**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

отчет

**по** учебной практике

(технологической (проектно-технологической) практике)

|  |  |  |
| --- | --- | --- |
| Студент гр. 3354 |  | Ершов Е.В. |
| Руководитель |  | Калмычков В.А. |

Санкт-Петербург

2025

**ЗАДАНИЕ**

**на Учебную Практику (Технологическую (Проектно-технологическую) Практику)**

|  |  |  |
| --- | --- | --- |
| Студент Ершов Е.В. | | |
| Группа 3354 | | |
| Тема практики: учебная (технологическая (проектно-технологическая)) практика | | |
| Задание на практику:   1. Подготовка приложения или отдельных приложений на основе ознакомительного материала. Выполнить последовательность действий по использованию элементов оформления пользовательского интерфейса. 2. Первое индивидуальное задание. Реализовать приложение, которое обеспечивает движение заданного объекта по заданной траектории. 3. Второе индивидуальное задание. Создание приложения, в котором реализуется графическое представление фрактала, где каждому уровню построения соответствует уровень дерева. | | |
| Сроки прохождения практики: 01.07.2025 – Х.07.2025 | | |
| Дата сдачи отчета: | | |
| Дата защиты отчета: | | |
|  | | |
| Студентка |  | Ершов Е.В. |
| Руководитель |  | Калмычков В.А. |

**Аннотация**

Во время выполнения практики требуется выполнить три задания. Первое задание является общим и включает изучение теоретического материала, а также выполнение задач для закрепления полученных знаний. Второе задание выполняется индивидуально и предполагает разработку приложения, которое моделирует движение объекта по заданной траектории. Третье задание также выполняется индивидуально и требует разработки приложения, которое отображает фрактал и связанное с ним дерево. Все задания должны быть реализованы с использованием пользовательского графического интерфейса.

**Summary**

During the internship, you are required to complete three assignments. The first assignment is a general one, which involves studying theoretical material and completing tasks to reinforce the acquired knowledge. The second assignment is an individual one, where you need to develop an application that simulates the movement of an object along a given trajectory. The third assignment is also individual and requires you to develop an application that displays a fractal and its associated tree. All assignments must be implemented using a graphical user interface.

**содержание**

Оглавление

[ВВЕДЕНИЕ 5](#_Toc202306577)

[ЦЕЛЬ РАБОТЫ 5](#_Toc202306578)

[1. ОЗНАКОМИТЕЛЬНОЕ ПРАКТИЧЕСКОЕ ЗАДАНИЕ 6](#_Toc202306579)

[1.1. Элементы button, textBox и label 6](#_Toc202306580)

[1.2. Элемент MessageBox, Подсказка ToolTip 6](#_Toc202306581)

[1.3. Изменение шрифта текста и цвета формы и элементов 7](#_Toc202306582)

[1.4. Элемент MenuStrip и свойство Anchor, Открытие и запись текстового файла 9](#_Toc202306583)

[1.5. Рисование линий, треугольника, эллипса и окружности в PictureBox 10](#_Toc202306584)

[1.6. Событие MouseHover 13](#_Toc202306585)

[1.7. Формирование траектории для движения простого геометрического объекта 13](#_Toc202306586)

[2. ДВИЖЕНИЕ ГЕОМЕТРИЧЕСКОГО ОБЪЕКТА ПО ТРАЕКТОРИИ 15](#_Toc202306587)

[3. ФОРМИРОВАНИЕ ГРАФИЧЕСКОГО ИЗОБРАЖЕНИЯ НА ОСНОВЕ МАТЕМАТИЧЕСКОГО ОПИСАНИЯ ФРАКТАЛА 15](#_Toc202306588)

ВВЕДЕНИЕ

Задачами учебной практики являются: формирование базовых практических понятий, лежащих в основе процесса разработки программного графического интерфейса пользователя, получение навыков применения средств визуализации при решении практических задач и использовании различных структур данных, освоение способов реализации программ на выбранном рабочем языке программирования C# и С++ с учётом особенностей реализации в конкретной системе программирования.

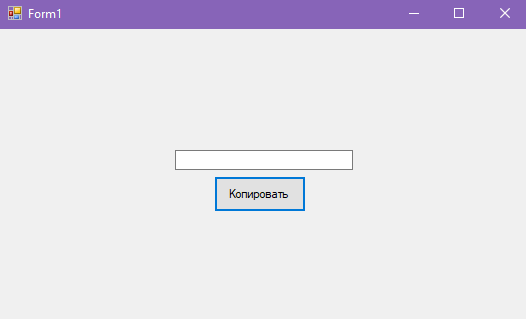
ЦЕЛЬ РАБОТЫ

Изучить и освоить базовые понятия и приемы использования инструментальных средств и технологий программирования при решении практических задач с выбором различных структур данных и организацией программного графического интерфейса пользователя, а также закрепить и приобрести новые знания и практические навыки программирования.

1. ОЗНАКОМИТЕЛЬНОЕ ПРАКТИЧЕСКОЕ ЗАДАНИЕ

1.1. Элементы button, textBox и label

В программе есть три элемента: кнопка (button), поле для ввода (textBox) и метка (label). Когда пользователь вводит текст и нажимает на кнопку, программа копирует этот текст в другое окошко. Пример работы показан на рисунках 1.1 и 1.2.

 Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.1 Рис. 1.2

На рис. 1.1 представлен пример «до», а на рис. 1.2 – «после».

1.2. Элемент MessageBox, Подсказка ToolTip

На рисунках 1.3-1.6 показано, как работает программа: когда нажимаешь на кнопку, появляется всплывающее окошко. В этом окошке отображается текст, который был введён в поле, а также даётся подсказка, какой именно текст нужно вводить.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.3

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, программное обеспечение, диаграмма

Содержимое, созданное искусственным интеллектом, может быть неверным.Рис. 1.4 Рис. 1.5

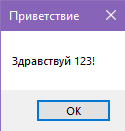


Рис. 1.6

На рис. 1.3 показано начальное состояние программы при запуске. Когда пользователь наводит курсор на текстовое поле, появляется всплывающая подсказка ToolTip (рис. 1.4). После ввода текста в поле (рис. 1.5) программа отображает окно MessageBox, которое использует введённый пользователем текст (рис. 1.6).

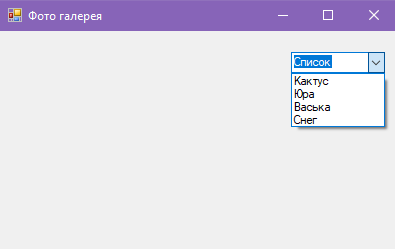
1.3. Изменение шрифта текста и цвета формы и элементов

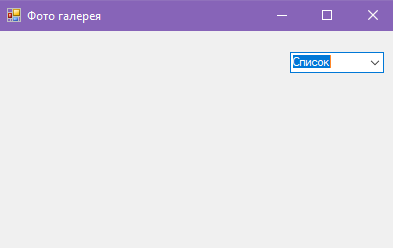
На рисунках 1.7-1.8 показана работа программы с графическим интерфейсом. В приложении реализована возможность наложения текста на фоновое изображение формы. На рис. 1.7 отображено начальное состояние программы при запуске, где уже установлено фоновое изображение (BackGroundImage) и изменен цвет кнопки. После того как пользователь вводит текст "Привет!" и подтверждает действие, программа отображает этот текст поверх фонового изображения, что демонстрируется на рис. 1.8.Изображение выглядит как текст, снимок экрана, Операционная система

Содержимое, созданное искусственным интеллектом, может быть неверным. Изображение выглядит как текст, снимок экрана, Операционная система, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.7 Рис. 1.8

 На рисунках 1.9-1.11 представлен пример работы программы с выпадающим списком comboBox.

 Рис. 1.9 Рис. 1.10

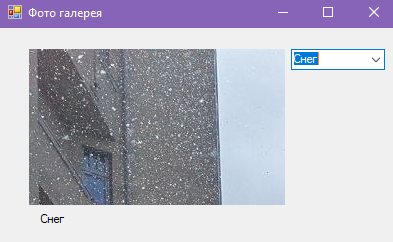


Рис. 1.11

На рис. 1.9 изображен первоначальный вид программы после запуска. При нажатии на кнопку "Albums" открывается выпадающий список с перечнем доступных альбмов (рис. 1.10). После выбора определенного альбма из списка, его обложка автоматически отображается в поле PictureBox, как продемонстрировано на рис. 1.11.

1.4. Элемент MenuStrip и свойство Anchor, Открытие и запись текстового файла

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным. На рисунках 1.13-1.14 показана работа программы с элементом MenuStrip. Этот элемент интерфейса представляет собой стандартное выпадающее меню, аналогичное тем, что встречаются в различных приложениях - например, в текстовых редакторах при выборе "Файл->Сохранить как" или в графических редакторах при использовании "Файл->Вставить". В данном примере меню содержит раздел "Файл" с двумя пунктами: "Открыть" и "Сохранить как", которые появляются при клике на соответствующий раздел меню.

Изображение выглядит как текст, линия, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.13 Рис. 1.14

На рис. 1.13 показан запуск программы, то есть исходное окно.

Далее представлен пример программы, которая работает с текстовым файлом, из которого может происходить чтение данных, а также в него могут быть записаны новые данные.

Изображение выглядит как текст, снимок экрана, линия, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт, белый

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.15 Рис. 1.16

На рисунках 1.15-1.16 показано открытие файла «1.txt».

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.17 Рис. 1.18

На рисунках 1.17-1.18 представлен пример работы программы, когда мы изменяем текст открытого файла и сохраняем его.

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, Шрифт, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.19 Рис. 1.20

Если ввести новый текст и выбрать пункт «Выход», но не выбрать перед этим «Сохранить как», то появится предупреждающее пользователя окно о потери новых несохраненных данных, что показано на рисунках 1.19-1.20.

1.5. Рисование линий, треугольника, эллипса и окружности в PictureBox

На рис. 1.21 представлен пример работы программы, которая рисует линию заданной длины, цвета и в заданном месте поля "PictureBox".

Изображение выглядит как текст, снимок экрана, диаграмма, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.21

В каждый textBox мы записываем начальное и конечное значение координат поля рисования: две координаты – одна точка, ещё две координаты – ещё одна точка, которая соединяется с предыдущей, образуя отрезок, заданного в коде программы цвета.

На рисунке 1.22 представлен пример работы программы, рисующей треугольник по трем заданным точкам. Координаты точек вводятся пользователем.

Изображение выглядит как снимок экрана, текст, диаграмма, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.22

Каждая точка является началом и концом двух отрезков, из-за чего нам достаточно трех точек для того, чтобы нарисовать треугольник. Каждый отрезок выделен своим цветом.

На рисунках 1.23-1.24 представлен пример работы программы, которая рисует эллипс.

Изображение выглядит как текст, снимок экрана, диаграмма, круг

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным. Рис. 1.23 Рис. 1.24

Чтобы нарисовать эллипс нужно указать координаты верхнего левого угла условного прямоугольника, в котором находится рисуемая фигура, и радиусы. Если горизонтальный и вертикальный радиусы равны, то получится окружность (рис. 1.24), а если нет, то получится эллипс (рис. 1.23).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.25

На рисунке 1.25 представлена программа, в которой с помощью comboBox можно выбирать фигуру, которая будет нарисована и залита одним цветом. В данном случае – синим.

1.6. Событие MouseHover

В данной программе при наведении курсора мыши на надпись label изменяется текст и его цвет. На рисунках 1.26-1.27 надпись «Не трогай.» заменяется на «ERROR!!!» и становится из черного красным цветом.

Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Содержимое, созданное искусственным интеллектом, может быть неверным.Изображение выглядит как снимок экрана, текст, дизайн

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.26 Рис. 1.27

Изображение выглядит как текст, снимок экрана, Шрифт, логотип

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. .28

Также выскакивает окно с сообщением об ошибке «Написано же НЕтрогать!», что показано на рис. 1.28.

1.7. Формирование траектории для движения простого геометрического объекта

На рисунках 1.29 и 1.30 демонстрируется работа программы, визуализирующей траекторию гипоциклоиды в зависимости от заданных параметров: радиуса большой окружности и количества областей.

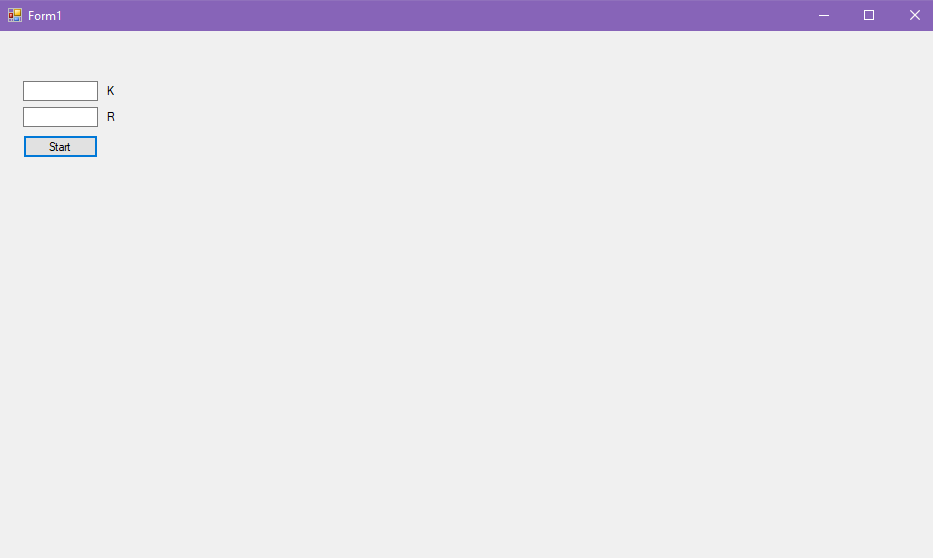


Рис. 1.29 – исходный вид при запуске программы.

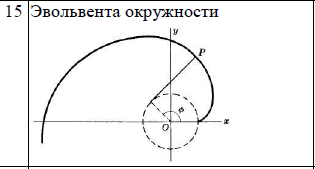
Изображение выглядит как диаграмма, текст, снимок экрана, круг

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 1.30 – пример траектории гипоциклоиды с радиусом большой окружности 200 и количеством областей 10.

2. ДВИЖЕНИЕ ГЕОМЕТРИЧЕСКОГО ОБЪЕКТА ПО ТРАЕКТОРИИ

# **Задание****:** Движение геометрического объекта параллелограмма по траектории контур черного цвета заливка заданного цвета

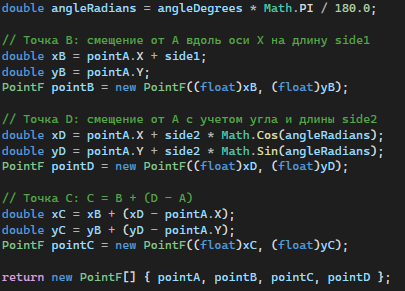


# **Математический метод решения**

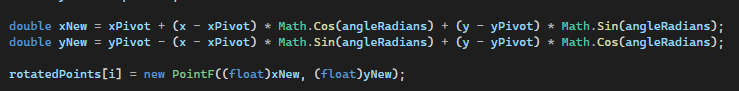
Для изображения траектории мы возьмём обще извечные формулы Эвольвента окружности:



Остальная часть решения является тривиальной с точки зрения математического решения.



Итоговые точки параллелограмма вычисляются умножение на синус и косинус углов.



# **Описание используемых элементов для интерфейса и их настройку**

В данной программе с помощью графических примитивов можно настроит: радиус начальной окружности вокруг которой строиться Эвольвента(любое число, подходящее под условия), можно скрыть или показать эту окружность, можно выбрать способ создания параллелограмма(во втором случае у нас будет ромб), для каждого случая можно указать параметры(любое число, подходящее под условия), также можно добавить или убрать пульсацию, настроить цвет через численные поля в соответствии с палитрой RGB, выбрать из списка вид линии которой рисуется траектория, указать скорость поворота( при 0 поворота нету).

**Описание используемых графических примитиво**

В данной программе используются:

textbox- примитив для ручного указания параметров.

Checkbox- примитив для выбора режимов либо есть, либо нету

ComboBox- примитив для выбора из уже предустановленных вариантов

PictureBox-примитив для изображения фигур и графиков для этого и используем

Button-притив для запуска процесса

**Программный код**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WindowsFormsApp3

{

public partial class Form1 : Form

{

public double InitT = 0, LastT = 6.3 \* 1.2;

public double Step = 0.1;

public int cX = 300, cY = 300;

public int R\_base = 20;

public List<PointF> p = new List<PointF>();

public float x, y;

//Данные пар:

public int par1 =50, par2=100, par3=50;

public float angel\_rotate =0;

public bool Puls =false;

public bool kons = true;

public bool Cir = false;

public bool TrDash = true;

public int red = 255, green = 128, blue = 0;

private PointF[] Parallelogram\_By\_2Sides\_And\_Angle(PointF pointA, float side1, float side2, float angleDegrees)

{

double angleRadians = angleDegrees \* Math.PI / 180.0;

// Точка B: смещение от A вдоль оси X на длину side1

double xB = pointA.X + side1;

double yB = pointA.Y;

PointF pointB = new PointF((float)xB, (float)yB);

// Точка D: смещение от A с учетом угла и длины side2

double xD = pointA.X + side2 \* Math.Cos(angleRadians);

double yD = pointA.Y + side2 \* Math.Sin(angleRadians);

PointF pointD = new PointF((float)xD, (float)yD);

// Точка C: C = B + (D - A)

double xC = xB + (xD - pointA.X);

double yC = yB + (yD - pointA.Y);

PointF pointC = new PointF((float)xC, (float)yC);

return new PointF[] { pointA, pointB, pointC, pointD };

}

PointF[] Parallelogram\_By\_Side\_And\_2Angles(PointF pointA, float side1, float angle1Degrees, float angle2Degrees)

{

// Проверка на корректность входных данных

if (side1 <= 0)

throw new ArgumentException("Длина стороны должна быть положительной.");

if (Math.Abs(angle2Degrees % 180) < 0.001 || Math.Abs(angle2Degrees % 180 - 180) < 0.001)

throw new ArgumentException("Угол между сторонами не должен быть 0 или 180 градусов, чтобы избежать вырождения.");

double angle1Radians = angle1Degrees \* Math.PI / 180.0;

double angle2Radians = angle2Degrees \* Math.PI / 180.0;

// Точка B: смещение от A по углу angle1 и длине side1

double xB = pointA.X + side1 \* Math.Cos(angle1Radians);

double yB = pointA.Y + side1 \* Math.Sin(angle1Radians);

PointF pointB = new PointF((float)xB, (float)yB);

// Точка D: смещение от A по углу (angle1 + angle2) и длине side1

double angleDRadians = angle1Radians + angle2Radians; // Угол для AD

double xD = pointA.X + side1 \* Math.Cos(angleDRadians);

double yD = pointA.Y + side1 \* Math.Sin(angleDRadians);

PointF pointD = new PointF((float)xD, (float)yD);

// Точка C: C = B + (D - A)

double xC = xB + (xD - pointA.X);

double yC = yB + (yD - pointA.Y);

PointF pointC = new PointF((float)xC, (float)yC);

return new PointF[] { pointA, pointB, pointC, pointD };

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

Cir=(Cir ? false :true);

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedItem != null)

{

string selectedText = comboBox1.SelectedItem.ToString();

if (selectedText== "2 стороны и угол")

{

kons = true;

label3.Text = "1 сторонa";

label4.Text = "2 сторонa";

label5.Text = "угол";

}

else

{

kons = false;

label3.Text = "сторона";

label4.Text = "1 угол";

label5.Text = "2 угол";

}

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox2.Text.All(char.IsDigit))

{

int.TryParse(this.textBox2.Text, out par1);

}

else

{

par1 = 10;

}

if (par1 < 2)

{

par1 = 2;

}

else if (par1 > 250)

{

par1 = 250;

}

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

if (textBox3.Text.All(char.IsDigit))

{

int.TryParse(this.textBox3.Text, out par2);

}

else

{

par2 = 10;

}

if (par2 < 0)

{

par2 = 2;

}

else if (par2 > 250)

{

par2 = 250;

}

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

if (textBox4.Text.All(char.IsDigit))

{

int.TryParse(this.textBox4.Text, out par3);

}

else

{

par3 = 10;

}

if (par3 < 0)

{

par3 = 2;

}

else if (par3 > 250)

{

par3 = 250;

}

}

private void checkBox2\_CheckedChanged(object sender, EventArgs e)

{

Puls = (Puls ? false : true);

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

if (textBox5.Text.All(char.IsDigit))

{

int.TryParse(this.textBox5.Text, out red);

}

else

{

red = 255;

}

if (red < 0)

{

red = 0;

}

else if (red > 255)

{

red = 255;

}

}

private void textBox6\_TextChanged(object sender, EventArgs e)

{

if (textBox6.Text.All(char.IsDigit))

{

int.TryParse(this.textBox6.Text, out green);

}

else

{

green = 255;

}

if (green < 0)

{

green = 0;

}

else if (green > 255)

{

green = 255;

}

}

private void textBox7\_TextChanged(object sender, EventArgs e)

{

if (textBox6.Text.All(char.IsDigit))

{

int.TryParse(this.textBox7.Text, out blue);

}

else

{

blue = 255;

}

if (blue < 0)

{

blue = 0;

}

else if (blue > 255)

{

blue = 255;

}

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox2.SelectedItem != null)

{

string selectedText = comboBox2.SelectedItem.ToString();

TrDash = (selectedText == "сплошная" ? true : false);

}

}

private void textBox8\_TextChanged(object sender, EventArgs e)

{

if (textBox8.Text.All(char.IsDigit))

{

float.TryParse(this.textBox8.Text, out angel\_rotate);

}

else

{

par1 = 0;

}

if (par1 < 0)

{

par1 = 0;

}

else if (par1 > 180)

{

par1 = 180;

}

}

private PointF[] RotateParallelogram(PointF[] parallelogram, PointF pivot, float angleDegrees)

{

double angleRadians = angleDegrees \* Math.PI / 180.0;

PointF[] rotatedPoints = new PointF[parallelogram.Length];

rotatedPoints[0] = pivot;

for (int i = 1; i < parallelogram.Length; i++)

{

double x = parallelogram[i].X;

double y = parallelogram[i].Y;

double xPivot = pivot.X;

double yPivot = pivot.Y;

double xNew = xPivot + (x - xPivot) \* Math.Cos(angleRadians) + (y - yPivot) \* Math.Sin(angleRadians);

double yNew = yPivot - (x - xPivot) \* Math.Sin(angleRadians) + (y - yPivot) \* Math.Cos(angleRadians);

rotatedPoints[i] = new PointF((float)xNew, (float)yNew);

}

return rotatedPoints;

}

private void Paint\_Graphic(List<PointF> p, int cX = 250, int cY = 250)

{

p.Clear();

Graphics Graf = pictureBox1.CreateGraphics();

double angle = InitT;

Graf.Clear(BackColor);

if (Cir==true) {

Graf.DrawEllipse(Pens.Red, cX - (int)R\_base, cY - (int)R\_base, (int)R\_base \* 2, (int)R\_base \* 2);

}

while (angle <= LastT)

{

x = (float)(R\_base \* (Math.Cos(angle) + angle \* Math.Sin(angle)));

y = (float)(R\_base \* (Math.Sin(angle) - angle \* Math.Cos(angle)));

p.Add(new PointF(cX + (int)x, cY + (int)y));

angle += Step;

}

PointF center = new PointF(250, 250);

PointF[] p\_arr = p.ToArray();

if (p\_arr.Length > 2)

{

Graf.DrawLines(Pens.Black, p\_arr); // траектория

using (Pen pen = new Pen(Color.Black, 2))

{

pen.DashStyle = (TrDash? DashStyle.Solid : DashStyle.Dash);

Graf.DrawLines(pen, p\_arr);

}

//using (Pen pen = new Pen(Color.Red, 2)) // Жирность 3 пикселя

//{

// pen.DashStyle = DashStyle.Dash; // Пунктирный стил

// foreach (PointF sp in p\_arr)

// {

// PointF[] line = new PointF[2];

// line[0] = center;

// line[1] = sp;

// Graf.DrawLines(pen, line);

// }

//}

}

}

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

Paint\_Graphic(p, cX, cY);

float start\_rot = 0;

Graphics Graf = pictureBox1.CreateGraphics();

PointF[] p\_arr = p.ToArray();

float koef = 1;

float step = 0;

foreach (var item in p\_arr)

{

if (Puls)

{

if (koef > 2)

{

step = (float)-0.1;

} else if (koef == 1)

{

step = (float)0.1;

}

}

Paint\_Graphic(p, cX, cY);

PointF[] par;

if (kons)

{

par = Parallelogram\_By\_2Sides\_And\_Angle(item, par1 \* koef, par2 \* koef, par3);

}

else

{

par = Parallelogram\_By\_Side\_And\_2Angles(item, par1 \* koef, par2, par3);

}

par = RotateParallelogram(par, item, start\_rot);

start\_rot += angel\_rotate;

using (SolidBrush brush = new SolidBrush(Color.FromArgb(red, green, blue)))

{

Graf.FillPolygon(brush, par); // Заливка

}

using (Pen pen = new Pen(Color.Black, 2)) // Жирность 3

{

pen.DashStyle = DashStyle.Solid; // Пунктир

Graf.DrawPolygon(pen, par);

}

koef += step;

Thread.Sleep(40);

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (textBox1.Text.All(char.IsDigit))

{

int.TryParse(this.textBox1.Text, out R\_base);

}

else

{

R\_base = 100;

}

if (R\_base < 2)

{

R\_base = 2;

}

else if (R\_base > 250)

{

R\_base = 250;

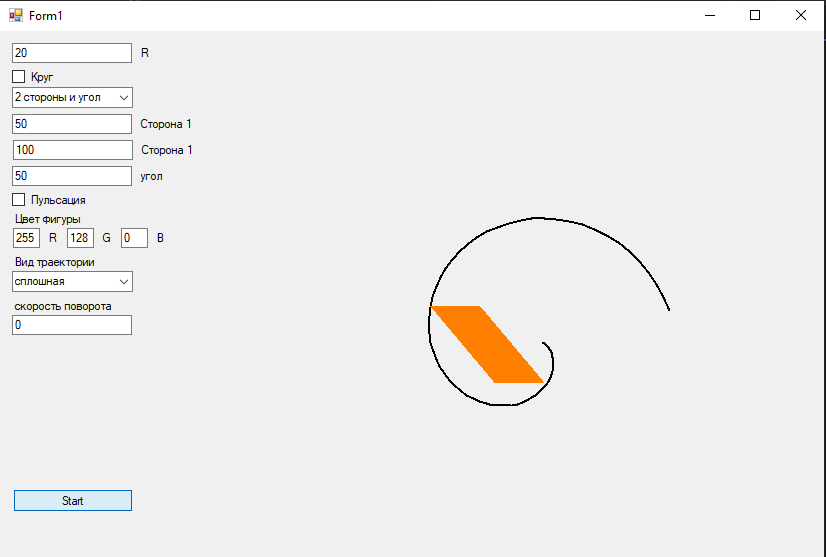
}

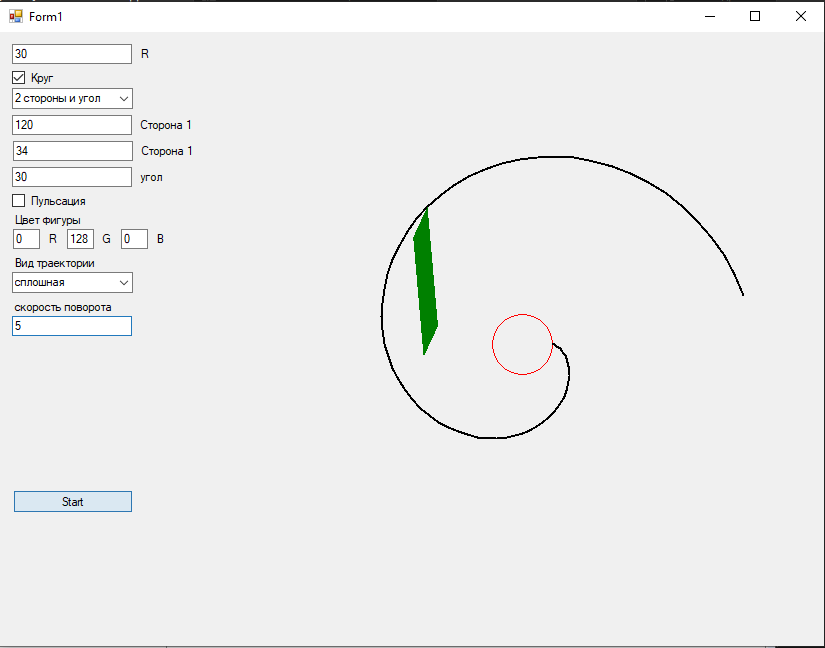
}

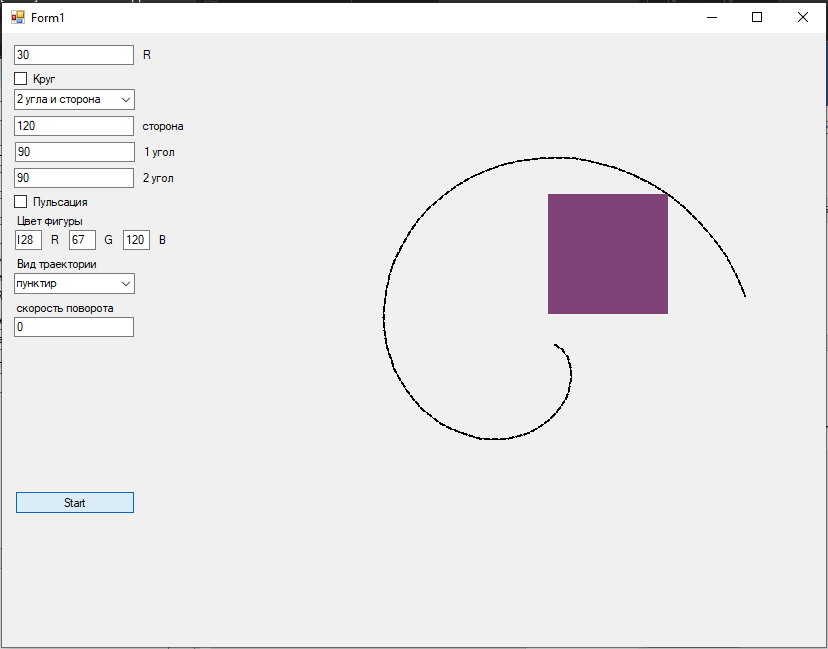
}

}

**Результаты работы программы:**







# **Вывод о проделанной работе**

В результате выполнения задания я научился работать с графическим интерфейсом. Освоил примитивы и варианты их использования для реализации простейшего пользовательского интерфейса.

3. ФОРМИРОВАНИЕ ГРАФИЧЕСКОГО ИЗОБРАЖЕНИЯ НА ОСНОВЕ МАТЕМАТИЧЕСКОГО ОПИСАНИЯ ФРАКТАЛА

**3.1. Формулировка задания**

Выбор на дереве и прорисовка определенного уровня n фракталов.

Заданный фрактал:

Вид фрактала: Кривая Госпера



Рис. 3.1

**3.2. Математическая постановка**

Расчетные формулы и пояснения: остается средняя треть и образуются еще 6 звеньев при вписывании в параллелограмм на основе равносторонних треугольников.

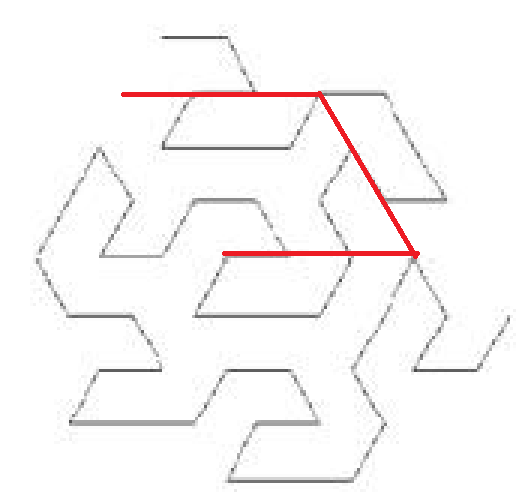
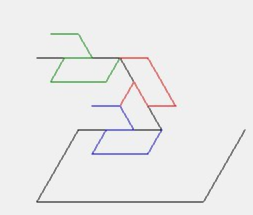
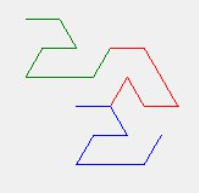
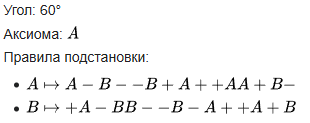


Рис. 3.2 Рис. 3.3 Рис. 3.4

Вывод на форму: к сожалению, данный способ даёт искажённый вид фрактал и не может использовать для реализации. Поэтому я для реализации я выбрал систему Линденмайера со следующими правилами:



Изначально имеется прямая потом при замене по правилам выше мы получаем закон для нахождения следующих точек

**3.3. Описание элементов интерфейса**

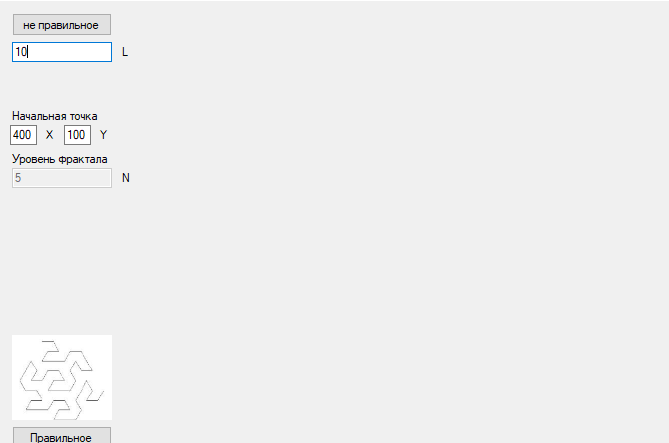


Рис. 3.5

Интерфейс представляет собой 2 кнопку, 4 textbox, 6 label и 3 picturebox.

При нажатии на конпку запуска программа просчитывает координаты дерева и рисует их на экран. В дальнейшем, нажимая на определенный уровень дерева в PictureBox, рисуется фрактал до выбранного уровня.

**3**.**4. Описание графических примитивов**

Работа программы основана на нажатии кнопки и указании уровня в textbox.

При выборе уровня и нажатии на кнопку «Построить дерево» строится максимальное дерево и каждый уровень отличается цветом. При нажатии на рисунок дерева на определенный уровень рисуется фрактал для выбранного уровня дерева. Через textbox можно задать начальные координаты и длину шага. На picturBox выводится сам фрактал и дерево, а также картинка для проверки корректности постановки.

**3.5. Текст программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WindowsFormsApp5

{

public partial class Form1 : Form

{

public float L = 10;

public float cX = 380, cY = 20;

public int levels = 5;

private List<(PointF center, float radius, Color color, int nodeId, int level)> nodes = new List<(PointF, float, Color, int, int)>();

private int nodeCounter = 0;

public PointF[] GetKochSegmentPoints(PointF pI, PointF pA)

{

double length = Math.Sqrt(Math.Pow(pI.X - pA.X, 2) + Math.Pow(pI.Y - pA.Y, 2));

double third = length / 3;

PointF pE = new PointF(

pA.X + (float)((pI.X - pA.X) / 3),

pA.Y + (float)((pI.Y - pA.Y) / 3)

);

PointF pD = new PointF(

pA.X + (float)(2 \* (pI.X - pA.X) / 3),

pA.Y + (float)(2 \* (pI.Y - pA.Y) / 3)

);

double baseAngle = Math.Atan2(pI.Y - pA.Y, pI.X - pA.X);

double segmentLength = third;

double angle1 = baseAngle - 2\* Math.PI / 3;

PointF pC = new PointF(

pD.X + (float)(segmentLength \* Math.Cos(angle1)),

pD.Y + (float)(segmentLength \* Math.Sin(angle1))

);

double angle2 = baseAngle + 2\*Math.PI / 3;

PointF pF = new PointF(

pE.X + (float)(segmentLength \* Math.Cos(angle2)),

pE.Y + (float)(segmentLength \* Math.Sin(angle2))

);

PointF pG = new PointF(

pF.X + (float)(segmentLength \* Math.Cos(baseAngle)),

pF.Y + (float)(segmentLength \* Math.Sin(baseAngle))

);

PointF pH = new PointF(

pG.X + (float)(segmentLength \* Math.Cos(baseAngle)),

pG.Y + (float)(segmentLength \* Math.Sin(baseAngle))

);

PointF pB = new PointF(

pC.X - (float)(segmentLength \* Math.Cos(baseAngle)),

pC.Y - (float)(segmentLength \* Math.Sin(baseAngle))

);

Graphics Graf = pictureBox1.CreateGraphics();

return new PointF[] { pB, pC, pD, pE, pF, pG,pH,pI };

}

public PointF[] GenerateKochFractal(PointF start, float stepSize, int iterations)

{

string axiom = "A";

string tempAx = "";

Dictionary<char, string> logic = new Dictionary<char, string>

{

{ 'A', "A-B--B+A++AA+B-" },

{ 'B', "+A-BB--B-A++A+B" }

};

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

{

foreach (char j in axiom)

{

tempAx += logic.ContainsKey(j) ? logic[j] : j.ToString();

}

axiom = tempAx;

tempAx = "";

}

List<PointF> points = new List<PointF> { start };

float currentX = start.X;

float currentY = start.Y;

double angle = 0;

foreach (char k in axiom)

{

if (k == '+')

{

angle -= Math.PI / 3;

}

else if (k == '-')

{

angle += Math.PI / 3;

}

else

{

currentX += (float)(stepSize \* Math.Cos(angle));

currentY += (float)(stepSize \* Math.Sin(angle));

points.Add(new PointF(currentX, currentY));

}

}

return points.ToArray();

}

public void DrawTreeOnPictureBox(int levels, PictureBox pictureBox2)

{

nodes.Clear(); // Очищаем список узлов

nodeCounter = 0; // Сбрасываем счётчик

using (Graphics g = pictureBox2.CreateGraphics())

{

g.Clear(Color.White); // Очищаем PictureBox

// Параметры для размещения

float startX = pictureBox2.Width / 2; // Центр по X

float startY = 50; // Начальная Y-координата

float levelHeight = 80; // Расстояние между уровнями

float baseWidth = pictureBox2.Width \* 0.75f; // Ширина дерева

float circleRadius = 5; // Радиус кружков

// Цвета для уровней

Color[] levelColors = new Color[]

{

Color.Red, // Уровень 1 (корень)

Color.Blue, // Уровень 2

Color.Green, // Уровень 3

Color.Orange, // Уровень 4

Color.Purple // Уровень 5

};

// Список для хранения линий

List<(PointF parent, PointF child)> lines = new List<(PointF, PointF)>();

// Рекурсивная функция для рисования узлов и линий

void DrawNode(int currentLevel, float x, float y, float width)

{

if (currentLevel > levels) return;

// Цвет для текущего уровня

Color nodeColor = levelColors[Math.Min(currentLevel - 1, levelColors.Length - 1)];

PointF currentPoint = new PointF(x, y);

// Рисуем кружок для узла

using (SolidBrush brush = new SolidBrush(nodeColor))

{

g.FillEllipse(brush, x - circleRadius, y - circleRadius, 2 \* circleRadius, 2 \* circleRadius);

}

// Сохраняем узел с его уровнем

nodes.Add((currentPoint, circleRadius, nodeColor, nodeCounter++, currentLevel));

if (currentLevel == levels) return;

// Вычисляем позиции для 6 детей

float childWidth = width / 6;

float offsetX = x - width / 2 + childWidth / 2;

float nextY = y + levelHeight;

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

{

float childX = offsetX + i \* childWidth;

PointF childPoint = new PointF(childX, nextY);

lines.Add((currentPoint, childPoint));

DrawNode(currentLevel + 1, childX, nextY, childWidth);

}

}

// Запускаем рисование

DrawNode(1, startX, startY, baseWidth);

// Рисуем линии

using (Pen pen = new Pen(Color.Black, 1))

{

foreach (var (parent, child) in lines)

{

g.DrawLine(pen, parent, child);

}

}

}

}

public void PictureBoxClick()

{

Graphics Graf = pictureBox1.CreateGraphics();

Graf.Clear(BackColor);

PointF[] all = GenerateKochFractal(new PointF(cX, cY), L, levels);

Graf.DrawLines(Pens.Black, all);

}

public Form1()

{

InitializeComponent();

}

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

{

PointF start = new PointF(350, 300);

PointF end = new PointF(100, 300);

PointF[] all = GetKochSegmentPoints(start, end);

Graphics Graf = pictureBox1.CreateGraphics();

Graphics Graf\_2 = pictureBox2.CreateGraphics();

Graf.Clear(BackColor);

Graf\_2.Clear(BackColor);

PointF[] all\_1 = GetKochSegmentPoints(all[1], all[0]);

PointF[] all\_2 = GetKochSegmentPoints(all[1], all[2]);

PointF[] all\_3 = GetKochSegmentPoints(all[2], all[3]);

//PointF[] all\_4 = GetKochSegmentPoints(all[3], all[4]);

Graf.DrawLines(Pens.Black, all);

Graf.DrawLines(Pens.Green, all\_1);

Graf.DrawLines(Pens.Red, all\_2);

Graf.DrawLines(Pens.Blue, all\_3);

//Graf.DrawLines(Pens.Red, all\_4);

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (textBox1.Text.All(char.IsDigit))

{

float.TryParse(this.textBox1.Text, out L);

}

else

{

L = 10;

}

if (L < 2)

{

L = 2;

}

else if (L > 250)

{

L = 250;

}

}

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

{

DrawTreeOnPictureBox(5, pictureBox2);

}

private void label6\_Click(object sender, EventArgs e)

{

}

private void textBox6\_TextChanged(object sender, EventArgs e)

{

if (textBox6.Text.All(char.IsDigit))

{

float.TryParse(this.textBox6.Text, out cY);

}

else

{

cY = 20;

}

if (cY < 0)

{

cY = 0;

}

else if (cY > 600)

{

cY = 600;

}

}

private void label7\_Click(object sender, EventArgs e)

{

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

if (textBox5.Text.All(char.IsDigit))

{

float.TryParse(this.textBox5.Text, out cX);

}

else

{

cX = 400;

}

if (cX < 0)

{

cX = 0;

}

else if (cX > 600)

{

cX = 600;

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox2.Text.All(char.IsDigit))

{

int.TryParse(this.textBox2.Text, out levels);

}

else

{

levels = 100;

}

if (levels < 0)

{

levels = 0;

}

else if (levels > 600)

{

levels = 600;

}

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

}

private void pictureBox2\_MouseClick(object sender, MouseEventArgs e)

{

// Проверяем, попал ли клик в узел

foreach (var (center, radius, color, nodeId,

level) in nodes)

{

// Расстояние от точки клика до центра узла

float distance = (float)Math.Sqrt(Math.Pow(e.X - center.X, 2) + Math.Pow(e.Y - center.Y, 2));

if (distance <= radius)

{

levels = level-1;

textBox2.Text = (levels).ToString();

PictureBoxClick();

break; // Выходим после первого найденного узла

}

}

}

private void pictureBox2\_LoadCompleted(object sender, AsyncCompletedEventArgs e)

{

}

private void pictureBox2\_Paint(object sender, PaintEventArgs e)

{

}

private void label8\_Click(object sender, EventArgs e)

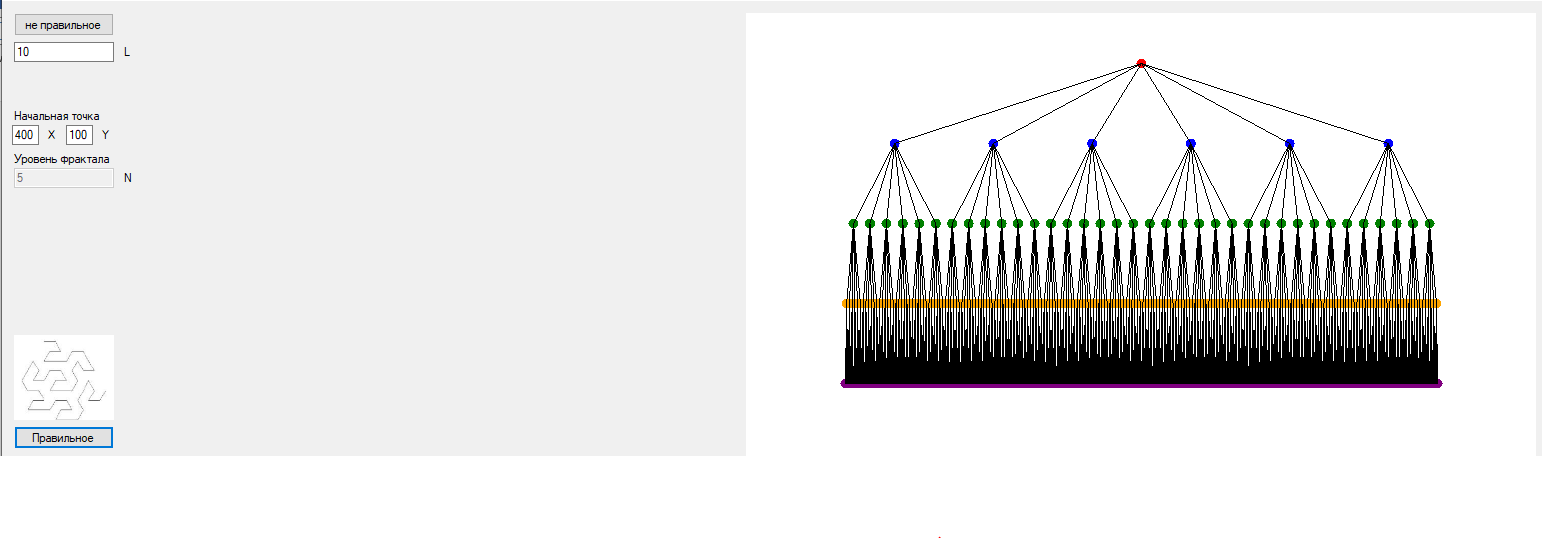
{

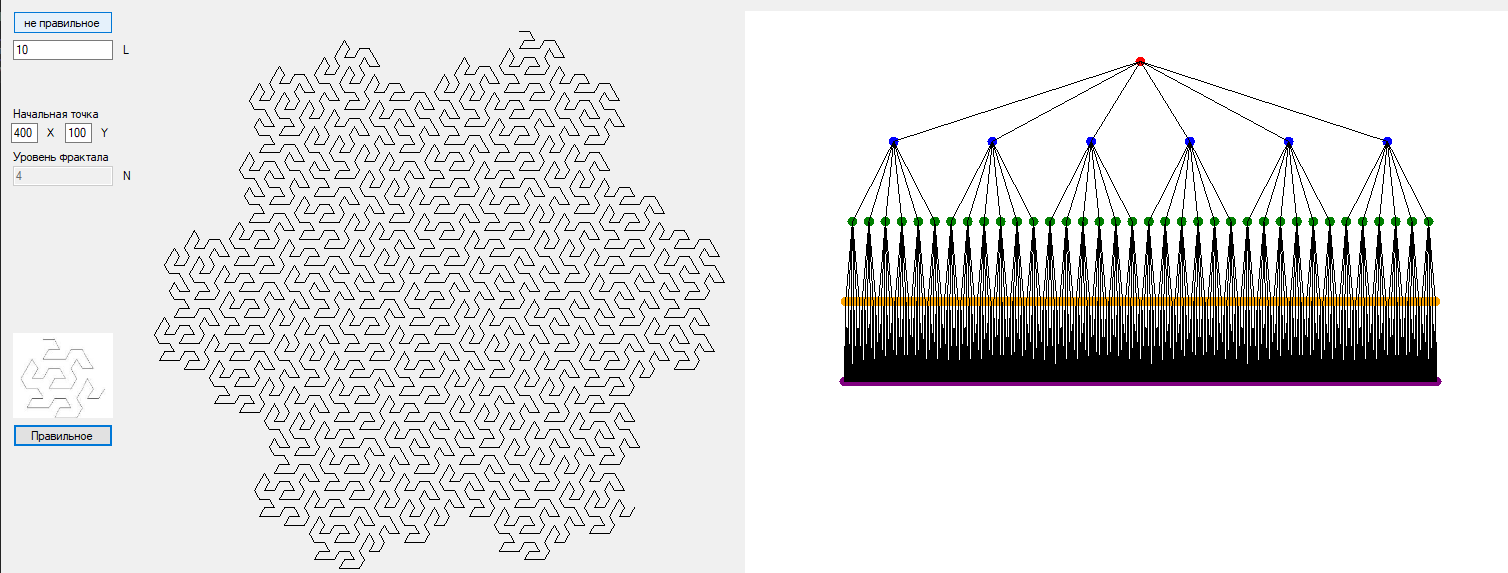
}

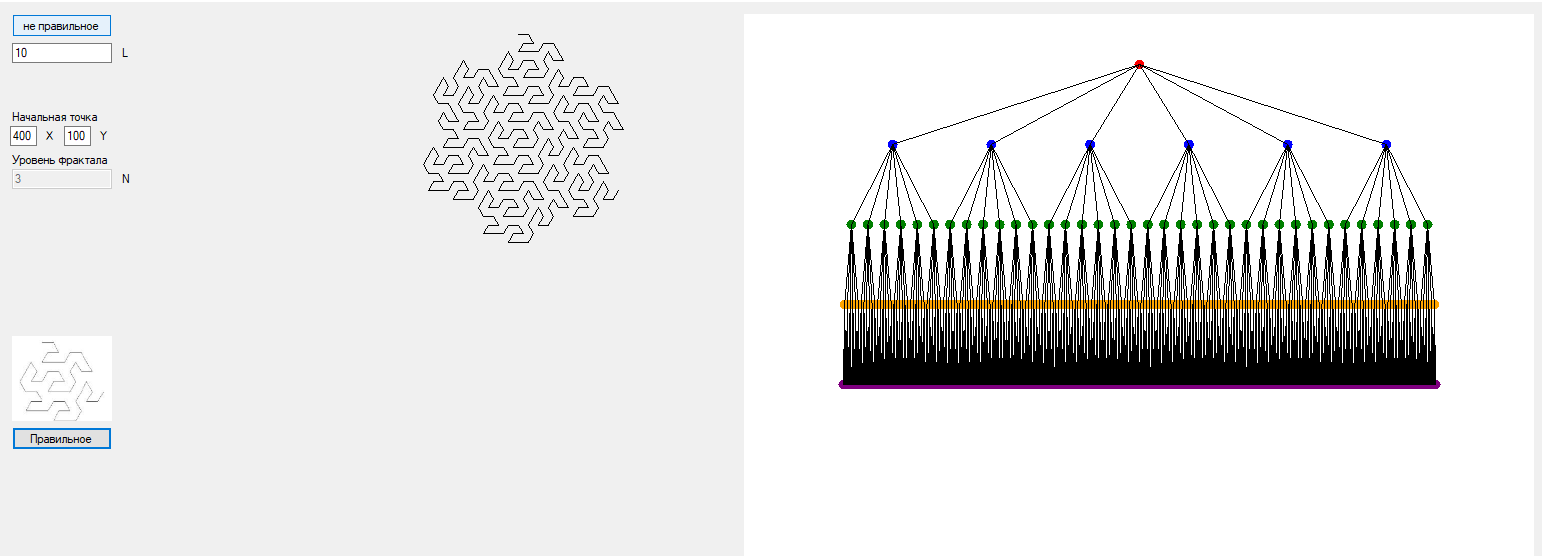
}

}

**3.6. Примеры работы программы**







**заключение**

В ходе учебной практики были освоены основные принципы работы с Windows Forms. Изучены новые подходы к разработке десктопных приложений на языке программирования C#. В первой части практики были рассмотрены теоретические основы, необходимые для выполнения индивидуальных заданий. Во второй части было реализовано задание по созданию интерактивного дерева с кликабельными узлами, отображаемого в PictureBox, с учётом уровней и цветового оформления. В третьей части выполнен проект по генерации и визуализации фрактала Госпера на основе L-системы, с выводом точек для построения в графическом интерфейсе.