МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«ИРКУТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ИГУ»)

Институт математики и информационных технологий
Кафедра вычислительной математики и оптимизации

КУРСОВАЯ РАБОТА

по направлению «Прикладная математика и информатика»
профиль «Математические методы и информационные технологии»

Студента 3 курса группы 02321-ДБ
направления 01.03.02 Прикладная
математика и информатика
Павлов Данила Сергеевич

Руководитель: к. ф.-м. н., доцент
_____ Черкашин Е. А.

Иркутск – 2022

Содержание

Введение

В данной курсовой работе мы рассмотрим язык программирования Haskell и его применение в создании фракталов. Haskell является функциональным языком программирования, который характеризуется строгой типизацией, легкостью написания чистых и безопасных программ, а также эффективностью выполнения.

Фракталы - это геометрические объекты, которые имеют сложную структуру и неограниченную подробность при любом масштабе. Они могут быть описаны математически и визуализированы на компьютере с помощью алгоритмов генерации фракталов. В этой работе мы рассмотрим различные алгоритмы генерации фракталов и реализуем их с помощью Haskell.

Целью нашей курсовой работы является изучение языка Haskell и возможностей, которые он предоставляет для реализации фракталов. Мы также намерены рассмотреть различные алгоритмы и методы, которые можно использовать для создания фракталов, а также сравнить эффективность различных подходов.

Основной задачей нашей работы является разработка программы на языке Haskell, которая способна создавать фракталы различных типов, например, Коха, Серпинского и т.д. Мы также планируем исследовать способ визуализации фракталов с использованием библиотеки Graphics.Gloss.

Фрактальная линия Коха

Одним из самых известных фракталов является кривая Коха, которая была изобретена в 1884 году французским математиком Пьером Коха. Давайте разберем, как построить фрактал на основе кривой Коха и какие уникальные свойства он обладает.

Этот код реализует рекурсивную функцию drawKoch, которая рисует фрактальную линию Коха. Фрактальная линия Коха - это кривая, которая строится путем рекурсивного деления отрезка на три части и построения треугольника на основе этих трех частей.

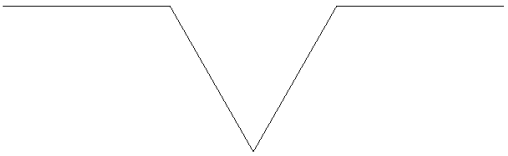
Функция drawKoch принимает три аргумента:

- (xa, ya) - координаты начала отрезка.
- (xe, ye) - координаты конца отрезка.
- i - уровень рекурсии

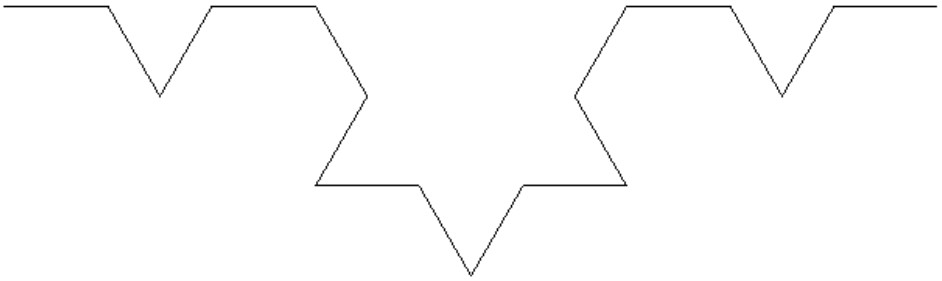
```
import Graphics.Gloss
-- Основная функция, отвечающая за отрисовку окна с изображением
main :: IO ()
main = display window background picture
  where
    window = InWindow "Koch curve" (600, 200) (10, 10)
-- Создание окна с заданными размерами и координатами верхнего левого угла
    background = white   -- Установка белого цвета фона
    picture = drawKoch (0, 0) (600, 0) 1--i
-- Отрисовка Коха с начальными координатами (0,0) и (600,0) и уровнем рекурсии
1 (i)
drawKoch :: Point -> Point -> Int -> Picture -- Функция отрисовки Коха
drawKoch (xa, ya) (xe, ye) i
-- Если уровень рекурсии равен 0, то отрисовываем прямую линию
  | i == 0 = Line [(xa, ya), (xe, ye)]
  | otherwise =
    let
        xb = xa + (xe - xa) / 3 -- Вычисляем координаты точки B
        yb = ya + (ye - ya) / 3
        xd = xa + (xe - xa) * 2 / 3 -- Вычисляем координаты точки D
        yd = ya + (ye - ya) * 2 / 3
        cos60 = 0.5 -- Вычисляем косинус 60 градусов
        sin60 = -0.866 -- Вычисляем синус 60 градусов
        xc = xb + (xd - xb) * cos60 - sin60 * (yd - yb) -- Вычисляем координаты точки
C
        yc = yb + (xd - xb) * sin60 + cos60 * (yd - yb)
```

```
in  Pictures -- Отрисовывание путем вызова функции  drawKoch
      [ drawKoch (xa, ya) (xb, yb) (i - 1)
      , drawKoch (xb, yb) (xc, yc) (i - 1)
      , drawKoch (xc, yc) (xd, yd) (i - 1)
      , drawKoch (xd, yd) (xe, ye) (i - 1)]
```

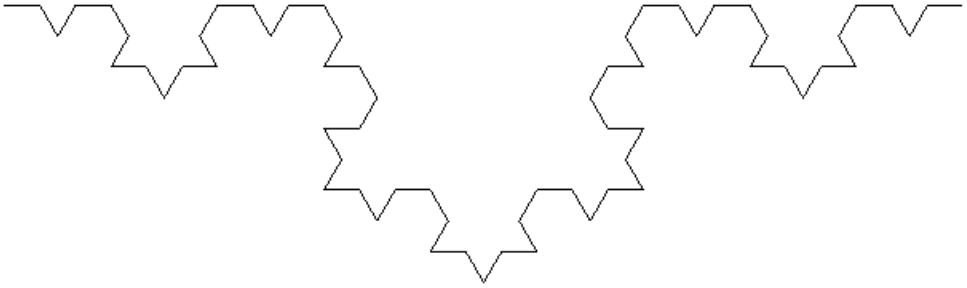

i=0



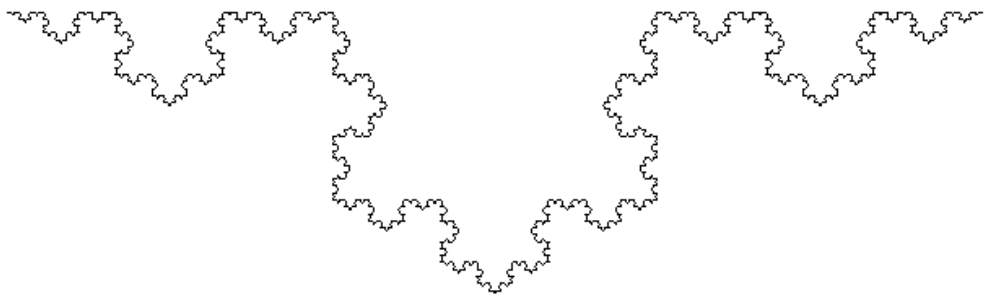
i=1



i=2



i=3



i=5

Треугольная салфетка Серпинского

Треугольная салфетка Серпинского - это фрактал, который строится на основе рекурсивного деления треугольника на четыре меньших треугольника. Обычно треугольная салфетка Серпинского строится с использованием треугольника с равнобедренными сторонами, но она также может быть строиться с использованием треугольника с равносторонними сторонами.

Этот код реализует рекурсивную функцию serpinskiTriangle, которая рисует треугольную салфетку Серпинского. Она принимает два аргумента:

- i - размер треугольника (сторона).
- (x, y) - координаты верхнего левого угла треугольника.

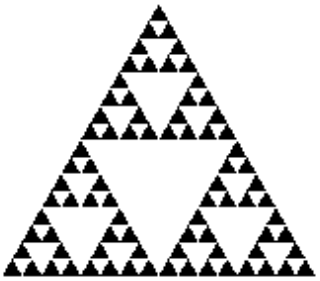
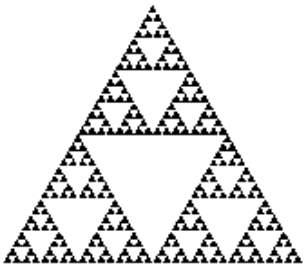
Функция рекурсивно вызывает себя для трех меньших треугольников, расположенных внутри текущего треугольника, с использованием координат (x, y), (x + i/2, y) и (x + i/4, y + i * sqrt 3 / 4). Когда размер треугольника становится меньше или равен 1, треугольник рисуется с помощью функции polygon.

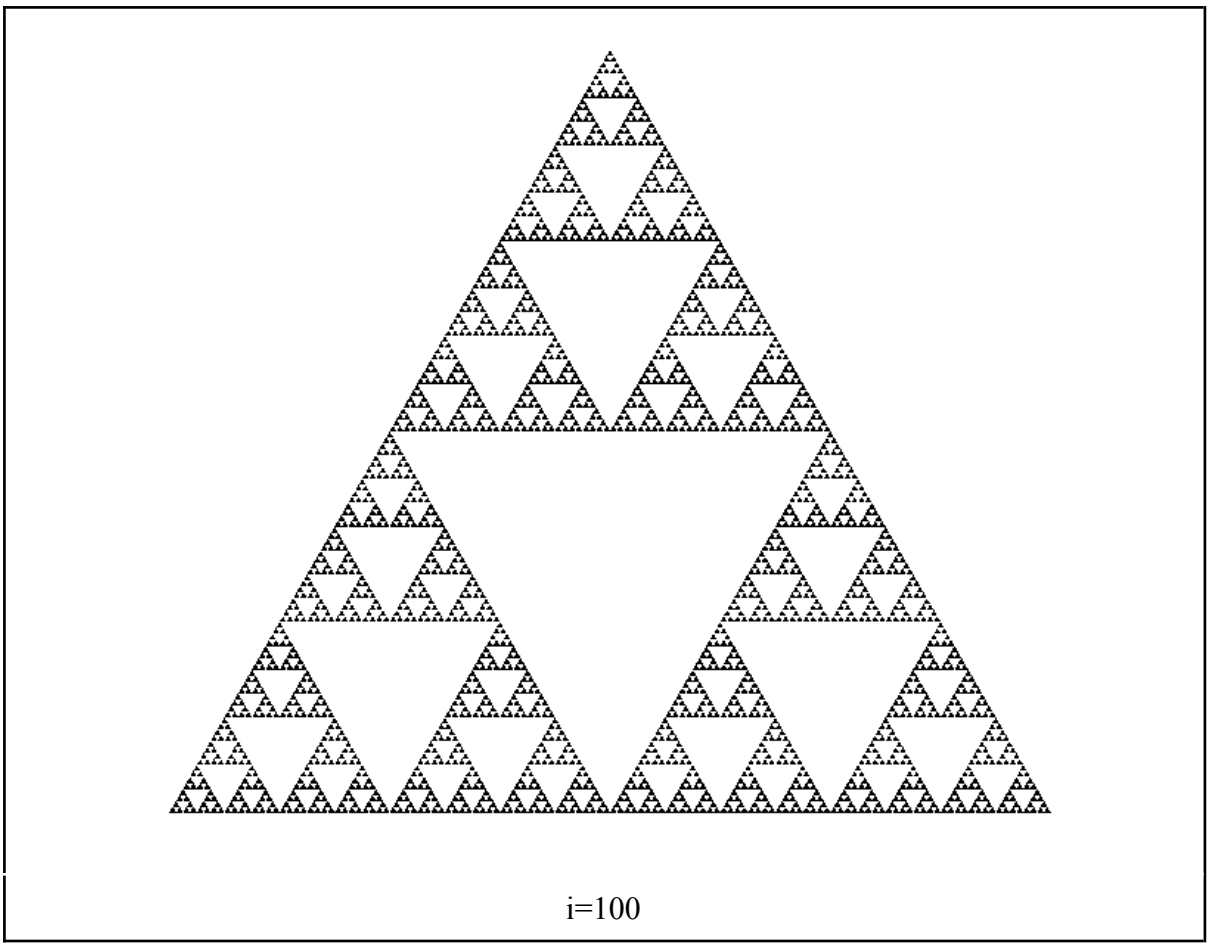
```
import Graphics.Gloss

main :: IO ()
main = display window background drawing
  where
    window = InWindow "Serpinski Triangle" (400, 400) (10, 10)
    background = white --  фон, который будет белым
    drawing = pictures [serpinskiTriangle 3 (0, 0)] -- рисунок, который мы получаем
из функции serpinskiTriangle

-- Функция, рисующая Серпинского треугольник
serpinskiTriangle :: Float -> (Float, Float) -> Picture -- Если размер треугольника
```

```
меньше или равен 1, то рисуем треугольник
serpinskiTriangle i (x, y)
  | i <= 1 = color black (polygon [(x, y), (x + i, y), (x + i / 2, y + i * sqrt 3 / 2)])
  | otherwise = pictures -- Иначе рекурсивно рисуем три меньших треугольника
      [ serpinskiTriangle (i / 2) (x, y)
      , serpinskiTriangle (i / 2) (x + i / 2, y)
      , serpinskiTriangle (i / 2) (x + i / 4, y + i * sqrt 3 / 4)]
```

		
i = 1	i = 2	i = 3
		
i=5	i=10	i=20



Ковер Серпинского

Ковер Серпинского - это геометрическая фигура, которая состоит из нескольких треугольников, соединенных вместе. Он был создан в 1903 году русским математиком Георгием Серпинским (Георг Серпинский).

Этот код рисует так называемый "ковер Серпинского". Он состоит из рекурсивной функции sierpinskiCarpet, которая рисует картину, а также функции main, которая отображает результат рисования.

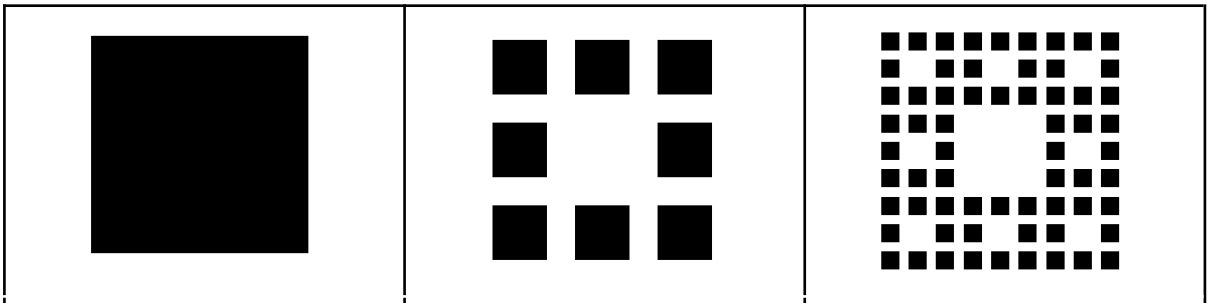
Функция sierpinskiCarpet принимает два аргумента:

- i - это уровень рекурсии. Чем больше n, тем больше деталей будет в изображении.
- size - это размер квадрата, из которого строится ковер Серпинского.

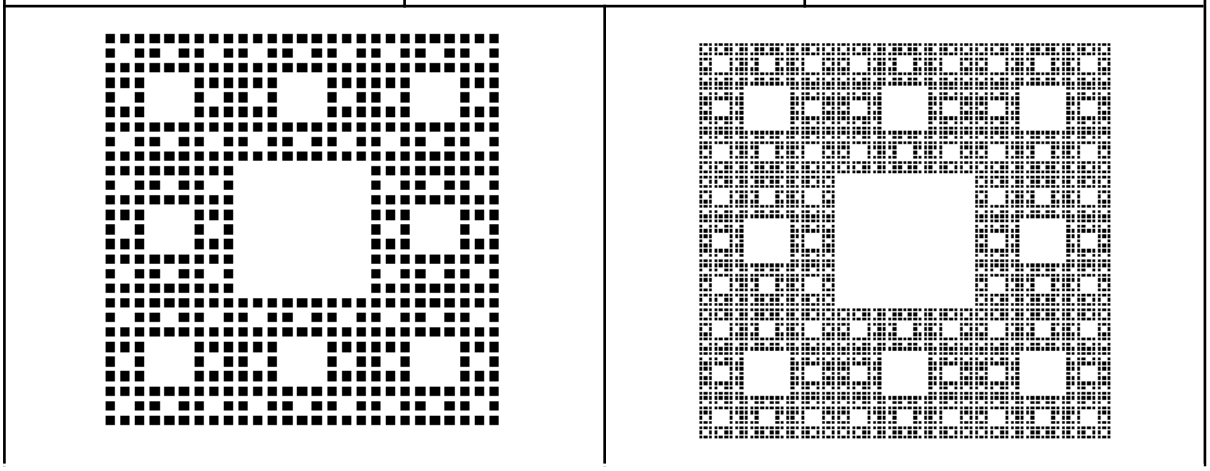
Каждый раз, когда функция вызывается с более низким уровнем рекурсии n, она рисует квадрат со стороной subSize, где subSize = size / 3, в центре каждого из восьми квадратов размера size, которые в свою очередь размещены в центре каждого из восьми квадратов размера size. Этот процесс повторяется, пока уровень рекурсии n не станет равным нулю.

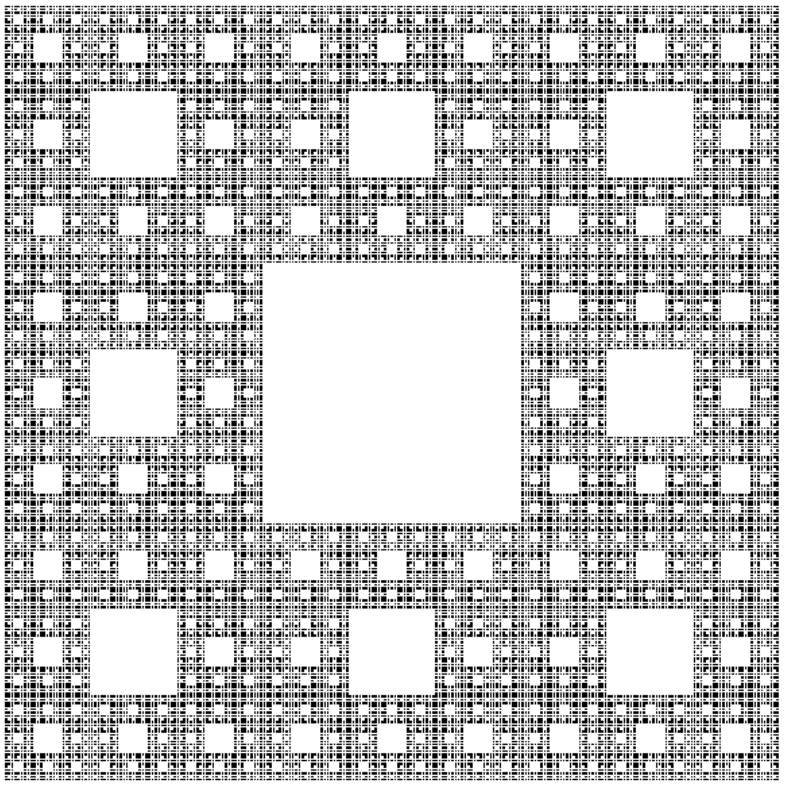
```
import Graphics.Gloss

sierpinskiCarpet :: Int -> Float -> Picture
sierpinskiCarpet 0 size = rectangleSolid size size -- Если n равно 0, то рисуем квадрат
-- Иначе рисуем список картинок, каждую из которых смещаем на
-- x и y с помощью функции translate, где x и y проходят через
-- все возможные значения из списка [-size/2, 0, size/2],
 -- a subSize равно size / 3
sierpinskiCarpet n size = Pictures [translate x y (sierpinskiCarpet (n-1) subSize) |
                            x <- [-size/2, 0, size/2],
                            y <- [-size/2, 0, size/2],
                            x /= 0 || y /= 0]
  where subSize = size / 3
main :: IO ()
main = display window background picture
  where
    window = InWindow "Sierpinski Carpet" (400, 400) (10, 10)
    background = white
    picture = sierpinskiCarpet 5 200
```



i=0 | i=1 | i=2



i=3	i=4
	
i=5	

Фрактальное дерево

Фрактальное дерево - это геометрическая фигура, которая похожа на дерево, но состоит из множества меньших копий самого себя. Оно часто используется в качестве примера фрактальной фигуры, так как оно имеет свойство самоподобия - то есть, его части похожи на целое структуру.

Этот код рисует фрактальное дерево. Он состоит из рекурсивной функции fractalTree, которая рисует фрактальное дерево, а также функции main, которая отображает результат рисования.

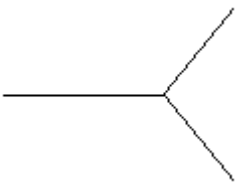
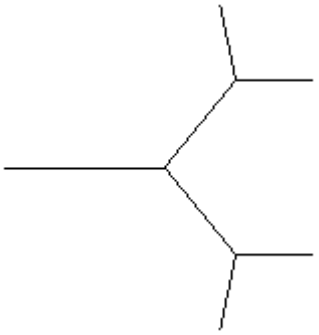
Функция fractalTree принимает следующие аргументы:

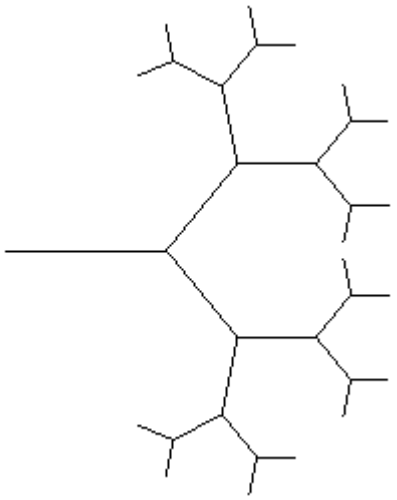
- i - это уровень рекурсии. Чем больше i, тем больше деталей будет в изображении.
- (x, y) - это координаты начала линии.
- angle - это угол, на который нужно повернуться от начальной линии.

- len - это длина линии.

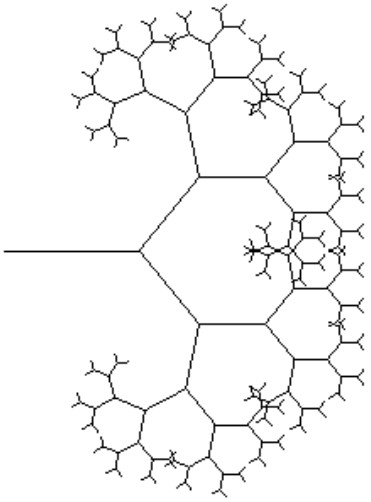
Каждый раз, когда функция вызывается с более низким уровнем рекурсии i, она рисует две линии, каждая из которых повернута на угол angle - 70 и angle + 70 соответственно, а затем рекурсивно вызывает саму себя с уровнем рекурсии i - 1, начальной точкой в конце каждой из этих линий, и новыми значениями.

```
import Graphics.Gloss
import Graphics.Gloss.Data.ViewPort
fractalTree :: Int -> Point -> Float  -> Float -> Picture
-- Если i равно 0, то рисуем пустую картинку
fractalTree 0 _ _ _ = blank
-- Иначе рисуем линию от точки (x, y) до точки (x', y')
-- и рекурсивно вызываем функцию для рисования двух дочерних деревьев
fractalTree i (x, y) angle len = pictures
  [ line [(x, y), (x', y')]
  , fractalTree (i - 1) (x', y') (angle - 70) (len * 0.7)
  , fractalTree (i - 1) (x', y') (angle + 70) (len * 0.7)
  ]
  where
    -- Координаты точки (x', y') рассчитываются по углу angle и длине len
    x' = x + len * cos angle
    y' = y + len * sin angle
exampleTree :: Picture
exampleTree = fractalTree 15 (0, 0) (0) 80
main :: IO ()
main = display (InWindow "Fractal Tree" (400, 400) (10, 10)) white exampleTree
```

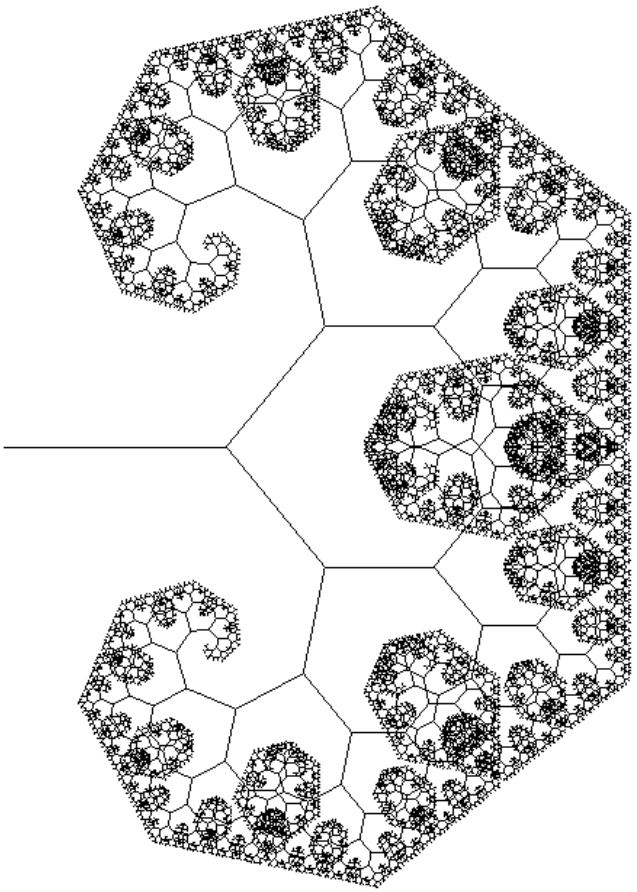
		
i = 1	i = 2	i = 3



i=5



i = 8



i = 15

Заключение

В курсовой работе был рассмотрен вопрос создания фрактальных изображений с помощью языка Haskell и библиотеки Gloss. Были рассмотрены различные алгоритмы создания фракталов, такие как фрактальная линия Коха, “ковер” и “салфетка” Серпинского и фрактальное дерево. Было показано, как с помощью языка Haskell и библиотеки Gloss можно удобно реализовывать эти алгоритмы и визуализировать результаты.

В ходе работы были изучены основные принципы работы с языком Haskell, а также библиотекой gloss, что позволило создать графически привлекательное приложение.

Результатом работы стали прогр

аммы, которые позволяют визуализировать фракталы и изучить их свойства, такие как структура и геометрические формы.

Список Литературы

1. Миран Липовача. Изучай Haskell во имя добра! / Пер. с англ. Леушина Д., Синицына А.,

Арсанукаева Я.– М.: ДМК Пресс, 2012. – 490 с.: ил. ISBN 978-5-94074-749-9

2. Г.М. Сергиевский, Н.Г. Волченков. Функциональное и логическое программирование. –

М.: Академия, 2010. – 320 с.: ил. ISBN: 978-5-7695-6433-8

3. Уилл Курт. Программируй на Haskell / пер. с англ. Я. О. Касюлевича, А. А. Романовского

и С. Д. Степаненко; под ред. В. Н. Брагилевского. – М.: ДМК Пресс, 2019. — 648 с.: ил.

ISBN 978-5-97060-694-0

4. Daniel Shiffman. The Nature of Code; 1st edition: The Nature of Code, 2012. – 520p.