Listas de imagens

Projeto 2 - Algoritmos e Técnicas de Programação 2

30 de outubro de 2023

1 Introdução

Listas encadeadas estão entre as estruturas de dados mais simples, mas isso não as torna simplistas. As aplicações das mesmas são das mais diversas, podendo ir desde a criação de *arrays* dinâmicos até a construção de bancos de dados.

Neste projeto, você deverá desenvolver uma estrutura de lista que será responsável por permitir a manipulação de imagens. A ideia é que as imagens sejam adicionadas à lista e, quando deseja-se realizar uma determinada operação sobre as imagens, esta é realizada sobre todas as imagens da lista. O software *Adobe Photoshop* implementa algo similar (embora muito mais avançado) chamado de *batch editing* (ver https://www.adobe.com/ph_en/creativecloud/design/discover/batch-edit.html).

Como já vimos em aula, isso será feito utilizando os recursos *argc* e *argv* para construir uma interface de usuário para gerenciamento da lista e aplicação das transformações às imagens. Dessa forma, o usuário poderá adicionar e remover itens da lista, além de aplicar as ditas transformações.

Diferentemente do primeiro projeto, neste você já conta com código relacionado ao mesmo, como é o caso das imagens e das listas. Você pode então utilizá-los como base e fazer as modificações plausíveis (inclusive nos códigos fornecidos pelo docente) para executar seu projeto.

Novamente, **compilação e comentários no código formam a base da sua nota e valem 3 pontos na sua nota final**. O restante da avaliação se dará de forma similar às atividades. A seguir, discute-se o que será implementado.

2 A estrutura básica de lista (3 pontos)

Você deve implementar uma estrutura de lista dinâmica na qual são armazenados os **nomes dos arquivos de imagem**. Ou seja, sua estrutura deve ser capaz de armazenar uma *string* com o nome do arquivo de imagem que deseja-se modificar.

Com base no que foi visto em aula, implemente as seguintes funcões:

- Uma função que cria a lista com um elemento inicial.
- Uma função que insere um elemento na lista
    1. Como trabalharemos apenas com imagens em formato *.pgm*, certifique-se sempre de que os arquivos passados possuem essa extensão, de forma a evitar a inserção de arquivos que não podem ser processados. Isto pode ser feito utilizando as funções de manipulação de *strings* vistas em aula.
- Uma função que remove um elemento da lista
- Uma função que imprime na tela **em ordem alfabética** as imagens que estão na lista. Você pode fazer isso de duas formas:
    1. A lista pode ser ordenada a cada chamada da função de impressão com um dos algoritmos vistos em aula.
    2. Você pode fazer uma função de inserção ordenada na lista, de forma que esta esteja sempre na ordem desejada.

3. Note que para fazer qualquer das opções acima você precisará de uma função de comparação de *strings*. Pense em como isso pode ser feito. Você, no entanto, não precisa pensar em como ordenar caracteres especiais, apenas letras e dígitos. Consultar a tabela ASCII pode ser útil para isso.

A interface com o usuário se dá da forma discutida anteriormente. Portanto, para cada uma destas funções será necessário criar um comando a ser utilizado a cada chamada do seu programa.

Note que os itens da lista armazenam apenas o nome (ou o endereço na memória) das imagens. A abertura e processamento das mesmas se dá em uma segunda etapa, quando o usuário chama uma das funções de gerenciamento de imagens.

3 A manipulação de imagens (2+1 pontos)

Vamos trabalhar com imagens *.pgm* com um único canal de cor, como já fizemos algumas vezes no decorrer do curso. Aqui, quando o usuário chamar uma transformação para as imagens, esta deve ser aplicada a todas as imagens da lista, o que pode ser feito iterando sobre a lista de imagens, transformando a mesma e em seguida salvando-a para um arquivo. Lembre-se sempre de gerenciar a memória utilizada.

Ao salvar a imagem modificada para o arquivo, adicione uma *tag* ao nome original da mesma de forma a distingui-la da imagem modificada. Em particular, crie uma *tag* para cada transformação. Novamente, você pode fazer isso utilizando as funções para manipulação de *strings* vistas em aula.

Quanto às transformações que você implementará, serão duas das já vistas no decorrer do custo e mais duas novas a serem discutidas a seguir. Considere então uma imagem de um canal $I_{m\times n}$ e com C tons de cinza. Implemente as funções a seguir:

- **Limiar**, na qual, para um valor L produz-se a imagem

$$I^{(L)}_{i,j} = \begin{cases} 0, \text{ se } I_{i,j} < L \\ C, \text{ caso contrário.} \end{cases} \tag{1}$$

- **Imagem inversa** (quanto às intensidades de cor), que resulta na imagem

$$I^{inv}_{i,j} = C - I_{i,j}. \tag{2}$$

- **Espelhamento vertical**, que produz a imagem

$$I^{ver}_{i,j} = I_{m-i-1,j}. \tag{3}$$

- E, analogamente, o **espelhamento horizontal**

$$I^{hor}_{i,j} = I_{i,n-j-1}. \tag{4}$$

Para receber um ponto a mais nessa parte da tarefa, faça a paralelização dos métodos de transformação discutidos utilizando os recursos do OpenMP. Não é necessário gerenciar o número de *threads* que será utilizado, apenas certifique-se de que o número máximo de *threads* será utilizado.

4 A interface de usuário (2 pontos)

Vimos em sala como utilizar os argumentos *argc* e *argv* para passar argumentos para a função *main* e como isto pode ser útil para programas que realizam manipulação de arquivos, como é o caso do programa que você irá elaborar neste projeto.

Portanto, para as funções pedidas anteriormente deverá ser criada uma interface de usuário na qual o usuário passa como argumentos da função *main* a função a ser chamada dentro do programa e também os argumentos da mesma (caso seja necessário).

Implemente também uma função de ajuda que imprime na tela os comandos que podem ser utilizados e certifique-se sempre de informar ao usuário qual erro foi cometido (quando ocorrer).