**КУРСОВАЯ РАБОТА**

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

по дисциплине «Языки и средства функционального программирования»

Выполнил

студент гр. 3530904/80002 Конев И.А.

Руководитель Лукашин А. А.

"17" декабря 2019 г.

Оглавление

[Описание задачи 3](#_Toc27500594)

[Описание решения 3](#_Toc27500595)

[Скриншот работы программы 4](#_Toc27500596)

[Вывод 5](#_Toc27500597)

# Описание задачи

Реализовать построение множества Мандельброта и представления его в виде картинки.

# Описание решения

//Создал класс комплексного числа и описал основные и необходимые действия для работы с ними

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

// Определение действий с комплексными числами

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

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

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

}

//Известно, что множество Мандельброта находится внутри круга радиуса 2 на комплексной плоскости.

//Если на каком-то шаге орбита точки выходит из этого круга, она уже никогда не вернется обратно.

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

def iterations(c : ComplexNumber, max : Int): Int = {

var compNum = new ComplexNumber(0, 0)

var numOfIter:Int = 0

do {

compNum = compNum\*compNum + c

numOfIter += 1

} while ((compNum.abs < 4) && (numOfIter < max))

if (compNum.abs > 4)

numOfIter;

else

0;

}

# Скриншот работы программы

# 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

## **ComplexNumber.scala**

class ComplexNumber (val real : Double, val imaginary : Double) { // Объявление класса комплексного числа

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

def -(c : ComplexNumber) = new ComplexNumber(real - c.real, imaginary - c.imaginary)

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

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

}

## **Main.scala**

import java.awt.\_

import javax.imageio.\_

import java.io.File

import java.awt.image.BufferedImage

object Main {

//Известно, что множество Мандельброта находится внутри круга радиуса 2.

//Если на каком-то шаге орбита точки выходит из этого круга, она уже никогда не вернется обратно.

//Поэтому определяем количество итераций, спустя которое точка останется в кругу или выйдет из него

def iterations(c : ComplexNumber, max : Int): Int = {

var compNum = new ComplexNumber(0, 0)

var numOfIter:Int = 0

do {

compNum = compNum\*compNum + c

numOfIter += 1

} while ((compNum.abs < 4) && (numOfIter < max))

if (compNum.abs > 4)

numOfIter;

else

0;

}

def main(args: Array[String]) {

val width = 1000

val height = 1000

val img = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

val g = img.getGraphics();

// Константы для перевода пикселей в координаты для проведения расчётов

val viewX = -2.0;

val viewY = -1.5;

val dx = 3.0/width; val dy = 3.0/height

for (y <- 0 to height) {

for (x <- 0 to width) {

val re:Double = dx\*x + viewX

val im:Double = dy\*y + viewY

val c = new ComplexNumber(re,im)

val numOfIter:Int = iterations(c, 200)

g.setColor(new Color(numOfIter, numOfIter/4, numOfIter))

g.drawLine(y,x,y,x)

}

}

ImageIO.write(img, "png", new File("Mandelbrot.png"));

}

}