МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доцент |  |  |  | А.В. Аграновский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ЛАБОРАТОРНАЯ РАБОТА №1  Исследование фрактальной графики |
|  |
| по дисциплине: Компьютерная графика |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4329 |  |  |  | Д.С. Шаповалова |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

Содержание

[1. Цель работы: 3](#_Toc177328079)

[2. Задание: 3](#_Toc177328080)

[3. Теоретические сведения: 3](#_Toc177328081)

[3.1 Множества Жюлиа и Мандельброта 4](#_Toc177328082)

[4. Алгоритм построения фрактала Жюлиа: 6](#_Toc177328083)

[5. Язык программирования и используемые библиотеки 6](#_Toc177328084)

[6. Описание программы построения фрактала: 7](#_Toc177328085)

[7. Скриншоты с изображениями фрактала с разной степенью детализации: 8](#_Toc177328086)

[8. Вывод: 11](#_Toc177328087)

[8.1 Полученные теоретические знания и навыки: 11](#_Toc177328088)

[8.2 Возникшие проблемы и пути их решения: 12](#_Toc177328089)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 14](#_Toc177328090)

[ПРИЛОЖЕНИЕ А 15](#_Toc177328091)

# 1. Цель работы:

Изучение теоретических основ фрактальной графики. Получение навыков визуализации фракталов.

# 2. Задание:

Реализовать программу построения фрактала на любом языке программирования высокого уровня в соответствии с заданием под вариантом 2, алгебраический фрактал Жюлиа. Предусмотреть возможность установки диапазона изменения координат. Предложить собственную цветовую гамму.

# 3. Теоретические сведения:

Фрактал – это объект или величина, демонстрирующие самоподобие (в формальном смысле) в любых масштабах. Объект демонстрирует при разных масштабах не идентичные структуры, но на всех уровнях фрактала должны проявляться структуры одного и того же «типа». В таком случае график, откладываемый в системе координат с логарифмическим масштабом, где по осям отсчитываются величина и масштаб, то график представляет собой прямую линию с наклоном, отражающим размерность фрактала.

Отрисовка фракталов сложна, так как глубинная природа фракталов определяется концепцией рекурсии. Говоря о графиках и их вычерчивании, мы обычно считаем, что они образованы пикселями или векторами, но количество пикселей или векторов всегда ограничено, а фракталы по определению бесконечно рекурсивны. Таким образом, попытавшись нанести фрактал на координатную сетку, мы в какой-то момент должны будем остановиться, и именно поэтому мы в данном случае говорим об «итерациях». На каждой итерации фрактал становится все сложнее, и в какой-то момент становится невозможно отличить две его итерации, следующие друг за другом (такой момент наступает, когда изменения происходят на уровне, сравнимом с размером пикселя). Здесь логично остановиться, но, как правило, форма фрактала вырисовывается быстрее, и остановиться можно еще раньше.

Типы фракталов:

Геометрические - строятся на основе исходной фигуры (линии, многоугольника или многогранника) путем ее дробления и выполнения различных преобразований полученных фрагментов. Самые понятные для обывателя, можно нарисовать руками на листочке или в растровом редакторе.

Алгебраические - строятся на основе математических формул — их можно превратить в геометрические, если построить графики на координатной плоскости. Среди алгебраических фракталов можно выделить фракталы Мандельброта, Жюлиа и бассейны Ньютона. Все они строятся на множестве комплексных чисел, которые состоят из действительной и мнимой части. Просто фракталы Мандельброта и Жюлиа строятся на основе квадратов комплексных чисел, а бассейны Ньютона — на основе их кубов.

Стохастические - строится на основе математических формул, но в процессе построения параметры в них случайным образом изменяются. Это приводит к появлению причудливых форм, очень похожих на природные (несимметричные деревья, изрезанные береговые линии и т.д.). Двумерные стохастические фракталы используются при моделировании рельефа местности и поверхности моря. В отличие от геометрических и некоторых алгебраических, стохастические фракталы можно построить лишь при помощи компьютера.

# 3.1 Множества Жюлиа и Мандельброта

Вероятно, два наиболее распространенных среди сложных фракталов. Множества Жюлиа образуются по той же самой формуле, что и множество Мандельброта. Первый вопрос, возникающий после визуального знакомства с множествами Мандельброта и Жюлиа это "если оба фрактала сгенерированы по одной формуле, почему они такие разные?"

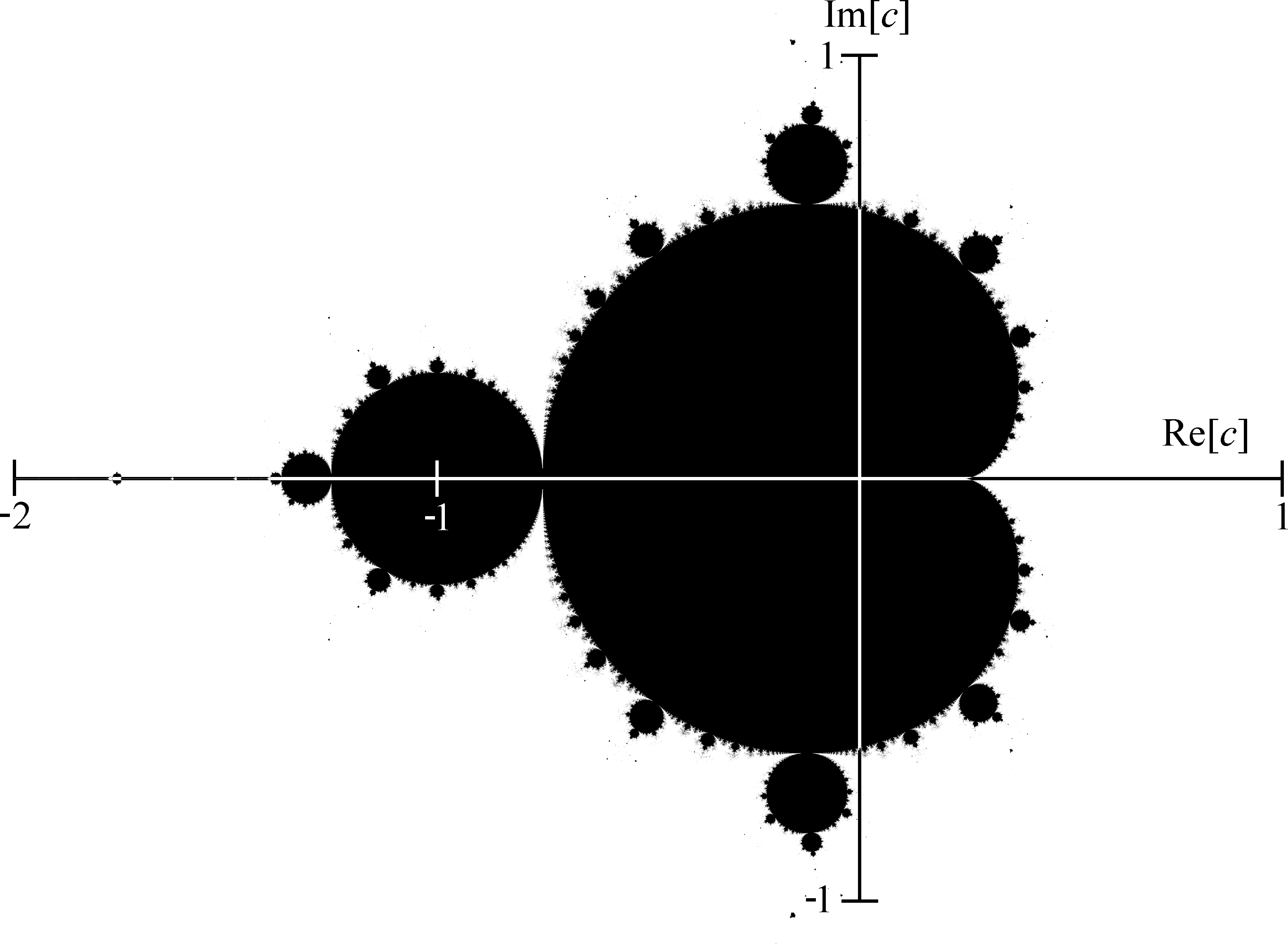


Рисунок 1.1. Множество Мандельброта

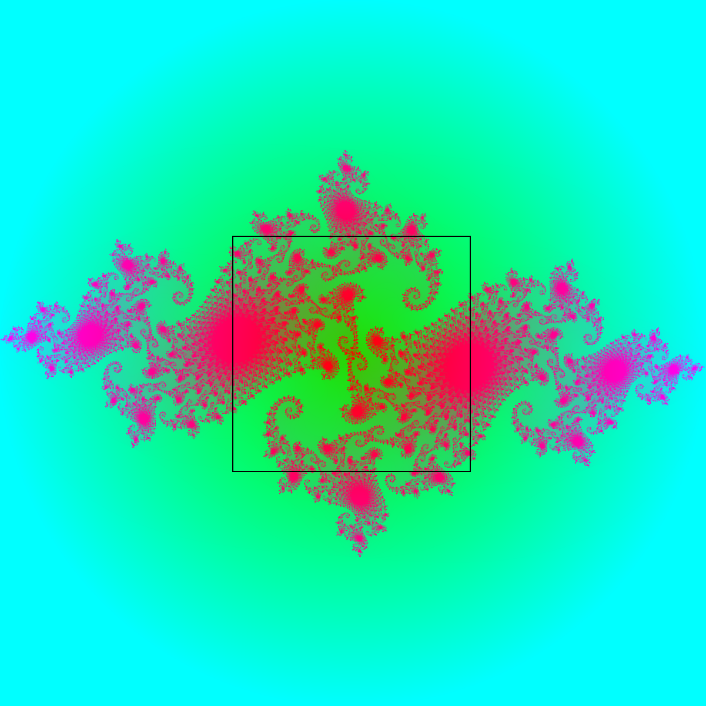


Рис 1.2. Множество Жюлиа

Достаточно странно, но существуют разные типы множеств Жюлиа. При рисовании фрактала с использованием различных начальных точек (чтобы начать процесс итераций), генерируются различные изображения. Это применимо только ко множеству Жюлиа.

Хотя это нельзя увидеть на картинке, фрактал Мандельброта - это, на самом деле, множество фракталов Жюлиа, соединенных вместе. Каждая точка (или координата) множества Мандельброта соответствует фракталу Жюлиа.

Их формула: z[n+1] = (z[n])^2+с. Отличие состоит в том, что в данном случае с является комплексной константой, а начальные координаты z[o]= x[o]+ iy[o] соответствуют координатам точек, принадлежность которых к фракталу мы определяем. Условием завершения расчета, как и для фрактала Мандельброта, является |z[n]|>2. Несмотря на незначительные отличия во входящих в формулу значениях, изображение фрактала Жюлиа существенно отличается от фрактала Мандельброта.

Точки пространства делятся на принадлежащие и не принадлежащие определенному множеству, что не предполагает наличия на изображении более двух цветов. Однако в настоящее время достаточно часто приходится видеть многоцветные фракталы, в том числе и фракталы Мандельброта и Жюлиа. На каждом шаге итерации проверяется условие завершения расчета |z[n]|>2, где |z[n1]=√(x[n]^2+у[n]^2). Данное условие для точек, не входящих в множество, на каком-то шаге оказывается выполненным. Номер такого шага фиксируют и преобразовывают в некоторый цвет. Цвет может выбираться из заранее принятой индексной палитры, формироваться в виде градаций серого цвета или по какому-то иному принципу, предусмотренному программистом. При этом в художественных целях обычно меняют цвет фона и цвет самого фрактала.

# 4. Алгоритм построения фрактала Жюлиа:

Опишем в общих чертах процедуру рисования множества Жюлиа многочлена *z*2 + *c* для конкретного значения комплексного параметра *c* = *p* + *iq*.

Будем считать, что экран прямоугольный и состоит из *a* × *b* точек и что изображение будет покрашено в *K* + 1 цвет (то есть цвета пронумерованы от 0 до *K*, причем цвет номер 0 — черный, а для других цветов условимся, что чем больше номер цвета, тем быстрее «убегает на бесконечность» точка, которую мы покрасим в этот цвет). Еще необходимо выбрать область плоскости, которую выведем на экран (для начала подойдет квадрат {|Re *z*| ≤ 1,5, |Im *z*| ≤ 1,5}; его нужно разрезать на *a* × *b* прямоугольников, каждый из которых будет выступать в роли точки экрана), и радиус *R* круга *D*, точки снаружи которого будем считать «бесконечно далекими» (можно взять *R* = 10).

Для каждой точки *z*0 = (*x*0; *y*0) экрана (то есть центра соответствующего прямоугольника) нужно в цикле последовательно вычислять *zk*+1 по *zk*, используя формулу  (в координатах это выглядит так: , *yk*+1 = 2*xkyk* + *q*). Признаком остановки цикла является выполнение одного из двух условий: либо на *k*-м шаге точка *zk* вышла из круга *D* (то есть верно неравенство , и тогда точку *z*0 нужно покрасить в цвет номер *k*, либо оказалось, что *k* = *K* + 1, тогда мы считаем, что точка *z*0 лежит внутри множества Жюлиа, и красим ее в черный.

В результате работы программы на экран будет выведена квадратная область комплексной плоскости {|Re *z*| ≤ 1,5, |Im *z*| ≤ 1,5}, на которой черным цветом будет изображено множество Жюлиа многочлена *z*2 + *c* для выбранного параметра *c* = *p* + *iq*, а остальные точки будут раскрашены в *K* цветов.

Увеличивая числа *a* и *b*, можно повышать разрешение экрана и тем самым улучшать качество изображения. Меняя *K* и подбирая соответствие между цветами и их номерами, можно добиться довольно красивых картинок.

# 5. Язык программирования и используемые библиотеки

В качестве языка программирования был выбран язык Python. В коде используются следующие библиотеки:

1) numpy (import numpy as np):

Библиотека Python, которую применяют для математических вычислений: начиная с базовых функций и заканчивая линейной алгеброй. В данном случае используется для создания и манипулирования массивами, а именно, для создания двумерного массива, представляющего изображение фрактала Жюлиа.

2) matplotlib (import matplotlib.pyplot as plt):

Библиотека на языке Python для визуализации данных. В ней можно построить двумерные (плоские) и трехмерные графики. В данном случае используется для отображения принадлежащих и не принадлежащих множеству Жюлиа точек.

Краткое описание действий в коде:

- Сначала функция `julia\_fractal` создает изображение фрактала Жюлиа с использованием numpy для обработки данных.

- Затем создается графическое окно Matplotlib с осями и холстом для отображения фрактала.

- Изображение фрактала выводится на экран с помощью matplotlib.pyplot

# 6. Описание программы построения фрактала:

Шаг 1: Область определения.

Мы начинаем с определения области комплексных чисел, в которой будем вычислять фрактал. Эта область задается диапазонами для вещественной (Re) и мнимой (Im) частей комплексного числа. – «xmin», «xmax»: границы по оси x (вещественная часть). – «ymin», «ymax»: границы по оси y (мнимая часть).

Шаг 2: Создание массива.

Мы создаем массив комплексных чисел «z» на основе указанного диапазона. Для этого: 1) Создаём равномерно распределённые точки по оси x и оси y с помощью «numpy.linspace». 2) Используем «numpy.meshgrid», чтобы создать сетку комплексных чисел z из двух одномерных массивов.

Шаг 3: Итерации.

Для каждого «z» (изначально представленного как «z = x + i\*y»):

- Начинаем с итерации (например, « z[0] = z» ).

- Повторяем процесс вычислений: «z[n+1] = z[n]^2 + c» до достижения максимального числа итераций или пока модуль «z» не превышает 2.

Условие выхода: Если модуль «z» превышает 2, это значит, что последовательность для данной точки «разбегается», и мы можем остановиться с итерациями. В противном случае, продолжаем до достижения максимального числа итераций («max\_iter»).

Подсчет итераций: Каждую итерацию записываем в массив, который позже будем использовать для визуализации.

Шаг 4: Визуализация.

По окончании вычислений массив указывает, сколько итераций понадобилось для каждого начального значения «z» до тех пор, пока условие выхода не было выполнено:

- Чем больше итераций, тем более «устойчива» точка.

- Мы визуализируем этот массив с помощью цветовой палитры (например, «cmap='inferno'»).

Изменение параметра «c» позволяет нам видеть разные формы фрактала. Разные значения «c» создают различные «изображения» фрактала, каждая из которых имеет свой уникальный вид.

# 7. Скриншоты с изображениями фрактала с разной степенью детализации:

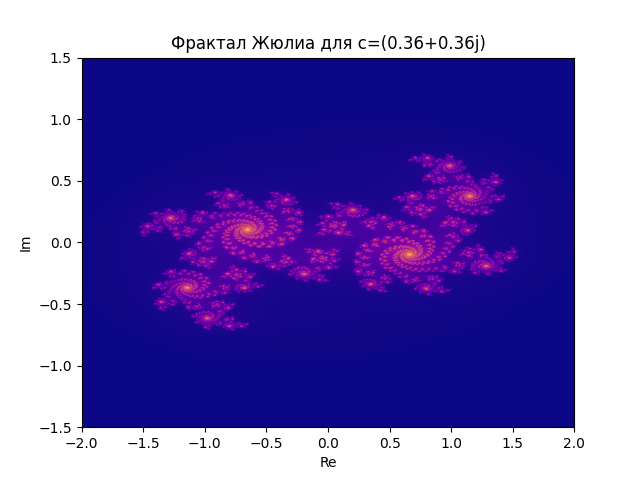


Рис 2.1. Изображение фрактала Жюлиа с c = 0,36 +0,36\*j, x\_min= -2, x\_max=2, y\_min=-1.5, y\_max=1.5

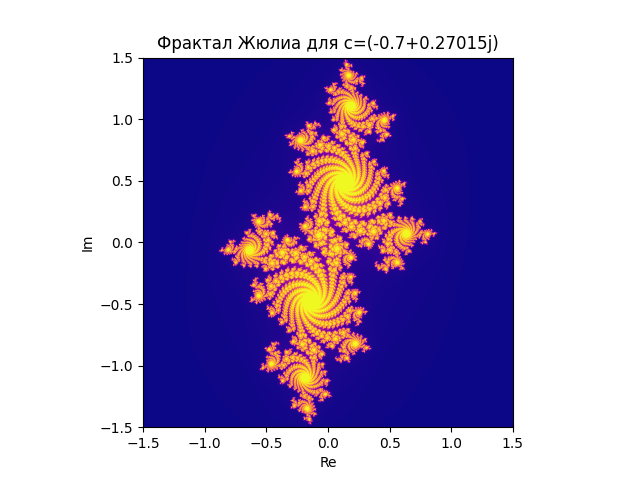


Рис 2.2. Изображение фрактала Жюлиа с «c = 0.7 +0.27015\*j, x\_min= -1.5, x\_max=1.5, y\_min=-1.5, y\_max=1.5

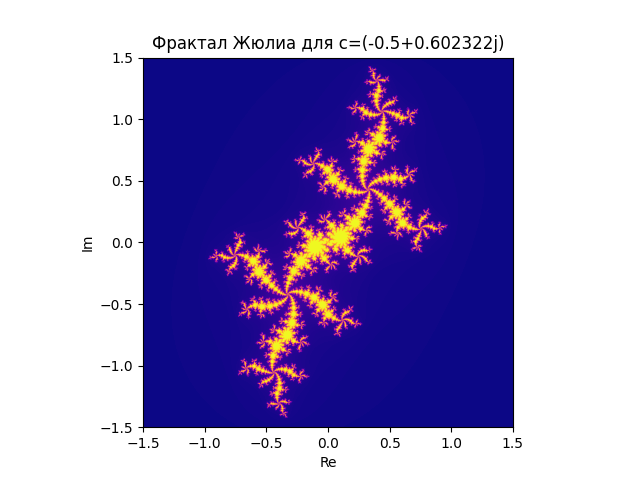


Рис 2.3. Изображение фрактала Жюлиа с c = -0.5 +0.602322\*j, x\_min= -1.5, x\_max=1.5, y\_min=-1.5, y\_max=1.5

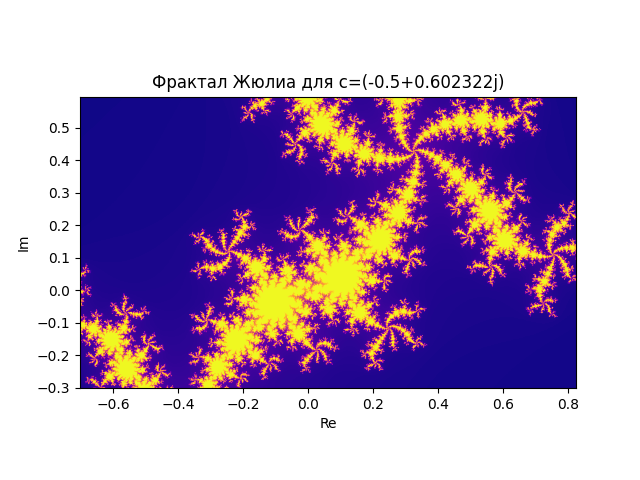


Рис 2.4. Изображение фрактала Жюлиа с c = -0.5 +0.602322\*j, x\_min= -1.5, x\_max=1.5, y\_min=-1.5, y\_max=1.5 - увеличенное

# 8. Вывод:

## 8.1 Полученные теоретические знания и навыки:

В процессе выполнения этой работы были получены следующие теоретические знания и навыки:

1. Фрактальная графика:

Были изучены основы фракталов, их виды(алгебраический, стохастический, геометрический), классификация(детерминированность). Была затронута причина их появления, основанная на самоподобии природы. Были освоены системы их построения, формулы и алгоритмы их реализации в компьютерной программе.

1. Фракталы алгебраического типа:

Были изучены фракталы алгебраического типа, основанные на использовании формул, участие рекурсии в их построении, роль нелинейных процессов в n-мерных пространствах и комплексных чисел в построении таких фракталов.

1. Фрактал Жюлиа:

Фракталы Жюлиа были изучены подробнее как один из фракталов алгебраического типа. Они создаются с использованием итеративных математических формул, а каждая точка применяет постоянные начальные настройки(количество итераций, диапазон по x,y, реальную и мнимую части комплексного числа с), изменяя которые, можно формировать разнообразные структуры фракталов Жюлиа. С изменением любого параметра, итоговый фрактал будет отличаться хотя бы немного.

1. Язык программирования высокого уровня Python:

В процессе выполнения работы был использован язык программирования Python. Был изучен алгоритм создания фракталов и визуализации их на экран, посредством графика, закрашивания каждой точки цветом, настолько ярким, насколько далека она от искомого множества. Так же были изучены новые, ранее не использовавшиеся библиотеки Python, такие как numpy и matplotlib, которые в будущем могут пригодиться в работе с графикой.

1. Библиотеки numpy, matplotlib:

Использование этих библиотек позволило реализовать алгоритм построения фрактала Жюлиа и визуализировать его в отдельном окошке, легко раскрасить, используя различные палитры. А конкретнее, модуль numpy был использован чтобы понять, какие точки фрактала Жюлиа будут оставаться в границах, а какие — уходить в бесконечность, в общем позволяет нам работать с многомерными массивами данных. В свою очередь matplotlib был использован для визуализации вычисленных данных (принадлежащих множеству Жюлиа точек).

## 8.2 Возникшие проблемы и пути их решения:

В процессе выполнения работы возникали определенные трудности, а именно:

1. Низкая производительность из-за библиотек:

В ходе создания программы для построения фрактала была попытка использовать модуль(библиотеку) pygame. Он предоставлял возможность двигать фрактал и менять масштаб прямо с клавиатуры, но работал с сильной задержкой и в целом медленно вычислял принадлежащие множеству точки. Было решено использовать библиотеку matplotlib, с которой изображение появлялось быстрее. К тому же разрешение, масштаб желаемого фрактала сильно влияет на производительность. В основном это задаётся такими параметрами как max\_iter, отвечающая за «изворотливость» фрактала, и width/height, отвечающие за разрешение изображения фрактала.

1. Выбор языка программирования:

Стоит упомянуть, что для разработки программы построения фрактала был использован язык программирования Python несмотря на то, что до этого полгода мы изучали C++, а сейчас изучаем C#. Такой выбор был обусловлен тем, что синтаксис Python прост до безумия, также для этого языка программирования существует множество удобных и простых в использовании библиотек и простых для понимания обучающих материалов. Хотя проблема с медленной скоростью работы могла бы быть решена с помощью использования эффективного алгоритма, написанном на с++.

1. Работа с библиотеками и отображение изображения:

Было бы неплохо, знай я до этого большинство существующих для графики в Python библиотек, но пришлось быстро изучать какие они есть и как работают, читать документацию. А их довольно много… Возможно, существуют более подходящие библиотеки, о которых я не знаю, но до их использования не дошла. Был испробован модуль «pygame», который не подошёл из-за его медлительности, вследствие чего был заменён на matplotlib, который меня устроил своей работой.

В результате выполнения этой работы, был получен практический опыт в создании алгоритма построения фракталов и их визуализации с использованием языка Python и его библиотек. Такой опыт может пригодиться для решения других задач в сфере информационных технологий, науке о данных и других областях, где используется вычислительное моделирование и визуализация.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Fractal -- from Wolfram MathWorld – URL: <https://mathworld.wolfram.com/Fractal.html> (дата обращения 15.09.2024)
2. Фракталы на Python. Пошаговое руководство – URL: <https://habr.com/ru/companies/piter/articles/496538/> (дата обращения 15.09.2024)
3. Шабанов П.А. Научная графика в python - URL: <https://github.com/whitehorn/Scientific_graphics_in_python>  (дата обращения 15.09.2024)
4. ФРАКТАЛЬНАЯ ГЕОМЕТРИЯ – ОБЛАСТИ ПРИМЕНЕНИЯ И ПЕРСПЕКТИВЫ РАЗВИТИЯ – URL: <https://dgng.pstu.ru/conf2015/papers/28/> (дата обращения 15.09.2024)
5. Фракталы: что это такое, виды фракталов, области применения – URL: <https://www.techinsider.ru/science/8906-krasota-povtora-fraktaly/> (дата обращения 15.09.2024)
6. Стохастические фракталы – URL: <https://studfile.net/preview/9026067/page:2/> (дата обращения 15.09.2024)
7. Как построить множества Жюлиа | Фракталы на Python – URL: <https://proproprogs.ru/fractals/fractals-kak-postroit-mnozhestva-zhyulia> (дата обращения 15.09.2024)

# ПРИЛОЖЕНИЕ А

Листинг Программы

import numpy as np  
import matplotlib.pyplot as plt  
  
# построение фрактала жюлиа  
def julia\_set(c, xmin, xmax, ymin, ymax, width, height, max\_iter):  
 # Создание массива комплексных чисел  
 x = np.linspace(xmin, xmax, width)  
 # равномерно распределяем точки по осям(ширине/высоте)  
 y = np.linspace(ymin, ymax, height)  
 z = np.array(np.meshgrid(x, y)).T.reshape(-1, 2)  
 # создаётся координатная сетка из двух одномерных массивов x и y.  
 # затем оборачивает матрицу в массив numpy  
 # T - чтобы транспонирвоать(строки->стобцы) массив  
 # reshape меняет форму массива. #  
 # Параметр -1 => кол-во строк будет авто рассчитано, сохранит общее количество элементов.  
 z = z[:, 0] + 1j \* z[:, 1] # двухмерный массив, пары (x,y) из сетки, чтобы построит график  
  
 # Инициализация массива для подсчета итераций (это холст для будущего рисунка)  
 img = np.zeros(z.shape, dtype=int) # массив заполненный целыми нулями, размера как массив z  
  
 # Основной цикл для вычисления фрактала  
 for i in range(max\_iter):  
 mask = np.abs(z) <= 2 # возвращает массив модулей для каждого элемента массива комплексных чисел z  
 # сравнивает каждое значения из массива модулей с числом 2 -> массив  
 img[mask] = i # записываем точки, которые принадлежат мн-ву  
 z[mask] = z[mask] \*\* 2 + c # Обновляем z согласно формуле Жюлиа  
 # вычисляем следующие точки!  
  
 return img.reshape((height, width))  
# создаём из одномерного массива полученных значений двумерный  
# матрицу, где каждая ячейка связана с пикселем изображения.  
  
# Чем ярче точка, тем ближе она к множеству Жюлиа и тем больше итераций ей нужно,  
# чтобы уйти от нуля на заданное большое расстояние  
  
def display\_fractal(c, xmin, xmax, ymin, ymax, width=1600, height=1600, max\_iter=256):  
 img = julia\_set(c, xmin, xmax, ymin, ymax, width, height, max\_iter)  
 # вызов функции вычисляющей точки, принадлежащие множеству жюлиа  
 plt.imshow(img, extent=(xmin, xmax, ymin, ymax), cmap='plasma') # cmap - меняет градиентную палитру из matplotlib  
# отображает изображение - массив с точками, принадлежащими мн-ву жюлиа  
# extent - задаёт границы картинки по осям, чтобы правильно сопоставить коорды и числа  
# plt.colorbar() # показывает полосу цвета-значения справа  
 plt.title(f'Фрактал Жюлиа для c={c}')  
 plt.xlabel('Re')  
 plt.ylabel('Im')  
 plt.show()  
  
# Дано:  
c = complex(-0.5, 0.602322) # комплексное число (реальная часть, мнимая часть)  
xmin, xmax = -1.5, 1.5 # Диапазон по x  
ymin, ymax = -1.5, 1.5 # Диапазон по y  
display\_fractal(c, xmin, xmax, ymin, ymax) # Вызов функции отображения фрактала