Documentação Trabalho Prático da Disciplina Estrutura de Dados – TW1

Samuel Assis Vieira – 2018109736

Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

samuelassis@dcc.ufmg.br

1. Introdução

As especificações do TP apresentavam um problema relacionado a medições de quantidades usando recipientes. Devia se fazer um programa que recebia do usuário os recipientes e calculava o número mínimo de operações para chegar no valor dado baseado nos recipientes disponíveis. O usuário poderia fazer 3 operações: adicionar recipientes, remover recipientes e calcular o número de operações para uma medição.

2. Implementação

Todo o código está feito em C++, a estrutura de dados utilizada para a resolução do problema foi uma Lista a qual implementei duplamente encadeada para facilitar a remoção de termos e a iteração da lista nos dois sentidos, início-fim e fim-início. Implementei como um *struct* para facilitar o acesso aos atributos em funções que não eram métodos da Lista.

2.1 Lista

A Lista foi implementada duplamente encadeada, os atributos são dois ponteiros para a estrutura Cell, um indicando o início e o outro o fim da lista, outro atributo é um inteiro que guarda o tamanho da lista. A estrutura Cell tem como atributos dois ponteiros para o elemento anterior e próximo da lista e dois inteiros, um guardando o valor e outro guardando o número de operações para aquele valor.

2.1.1 Métodos

- **Construtor:** A lista conta com um construtor vazio que inicializa a lista com uma célula cabeça e zera o atributo de número de elementos.
- **add_element():** Este método adiciona um elemento sempre no final da lista e recebe como parâmetro o valor da célula a ser adicionado e o número de operações necessárias para o valor.
- **remove_element():** Este método remove da lista a célula que contém o valor passado como parâmetro.
- **print_list():** Este método itera a lista e imprime o valor e o número de operações de cada célula da lista

2.2 Functions

O arquivo functions.cpp possui apenas a função operation(), que realiza o cálculo das operações baseado na lista de recipientes disponíveis e é a função principal do trabalho prático.

- **operation():** A função recebe um inteiro referente a quantidade a ser calculada e uma lista contendo os recipientes disponíveis. Primeiro é testado se a quantidade buscada não pode ser atingida com apenas uma operação, caso não seja, os valores são armazenados com o valor de operações incrementado em outra lista de operações acumuladas. Então começa um processo de soma e subtração dos elementos da lista de recipiente com os da lista de operações acumuladas. Sempre que o resultado dessas somas e subtrações é maior que zero, o valor é armazenado na lista de operações acumuladas com a quantidade de operações necessárias para aquele valor, é testado se o valor resultante da soma ou da subtração é o valor de interesse, o passado como parâmetro para a função, caso seja, os dois últimos elementos da lista de operações acumuladas são testados e é retornado a quantidade mínima de operações necessárias armazenadas na célula do valor de interesse.

2.3 Main

A função main do programa cria uma Lista para os recipientes e lê as entradas do usuário, em caso de adição ou remoção de recipiente, chama os métodos respectivos da Lista e em caso de calcular uma quantidade chama a função operation(), passando a entrada e a Lista com os recipientes

3. Instruções de Compilação e Execução

O programa é compatível apenas para sistemas Mac e Linux. Deve ser feito o Download do arquivo samuel_assis.zip na máquina desejada e extrair o arquivo. Após isto, deve acessar o terminal, realizar o caminho até a pasta samuel_assis/src, e digitar o comando "make". Em algumas versões de Linux e Mac, deve ser acrescentado "-B" junto ao comando make. Tendo isto concluído, o executável será gerado na pasta "bin". Para rodar o programa sem trocar de diretório é só utilizar "./tp1" (sem as aspas). Para a realização dos testes a compilação é feita de forma semelhante, porém, adicionando "test" depois do make.

5. Conclusão

A implementação das estruturas de dados necessárias foi feita sem muitas dificuldades, a maior dificuldade foi abstrair o problema e converter isso para código na implementação da função operation(). Os resultados do programa foram satisfatórios, mas creio que poderiam ser ainda mais otimizados quanto a sua complexidade. A quantidade de memória escala rapidamente.