UNIVERSIDADE FEDERAL DE PELOTAS
Curso de Graduação em Ciência da Computação

Implementação com visualização gráfica e duas linguagens de programação

T1 - Conceitos de Linguagem de Programação

Implementação de Fractal de Mandelbrot com Python e C

Pâmela Braga dos Santos
pamela.bds@inf.ufpel.edu.br

Pelotas, 2024

1. Fractal de Mandelbrot

A aplicação escolhida foi Fractal de Mandelbrot, que é um conjunto matemático de pontos no plano complexo que forma uma imagem infinitamente complexa e auto-semelhante. Ele é gerado através de uma iteração de uma função quadrática complexa, onde para cada ponto no plano, a função é iterada repetidamente, e o comportamento resultante determina se o ponto pertence ao conjunto de Mandelbrot ou não.

Se a magnitude do ponto não diverge para o infinito durante as iterações, ele é considerado parte do conjunto e é geralmente colorido em preto. Os pontos que divergem são coloridos com base na rapidez com que divergem, resultando em padrões altamente detalhados e coloridos (no caso desse projeto, foi realizado apenas em preto e branco), que exibem complexidade em qualquer nível de zoom.

2. Linguagens escolhidas

As linguagens escolhidas foram Python para a interação com o usuário, geração e visualização da imagem gerada do fractal. C foi utilizado por sua eficiência para o cálculo do fractal.

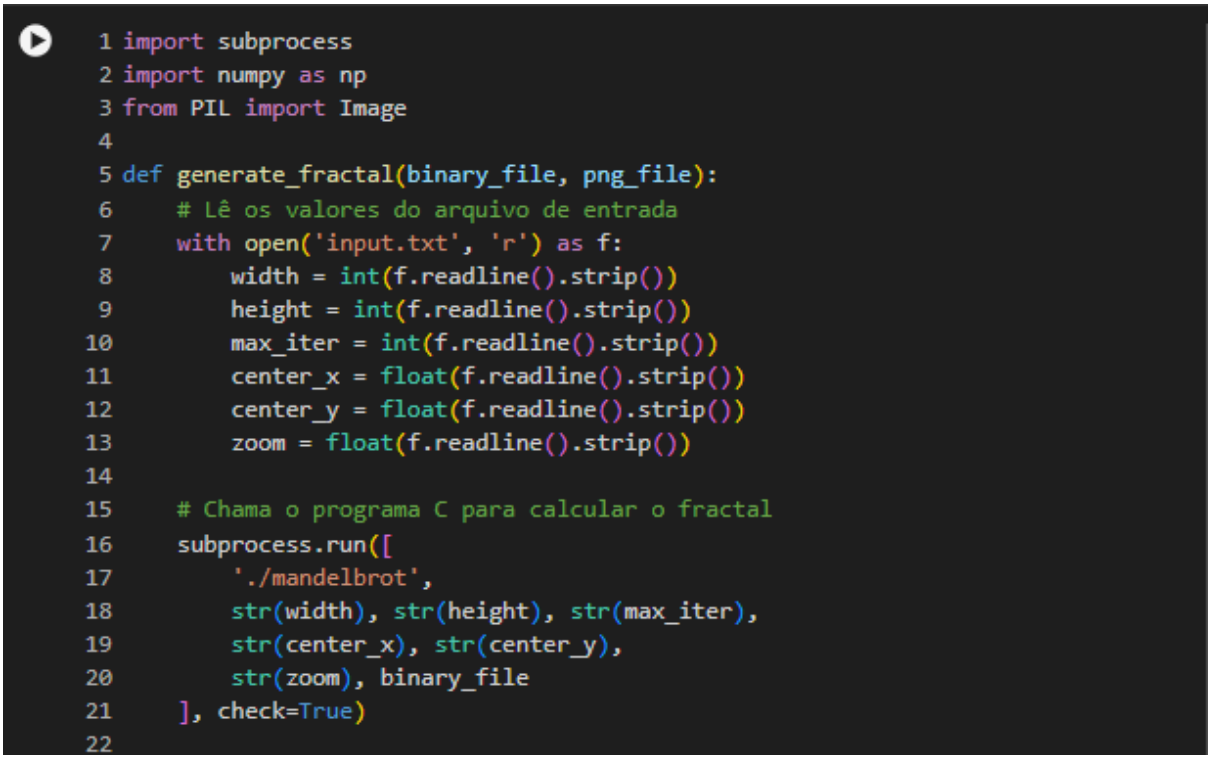
3. Subprocessos

No Python 3, temos a possibilidade de utilizar os subprocessos para invocar programas externos, podendo assim, chamar executáveis, invocar git, etc, no código Python. Nesse projeto, optei por utilizar o %%makefile para criar meu arquivo.c com as funções necessárias para fazer os cálculos:

```
%%writefile mandelbrot.c

#include <stdio.h>
#include <stdlib.h>

void mandelbrot(int width, int height, int max_iter, double center_x, double center_y, double zoom, const char* filename) {
    FILE* fp = fopen(filename, "wb");
    if (!fp) {
        perror("fopen");
        exit(EXIT_FAILURE);
    }

    double real_range = 4.0 / zoom;
    double imag_range = 4.0 / zoom;
    double real_min = center_x - (real_range / 2);
    double real_max = center_x + (real_range / 2);
    double imag_min = center_y - (imag_range / 2);
    double imag_max = center_y + (imag_range / 2);

```

Imagem retirada do Colab: projeto Fractal_Mandelbrot.pynb

Na célula 4, temos o código responsável pela interação com o usuário e geração da imagem a partir dos dados calculados. Para a integração entre as duas linguagens, e consequentemente o cálculo do fractal, foi utilizado o método *subprocess.run para*



```python
import subprocess
import numpy as np
from PIL import Image

def generate_fractal(binary_file, png_file):
    # Lê os valores do arquivo de entrada
    with open('input.txt', 'r') as f:
        width = int(f.readline().strip())
        height = int(f.readline().strip())
        max_iter = int(f.readline().strip())
        center_x = float(f.readline().strip())
        center_y = float(f.readline().strip())
        zoom = float(f.readline().strip())

    # Chama o programa C para calcular o fractal
    subprocess.run([
        './mandelbrot',
        str(width), str(height), str(max_iter),
        str(center_x), str(center_y),
        str(zoom), binary_file
    ], check=True)

```

.

Esse foi o método escolhido por motivos de eficiência e facilidade de implementação, que nesse caso, cumpriu com o propósito de maneira prática e sem necessidade de instalação de alguma ferramenta ou algo do tipo.