Fractal de Mandelbrot com Python e C

Nome: Henrique Betemps
Disciplina: Conceito de Linguagem de Programação

1. Aplicação Escolhida

A aplicação implementada é um visualizador para o **Fractal de Mandelbrot**. A escolha foi motivada pela natureza do problema, que possui uma clara separação de responsabilidades:

1. **Cálculo Numérico:** Uma tarefa computacionalmente intensiva, onde cada pixel da imagem final requer um número significativo de iterações e operações de ponto flutuante.
2. **Apresentação Gráfica:** Uma tarefa de I/O e gerenciamento de interface, que consiste em criar uma janela, alocar memória para a imagem e exibir o resultado para o usuário.

Essa dualidade torna o problema ideal para a abordagem proposta no trabalho, utilizando duas linguagens com vocações distintas.

2. Divisão de Tarefas e Linguagens

As responsabilidades do projeto foram divididas da seguinte forma:

- **Linguagem de Interface (Python):** Python foi escolhido para orquestrar a aplicação. Suas responsabilidades incluem:
  - Criar e gerenciar a janela da interface gráfica (usando a biblioteca `Tkinter`).
  - Alocar o buffer de memória para conter os dados da imagem.
  - Carregar e interagir com a biblioteca C.
  - Converter os dados brutos recebidos do C em uma imagem visível e apresentá-la na tela (usando a biblioteca `Pillow`).
  - **Justificativa:** Python é uma linguagem de alto nível excelente para desenvolvimento rápido, scripting e criação de interfaces de usuário, simplificando a gestão geral do programa.
- **Linguagem de Cálculo (C):** A linguagem C foi utilizada para implementar o "serviço de cálculo".
  - Sua única responsabilidade é a função `calculate_mandelbrot`, que realiza o loop de cálculo intensivo para cada pixel.

- O código C é compilado como uma biblioteca compartilhada (`.so` em sistemas Linux/macOS ou `.dll` em Windows).
- **Justificativa:** C é uma linguagem compilada de baixo nível, conhecida por seu desempenho excepcional em tarefas numéricas, tornando-a a escolha perfeita para a parte computacionalmente pesada do projeto.

3. Método de Interface entre as Linguagens

O método utilizado para a comunicação entre Python e C foi uma **Foreign Function Interface (FFI)**, provida pela biblioteca padrão do Python chamada `ctypes`.

O fluxo de interação ocorre da seguinte maneira:

1. **Compilação:** O código C é compilado de forma independente em uma biblioteca de vínculo dinâmico (`libmandelbrot.so`). Isso o torna um componente reutilizável que pode ser carregado por outros processos.
2. **Carregamento:** O script Python, ao ser executado, utiliza `ctypes.CDLL('./C/libmandelbrot.so')` para carregar a biblioteca C em seu próprio espaço de memória. A partir deste momento, as funções exportadas pela biblioteca C estão acessíveis ao Python.
3. **Alocação de Memória:** Python aloca um bloco de memória contíguo (`image_buffer`) com o tamanho exato necessário para a imagem (`WIDTH * HEIGHT` bytes). Esse buffer é criado com tipos compatíveis com C, usando `(ctypes.c_ubyte * size)()`.
4. **Chamada de Função e Passagem de Ponteiro:** O script Python chama a função C `calculate_mandelbrot` e passa os parâmetros necessários: a largura e altura da imagem, o número de iterações e, crucialmente, um **ponteiro** para o buffer de memória alocado no passo anterior.
5. **Execução Nativa:** A função C recebe o ponteiro e obtém acesso direto ao bloco de memória do Python. Ela executa o

cálculo de alta performance e escreve o valor de cinza de cada pixel diretamente nesse endereço de memória compartilhado.

6. **Retorno e Renderização:** Após a conclusão da função C, o controle retorna ao Python. O `image_buffer` agora está preenchido com os dados do fractal. A biblioteca `Pillow` é então utilizada para interpretar esses bytes brutos e criar um objeto de imagem, que é finalmente exibido na janela do `Tkinter`.