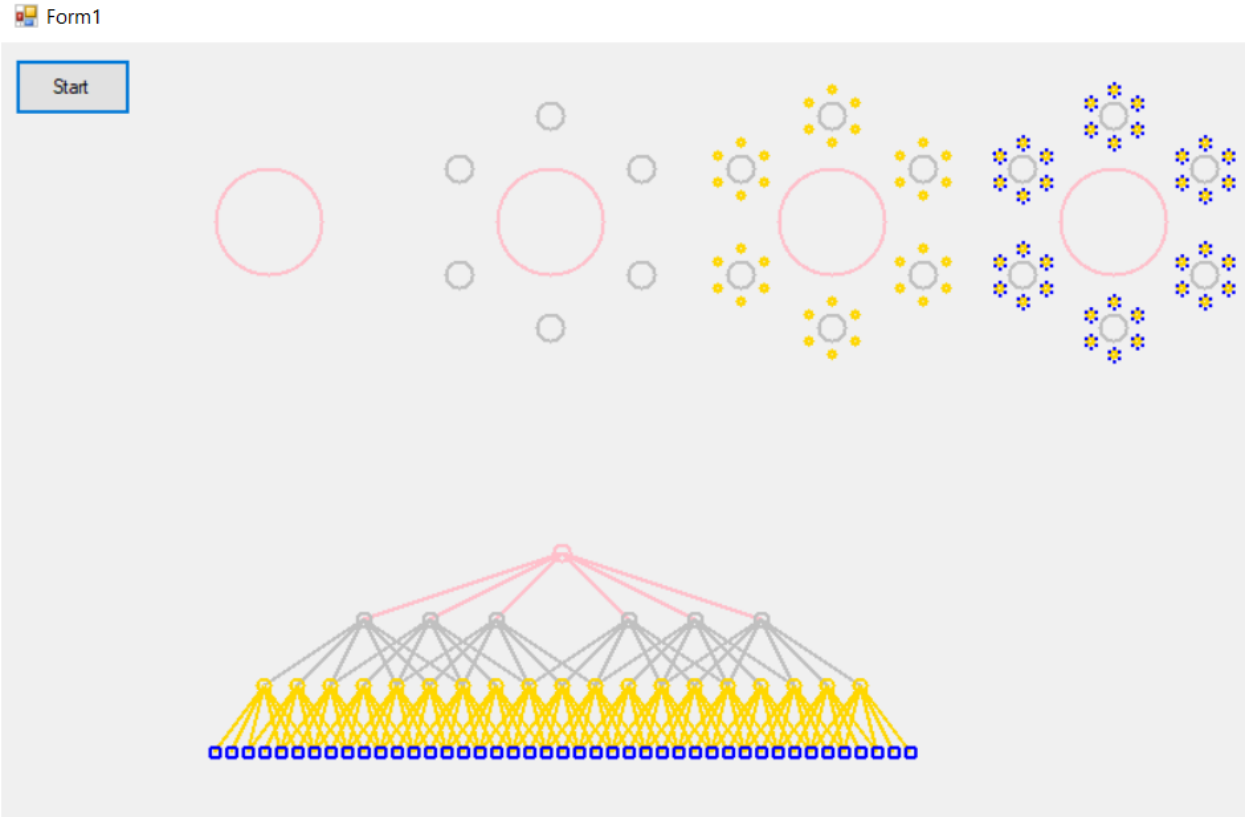
3 этап:



4 этап:



Результат работы программы:



Выводы

В ходе выполнения данной практической работы были получены навыки по реализации графического представления фрактала «Круговая снежинка» на основе дерева с потомками.

Ссылки на источники:

https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D1%83%D0%B3%D0%BE%D0%B2%D0%BE%D0%B9_%D1%84%D1%80%D0%B0%D0%BA%D1%82%D0%B0%D0%BB

https://metanit.com/sharp/windowsforms/1.1.php