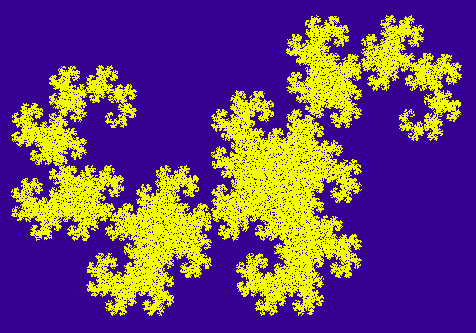
**task Practice «Dragon curve»**

In this task, you will need to draw a shape like this:



It is unlikely that this will be useful to you in the future, but it is beautiful! :)

You can read more about this fractal, for example, in Wikipedia.

Read the fractal construction algorithm in the comments in the DragonFractalTask class

By the way, many more fractals can be built in a similar way, in particular, a fractal fern:

**How to generate random numbers?**

For this, there is a Random class in the System namespace. You need to work with him like this:

// 1. Creating a new random number sequence generator:

var random = new Random(seed);

// seed - a number that completely defines the entire sequence of pseudo-random numbers of this generator.

// 2. Getting the next pseudo-random number from 0 to 9:

var nextNumber = random.Next(10);

If you do not specify seed when initializing the random number generator, then the current computer time is used with millisecond precision. Therefore, if you create two generators in a row, then with a high probability they will be initialized in the same way and will produce the same sequence.

A typical mistake for beginners is to place both operations inside the loop, while it is correct to move the generator creation outside the loop, leaving only the receipt of the next number inside.

// Paste the final file content here 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;

}

}

}