Estruturas de Dados

Matrizes: Processamento de Imagens

Davi Martins Senra, Vinícius Caetano

Planejamento

1. Antes de iniciar o desenvolvimento da atividade revisamos alguns conceitos que seriam necessários para executar a tarefa. Exemplos:
    a. Matrizes
    b. Ponteiros
    c. Leitura, escrita e manipulação de arquivos
    d. Modularização de funções
2. A primeira etapa consistiu na seleção das imagens e na conversão para o formato .pgm por meio de um pequeno script escrito em Python.
3. Em posse dos arquivos .pgm iniciamos o desenvolvimento do código em C por meio de pequenas funções e estruturas base. Exemplos:
    a. struct imagem (estrutura principal, responsável por representar um arquivo .pgm)
    b. ler_imagem (função que recebe uma string representando o caminho até o .pgm e retorna um struct “imagem”)

Planejamento

1. Tendo um .pgm carregado em memória por meio da função “ler_imagem”, demos prosseguimento ao desenvolvimento com o croqui da interface de usuário (CLI) e com os primeiros filtros.
2. Optamos por começar com o filtro de negativo e demos sequência com o filtro de espelhamento.
3. Com os filtros implementados iniciamos o desenvolvimento da função para gravar os resultados em disco, a função responsável por essa tarefa é a “gravar_imagem”, que recebe um *struct* do tipo “imagem” e uma *string* representando o caminho de *output*.
4. Por fim implementamos os filtros de borramento e por último o filtro de clareamento (*brightening*).

Decisões

1. Buscamos desenvolver um programa modular na medida do possível, evitando a repetição de código e o acoplamento das seções.
2. Separamos as principais funções em arquivos *header* externos, deixamos no *main* apenas o conteúdo principal de interação com o usuário e as chamadas para as funções.
3. Optamos por usar um *struct* para representar os arquivos .pgm, dessa forma temos uma “interface” de acesso aos dados unificada em todos os filtros e conseguimos mais legibilidade do código.

Desafios

1. Enfrentamos diversas adversidades na construção da aplicação, em parte pela dificuldade lógica do desafio e em outra parte (talvez a maior) pela pouca familiaridade com uma linguagem de mais baixo nível como o C.
2. Em diversos momentos o programa funcionava em uma máquina, porém não funcionava em outra, mesmo utilizando o mesmo compilador (GCC).
3. Além disso, mesmo quando o programa rodava de forma esperada, o GCC mostrou diversos *warnings* durante a compilação, porém entendemos que tratam-se de possíveis pontos de melhoria e não de erros em si.
4. Outro problema é que os erros de compilação nem sempre são claros sobre o motivo de ocorrerem, restando muita pesquisa e tentativa e erro para solucioná-los.

Sugestões

1. Talvez fosse interessante trabalhar com duas linguagens ao invés de construir a implementação apenas em C, dessa forma a diferença entre uma linguagem de baixo nível e uma de mais alto nível poderia ser elencada de forma mais nítida.
2. Por exemplo, implementar 2 dos 4 filtros em Python e 2 dos 4 filtros em C.
3. Ao final poderíamos comparar a diferença de código e de dificuldade na implementação.

Resultados

Negativo

Resultados

Espelhamento

Resultados

Borramento

Resultados

Brightening