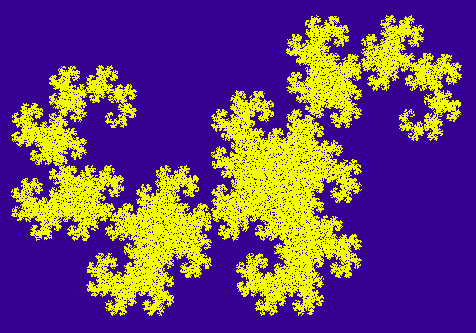
**Практика «Dragon curve»**

В этой задаче вам нужно будет нарисовать вот такую фигуру:



Вряд ли это пригодится вам в будущем, но зато красиво! :)

Подробнее про этот фрактал можете почитать, например, [в википедии](http://en.wikipedia.org/wiki/Dragon_curve).

Алгоритм построения фрактала читайте в комментариях в классе DragonFractalTask

Кстати, похожим образом можно построить ещё множество фракталов, в частности, [фрактальный папоротник](https://en.wikipedia.org/wiki/Barnsley_fern):

**Как генерировать случайные числа?**

Для этого в пространстве имен System есть класс Random. Работать с ним нужно так:

// 1. Создание нового генератора последовательности случайных чисел:

var random = new Random(seed);

// seed — число полностью определяющее все последовательность псевдослучайных чисел этого генератора.

// 2. Получение очередного псевдослучайного числа от 0 до 9:

var nextNumber = random.Next(10);

Если при инициализации генератора случайных чисел не указывать seed, то используется текущее время компьютера с точностью до миллисекунд. Поэтому если вы создадите два генератора подряд, то с большой вероятностью они проинициализируются одинаково и будут выдавать одну и ту же последовательность.

Типичная ошибка начинающих — поместить обе операции внутрь цикла, тогда как правильно вынести создание генератора за пределы цикла, оставив внутри только получение следующего числа.

// Вставьте сюда финальное содержимое файла DragonFractalTask.cs

using System;

using System.Drawing;

namespace Fractals

{

internal static class DragonFractalTask

{

public static void DrawDragonFractal(Pixels pixels, int iterationsCount, int seed)

{

var x = 1.0;

var y = 0.0;

var angle45 = Math.PI \* 45 / 180;

var angle135 = Math.PI \* 135 / 180;

var random = new Random(seed);

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

{

var nextNumber = random.Next(1, 3);

if (nextNumber == 1)

{

var x1 = DrawX(x, y, angle45);

var y1 = DrawY(x, y, angle45);

x = x1;

y = y1;

}

if (nextNumber == 2)

{

var x1 = DrawX(x, y, angle135) + 1;

var y1 = DrawY(x, y, angle135);

x = x1;

y = y1;

}

pixels.SetPixel(x, y);

}

}

public static double DrawX(double a, double b, double angle)

{

var x1 = (a \* Math.Cos(angle) - b \* Math.Sin(angle)) / Math.Sqrt(2);

return x1;

}

public static double DrawY(double a, double b, double angle)

{

var y1 = (a \* Math.Sin(angle) + b \* Math.Cos(angle)) / Math.Sqrt(2);

return y1;

}

}

}