**Национальный исследовательский университет «Высшая школа экономики»**

**Факультет компьютерных наук**

Департамент

**Программной инженерии**

***Контрольное домашнее задание***
***по дисциплине***
***«Программирование»***

Тема работы: Построение фракталов

Выполнил(а): студент группы 185 (1)
_____ Мануйлов А.А.

тел. +7(912) 252 334 40
e-mail адрес: aamanuylov@edu.hse.ru

Преподаватель: Чуйкин Н.К.

Москва, 2018 год. Модуль 2

# Оглавление

# 1. Условие задачи

Вариант №101:
Разработать оконное приложение (шаблон Windows Forms Application) для отрисовки трех фракталов, позволяющее
- выбирать тип фрактала;
- устанавливать глубину рекурсии;
- изменять размеры окна;
- выбирать начальный и конечный цвет отрисовки, определяющие градиент;
- сообщать о некорректном вводе данных;
- сохранять изображение фрактала в формате .jpg;
- изменение масштаба изображения;
- перемещать фрактал, в том числе и в увеличенном виде;

Все исходные данные вводит пользователь с помощью экранных форм, содержащих поля для текстового ввода или списки значений для выбора пользователем.

# 2. Функции разрабатываемого приложения

## 2.1 Варианты использования

Программа может использоваться для наглядной демонстрации сущности фракталов на конкретных примерах.

## 2.2 Описание интерфейса пользователя

При открытии программы пользователь попадает в основное окно, имеющее пустой TextBox для установки глубина рекурсии, ComboBox для выбора увеличения, две кнопки Button для выбора цветов, ComboBox для выбора типа фрактала и кнопка Button для сохранения изображения

1. При нажатии на кнопку «Нарисовать» на объекте типа PictureBox отрисовывается фрактал соответствующий текущим введенным параметрам.
2. Кнопки «Начальный цвет» и «Конечный цвет» вызывают объекты типа ColorDialog, позволяющие выбрать цвет.
3. При нажатии кнопки «Сохранить» текущее изображение в PictureBox сохраняется в выбранную папку.
4. ComboBox «Тип фрактала» предоставляет выбор трех возможный типов фракталов.
5. ComboBox «Увеличение» предоставляет выбор трех возможный коэффициентов увеличения (1, 2, 3, 5).
6. TextBox «Глубина рекурсии» позволяет вводить значение глубины рекурсии отрисовки.

# Структура приложения

## 2.3 Диаграмма классов

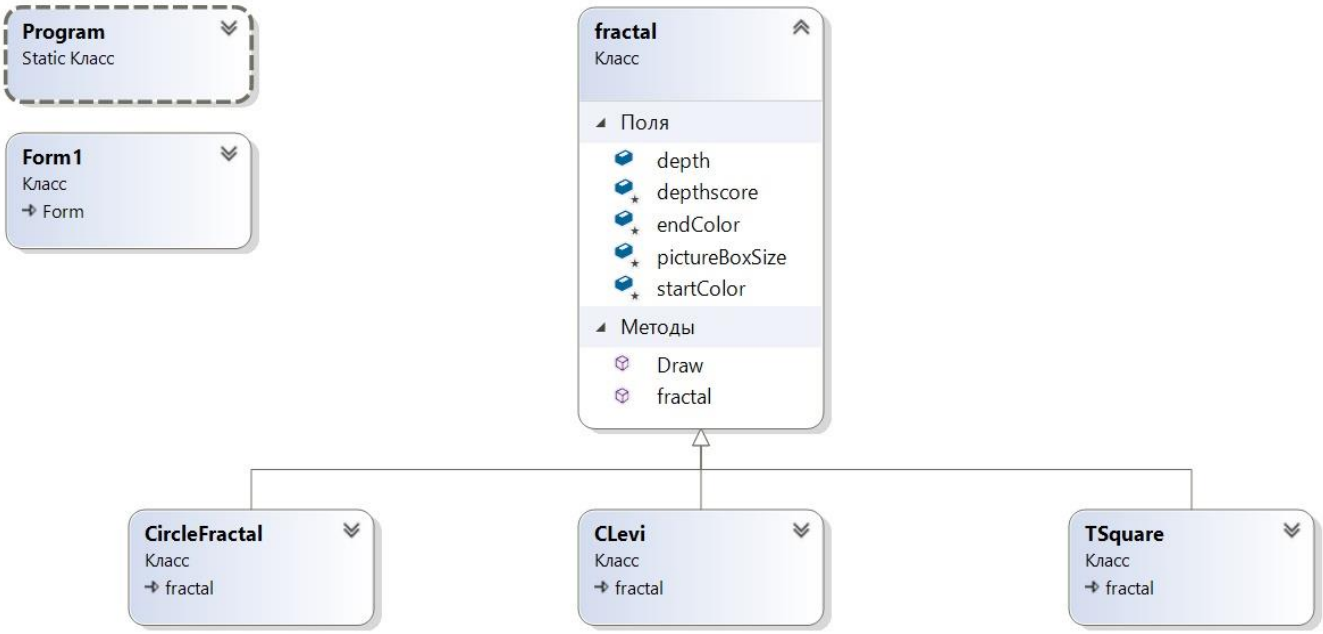


**Рисунок 1: Диаграмма классов**

## 2.4 Описание классов, их полей и методов

**Классы:**

- **`static class Program`** класс, содержащий метод Main()
- **`public class fractal`** абстрактный класс, содержащий метод Draw, переопределенный в производных
- **`public partial class Form1: Form`** форма, содержащая основные методы для отрисовки фрактала
- **`class CLevi: fractal`** класс-наследник, переопределяющий метод Draw
- **`class CircleFractal: fractal`** класс-наследник, переопределяющий метод Draw
- **`class TSquare: fractal`** класс-наследник, переопределяющий метод Draw

**Методы:**

- **`public class fractal`**
  - **`public virtual void Draw()`** шаблон для вызова рекурсивной функции отрисовки фрактала (переопределен в производных классах)
- **`class CLevi: fractal`**
  - **`public override void Draw()`** вызов рекурсивной функции отрисовки фрактала
- **`class CircleFractal: fractal`**
  - **`public override void Draw()`** вызов рекурсивной функции отрисовки фрактала
- **`class TSquare: fractal`**
  - **`public override void Draw()`** вызов рекурсивной функции отрисовки фрактала

- **public partial class Form1: Form**
  - **public void Gradient(List<Color> colorList)** заполнение списка цветов цветами градиента между начальным и конечным цветом
  - **private void CFractal()** вызов отрисовки С-кривой Леви
  - **private void TFractal()** вызов отрисовки Т-квадрата
  - **private void CircleFractal()** вызов отрисовки кругового фрактала
  - **void Clear()** закрашивание PictureBox'а цветом фона
  - **private void CheckDepth()** проверка на корректность глубины фрактала и вызов одной из функций вызывающий отрисовку
  - **private void SelectFracral()** узнает из ComboBox какой фрактал выбран

**Поля:**

- **public class fractal**
  - **protected Color startColor** поле, содержащее начальный цвет градиента
  - **protected Color endColor** поле, содержащее конечный цвет градиента
  - **protected int depthscore** текущая глубина рекурсии
  - **public int depth** глубина рекурсии
  - **protected Size pictureBoxSize** размер области рисования
- **class CLevi : fractal**
  - **PointF newPoint** вспомогательная точка для отрисовки
  - **PointF center** центр PictureBox впоследствии использующийся для рисования
  - **PointF A** вспомогательная точка для рисования
  - **Graphics gr** объект графики для рисования
  - **List<Color> colorList** динамический массив для хранения цветов градиента
  - **Pen pn** ручка для рисования
  - **Float size** коэффициент для масштабирования
- **class TSquare : fractal**
  - **double d** вспомогательная переменная – четверть стороны предыдущего квадрата
  - **PointF center** центр PictureBox впоследствии использующийся для рисования
  - **PointF A** вспомогательная точка для рисования
  - **Graphics gr** объект графики для рисования
  - **List<Color> colorList** динамический массив для хранения цветов градиента
  - **Pen pn** ручка для рисования
  - **Float size** сторона квадрата
  - **PointF[] M** вспомогательный массив, хранящий точки для отрисовки порожденных квадратов
  - **SolodBrush br** кисть, необходимая для закрашивания прямоугольников

- **class CircleFractal : fractal**
  - **PointF center** центр PictureBox впоследствии использующийся для рисования
  - **PointF A** вспомогательная точка для рисования
  - **Graphics gr** объект графики для рисования
  - **List<Color> colorList** динамический массив для хранения цветов градиента
  - **Pen pn** ручка для рисования
  - **Float size** радиус окружности
  - **PointF[] M** вспомогательный массив, хранящий точки для отрисовки порожденных окружностей

- **public partial class Form1: Form**
  - **Bitmap map** объект для запоминания изменений графики
  - **Graphics gr** объект графики для рисования
  - **Pen myPen** ручка для рисования
  - **PointF center** центр PictureBox впоследствии использующийся для рисования
  - **PointF nowcenter** точка для изменения положения изображения
  - **PointF mousePos** место где была зажата клавиша мыши. Используется при перемещении изображения
  - **List<Color> colorList** динамический массив для хранения цветов градиента
  - **int depth** глубина рекурсии
  - **int koef** коэффициент увеличения
  - **bool flag_depth** флаг для понимания была ли ошибка
  - **Color defaultColor** цвет по умолчанию – черный
  - **String typeFractal** строка, определяющая какой фрактал выбран

# 3. Распределение исходного кода по файлам проекта

1. Form1.cs – файл содержит главную форму программы и методы для обработки вводимых данных, отрисовки изображения и управления главными элементами формы.
2. Program.cs – файл содержит метод Main() – стандартную точку входа в выполнение
3. fractal.cs – файл, содержащий информацию об родительском классе
4. CLevi.cs – файл содержит дочерний класс fractal, хранящий информацию об отрисовке С-кривой Леви
5. TSquare.cs – файл содержит дочерний класс fractal, хранящий информацию об отрисовке Т-квадрата
6. CircleFractal.cs– файл содержит дочерний класс fractal, хранящий информацию об отрисовке кругового фрактала

# 4. Контрольный пример и описание результатов

Для проверки работоспособности программы были проведены множественные тесты.

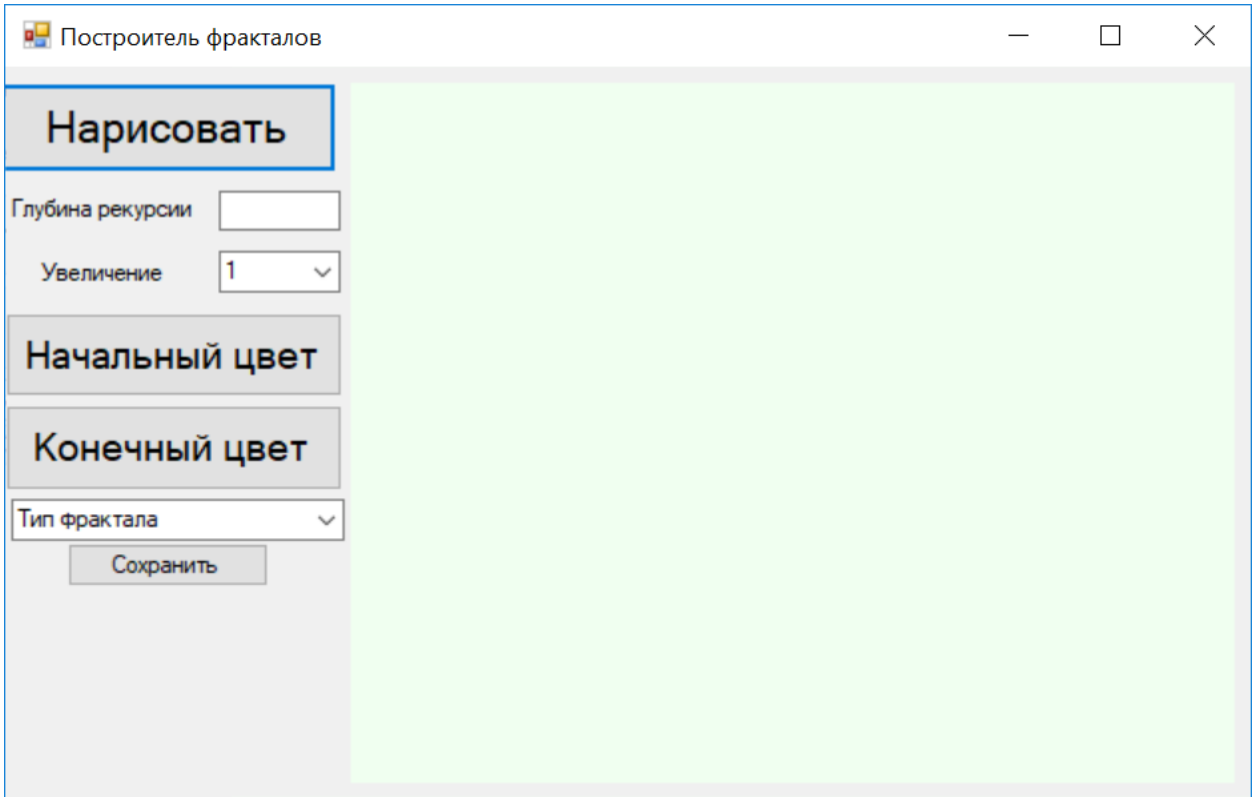


**Рисунок 2**
Начальный вид окна программы. По умолчанию стоит увеличение в 1 раз.

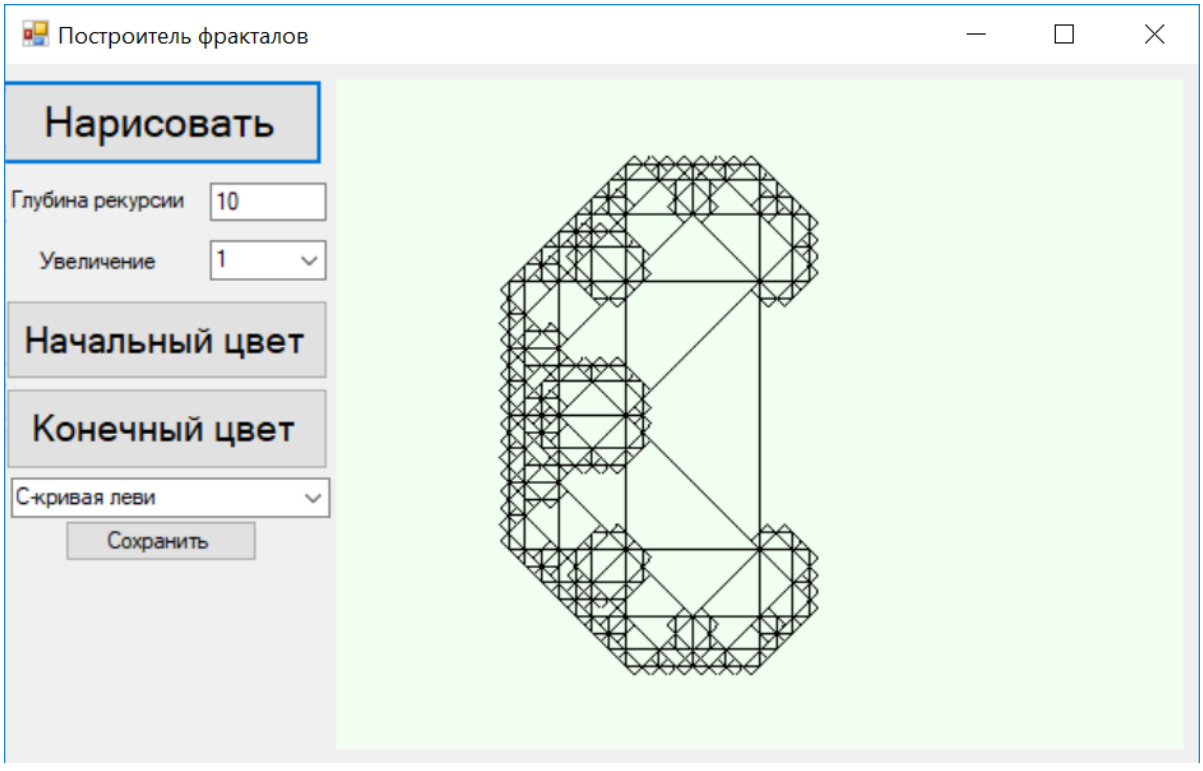

**Рисунок 3**
Пример построения фрактала (с-кривая Леви)


**Рисунок 4**
Пример градиента на т-квадрате

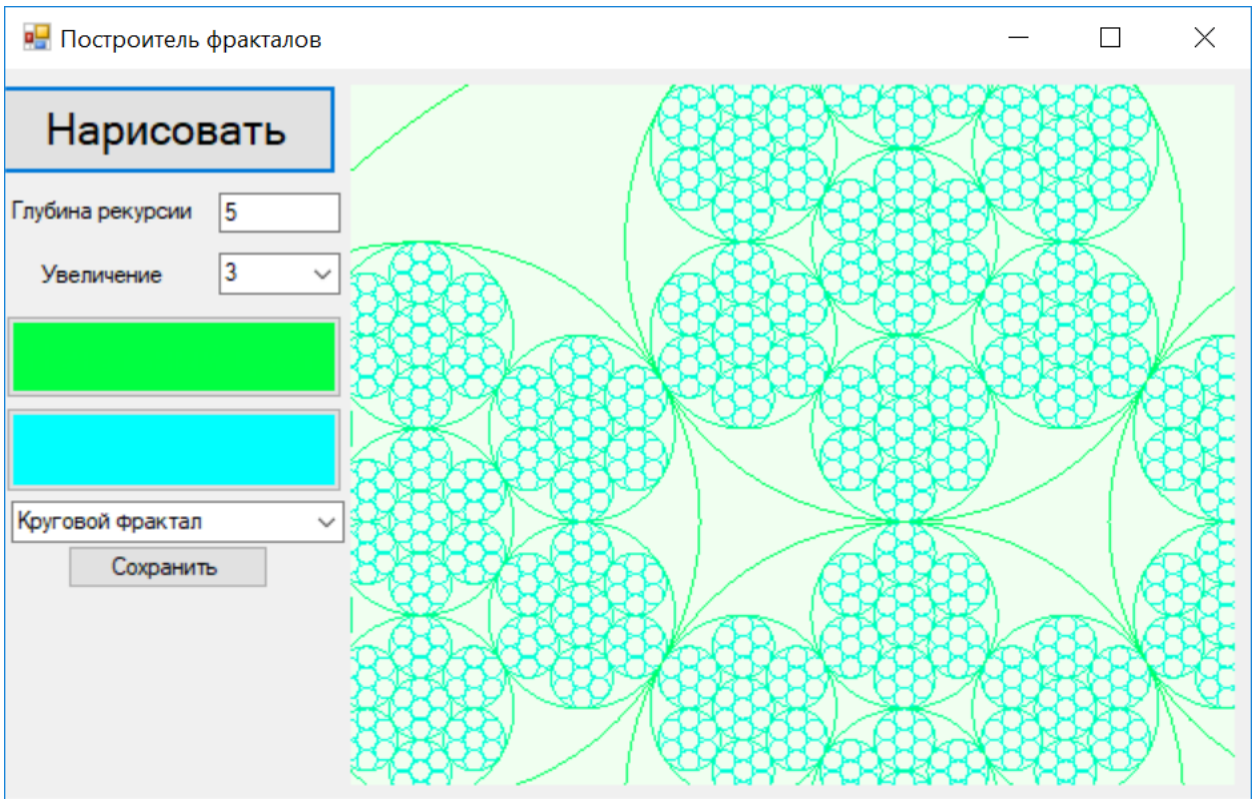


**Рисунок 5**
Демонстрация увеличения

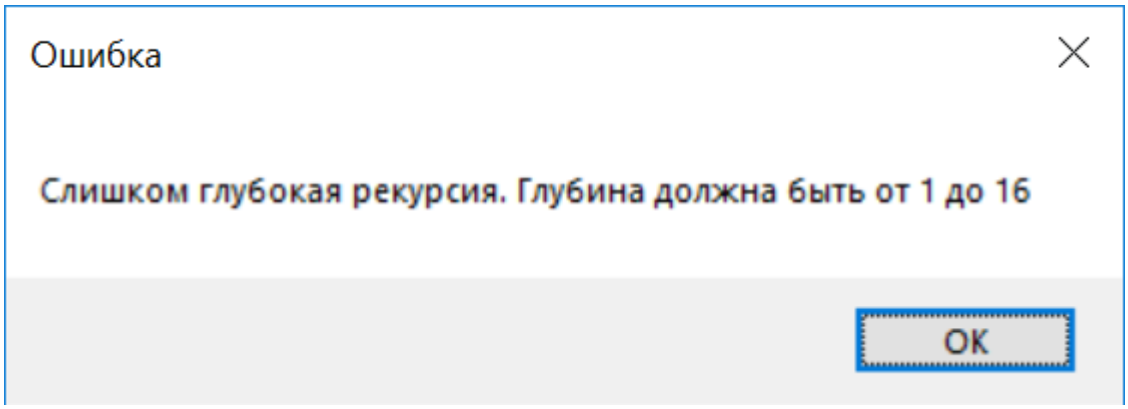


**Рисунок 6**

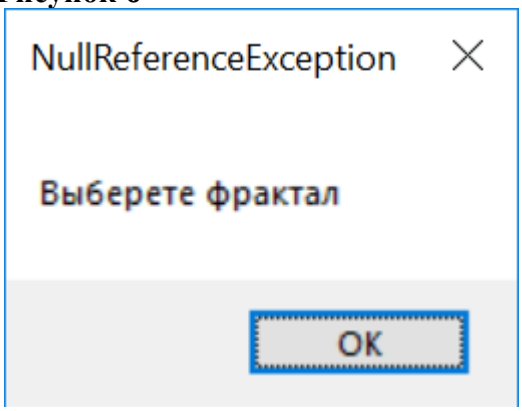


**Рисунок 7**

## 5. Текст (код) программы

Файл fractal.cs

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

namespace fractals
{
    public class fractal
    {
        //public float leng; // коэфициент увеличения
        protected Color startColor; // начальный цвет
        protected Color endColor; // конечный цвет
        protected int depthscore; // счетчик рекурсии
        public int depth; // глубина рекурсии
        protected Size pictureBoxSize;

        /// <summary>
        /// конструктор класса фрактала
        /// </summary>
        /// <param name="startColor"></param>
        /// <param name="endColor"></param>
        /// <param name="depthmax"></param>
```

```
        /// <param name="leng"></param>
        public fractal(Color startColor, Color endColor, int depth, Size pictureBoxSize)
        {
            this.startColor = startColor;
            this.endColor = endColor;
            this.depth = depth;
            this.depthscore = 0;
            this.pictureBoxSize = pictureBoxSize;
        }

        //переопределим в классе конкретного фрактала функция отрисовки
        public virtual void Draw(PointF center, PointF A, ref Graphics gr, List<Color>
colorList, Pen pen, float size) { }
    }
}
```

Файл CLevi.cs

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

namespace fractals
{
    class CLevi : fractal
    {
        // конструктор от родительского класса
        public CLevi(Color startColor, Color endColor, int depth, Size pictureBoxSize)
            : base(startColor, endColor, depth, pictureBoxSize){ }

        //PointF A нужна только здесь
        public override void Draw(PointF center,PointF A, ref Graphics gr, List<Color>
colorList, Pen pn, float size)
        {
            pn.Color = colorList[depthscore];
            if (depthscore == 0)
            {// верхняя точка первой линии
                A = new PointF(center.X, center.Y + size / 2);
                center = new PointF(center.X, center.Y - size / 2);
            }
            if (depthscore == depth - 1)
            {
                gr.DrawLine(pn, center.X, center.Y, A.X, A.Y);
                return;
            }
            else
            {
                PointF newPoint = new PointF((center.X + A.X) / 2 - (A.Y - center.Y) / 2,
                                             (center.Y + A.Y) / 2 + (A.X - center.X) / 2);
                depthscore++;
                Draw(center, newPoint, ref gr, colorList, pn, size);
                Draw(newPoint, A, ref gr, colorList, pn, size);
                depthscore--;
            }
            // рисуем исходную линию для градиента
            pn.Color = colorList[depthscore];
            gr.DrawLine(pn, center.X, center.Y, A.X, A.Y);
```

```
            return;
        }
    }
}
```

Файл TSquare.cs

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

namespace fractals
{
    class TSquare : fractal
    {
        // конструктор от родительского класса
        public TSquare(Color startColor, Color endColor, int depth, Size pictureBoxSize)
            : base(startColor, endColor, depth, pictureBoxSize){ }

        /// <summary>
        /// переопределенный метод отрисовки
        /// </summary>
        /// <param name="x"></param>
        /// <param name="y"></param>
        /// <param name="gr"></param>
        /// <param name="colorList"></param>
        /// <param name="pn"></param>
        public override void Draw(PointF center, PointF A, ref Graphics gr, List<Color>
colorList, Pen pn, float size)
        {
            SolidBrush br = new SolidBrush(colorList[(int)colorList.LongCount() - 1]);
            if (depthscore == 0)
            {
                center = new PointF(center.X - size / 2, center.Y - size / 2);
            }
            //Рекурсивная функция отрисовки фрактала
            //size - длина стороны

            //База рекурсии
            //Если итерация одна, просто рисуем заполненный прямоугольник
            if (depthscore == depth - 1)
            {
                gr.FillRectangle(br, center.X, center.Y, size, size);
                return;
            }

            float d = size / 4; //Вспомогательная переменная, четверть длины исходного
квадрата
            PointF[] M = new PointF[4];  //Координаты левых верхних углов порожденных
квадратов

            for (int i = 0; i < 4; i++)
            {
                M[i] = new PointF();
            }

            //Левый верхний квадрат
```

```
            M[0].X = center.X - d;
            M[0].Y = center.Y - d;

            //Левый нижний квадрат
            M[1].X = center.X - d;
            M[1].Y = center.Y + size - d;

            //Правый верхний квадрат
            M[2].X = center.X + size - d;
            M[2].Y = center.Y - d;

            //Правый нижний квадрат
            M[3].X = center.X + size - d;
            M[3].Y = center.Y + size - d;

            //Вызываем рекурсивно для каждого квадрата
            for (int i = 0; i < 4; i++)
            {
                depthscore++;
                Draw(M[i], M[i], ref gr, colorList, pn, size / 2);
                depthscore--;
            }
            // меняем цвет кисти в зависимости от глубины рекурсии
            br = new SolidBrush(colorList[depthscore]);

            //Отрисовываем исходный квадрат
            gr.FillRectangle(br, center.X, center.Y, size, size);
            return;
        }
    }
}
```

Файл CircleFractal.cs

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

namespace fractals
{
    class CircleFractal : fractal
    {
        // конструктор от родительского класса
        public CircleFractal(Color startColor, Color endColor, int depth, Size
pictureBoxSize)
            : base(startColor, endColor, depth, pictureBoxSize) { }


        // size  используем как радиус
        public override void Draw(PointF center, PointF A, ref Graphics gr, List<Color>
colorList, Pen pn, float size)
        {
            pn.Color = colorList[depthscore];
            if (depthscore == depth - 1)
            {
                //pn.Color = colorList[(int)colorList.LongCount() - 1];
                gr.DrawEllipse(pn, center.X - size / 2, center.Y - size / 2, size, size);
                return;
```

```
            }

            float d = size / 3; //Вспомогательная переменная, треть радиуса исходной
окружности
            PointF[] M = new PointF[7];  //Координаты центров порожденных окружностей

            for (int i = 0; i < 7; i++)
            {
                M[i] = new PointF();
            }

            M[0].X = center.X;
            M[0].Y = center.Y;

            M[1].X = center.X;
            M[1].Y = center.Y - d;

            M[2].X = center.X;
            M[2].Y = center.Y + d;

            M[3].X = (float)((M[1].X - center.X) * Math.Cos(Math.PI / 3) - (M[1].Y -
center.Y) * Math.Sin(Math.PI / 3) + center.X);
            M[3].Y = (float)((M[1].X - center.X) * Math.Sin(Math.PI / 3) + (M[1].Y -
center.Y) * Math.Cos(Math.PI / 3) + center.Y);

            M[4].X = (float)((M[1].X - center.X) * Math.Cos(-Math.PI / 3) - (M[1].Y -
center.Y) * Math.Sin(-Math.PI / 3) + center.X);
            M[4].Y = (float)((M[1].X - center.X) * Math.Sin(-Math.PI / 3) + (M[1].Y -
center.Y) * Math.Cos(-Math.PI / 3) + center.Y);

            M[5].X = (float)((M[2].X - center.X) * Math.Cos(Math.PI / 3) - (M[2].Y -
center.Y) * Math.Sin(Math.PI / 3) + center.X);
            M[5].Y = (float)((M[2].X - center.X) * Math.Sin(Math.PI / 3) + (M[2].Y -
center.Y) * Math.Cos(Math.PI / 3) + center.Y);

            M[6].X = (float)((M[2].X - center.X) * Math.Cos(-Math.PI / 3) - (M[2].Y -
center.Y) * Math.Sin(-Math.PI / 3) + center.X);
            M[6].Y = (float)((M[2].X - center.X) * Math.Sin(-Math.PI / 3) + (M[2].Y -
center.Y) * Math.Cos(-Math.PI / 3) + center.Y);

            for (int i = 0; i < 7; i++)
            {
                depthscore++;
                Draw(M[i], M[i], ref gr, colorList, pn, size / 3);
                depthscore--;
            }
            //Отрисовываем исходную окружность
            pn.Color = colorList[depthscore];
            gr.DrawEllipse(pn, center.X - size / 2, center.Y - size / 2, size, size);
            return;
        }
    }
}
```

Файл Form1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using System.Windows.Forms;

namespace fractals
{
    public partial class Form1 : Form
    {
        Bitmap map;
        Graphics gr;
        Pen myPen;
        PointF center, nowcenter, mousePos;
        List<Color> colorList;
        int depth;
        int koef;
        bool flag_depth;
        Color defaultColor;
        string typeFractal;

        public Form1()
        {
            InitializeComponent();
            defaultColor = Color.Black;
            comboBox1.SelectedItem = "1";
            koef = 1;
            myPen = new Pen(Color.Black);
            MinimumSize = new Size(Screen.PrimaryScreen.Bounds.Size.Width / 2,
                Screen.PrimaryScreen.Bounds.Size.Height / 2);
            map = new Bitmap(pictureBox1.Width, pictureBox1.Height);//Подключаем Битмап
            gr = Graphics.FromImage(map);
            colorList = new List<Color>();
        }

        /// <summary>
        /// кнопка нажать, приводящая к отрисовке фрактала
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void button2_Click(object sender, EventArgs e)
        {
            center = new PointF(pictureBox1.Width / 2, pictureBox1.Height / 2);

            SelectFractal();
            CheckDepth();
            if (flag_depth)
            {
                flag_depth = false;
                return;
            }// если ошибка то выходим
        }

        /// <summary>
        /// функция узнающая какой фрактал выбрал пользователь
        /// </summary>
        private void SelectFractal()
        {
            try
            {
                if (comboBox2.Text == "Тип фрактала")
                {
                    throw new NullReferenceException();
                }
                typeFractal = comboBox2.Text;
            }
            catch (NullReferenceException)
```

```
            {
                MessageBox.Show("Выберете фрактал", "NullReferenceException");
                return;
            }
        }

        /// <summary>
        /// функция проверяющая корректность ввода глубины рекурсии и вызывающая
отрисовку
        /// </summary>
        private void CheckDepth()
        {
            try
            {
                depth = int.Parse(textBox1.Text);
                if (typeFractal == "Т-квадрат")
                {
                    if (depth <= 0 || depth > 11)
                    {
                        MessageBox.Show("Слишком глубокая рекурсия. Глубина должна быть
от 1 до 11", "Окошко"); return;
                    }
                    TFractal();
                }
                if (typeFractal == "С-кривая леви")
                {
                    if (depth <= 0 || depth > 16)
                    {
                        MessageBox.Show("Слишком глубокая рекурсия. Глубина должна быть
от 1 до 16", "Окошко"); return;
                    }
                    CFractal();
                }
                if(typeFractal == "Круговой фрактал")
                {
                    if (depth <= 0 || depth > 6)
                    {
                        MessageBox.Show("Слишком глубокая рекурсия. Глубина должна быть
от 1 до 6", "Окошко"); return;
                    }
                    CircleFractal();
                }
            }
            catch (NullReferenceException)
            {
                MessageBox.Show("Неверное значение глубины рекурсии",
"NullReferenceException");
                flag_depth = true;
                return;
            }
            catch (FormatException)
            {
                MessageBox.Show("Некорректное значение глубины рекурсии",
"FormatException");
                flag_depth = true;
                return;
            }
        }

        /// <summary>
        /// закрашиваем все цветом фона
        /// </summary>
        void Clear()
        {
```

```
            gr.Clear(Color.Honeydew);
            pictureBox1.Image = map;
        }

        /// <summary>
        /// вызов отрисовки т-квадрата
        /// </summary>
        private void TFractal()
        {
            Clear();
            Gradient(colorList);
            fractal sq = new TSquare(Color.Black, Color.Black, depth, pictureBox1.Size);
            sq.Draw(center, center, ref gr, colorList, myPen, (pictureBox1.Height / 2 -
pictureBox1.Height / 10) * koef);
            pictureBox1.Image = map;
        }

        /// <summary>
        /// вызов отрисовки кругового фрактала
        /// </summary>
        private void CircleFractal()
        {
            Clear();
            Gradient(colorList);
            fractal cf = new CircleFractal(Color.Black, Color.Black, depth,
pictureBox1.Size);
            cf.Draw(center, center, ref gr, colorList, myPen, (pictureBox1.Height -
pictureBox1.Height / 5) * koef);
            pictureBox1.Image = map;
        }

        /// <summary>
        /// вызов отрисовки с-кривой леви
        /// </summary>
        private void CFractal()
        {
            Clear();
            Gradient(colorList);
            fractal cl = new CLevi(Color.Black, Color.Black, depth, pictureBox1.Size);
            PointF A = new PointF(center.X, center.Y + pictureBox1.Width / 6);
            cl.Draw(center, A, ref gr, colorList, myPen, (pictureBox1.Height / 2 -
pictureBox1.Height / 10) * koef);
            pictureBox1.Image = map;
        }

        /// <summary>
        ///  событие дмижения мыши, перерисовывающее фрактал
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                DoubleBuffered = true;
                center.X = nowcenter.X - mousePos.X + MousePosition.X;
                center.Y = nowcenter.Y - mousePos.Y + MousePosition.Y;
                CheckDepth();
            }
        }

        /// <summary>
        /// событие изменения коэффициента приближения
        /// </summary>
```

```
/// <param name="sender"></param>
/// <param name="e"></param>
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    koef = int.Parse(comboBox1.Text);
}

// события нажатия на кнопку вызывают колор диалог, где мы выбираем цвета
private void button1_Click(object sender, EventArgs e)
{
    colorDialog1.ShowDialog();
    button1.ForeColor = colorDialog1.Color;//красим буквы
    button1.BackColor = colorDialog1.Color;//красим кнопку
}

private void button3_Click(object sender, EventArgs e)
{
    colorDialog2.ShowDialog();
    button3.ForeColor = colorDialog2.Color;//красим буквы
    button3.BackColor = colorDialog2.Color;//красим кнопку
}

/// <summary>
/// функция создания градиента в массиве цветов по начальному и конечному цвету
/// </summary>
/// <param name="colorList"></param>
public void Gradient(List<Color> colorList)
{
    colorList.Clear();
    int rMin = colorDialog1.Color.R;
    int gMin = colorDialog1.Color.G;
    int bMin = colorDialog1.Color.B;
    int argMin = colorDialog1.Color.ToArgb();

    if (argMin == 0)
    {
        argMin = defaultColor.ToArgb();
    }
    int rMax = colorDialog2.Color.R;
    int gMax = colorDialog2.Color.G;
    int bMax = colorDialog2.Color.B;
    int argMax = colorDialog2.Color.ToArgb();

    if (argMax == 0)
    {
        argMax = defaultColor.ToArgb();
    }
    for (int i = 0; i <= depth; i++)
    {
        var rAverage = rMin + ((rMax - rMin) * i / depth);
        var gAverage = gMin + ((gMax - gMin) * i / depth);
        var bAverage = bMin + ((bMax - bMin) * i / depth);
        colorList.Add(Color.FromArgb(rAverage, gAverage, bAverage));
    }
}

/// <summary>
/// сохранение изображения в формате .jpg
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void button4_Click(object sender, EventArgs e)
{
    SaveFileDialog save = new SaveFileDialog();
```

```
            save.ShowDialog();
            string filename = save.FileName + ".jpg";
            map.Save(filename, System.Drawing.Imaging.ImageFormat.Jpeg);
        }

        /// <summary>
        /// изменение размера формы с последующей перерисовкой фрактала
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Form1_SizeChanged(object sender, EventArgs e)
        {
            try
            {
                map = new Bitmap(pictureBox1.Width, pictureBox1.Height);
                gr = Graphics.FromImage(map);
            }
            catch (ArgumentException ext)
            {
                MessageBox.Show(ext.Message, "Exception");
                return;
            }
            if (depth != 0)
                CheckDepth();
        }

        /// <summary>
        /// событие нажатия мыши для перемещения изображения
        /// запоминаем где нажали и где был центр
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
        {
            mousePos = MousePosition;
            nowcenter = center;
        }
    }
}
```

# 6. Список литературы


1. Герберт Шилдт: C# 4.0. Полное руководство, 2011.
2. Руководство по программированию на C# - MSDN – Microsoft [Электронный ресурс]: русский ресурс, содержащий информацию по документации Visual Studio 2008 URL: https://msdn.microsoft.com/ru-ru/
3. [Электронный ресурс]: русский ресурс, содержащий информацию по визуализации фракталов, алгоритмах из отрисовки и возможностям Windows Forms: http://grafika.me/node/412
4. [Электронный ресурс]: русский ресурс, содержащий информацию по визуализации фракталов, алгоритмах из отрисовки и возможностям Windows Forms: http://grafika.me/node/644
5. [Электронный ресурс]: русский ресурс, содержащий информацию по визуализации фракталов, алгоритмах из отрисовки и возможностям Windows Forms: http://grafika.me/node/598
6. [Электронный ресурс]: русский ресурс, форум, посвященный проблемам, возникающим у программистов http://www.cyberforum.ru