КУРСОВАЯ РАБОТА  
по дисциплине   
«Языки и средства функционального программирования»

А.А. Лукашин

Руководитель

В.А. Лукина

Выполнил студент   
гр. 3530904/80001

# Задание

Реализовать построение множества Мандельброта и представления его в виде картинки.

# Решение

Для решения поставленной задачи были подключены необходимые библиотеки, написан класс Complex для работы с комплексными числами, написана функция mandelbort для перебора точек, каждая из которых проверяется функцией haslim на наличие предела. Далее полученные точки записываются в список и закрашиваются на графике.

# Скриншоты

Изображение выглядит как устройство, датчик

Автоматически созданное описание

# Изображение выглядит как текст, карта Автоматически созданное описание

Результат работы программы

# Выводы

Реализовано построение множества Мандельброта и получено его представление в виде картинки.

# Приложение

package main

import scalax.chart.api.\_

import scala.collection.mutable.ListBuffer

/////////////////////////////////

class Complex(val real: Double, val image: Double) {

override def toString = real + (if (image < 0) "-" + -image else "+" + image) + "\*i"

def +(rhs: Complex) = new Complex(rhs.real + real, rhs.image + image)

def \*(rhs: Complex) = new Complex(real \* rhs.real - image \* rhs.image, image \* rhs.real + real \* rhs.image)

def module: Double = real \* real + image \* image

}

/////////////////////////////////

object Main {

def main(args: Array[String]) {

val z = new Complex(0, 0)

val data = mandelbort(ListBuffer())

val title: String = "Mandelbort"

val chart = XYLineChart(data)

chart.plot.setRenderer(new org.jfree.chart.renderer.xy.XYLineAndShapeRenderer(false, true))

chart.saveAsPNG("mandelbort.png")

}

/////////////////////////////////

def mandelbort(container: ListBuffer[(Double, Double)]): List[(Double, Double)] = {

for (re <- -3.0 until 3.0 by 0.0005)

for (im <- 3.0 until -3.0 by -0.0005)

if (haslim(new Complex(0.0, 0.0), new Complex(re, im), 0))

container += ((re, im))

container.toList

}

def haslim(z: Complex, c: Complex, count: Int): Boolean =

{

if (z.module < 4 && count < 100)

haslim(z \* z + c, c, count + 1)

else if (count == 100)

true

else

false

}

}