Такая организация кода, при которой одна рекурсивная функция вызывает другую рекурсивную функцию, называется **косвенной** или **взаимной рекурсией**.

Проект «Дракон Хартера–Хейтуэя»

Рассмотрим еще один проект, для выполнения которого требуется косвенная рекурсия, — «Дракон Хартера–Хейтуэя».

Генератором этого фрактала является самая простая фигура — отрезок. А процедура, с помощью которой получаются следующие фигуры, — построение на отрезке, как на гипотенузе, равностороннего прямоугольного треугольника (рис. 10.12).

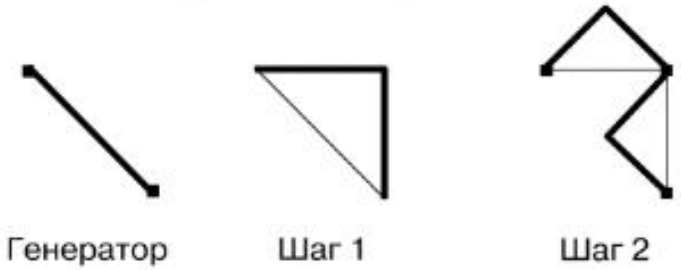


Рис. 10.12

На следующем шаге, когда отрезков становится уже два (см. рис. 10.12, шаг 1), первый треугольник строится с левой стороны от основания, а второй — с правой (здесь слова «лево» и «право» используются относительно направления обхода отрезков).

На шаге 2 отрезков становится 4, на шаге 3 — 8.

1. Определи, какая формула связывает количество отрезков ломаной **K** и номер шага **n**:

Если длину отрезка **n**-го шага обозначить x_n, то длину отрезка следующего шага x_{n+1} можно найти по теореме Пифагора (рис. 10.13).

$$x_{n+1} = \frac{x_n}{\sqrt{2}}$$

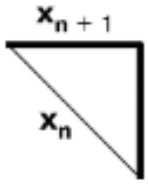


Рис. 10.13

2. Учитывая правила построения, получи (дорисуй) из фигуры шага 2 фигуру шага 3 (рис. 10.14).

Рис. 10.14

3. А теперь получи серию фигур, порождаемую противоположно ориентированным треугольником шага 1 и той же процедурой получения фигур следующих ша-

гов (первый треугольник строится слева, а второй — справа) (рис. 10.15).

Рис. 10.15

4. Проверь себя.
   - Являются ли фигуры шага 2 из разных серий подобными? Каким способом можно получить из одной фигуры другую? _____
   - Состоит ли фигура шага 3 из двух фигур шага 2? Каких именно — одноименных или разноименных? _____

Как видишь, здесь, как и в задаче построения кривой Гильберта, переплетаются противоположно ориентированные фигуры предыдущих шагов. А это значит, что без взаимной рекурсии нам не обойтись!

Договоримся: фигуры, порожденные первым треугольником (см. рис. 10.12), называть *левыми*, а фигуры, порожденные вторым треугольником (см. рис. 10.15), — *правыми*.

Проект «Дракон»

1. Создай невидимый спрайт-рисовальщик.
2. Создай блок **элемент** с числовым параметром **длина**.
3. Определи блок **элемент** так, чтобы:
   а) рисовальщик изначально был ориентирован по направлению отрезка-генератора;
   б) затем поворачивался на угол 45° и рисовал стороны порождаемого им левого равностороннего прямоугольного треугольника (длина основания равна значению параметра **длина**);

в) возвращался в исходную точку, не оставляя следов.

4. Создай запускающий блок в соответствии со следующими требованиями:

а) рисовальщик должен стартовать из центра в горизонтальном направлении;

б) сцена должна быть очищена от предыдущей графики;

в) вызывается блок **элемент** с таким значением числового параметра, чтобы получившийся рисунок целиком умещался на сцене.

5. Запусти проект и убедись, что в результате рисуется прямой угол (углом вверх).

6. Создай еще один блок — **отражение** с числовым параметром **длина**.

7. Определи блок **отражение** так же, как блок **элемент**, поменяв в нем углы поворота на противоположные.

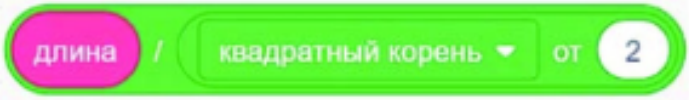
8. Поменяй в запускающем блоке вызов команды **элемент** на вызов команды **отражение** и запусти проект.

9. Проверь: получился ли в результате правый прямой угол? Если нет, устрани ошибки.

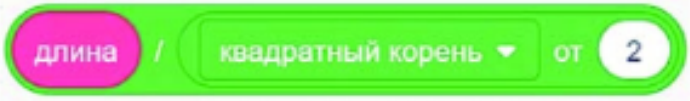
Теперь нам надо повторить команды блока **элемент** для построения отрезков следующего шага.

Чтобы нам было легче осуществлять пошаговую проверку работы кода, свяжи цвет пера со значением параметра **длина**.

10. Скопируй содержимое блока **элемент**.

11. Поскольку длина отрезка следующего шага связана с длиной данного отрезка по формуле (2), замени в командах скопированной части значение параметра **длина** на выражение 

.

12. Вставь в скопированную часть после первой команды поворота вызов блока **элемент**, а после второй — вызов блока **отражение** со значением параметра, равным 

.

13. Соедини оба фрагмента блока **элемент**.

14. Вложи получившийся код в конструкцию с условием, ограничивающим рекурсию заданным значением параметра **длина**.

15. Каким должно быть это условие, чтобы код остановился после второго рекурсивного вызова?

---

16. Верни в запускающем блоке **вызов** команды **элемент** и проверь:

а) произошел ли рекурсивный вызов блоков **элемент** и **отражение**;

б) остановился ли код после второго рекурсивного вызова;

в) получилась ли в результате фигура шага 2 (сравни с рис. 10.14)?

17. Если не выполнилось:

а) первое условие, уменьши значение, ограничивающее рекурсию параметром **длина** (см. п. 15);

б) второе условие, увеличь значение, ограничивающее рекурсию параметром **длина** (см. п. 15);

в) третье условие, найди ошибки в самом коде и исправь их.

18. Если фигура второго шага в результате работы кода нарисована верно, то:

а) скопируй фрагмент блока **элемент**, отвечающий за рекурсивные вызовы;

б) замени в нем все углы поворота на противоположные;

в) добавь получившийся фрагмент к командам блока **отражение**;

г) вложи содержимое блока **отражение** в условие, аналогичное условию выполнения блока **элемент**.

19. Замени в запускающем блоке вызов команды **элемент** на вызов команды **отражение** и проверь, получилась ли правая фигура второго шага. Если нет, найди и устрани ошибки.

20. Меняя значение, ограничивающее рекурсию в обоих блоках, начальное значение параметра **длина** и начальное положение спрайта-рисовальщика, получи фигуру максимально возможного шага (рис. 10.16).

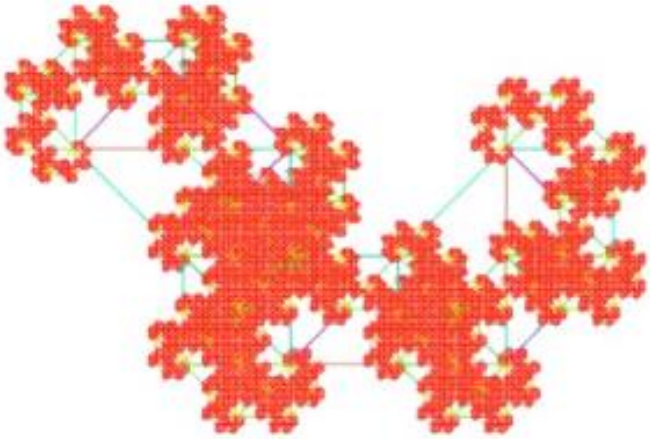


Рис. 10.16

21. Как ты думаешь, можно ли изменить код так, чтобы вспомогательные построения оказались скрыты? Если нет, то почему? Если да, то как именно?

---

22. Если ты утвердительно ответил на предыдущий вопрос, то внеси необходимые изменения в код и проверь свои предположения.

23. Сохрани проект.

Задания для самостоятельной работы

2. Создай копию проекта «Дракон».
Измени код так, чтобы все построенные треугольники были одинаково ориентированы.
Такая фигура получила название *кривой Леви* (рис. 10.18).