Отчёт к лабораторной работе:
Построение и визуализация фрактальных множеств

Выполнили:

- Зинченко Иван
- Рублёв Валерий
- ✨Марья(ु•ᴗ•ु❁)✨(я не знаю фамилию 👉👈)

Доказательства свойств для множества Мандельброта

1. Множество Мандельброта переходит само в себя при сопряжении. Иными словами, оно симметрично относительно вещественной оси

Рассмотрим две последовательности: $z_n(c) = \{0; c; c + c^2; c + c^2 + 2c^3 + c^4; \ldots\}$, $z_n(\overline{c}) = \{0; \overline{c}; \overline{c} + \overline{c}^2; \overline{c} + \overline{c}^2 + 2\overline{c}^3 + \overline{c}^4; \ldots\}$ Воспользуемся свойствами: 1) $\overline{a} + \overline{b} = \overline{a + b}$; 2) $\overline{a} \cdot \overline{b} = \overline{a \cdot b}$. Так как множество состоит из многочленов мы можем воспользоватся свойствами, чтобы сказать, что $z_n(\overline{c}) = \overline{z_n(c)}$, ч.т.д.

2. Если $|c| > 2$, то c не принадлежит множеству Мандельброта

Пусть для некоторого n выполнено $|z_n| > 2$. Тогда мы можем оценить $|z_n + 1|$ следующим образом: $|z_n + 1| = |z_n^2 + c| \geq |z_n|^2 - |c|$.

Поскольку $|z_n| > 2|$, то $|z_n|^2 > 4$. Подставляя это в неравенство, получаем: $|z_n + 1| > 4 - |c|$.

Так как $|c| > 2$, то $4 - |c| < 2$. Следовательно, $|z_n + 1| > 2$.

Из предыдущего рассуждения видно, что как только $|z_n|$ станет больше 2, все последующие значения $|z_n + 1|, |z_n + 2|, \ldots$ будут также больше 2 и будут неограниченно расти, так как каждое следующее значение $|z_n + 1|$ по крайней мере на единицу больше, чем предыдущее. Следовательно, последовательность $|z_n|$ не будет ограничена.

Код программ для построения множеств Мандельброта и Жюлиа

- Мандельброта

```
import numpy as np
import matplotlib.pyplot as plt
import os
from typing import NamedTuple, Tuple

class MandelbrotConfig(NamedTuple):
    xmin: float
    xmax: float
    ymin: float
    ymax: float
    width: int
    height: int

def mandelbrot(c: complex, max_iter: int) -> int:
    z = 0
    n = 0
    while abs(z) <= 2 and n < max_iter:
        z = z * z + c
        n += 1
    return n

def mandelbrot_set(config: MandelbrotConfig, max_iter: int) -> Tuple[np.ndarray,
```

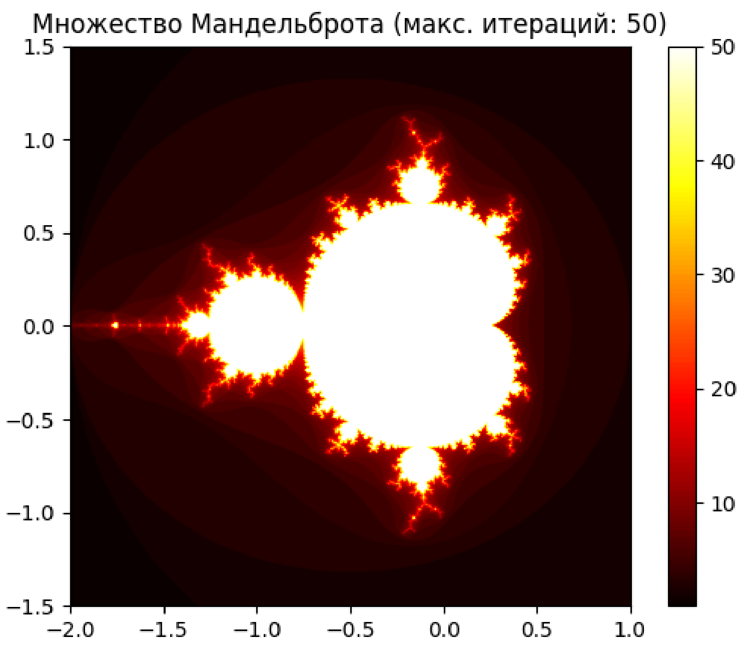
```
np.ndarray, np.ndarray]:
    r1 = np.linspace(config.xmin, config.xmax, num=config.width)
    r2 = np.linspace(config.ymin, config.ymax, num=config.height)
    return r1, r2, np.array([[mandelbrot(complex(r, i), max_iter) for r in r1] for i
in r2])

xmin, xmax = -2.0, 1.0
ymin, ymax = -1.5, 1.5
width, height = 800, 600
max_iters = (50, 100, 200, 400)

# Папка для хранения результатов
folder_name = "img/mandelbrot"
os.makedirs(folder_name, exist_ok=True)

config = MandelbrotConfig(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax, width=width,
height=height)

for count, max_iter in enumerate(max_iters, start=1):
    r1, r2, mandelbrot_image = mandelbrot_set(config, max_iter)

    # Путь для сохранения файла
    file_path = os.path.join(folder_name, f"experiment_{count}.png")

    # Построение и сохранение изображения
    plt.imshow(mandelbrot_image, extent=(xmin, xmax, ymin, ymax), cmap='hot')
    plt.colorbar()
    plt.title(f"Множество Мандельброта (макс. итераций: {max_iter})")
    plt.savefig(file_path)
    plt.close()
```

- Жюлиа

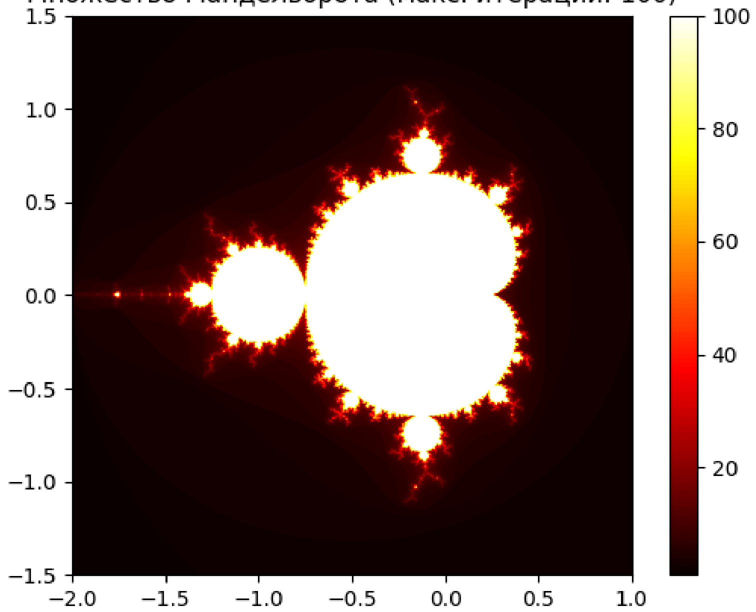
```
import numpy as np
import matplotlib.pyplot as plt
import os
from typing import NamedTuple, Tuple

class JuliaSetConfig(NamedTuple):
    c: complex
    xmin: float
    xmax: float
    ymin: float
    ymax: float
    width: int
    height: int
    max_iter: int

def julia(c: complex, max_iter: int, z: complex) -> int:
    n = 0
    while abs(z) <= 2 and n < max_iter:
        z = z * z + c
        n += 1
    return n

def julia_set(config: JuliaSetConfig) -> Tuple[np.ndarray, np.ndarray, np.ndarray]:
    r1 = np.linspace(config.xmin, config.xmax, num=config.width)
```

```
    r2 = np.linspace(config.ymin, config.ymax, num=config.height)
    return r1, r2, np.array([[julia(config.c, config.max_iter, complex(r, i)) for r
in r1] for i in r2])

c_values = (complex(-0.7, 0.27015), complex(0.355, 0.355), complex(-0.4, 0.6))
max_iters = (50, 100, 200, 400)
xmin, xmax = -1.5, 1.5
ymin, ymax = -1.5, 1.5
width, height = 800, 600

for idx, c in enumerate(c_values, start=1):
    folder_name = f"img/julya/experiment_{idx}"
    os.makedirs(folder_name, exist_ok=True)

    for count, max_iter in enumerate(max_iters, start=1):
        config = JuliaSetConfig(c=c, xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax,
width=width, height=height, max_iter=max_iter)
        r1, r2, julia_image = julia_set(config)
        file_path = os.path.join(folder_name, f"{count}.png")
        plt.imshow(julia_image, extent=(xmin, xmax, ymin, ymax), cmap='hot')
        plt.colorbar()
        plt.title(f"Множество Жюлиа для c={c} (макс. итераций: {max_iter})")
        plt.savefig(file_path)
        plt.close()
```

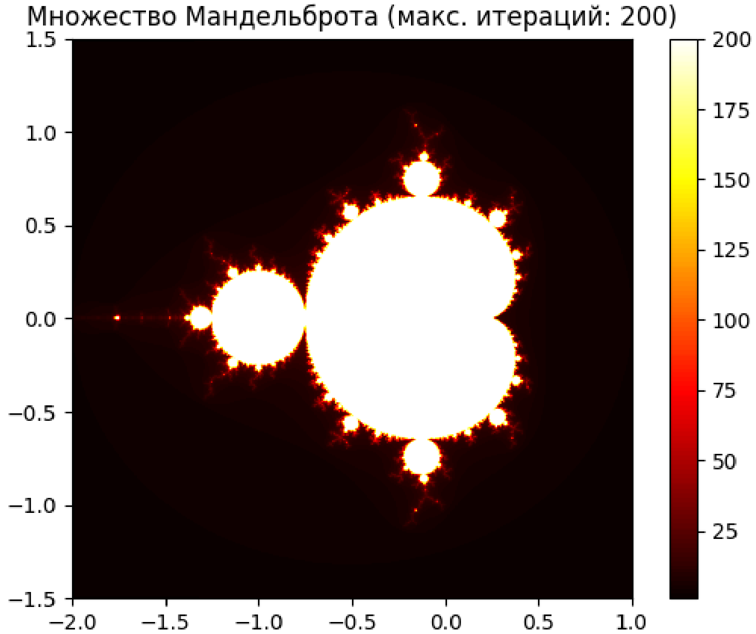
Набор изображений, построенных при разном числе итераций и приближении

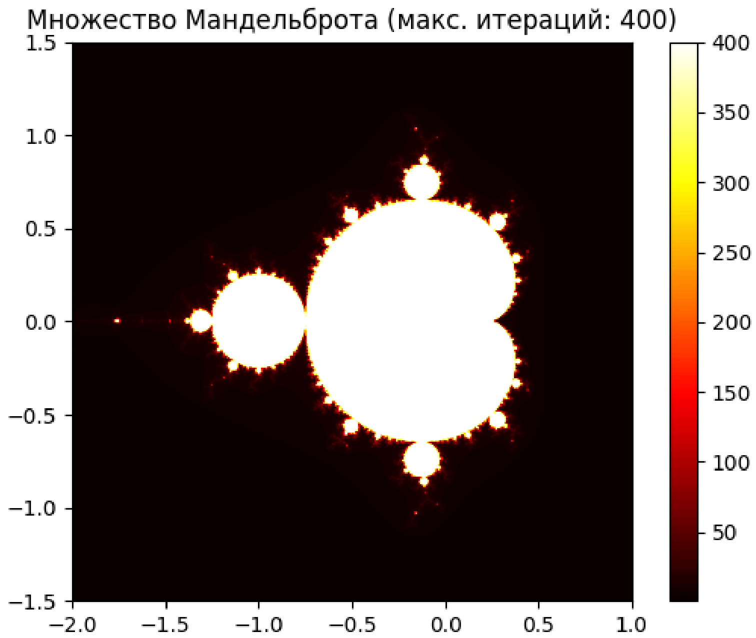


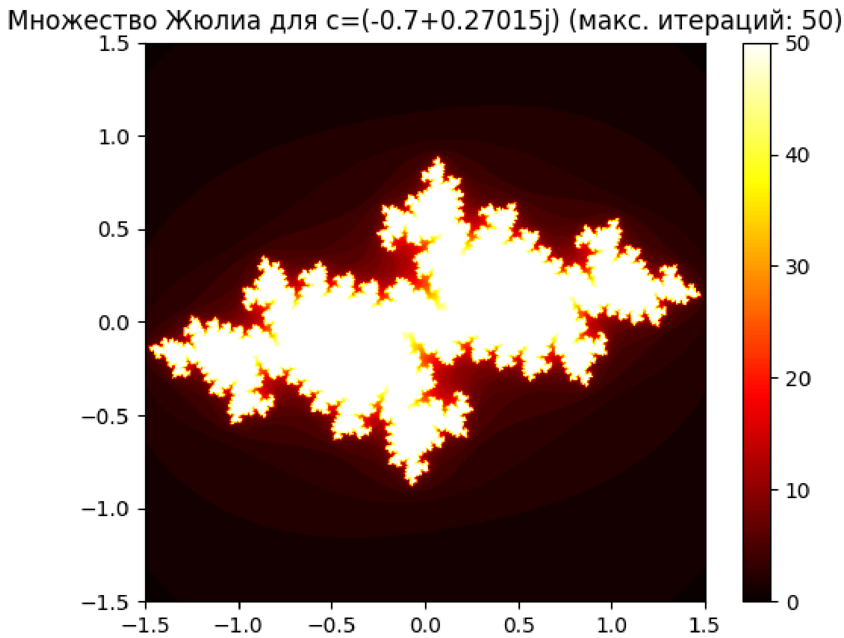
Множество Мандельброта (макс. итераций: 100)



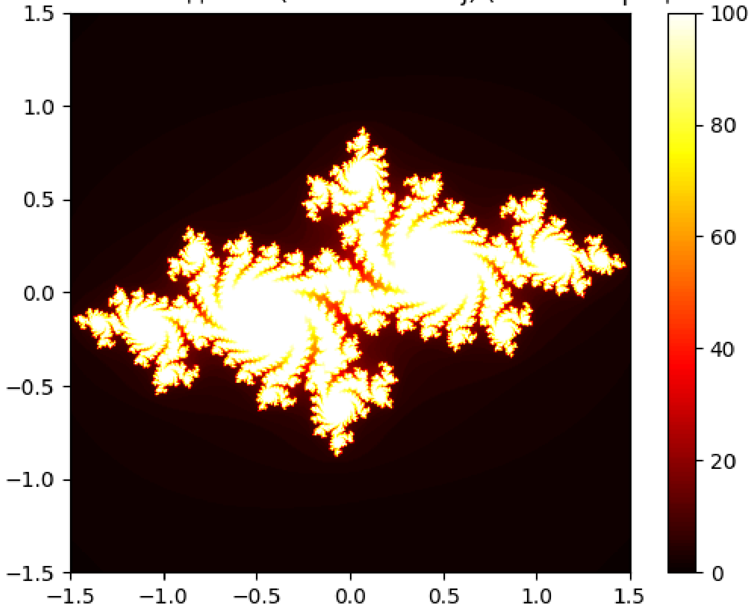
Множество Мандельброта (макс. итераций: 200)



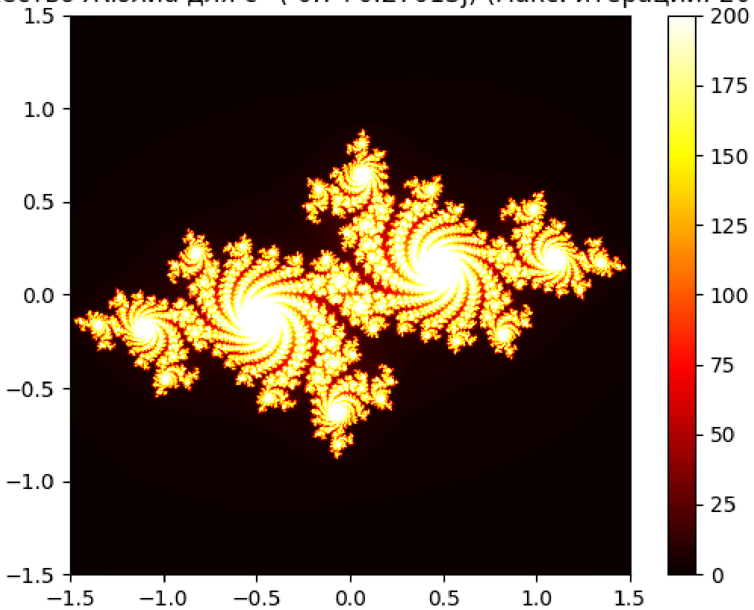

Множество Мандельброта (макс. итераций: 400)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
400
350
300
250
200
150
100
50


Множество Жюлиа для c=(-0.7+0.27015j) (макс. итераций: 50)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
50
40
30
20
10
0

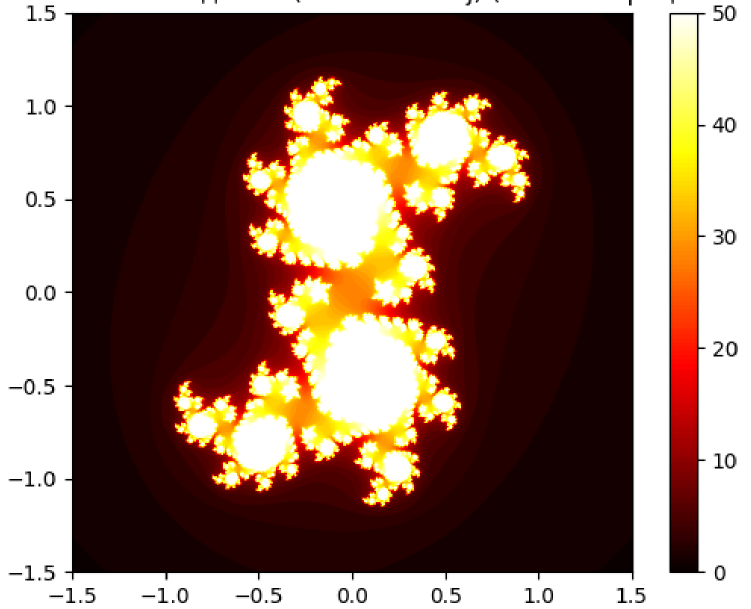
Множество Жюлиа для c=(-0.7+0.27015j) (макс. итераций: 100)



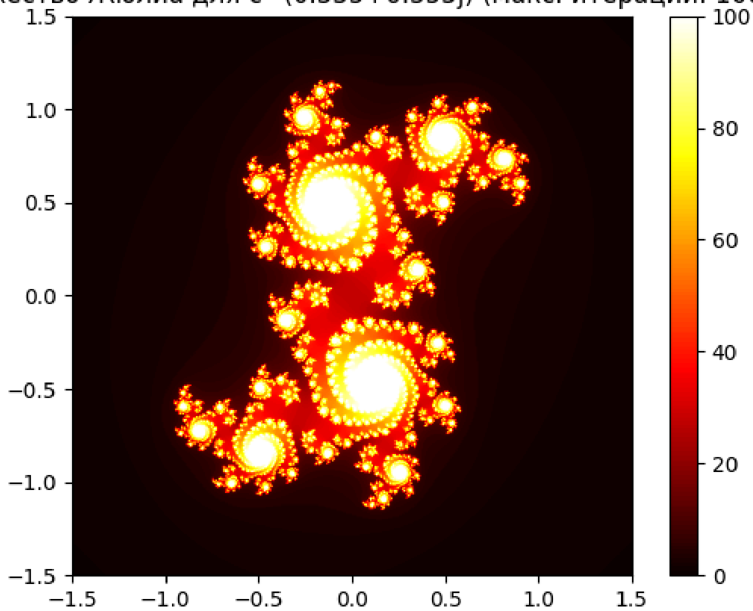
Множество Жюлиа для c=(-0.7+0.27015j) (макс. итераций: 200)



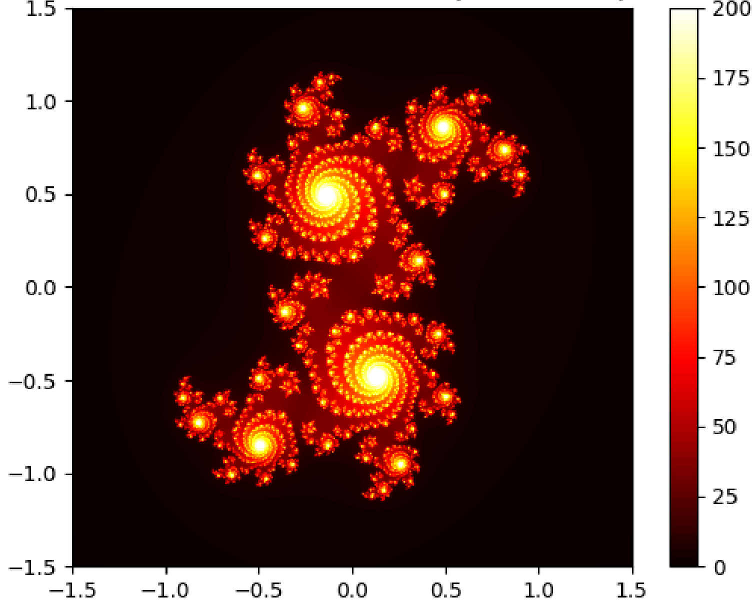
Множество Жюлиа для c=(0.355+0.355j) (макс. итераций: 50)



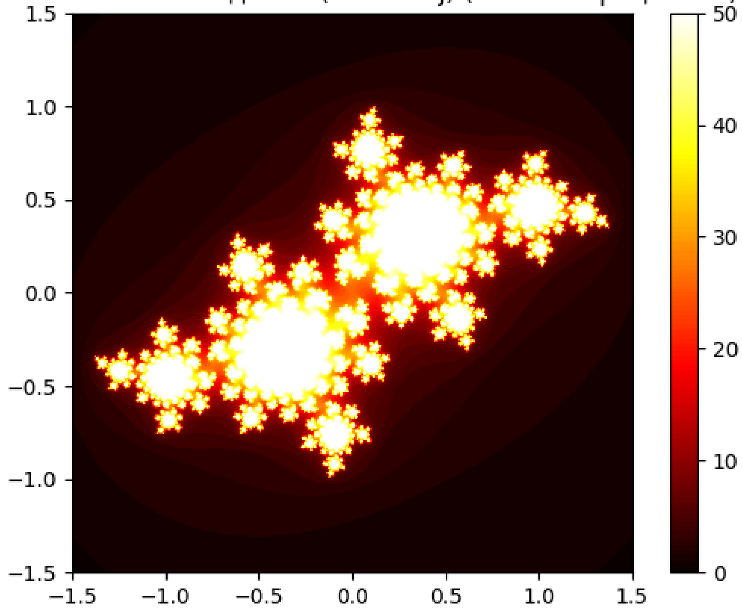
Множество Жюлиа для c=(0.355+0.355j) (макс. итераций: 100)

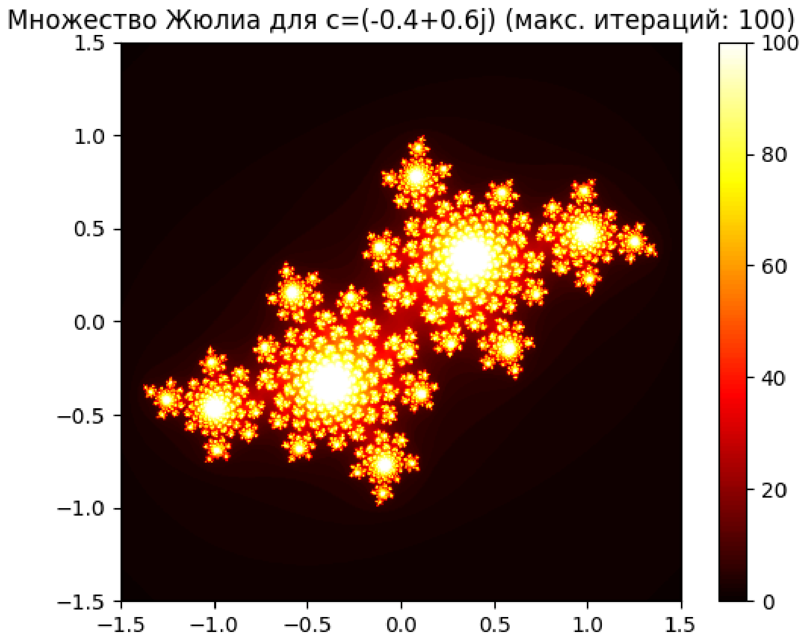


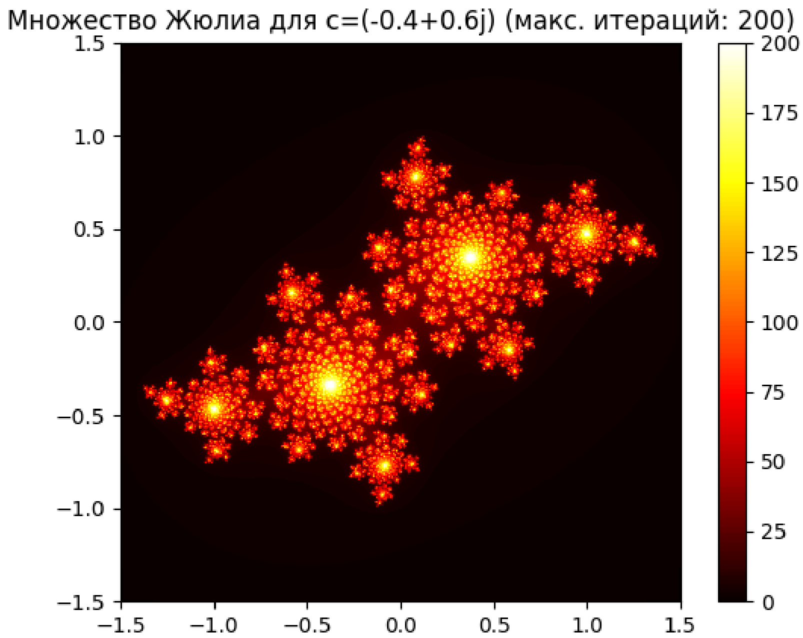
Множество Жюлиа для c=(0.355+0.355j) (макс. итераций: 200)



Множество Жюлиа для c=(-0.4+0.6j) (макс. итераций: 50)







Текст-описание структуры и построения ранее неразобранного фрактала. Его визуализации

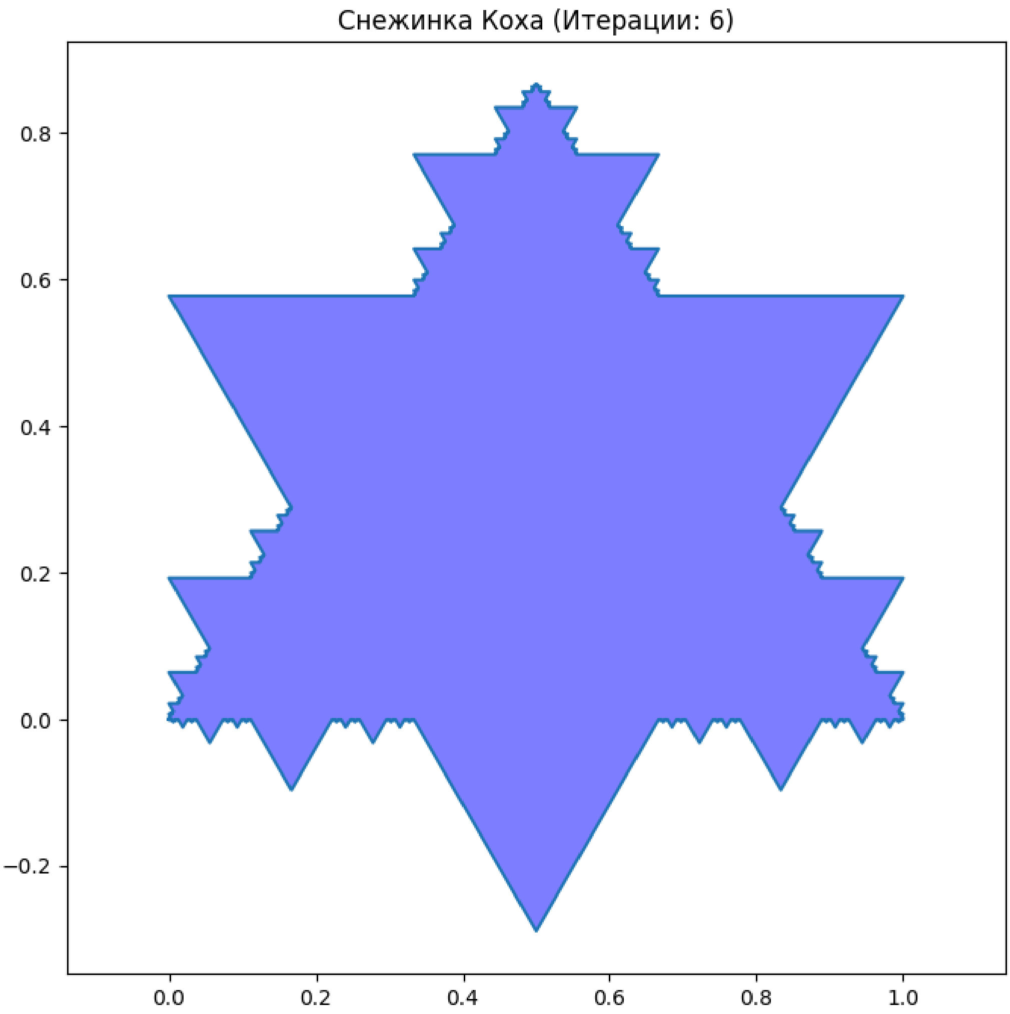
Фрактал "Снежинка Коха"

Кривая Коха является типичным геометрическим фракталом. Процесс её построения выглядит следующим образом: берём единичный отрезок, разделяем на три равные части и заменяем средний интервал равносторонним треугольником без этого сегмента. В результате образуется ломаная, состоящая из четырёх звеньев длины 1/3. На следующем шаге повторяем операцию для каждого из четырёх получившихся звеньев и т. д... Предельная кривая и есть кривая Коха.

Снежинка Коха, построенная в виде замкнутой кривой на базе равностороннего треугольника, впервые была описана шведским математиком Хельге фон Кохом в 1904 году. В некоторых работах она получила название «остров Коха».

Код построения снежинки Коха:

```
import matplotlib.pyplot as plt

def koch_snowflake(iterations):
    # Начальные точки равностороннего треугольника
    points = [[0, 0], [1, 0], [0.5, (3 ** 0.5) / 2]]

    def koch_segment(start, end, depth):
        if depth == 0:
            return [start, end]

        # Получаем точки на сегменте
        s = (start[0] + end[0]) / 2, (start[1] + end[1]) / 2  # Центр
        d = ((end[0] - start[0]) / 3, (end[1] - start[1]) / 3)  # Направление
        p1 = (start[0] + d[0], start[1] + d[1])  # 1/3 из точки начала
        p2 = (end[0] - d[0], end[1] - d[1])      # 2/3 из точки конца

        # Расчет вершины нового треугольника
        peak = (s[0] + (d[1] * (3 ** 0.5) / 2), s[1] - (d[0] * (3 ** 0.5) / 2))
        return koch_segment(start, p1, depth - 1) + [peak] + koch_segment(p2, end,
depth - 1)

    # Генерация всей снежинки
    snowflake = []
    for i in range(3):
        start = points[i]
        end = points[(i + 1) % 3]
        snowflake += koch_segment(start, end, iterations)

    return snowflake

# Параметры
iterations = 5  # Количество итераций
snowflake = koch_snowflake(iterations)

# Визуализация фрактала
plt.figure(figsize=(8, 8))
x, y = zip(*snowflake)  # Распаковка координат
plt.plot(x, y)
plt.fill(x, y, 'b', alpha=0.5)  # Заполнение снежинки цветом
plt.axis('equal')
plt.title(f"Снежинка Коха (Итерации: {iterations})")
plt.show()
```


Снежинка Коха (Итерации: 6)
0.8
0.6
0.4
0.2
0.0
−0.2
0.0
0.2
0.4
0.6
0.8
1.0