КУРСОВАЯ РАБОТА

по дисциплине «Функциональное программирование»

Выполнил: студент гр. 3530904/80003

Афанасьев А.А.

Преподаватель: Лукашин А.А.

Санкт-Петербург 2019 г.

Оглавление

[1. Задача 3](#_Toc27518901)

[2. Ход работы 3](#_Toc27518902)

[3. Скриншоты 5](#_Toc27518903)

[4. Выводы 5](#_Toc27518904)

# **Задача**

Реализовать построение множества Мандельброта и представления его в виде картинки.

# **Ход работы**

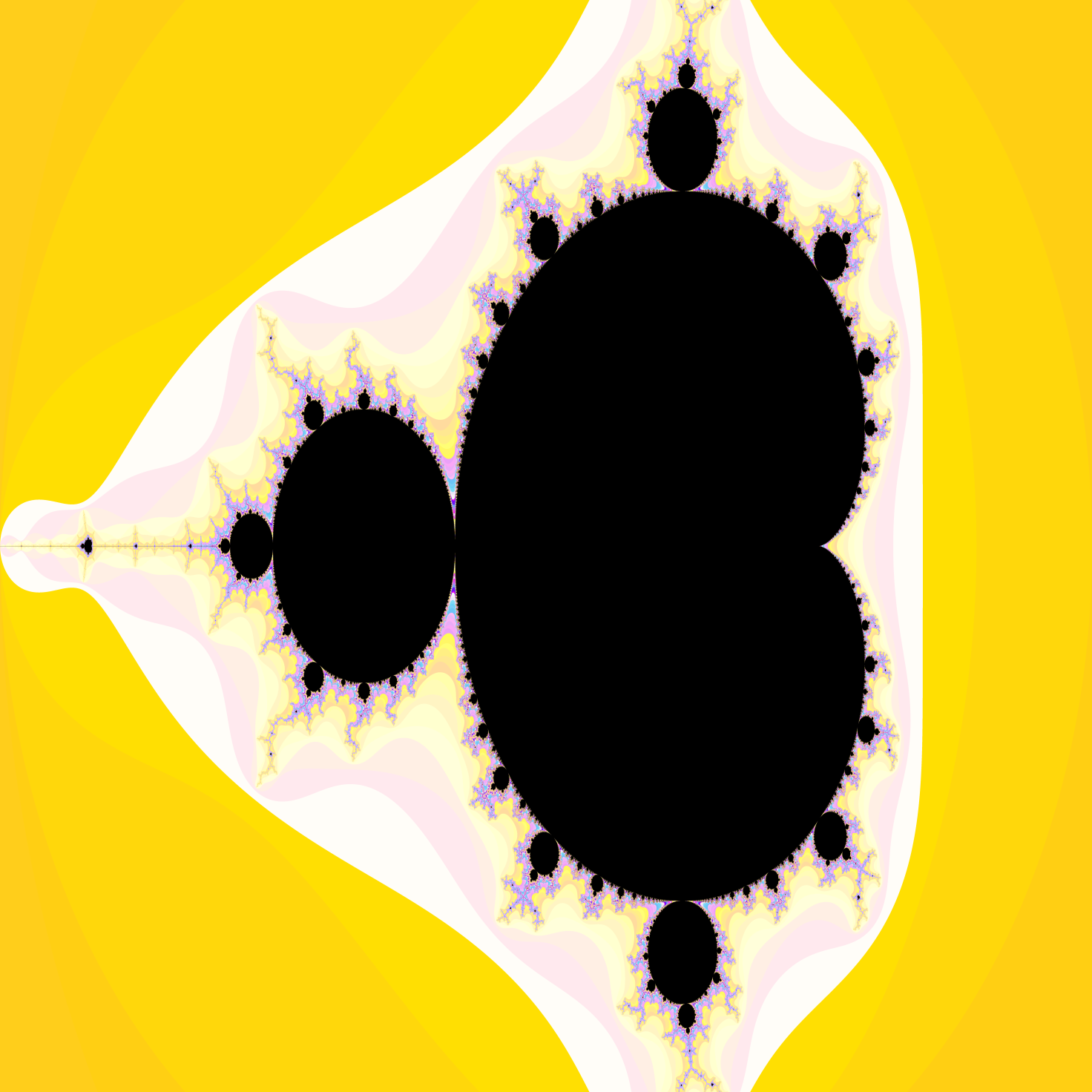
Реализован метод определения количества итераций входа/выхода точки из круга. Для отрисовки изображения были использованы библиотеки Java:

package common  
  
import java.awt.Color  
import java.awt.image.BufferedImage  
import java.io.File  
//import java.lang.reflect.Array  
  
import javax.imageio.ImageIO  
import common.Complex.\_  
  
object Solution {  
  
 def main(args: Array[String]): Unit = {  
  
 val width = 10000  
 val height = 10000  
 val out = new BufferedImage(width, height, BufferedImage.*TYPE\_INT\_RGB*)  
  
 (0 until width).par.foreach { x =>  
 (0 until height).par.foreach { y =>  
 val c = *pixelToMandelbrotComplex*(x, y, width, height)  
  
 val iterations = Complex.*divergentIterations*(c)  
  
 out.setRGB(x, y, Color.*HSBtoRGB*(iterations / 130.0f, 156, iterations / (iterations + 8f)))  
 }  
 }  
  
 ImageIO.*write*(out, "png", new File("set.png"))  
 }  
  
 def pixelToMandelbrotComplex(x: Int, y: Int, width: Int, height: Int): Complex = {  
  
 val real = (x \* 3.0 / width) - 2.0  
 val imaginary = (y \* 2.0 / height) - 1.0  
  
 *Complex*(real, imaginary)  
 }  
}

Создан класс комплексных чисел и определены необходимые методы операций с ними:

package common  
  
object Complex {  
 def abs(complex: Complex): Double = scala.math.*sqrt*(complex.real \* complex.real + complex.imaginary \* complex.imaginary)  
  
 def divergentIterations(c: Complex, z: Complex = *Complex*(0, 0), iterations: Int = 256, absLimit: Double = 2): Int = {  
 val z2 = z \* z + c  
  
 if (*abs*(z2) > absLimit || iterations == 0) {  
 iterations  
 } else {  
 *divergentIterations*(c, z2, iterations - 1, absLimit)  
 }  
 }  
}  
  
case class Complex(real: Double, imaginary: Double) {  
 def +(that: Complex) = Complex(real + that.real, imaginary + that.imaginary)  
  
 def \*(that: Complex) = Complex(real \* that.real - imaginary \* that.imaginary, real \* that.imaginary + imaginary \* that.real)  
  
 override def toString: String = s"**$**real + **$**{imaginary}i"  
}

# **Скриншоты**



# **Выводы**

В ходе выполнения работы были изучены некоторые алгоритмы функционального программирования и подходы к программирования на Scala, а также получены навыки работы с библиотеками Java.