|  |  |  |
| --- | --- | --- |
| Министерство образования и науки РФ  Санкт-Петербургский политехнический университет Петра Великого  Институт компьютерных наук и технологий  Высшая школа программной инженерии | | |
| КУРСОВАЯ РАБОТА  построение множества Мандельброта | | |
|  | | |
| по дисциплине «Языки и средства функционального программирования» | | |
| Выполнила | | |
| Студентка гр. 3530904/80004 |  | Морева Е. И. |
|  | | |
| Руководитель |  | Лукашин А. А. |
| «17» декабря 2019 г. | | |
| Санкт-Петербург | | |
| 2019 | | |

Оглавление

[Задание 3](#_Toc27913230)

[Пояснение к заданию 3](#_Toc27913231)

[Построение множества 3](#_Toc27913232)

[Реализация 4](#_Toc27913233)

[Результаты](#_Toc27913234) 6

[Выводы](#_Toc27913235) 7

# **Задание**

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

# **Пояснение к заданию**

Множество Мандельброта – это множество таких точек на комплексной плоскости *c*, для которых существует такое действительное *R*, что неравенство *|zn| < R* выполняется при всех натуральных *n*. Множество названо в честь математика Бенуа Мандельброта.

Множество Мандельброта является одним из самых известных фракталов. Его фрагменты не строго подобны исходному множеству, но при многократном увеличении определённые части всё больше похожи друг на друга.

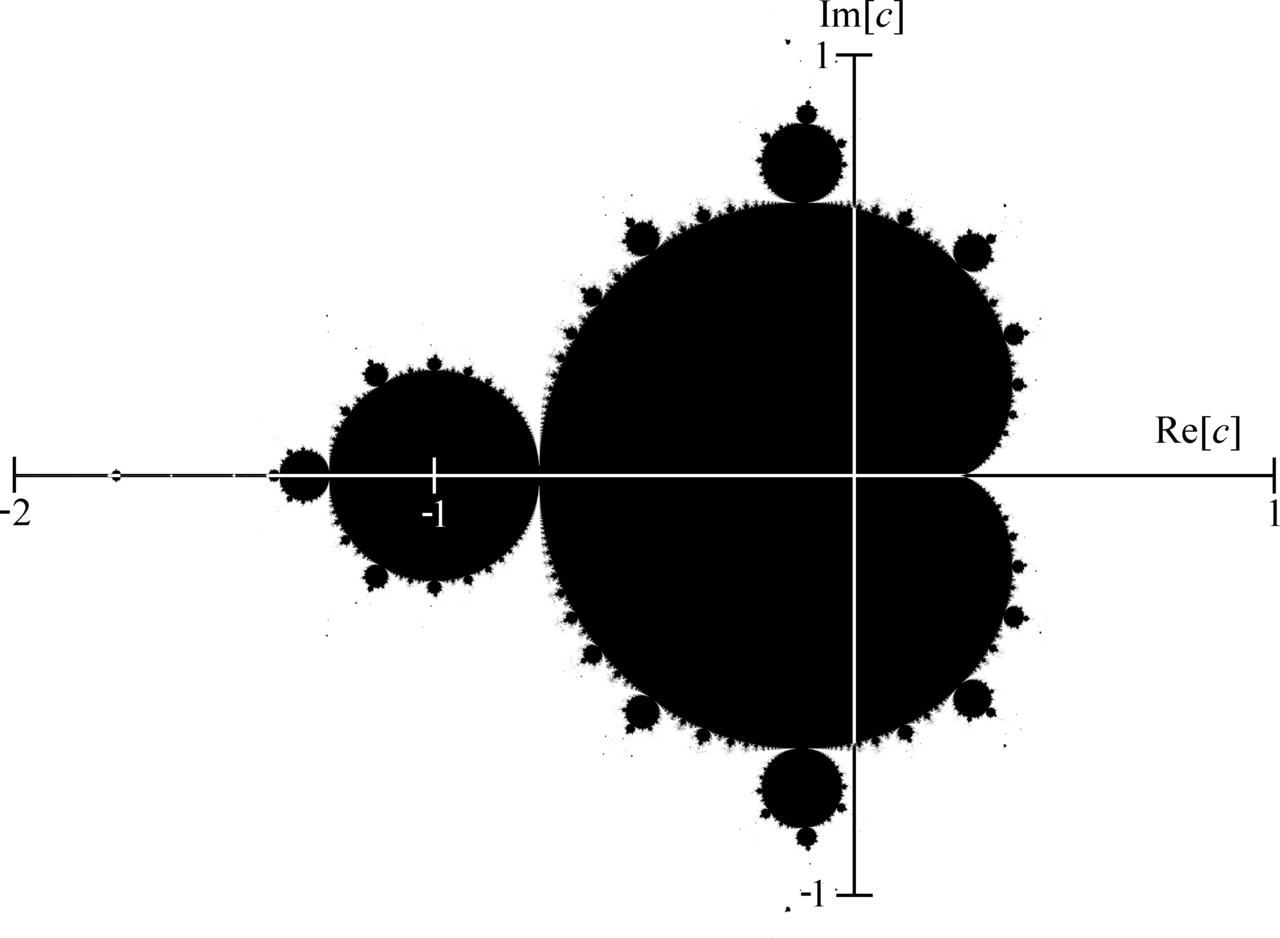
****

Рис. 1 Множество Мандельброта

## Построение множества

Как только модуль *zn* окажется больше 2, последовательность станет стремиться к бесконечности. В случае |*c*| ≤ 2 это можно доказать с помощью метода математической индукции. При |*c*| > 2 точка *c* заведомо не принадлежит множеству Мандельброта, что также можно вывести методом индукции, используя равенство= 0.

Сравнение |*zn*| с этим числом позволяет выделять точки, не попадающие внутрь множества. Для точек, лежащих внутри множества, последовательность не будет иметь тенденции к бесконечности и никогда не достигнет этого числа, поэтому после определённого числа итераций расчёт необходимо принудительно завершить. Максимальное число итераций, после которых число считается попавшим внутрь множества, задается в программе.

Изображение, полученное таким способом, является лишь приближением к реальному множеству Мандельброта. Более качественные результаты можно получать, увеличивая максимальное количество итераций, однако при этом пропорционально вырастает и время расчётов.

При построении можно учитывать количество итераций, за которое точка уходит в «бесконечность» или сточки зрения программы, на определенное расстояние от нуля.

Порядок определения, попадает ли точка внутрь множества следующий: на каждой итерации вычисляется значение , которое сравнивается с «границей бесконечности» (обычно значение равно 2). Таким образом, если при любом числе итераций, то цвет точки чёрный, в противном случае он зависит от последнего значения *n*, при котором выполнялось это неравенство. Значение *n*, фактически, обозначает скорость движения *zn* в бесконечность. Данный алгоритм определяет, что если точка удаляется больше чем на 2 от начала координат, то она лежит снаружи множества Мандельброта. Если точка не вышла за указанную границу, можно считать, что она находится внутри множества.

Точкам около границы множества нужно больше итераций для ухода в бесконечность. Поэтому такие области прорисовываются заметно дольше. Чем дальше от границ множества, тем выше скорость ухода в бесконечность. Для таких точек требуется меньше итераций.

# **Реализация**

Для реализации программы был выбран язык программирования Scala.

Scala – строгий статически типизированный JVM-based язык, разработанный Мартином Одерски. Язык успешно совмещает парадигмы объектно-ориентированного и функционального программирования.

Каждое значение в Scala является объектом. Значения и поведение описываются классами и трейтами. Классы могут расширяться с помощью наследования, а механизм композиции служит заменой множественного наследования. Каждая функция является значением и каждое значение объектом.

Scala обеспечивает простой синтаксис для определения анонимных функций и поддерживает функции высшего порядка.

Данный язык программирования поддерживает многопоточность.

Результатом компиляции кода, написанного на Scala, является Java байт-код, который исполняется JVM.

В процессе реализации программы был создан собственный класс для работы с комплексными числами.

class Complex(val a: Double, val b: Double) {

// represents the complex number a + b\*i

def +(that: Complex) = new Complex(this.a+that.a,this.b+that.b)

def \*(that: Complex) = new Complex(this.a\*that.a-this.b\*that.b,this.a\*that.b+that.a\*this.b)

def abs() = Math.sqrt(this.a\*this.a + this.b\*this.b)

}

При построении множества было решено учитывать количество итераций, за которое точка уходит в «бесконечность» (пиксели окрашиваются в белый цвет).

do {

z = z\*z + c

i += 1

} while (z.abs < 2 && i < level)

if (z.abs < 2)

out.write(0);

else

out.write((i\*255.0/level).toInt);

При запуске программы пользователь указывает размер изображения и количество итераций.

def main(args: Array[String]) {

run(Integer.parseInt(args(0)), Integer.parseInt(args(1)))

}

Ссылка на полный исходный код программы: <https://github.com/mycelium/hsse-fp-2019-2/tree/3530904_80004_Moreva_Ekaterina/Mandelbrot>

# **Результаты**

После исполнения программы пользователь получает графическое представление постарения множества Мандельброта

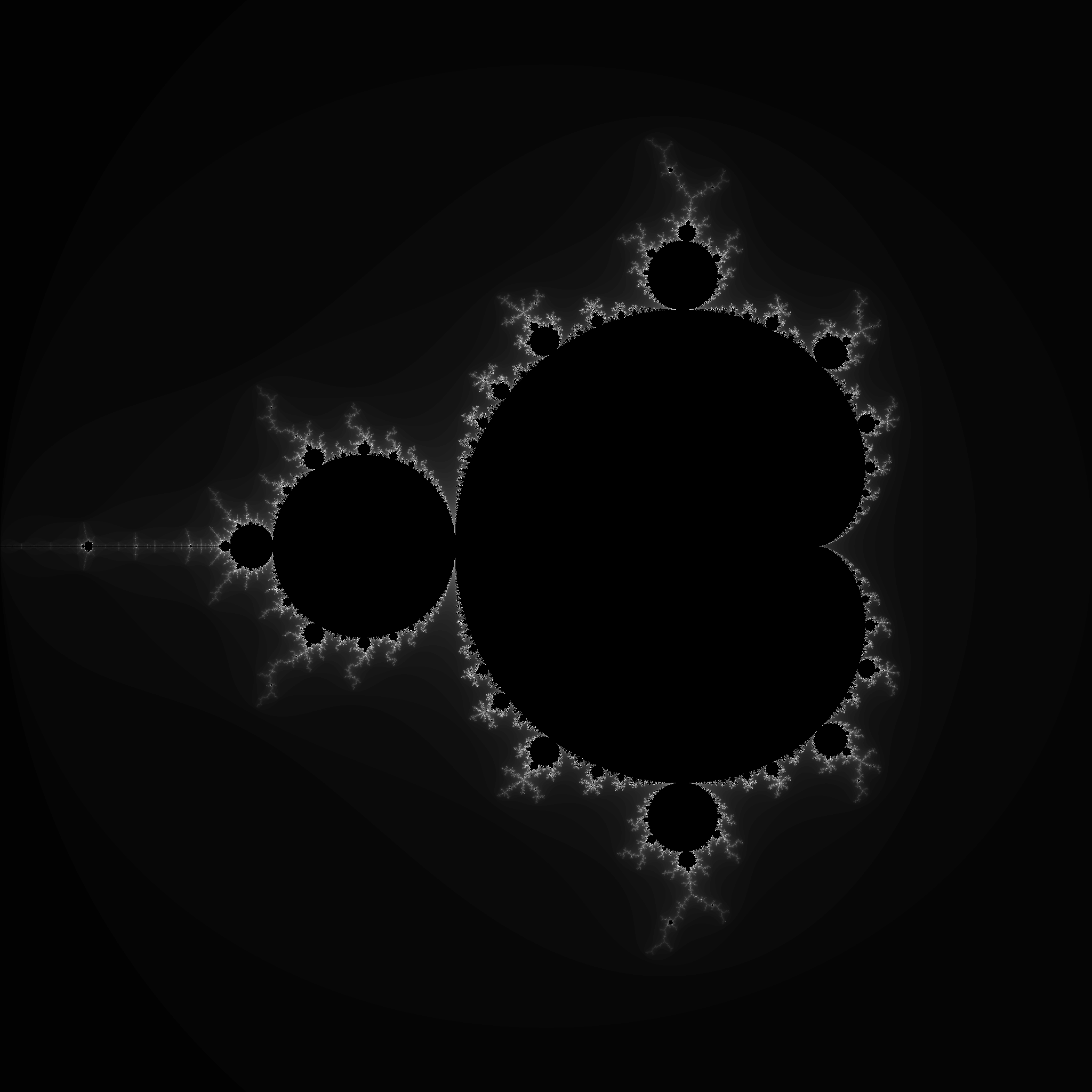


Рис. 2 Результат программы.

# **Выводы**

В процессе выполнения курсовой работы были применены знания, полученные в ходе изучения курса функционального программирования. При разработке программы на практике были более подробно изучены возможности языка Scala. Реализовано построение и представление в графическом виде множества Мандельброта.