**КУРСОВАЯ РАБОТА**

**Реализация построения множества Мандельброта и представление его в виде картинки**

по дисциплине «Языки и средства функционального программирования»

Выполнил

студент гр. 3530904/80002 Моисеева А.С.

Руководитель Лукашин А. А.

"22" декабря 2019 г.

Оглавление

[Описание задачи 3](#_Toc28015800)

[Описание решения 3](#_Toc28015801)

[Скриншот работы программы 4](#_Toc28015802)

[Вывод 5](#_Toc28015803)

[Приложение 6](#_Toc28015804)

# Описание задачи

Реализовать построение множества Мандельброта и представления его в виде картинки.

# Описание решения

//Создала класс комплексного числа и описала необходимые действия для работы с ним

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

def +(c : Complex) = new Complex(real + c.real, imaginary + c.imaginary)

def \*(c : Complex) = new Complex(real \* c.real - imaginary \* c.imaginary, real \* c.imaginary + imaginary \* c.real)

def abs() = real\*real + imaginary\*imaginary

}

//Множество Мандельброта находится **внутри** круга радиусом=2 на комплексной плоскости, при выходе из которого точка уже не возвращается.

//Поэтому определяем количество итераций, спустя которое точка останется в кругу или выйдет из него с помощью следующей функции:

def steps(c : Complex, max : Int): Int = {

var newNumber = new Complex (0, 0)

for (i <- 0 to max) {

newNumber = newNumber\* newNumber + c

if (newNumber.abs > 4) {

return i

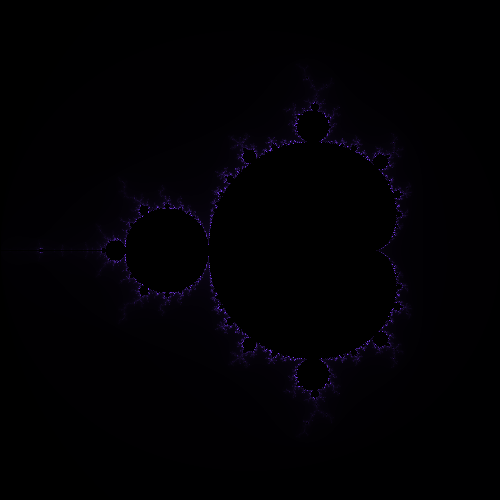
}

}

return 0;

}

# Скриншот работы программы

a

# Вывод

В ходе выполнения данной курсовой работы были получены навыки работы с языком функционального программирования scala и навыки работы с библиотеками для вывода изображения из языка java

Были выполнены следующие задания:

1. Реализована функция, определяющая количество итераций для каждой точки;
2. Реализован вывод изображения множества Мандельброта

# Приложение

Код программы

**Main.scala**

|  |
| --- |
|  |
|  | import javax.awt.\_  import javax.imageio.\_ |
|  | import java.io.File |
|  | import java.awt.image.BufferedImage |
|  |  |
|  | class Complex (val real : Double, val imaginary : Double) { |
|  | def +(c : Complex) = new Complex(real + c.real, imaginary + c.imaginary) |
|  | def \*(c : Complex) = new Complex(real \* c.real - imaginary \* c.imaginary, real \* c.imaginary + imaginary \* c.real) |
|  | def abs() = real\*real + imaginary\*imaginary |
|  | } |
|  |  |
|  | object Main { |
|  | def steps(c : Complex, max : Int): Int = { |
|  | var newNumber = new Complex(0, 0) |
|  | for (i <- 0 to max) { |
|  | newNumber = newNumber\*newNumber + c |
|  | if (newNumber.abs > 4) { |
|  | return i |
|  | } |
|  | } |
|  | return 0 |
|  | } |
|  |  |
|  | def main(args: Array[String]) { |
|  | val w = 500 |
|  | val h = 500 |
|  |  |
|  | val outputFile = new BufferedImage(w, h, BufferedImage.TYPE\_INT\_RGB); |
|  | val graphic = outputFile.getGraphics(); |
|  |  |
|  | //Умножаем х и у на константы для перевода пикселей в координаты для проведения расчётов |
|  | for (y <- 0 to h) { |
|  | for (x <- 0 to w) { |
|  | val c = new Complex((x\*3.0)/w - 2.0, (y\*3.0)/h - 1.5) |
|  | // Определение цвета точек |
|  | graphic.setColor(new Color(steps(c, 255)/2, steps(c, 255)/4, steps(c, 255))) |
|  | // Прорисовка точки как прямой с концами в одной точке |
|  | graphic.drawLine(x,y,x,y) |
|  | } |
|  | } |
|  | ImageIO.write(outputFile, "png", new File("Mandelbrot.png")); |
|  | } |
|  | } |