Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Лабораторная работа №7

По дисциплине: "СПП"

Выполнил:
Студент 3 курса
Группы ПО-3
Лущ М. Г.
Проверил:
Крощенко А. А.

Брест 2019

Цель работы: освоить возможности языка программирования C# в построении графических приложений.

Вариант 14

Задание №1:

Построение графических примитивов и надписей

Требования к выполнению

• Реализовать соответствующие классы, указанные в задании;

• Организовать ввод параметров для создания объектов (можно использовать файлы);

• Осуществить визуализацию графических примитивов, решить поставленную задачу

Создать классы Point и Line. Объявить массив из n объектов класса Point и определить в методе, какая из точек находится дальше всех от прямой линии.

Код программы:

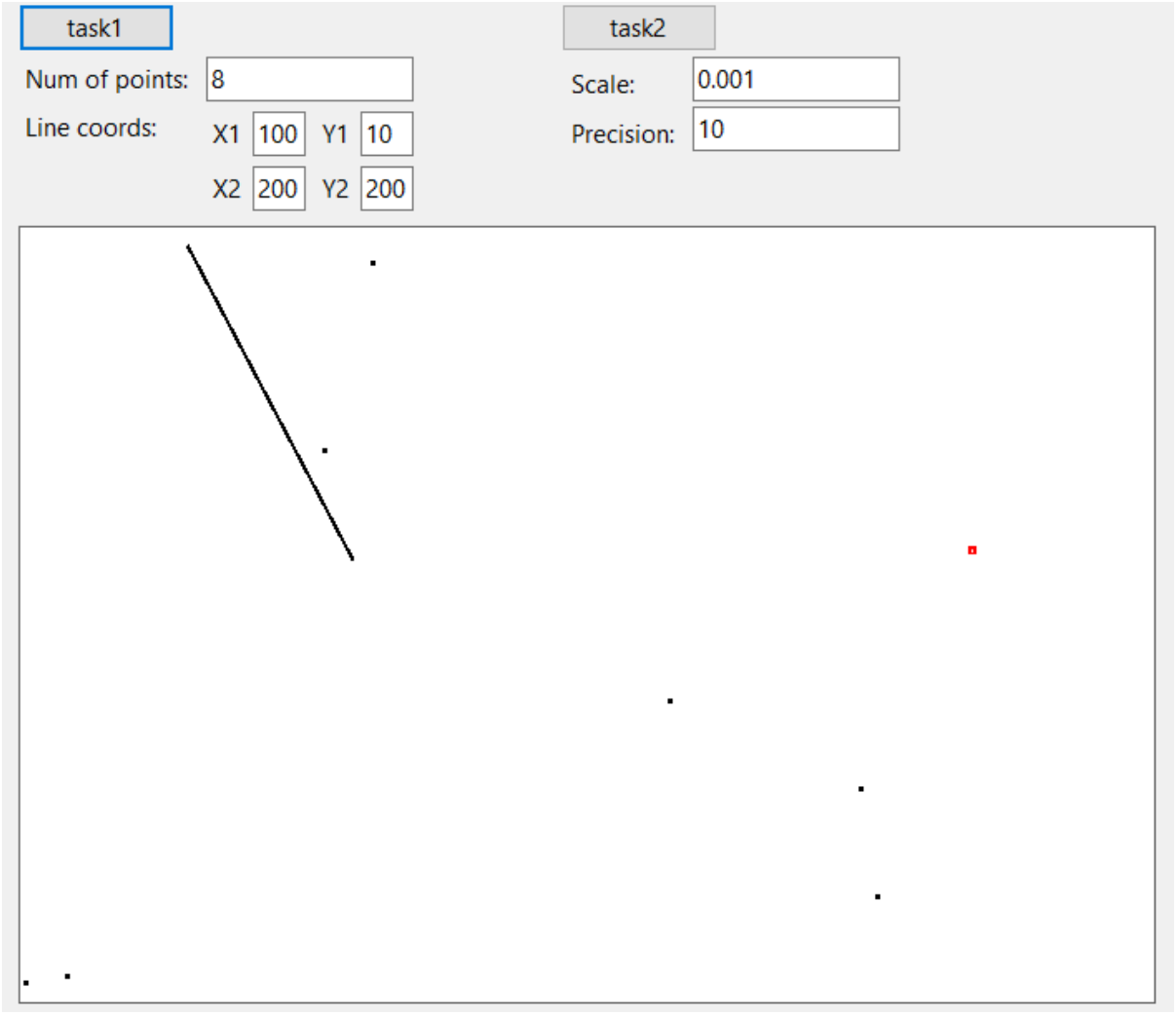
```
private void task1_Click(object sender, EventArgs e)
    {
        status.Text = string.Empty;
        using (Graphics graphics = drawingField.CreateGraphics())
        {
            graphics.Clear(Color.White);
            Pen pen = new Pen(Color.Black, 2);
            int pointsCount = 0;

            Line line = null;
            try
            {
                pointsCount = Convert.ToInt32(numOfPoints.Text);
                line = new Line(
                    new Point(Convert.ToInt32(x1.Text), Convert.ToInt32(y1.Text)),
                    new Point(Convert.ToInt32(x2.Text), Convert.ToInt32(y2.Text)));
```

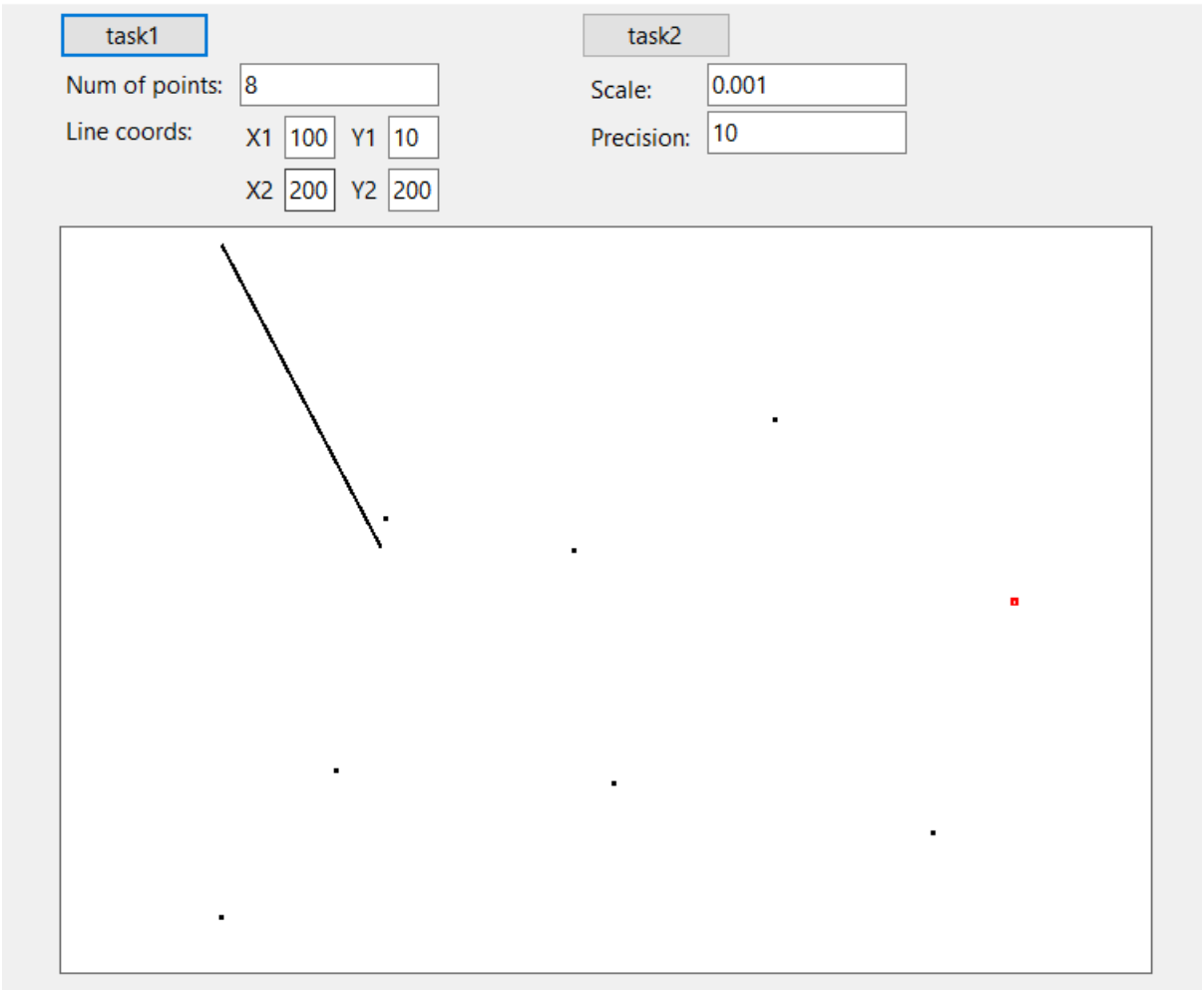
```
            }
            catch (FormatException)
            {
               line = new Line(new Point(0, 0), new Point(0, 0));
               status.Text = "incorrect format of input data";
            }

            Point[] points = new Point[pointsCount];



            Random random = new Random(DateTime.Now.Millisecond);

            for (int i = 0; i < points.Length; i++)
            {
               points[i] = new Point(random.Next(0, drawingField.Width), random.Next(0,
drawingField.Height));
            }

            Point furthest = FurthestPoint(line, points);

            line.Draw(graphics, pen);
            foreach (Point point in points)
            {
               if (point == furthest)
               {
                  point.Draw(graphics, new Pen(Color.Red, 4));
               }
               else
               {
                  point.Draw(graphics, pen);
               }
            }
         }
      }
```

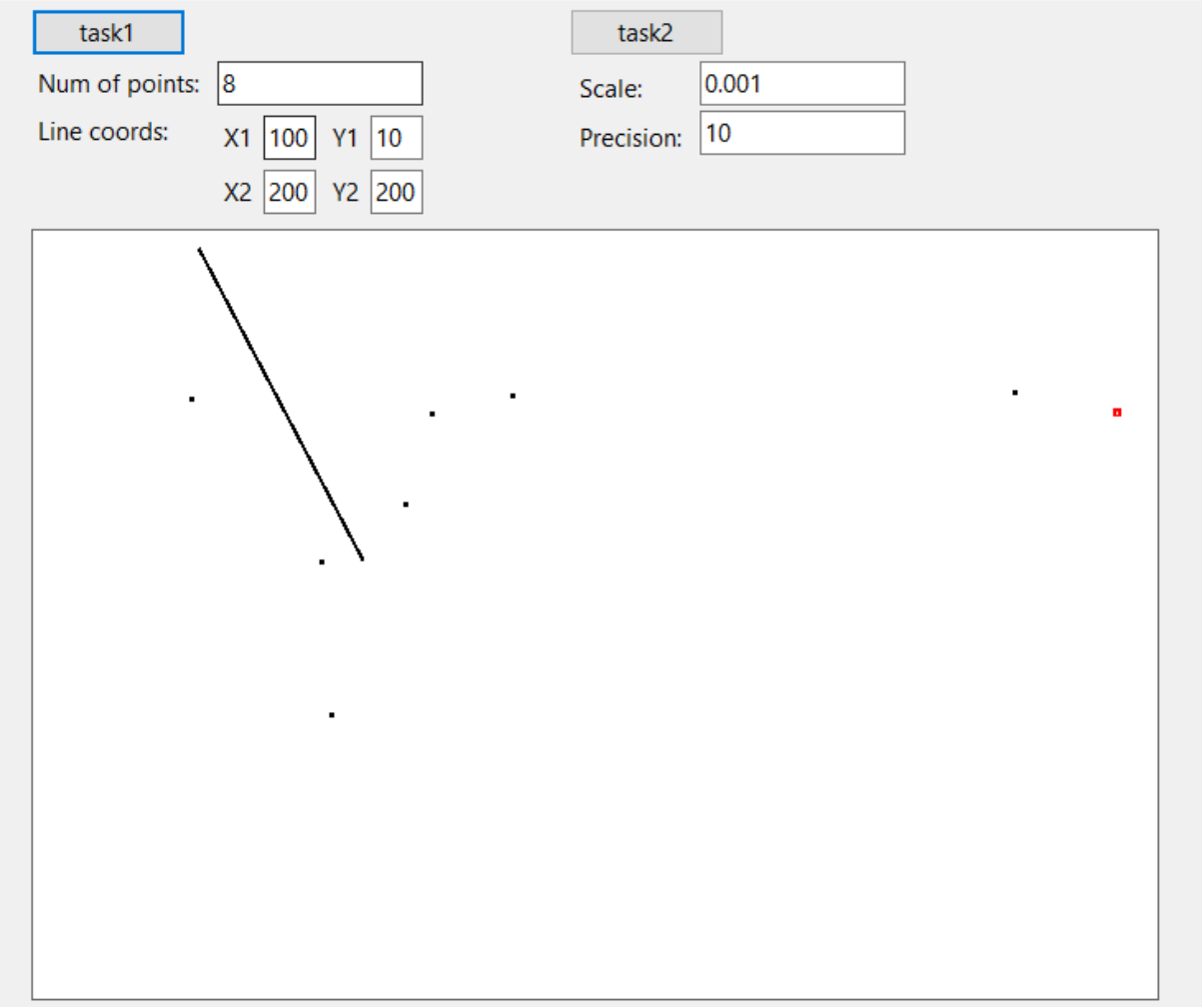
```
        private Point FurthestPoint(Line line, Point[] points)
        {
            int aCoof = line.Second.Y - line.First.Y;
            int bCoof = line.First.X - line.Second.X;
            int cCoof = line.Second.X * line.First.Y - line.First.X * line.Second.Y;
            float lendth = MathF.Sqrt(MathF.Pow(aCoof, 2) + MathF.Pow(bCoof, 2));

            Point furhest = null;
            float furhestDistance = 0, distance = 0;

            foreach (Point point in points)
            {
                distance = MathF.Abs(aCoof * point.X + bCoof * point.Y + cCoof) / lendth;
                if (distance > furhestDistance)
                {
                    furhestDistance = distance;
                    furhest = point;
                }
            }
            return furhest;
        }
    class Point
       {
        private int _x;
        private int _y;

        public int X { get => _x; }
        public int Y { get => _y; }

        public Point(int x, int y)
        {
            _x = x;
            _y = y;
        }
```

```
    public static implicit operator System.Drawing.Point(Point point)
    {
        return new System.Drawing.Point(point.X, point.Y);
    }

    public void Draw(Graphics graphics, Pen pen)
    {
        graphics.DrawRectangle(pen, new Rectangle(this, new Size(1, 1)));
    }
}

class Line
{
    private Point _first;
    private Point _second;

    internal Point First { get => _first; }
    internal Point Second { get => _second; }

    public Line(Point first, Point second)
    {
        _first = first;
        _second = second;
    }

    public void Draw(Graphics graphics, Pen pen)
    {
        graphics.DrawLine(pen, First, Second);
    }
}
```

Результаты работы программы:


task1
Num of points:
8
Line coords:
X1
100
Y1
10
X2
200
Y2
200
task2
Scale:
0.001
Precision:
10


task1
Num of points:
8
Line coords:
X1
100
Y1
10
X2
200
Y2
200
task2
Scale:
0.001
Precision:
10



Задание №2:

Реализовать построение заданного типа фрактала по варианту

Везде, где это необходимо, предусмотреть ввод параметров, влияющих на внешний вид фрактала

Бассейны Ньютона.

Итерационная формула для построения фрактала:

$$z_{k+1} = z_k - \frac{z_k^3 - 1}{3z_k^2}$$

Код программы:

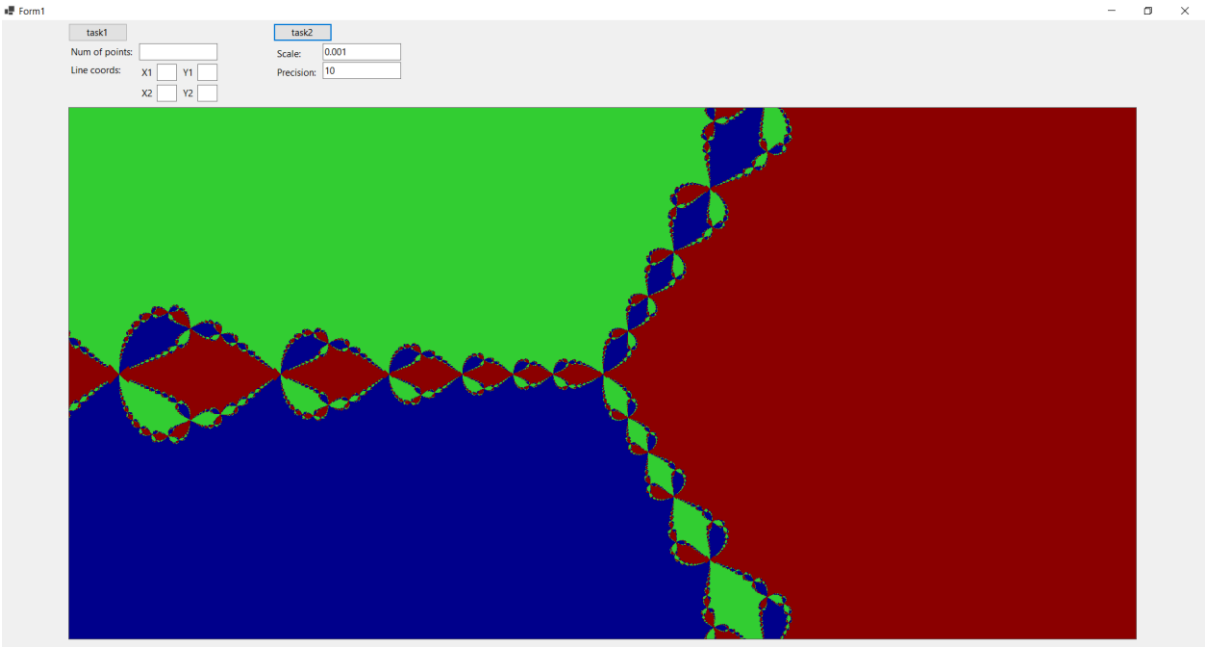
Метод для построения фрактала:

```
private void DrawFractal()
{
```

```
double fractalScale, fractalPrecision;
try
{
   fractalScale = Convert.ToDouble(scale.Text);
   fractalPrecision = Convert.ToDouble(precision.Text);
}
catch (FormatException)
{
   fractalScale = 0.01;
   fractalPrecision = 50;
}

Bitmap image = new Bitmap(drawingField.Width, drawingField.Height);

for (int i = 0; i < drawingField.Width; i++)
{
   for (int j = 0; j < drawingField.Height; j++)
   {
      double x = (i - drawingField.Width / 2) * fractalScale;
      double y = (j - drawingField.Height / 2) * fractalScale;

      Complex z = new Complex(x, y);
      int it = 0;

      do
      {
         it++;
         z = z - (Complex.Pow(z, 3) - 1) / (3 * Complex.Pow(z, 2));

         if (z.Magnitude > fractalPrecision)
            break;

      } while (it < 100);
```

```
                Color color = Color.Black;
                switch ((int)(Math.Atan2(z.Imaginary, z.Real) / (Math.PI / 2)))
                {
                    case 0:
                        color = Color.DarkRed;
                        break;
                    case 1:
                        color = Color.DarkBlue;
                        break;
                    case -1:
                        color = Color.LimeGreen;
                        break;
                }
                lock (_locker)
                {
                    image.SetPixel(i, j, color);
                }
            }
        }
        lock (_locker)
        {
            drawingField.Image = image;
        }
    }
```

Результаты работы программы:


Form1
task1
Num of points:
Line coords:
X1
Y1
X2
Y2
task2
Scale:
0,001
Precision:
10