КУРСОВАЯ РАБОТА

по дисциплине «Языки и средства функционального программирования»

Построение множества Мандельброта

Выполнил студент гр. 3530904/80006 Поддубный А.А.

Руководитель Лукашин А.А.

Оглавление

[Описание задачи 3](#_Toc27679934)

[Описание решения 3](#_Toc27679935)

[Результат выполнения 4](#_Toc27679936)

[Вывод 5](#_Toc27679937)

# Описание задачи

Реализовать построение множества Мандельброта и представления его в виде картинки (Без использования actors).

# Описание решения

Был создан класс комплексного числа с описанием действий над ними:

object Complex {  
 case class Complex(var re: Double, var im: Double) {  
 def modulus: Double = *sqrt*(*pow*(re, 2) + *pow*(im, 2))  
 def +(c: Complex) = new Complex(re + c.re, im + c.im)  
 def \*(c: Complex) = new Complex(re \* c.re - im \* c.im, im \* c.re + re \* c.im)  
 }  
}

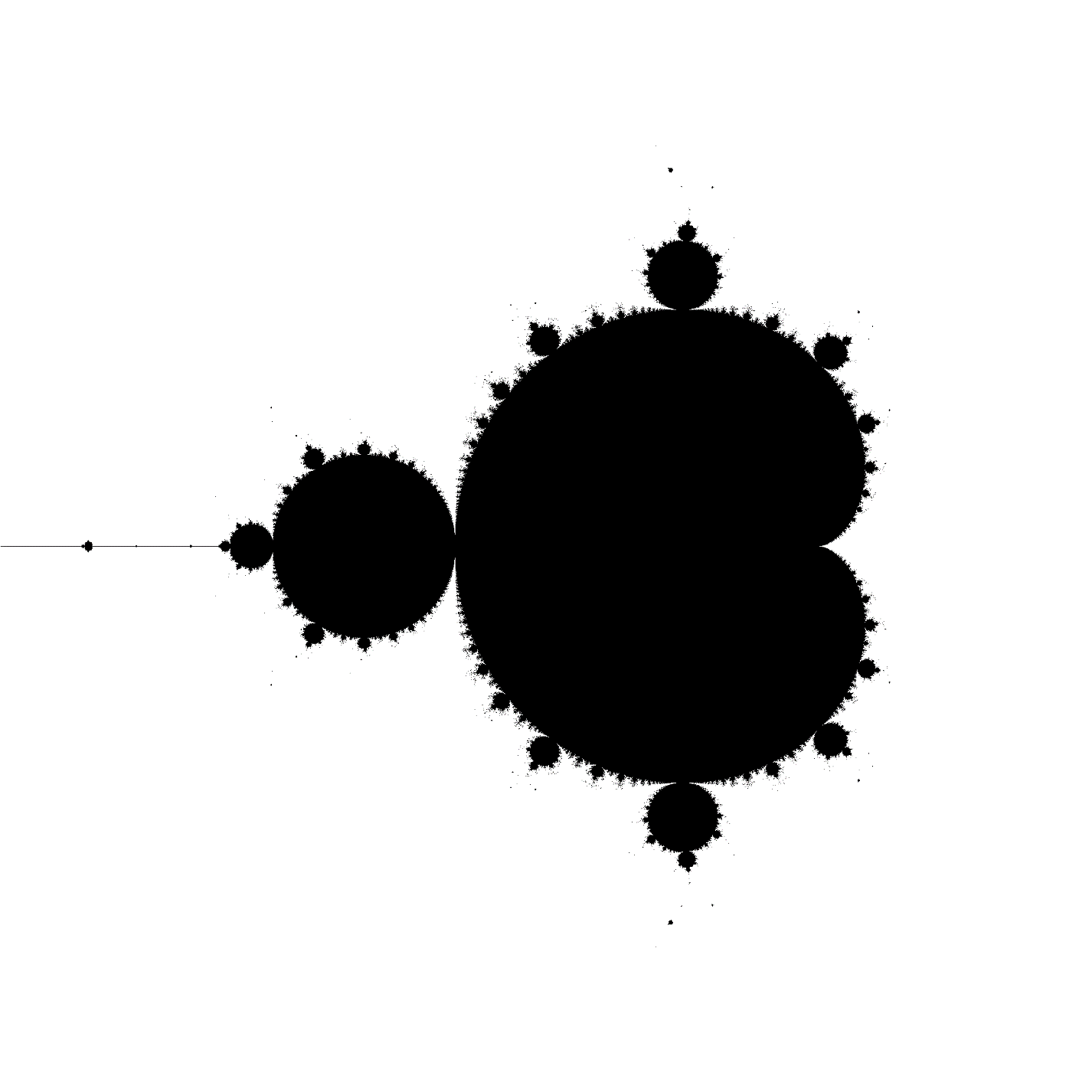
Для построения графического изображения множества Мандельброта был использован алгоритм escape-time. Доказано, что всё множество целиком расположено внутри круга радиуса 2 на плоскости. Поэтому будем считать, что если для точки *c* последовательность итераций функции *fc* = z2 + *c* с начальным значением *z* = 0 после некоторого числа *N* не вышла за пределы этого круга, то точка принадлежит множеству и красится в черный цвет. Соответственно, если на каком-то этапе, меньшем *N*, элемент последовательности по модулю стал больше 2, то точка множеству не принадлежит и остается белой.

for (i <- 0 until level)  
 z = z \* z + c  
 if (z.modulus < 2)  
 png.setRGB(x0, y0, *white*);  
 else  
 png.setRGB(x0, y0, *black*)  
}

main принимает некоторое число N для выполнения алгоритма и сторону квадратного изображения множества Мандельброта (в пикселях)

def main(args: Array[String]) {

# Результат выполнения



# Вывод

В ходе выполнения курсовой работы были освоены навыки работы с функциональным языком программирования Scala и библиотеками Java, необходимыми для вывода результата в виде картинки.