Федеральное государственное автономное образовательное учреждение высшего образования Национальный исследовательский университет ИТМО

Факультет программной инженерии и компьютерной техники

# Теория функций комплексного переменного

Лабораторная работа №1

Выполнили: студенты группы P3208,
Васильев Н. А., Елисеев К. И.,
Есоян В. С.

Преподаватель: Бойцев А. А.

Санкт-Петербург 2024

## Текст задания

1. Докажите свойства 1 и 2 для множества Мандельброта.
2. Напишите программу, которая будет строить визуализацию множества Мандельброта. Выберете разумные ограничения, поварьируйте максимальное количество итераций. Попробуйте приблизить отдельные части множества, чтобы увидеть фрактальную структуру.
3. Напишите программу, которая по заданному $c$ строит заполненное множество Жюлиа. Поварьируйте максимальное количество итераций, попробуйте пронаблюдать фрактальную структуру, рассмотрите множество при разных $c$.
4. Найдите какой-нибудь неразобранный фрактал. Опишите его структуру, построение. Нарисуйте визуализации.

## Доказательство

Свойства множества Мандельброта:

1. Множество Мандельброта переходит само в себя при сопряжении.

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

Пусть $z_n + 1 = z_n^2 + c \qquad z_0 = 0 \qquad c = x_0 + y_0 i$

Тогда $x_{n+1} + y_{n+1} i = (x_{n+1} + y_{n+1} i)^2 + x_0 + y_0 i$

Следовательно, $x_{n+1} = x^2 - y^2 + x \qquad y_{n+1} = 2x_n y_n + y_0$

При сопряжении значение $x$ останется прежним, то есть координата $x$ не поменяется, а значение $y$ заменяется на $-y$, поэтому точки симметричны относительно оси O$x$

◄

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

► Пусть $z_{n+1} = z_n^2 + c \qquad z_0 = 0$

Пусть $|c| = 2 + \delta$, где $\delta > 0$

С помощью индукции нужно показать, что $z_n > 2 + (n-1)\delta$ для $n \geq 2$

База индукции $n = 2$: $z_2 = z_2 > 2 + (2-1)\delta = 2 + \delta$

Предположение индукции $n = k$: $z_k > 2 + (k-1)\delta$

Переход индукции $n = k + 1$: $z_{k+1} > 2 + (k+1-1)\delta = 2 + (k-1)\delta + \delta > z_k + \delta$

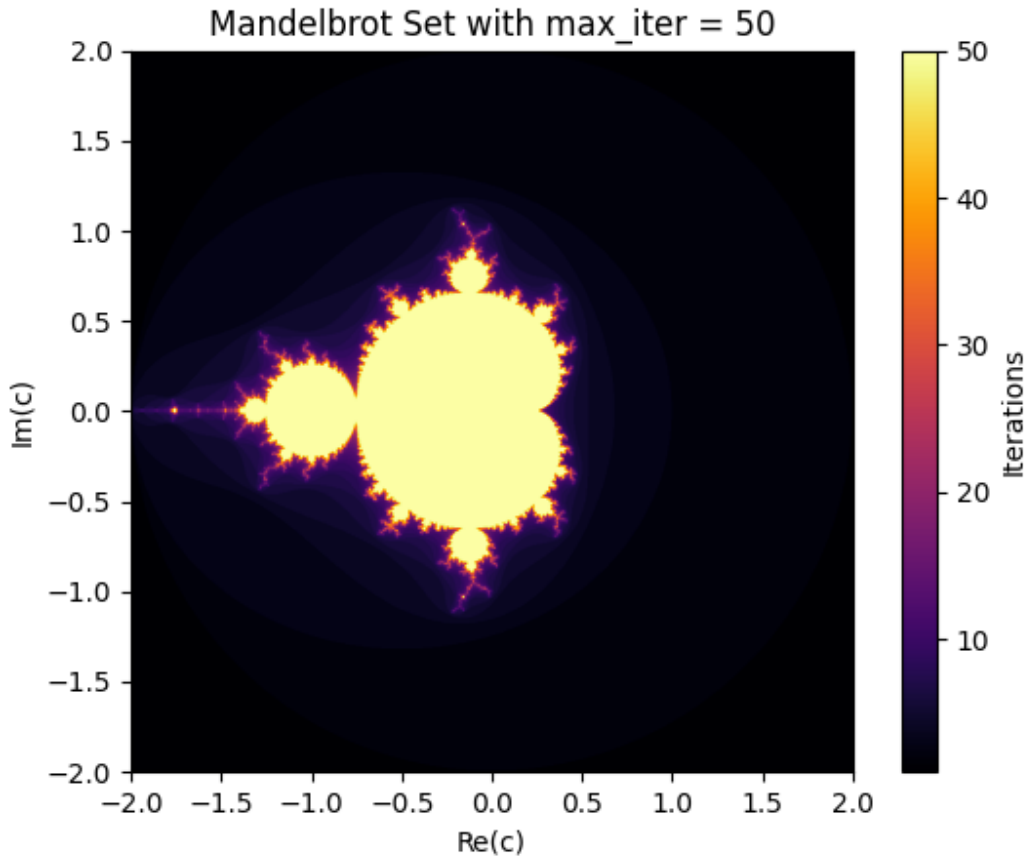
Это означает, что $|z_n| \to \infty$ при $n \to \infty$, когда $|z_n| > 2$, и поэтому $|z_n|$ не может быть ограниченным для всех $n \geq 1$

◄

## Исходный код программы визуализации множества Мандельброта на Python

```
import numpy as np
import matplotlib.pyplot as plt
```

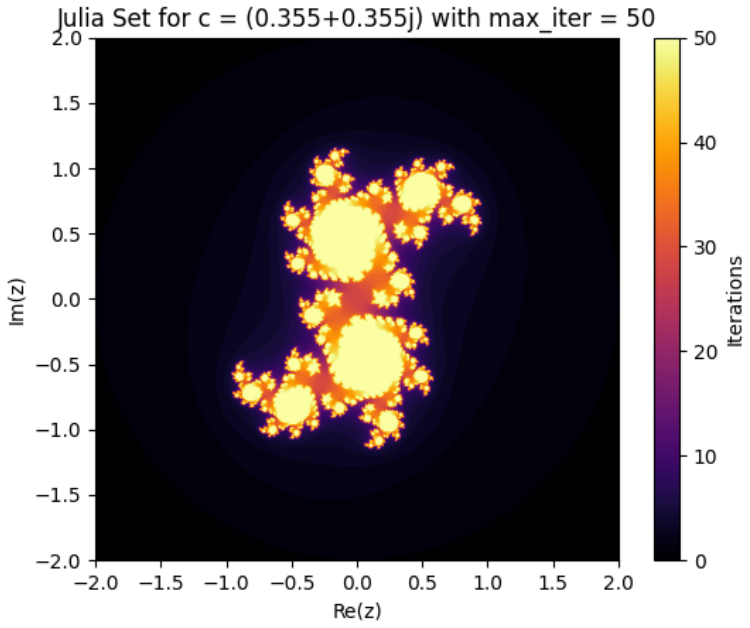
```
def main():
    center = (0.0, 0.0)
    scale = 2.0
    resolution = (800, 800)
    max_iters = [50, 100, 150]

    plot_mandelbrot_set(center, scale, resolution, max_iters)


def mandelbrot_set(center, scale, resolution, max_iter):
    width, height = resolution
    x_min = center[0] - scale
    x_max = center[0] + scale
    y_min = center[1] - scale
    y_max = center[1] + scale

    x_vals = np.linspace(x_min, x_max, width)
    y_vals = np.linspace(y_min, y_max, height)

    image = np.zeros((height, width))

    for x_idx, x in enumerate(x_vals):
        for y_idx, y in enumerate(y_vals):
            c = x + 1j * y
            z = 0
            iteration = 0
            while abs(z) <= 2 and iteration < max_iter:
                z = z ** 2 + c
                iteration += 1
            image[y_idx, x_idx] = iteration

    return image


def plot_mandelbrot_set(center, scale, resolution, max_iters):
    for max_iter in max_iters:
        mandelbrot_image = mandelbrot_set(center, scale, resolution,
max_iter)
        plt.imshow(mandelbrot_image, cmap="inferno", extent=(
            center[0] - scale, center[0] + scale, center[1] - scale,
center[1] + scale))
        plt.colorbar(label='Iterations')
        plt.title(f"Mandelbrot Set with max_iter = {max_iter}")
        plt.xlabel("Re(c)")
        plt.ylabel("Im(c)")
        plt.show()


if __name__ == "__main__":
    main()
```

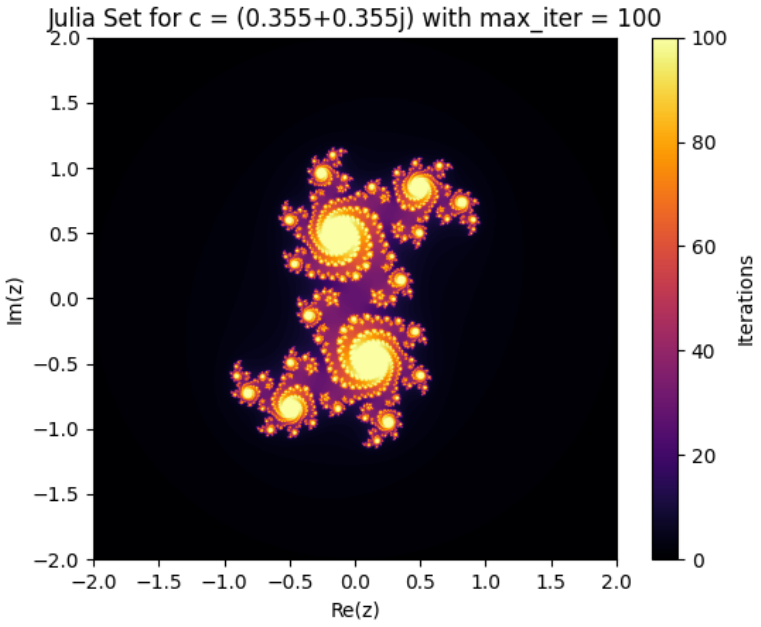


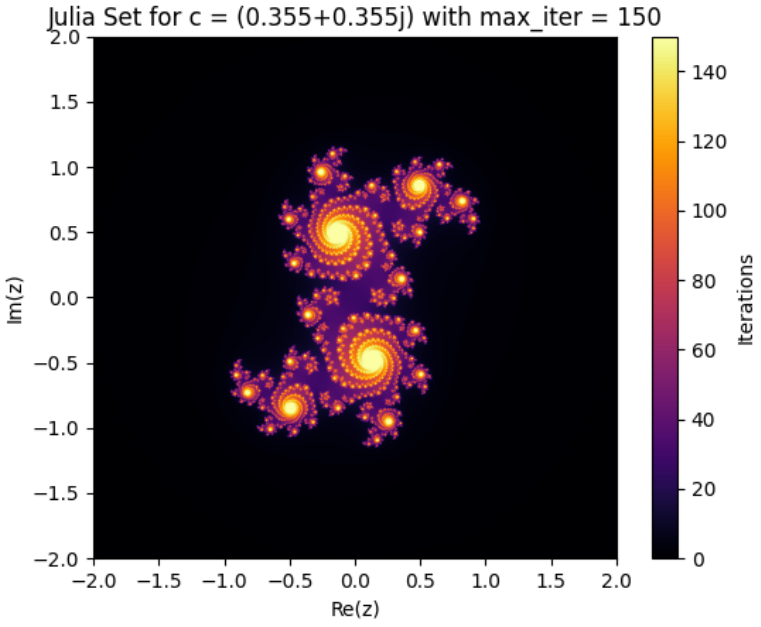

Исходный код программы визуализации множества Жюлиа на Python

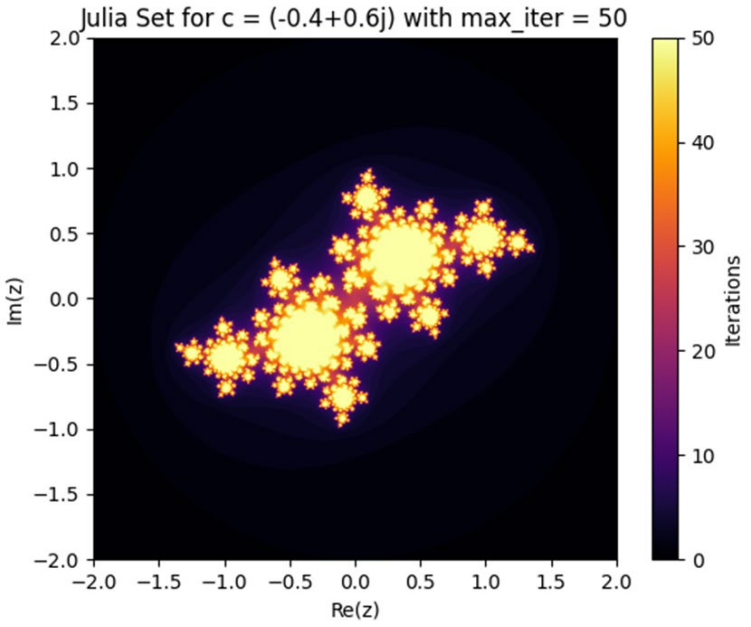
```
import numpy as np
import matplotlib.pyplot as plt


def main():
    center = (0.0, 0.0)
    scale = 2.0
    resolution = (1600, 1600)
    c_values = [0.355 + 0.355j, -0.4 + 0.6j, -0.70176 - 0.3842j, -0.5251993 +
0.5251993j]
    max_iters = [50, 100, 150]

    plot_julia_set(center, scale, resolution, c_values, max_iters)


def julia_set(center, scale, resolution, c, max_iter):
    width, height = resolution
    x_min = center[0] - scale
    x_max = center[0] + scale
    y_min = center[1] - scale
    y_max = center[1] + scale

    x_vals = np.linspace(x_min, x_max, width)
    y_vals = np.linspace(y_min, y_max, height)
```

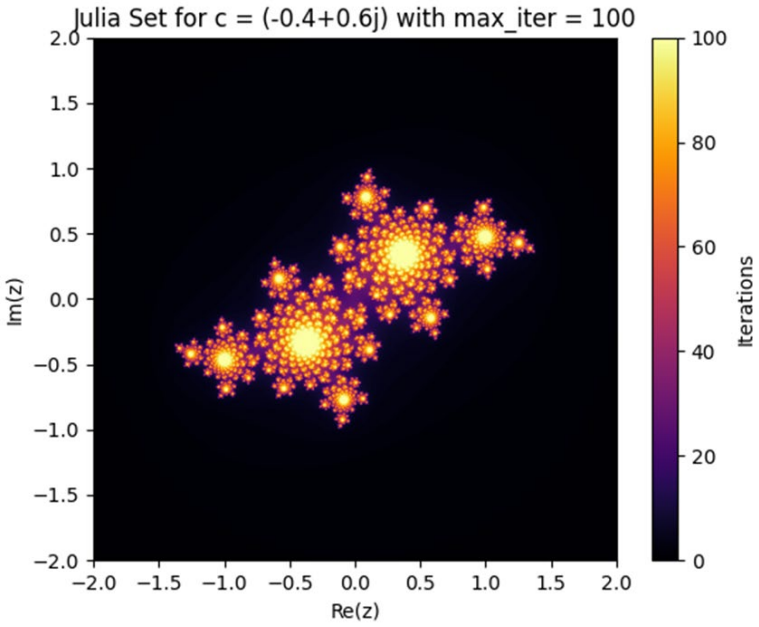
```
    image = np.zeros((height, width))

    for x_idx, x in enumerate(x_vals):
        for y_idx, y in enumerate(y_vals):
            z = x + 1j * y
            iteration = 0
            while abs(z) <= 2 and iteration < max_iter:
                z = z ** 2 + c
                iteration += 1
            image[y_idx, x_idx] = iteration

    return image


def plot_julia_set(center, scale, resolution, c_values, max_iters):
    for c in c_values:
        for max_iter in max_iters:
            julia_image = julia_set(center, scale, resolution, c, max_iter)
            plt.imshow(julia_image, cmap="inferno", extent=(
                center[0] - scale, center[0] + scale, center[1] - scale,
center[1] + scale))
            plt.colorbar(label='Iterations')
            plt.title(f"Julia Set for c = {c} with max_iter = {max_iter}")
            plt.xlabel("Re(z)")
            plt.ylabel("Im(z)")
            plt.show()


if __name__ == "__main__":
    main()
```

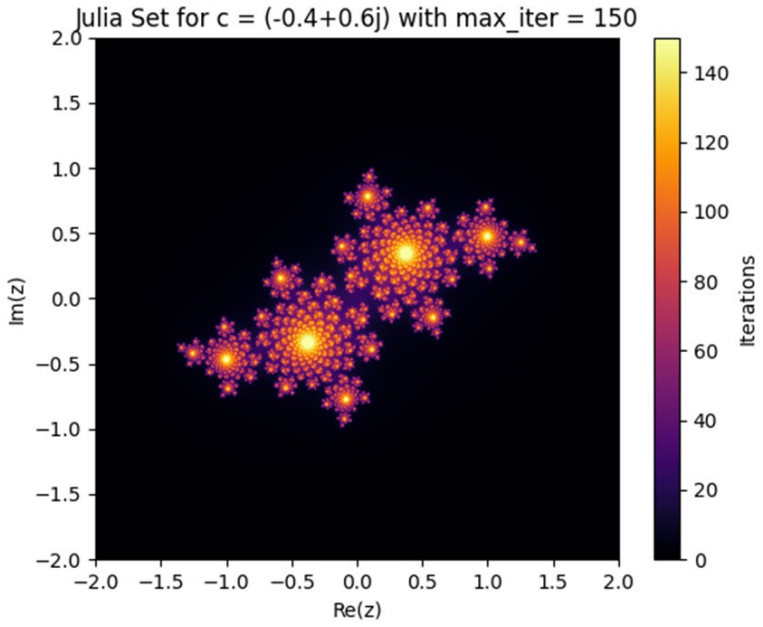


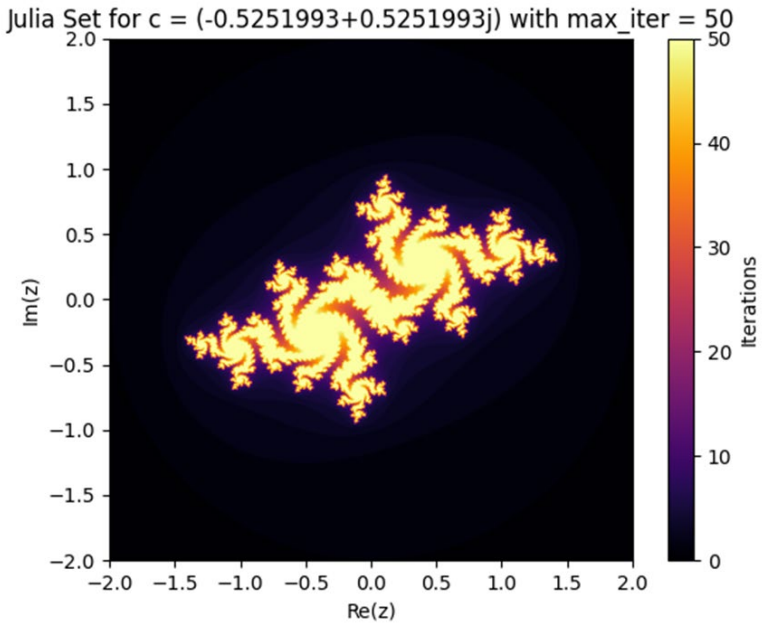

Julia Set for c = (0.355+0.355j) with max_iter = 100
Im(z)
Re(z)
Iterations
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
100
80
60
40
20
0

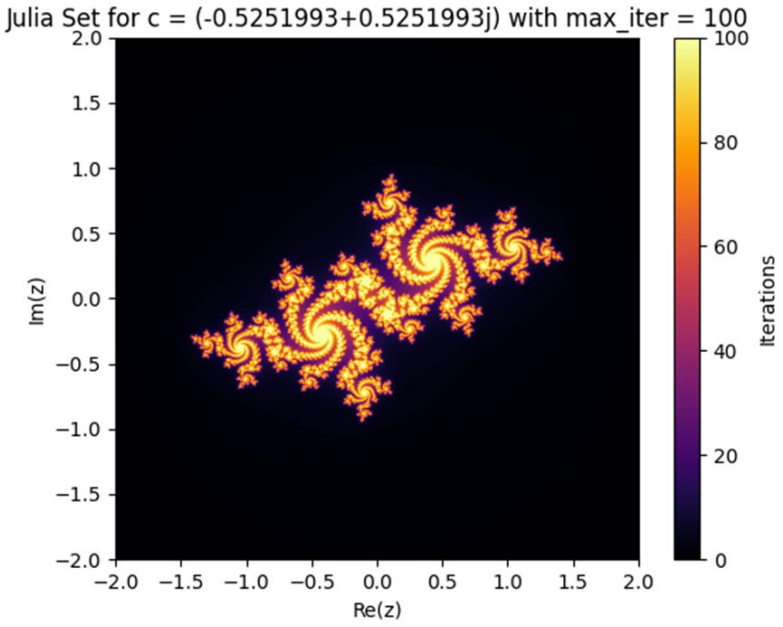

Julia Set for c = (0.355+0.355j) with max_iter = 150
Im(z)
Re(z)
Iterations
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
140
120
100
80
60
40
20
0

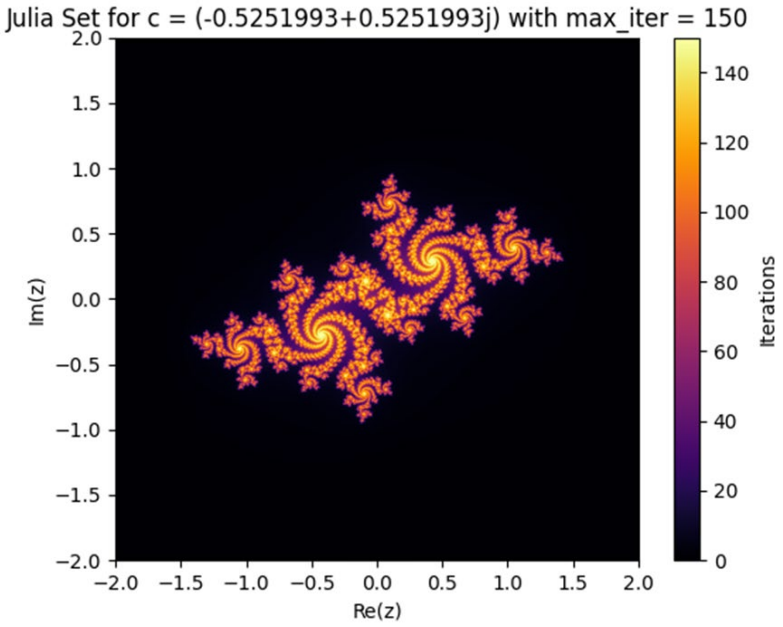

Julia Set for c = (-0.4+0.6j) with max_iter = 50
Im(z)
Re(z)
Iterations
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
50
40
30
20
10
0


Julia Set for c = (-0.4+0.6j) with max_iter = 100
Im(z)
Re(z)
Iterations
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
100
80
60
40
20
0


Julia Set for c = (-0.4+0.6j) with max_iter = 150
Im(z)
Re(z)
Iterations
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
140
120
100
80
60
40
20
0


Julia Set for c = (-0.5251993+0.5251993j) with max_iter = 50
Im(z)
Re(z)
Iterations
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
50
40
30
20
10
0




## Фрактал

**Фрактал Burning Ship** был открыт Бренданом Боуэрсом в 1992 году. Он отличается от множества Мандельброта тем, что в процессе итерации используются абсолютные значения действительной и мнимой частей z, что придаёт фракталу форму, напоминающую горящее судно.

Формула итерации для фрактала Burning Ship выглядит следующим образом: $z_{n+1} = (|Re(z_n)| + i|Im(z_n)|)^2$

## Исходный код программы визуализации фрактала Burning Ship на Python

```
import numpy as np
import matplotlib.pyplot as plt
```

```
from numba import jit


@jit(nopython=True)
def burning_ship(xmin, xmax, ymin, ymax, width, height, max_iter):
    real = np.linspace(xmin, xmax, width)
    imag = np.linspace(ymin, ymax, height)
    div_time = np.zeros((height, width), dtype=np.float64)

    for i in range(height):
        for j in range(width):
            c = complex(real[j], imag[i])
            z = complex(0, 0)
            n = 0
            while abs(z) <= 2 and n < max_iter:
                z = complex(abs(z.real), abs(z.imag))
                z = z * z + c
                n += 1
            if n < max_iter:
                div_time[i, j] = n + 1 - np.log(np.log2(abs(z)))
            else:
                div_time[i, j] = n
    div_time = div_time / div_time.max()
    return div_time


def plot_burning_ship(fractal, xmin, xmax, ymin, ymax, max_iter,
cmap='inferno'):
    plt.figure(figsize=(10, 10))
    plt.imshow(fractal, extent=[xmin, xmax, ymin, ymax], cmap=cmap,
interpolation='bilinear')
    plt.colorbar(label='Iterations')
    plt.title(f"Burning ship with max_iter = {max_iter}")
    plt.xlabel("Re(z)")
    plt.ylabel("Im(z)")
    plt.show()


def main():
    xmin, xmax = -2.0, -1.7
    ymin, ymax = -0.08, 0.025
    width, height = 1000, 1000
    max_iter = 1000

    fractal = burning_ship(xmin, xmax, ymin, ymax, width, height, max_iter)
    plot_burning_ship(fractal, xmin, xmax, ymin, ymax, max_iter)


if __name__ == "__main__":
    main()
```


Фрактал Burning Ship
0.02
0.00
−0.02
−0.04
−0.06
−0.08
Мнимая часть
−2.00
−1.95
−1.90
−1.85
−1.80
−1.75
−1.70
Действительная часть