MC202GH - Estrutura de Dados - Turmas G e H

Laboratório 6 - *Loja de Elementos Químicos*

Docente: Marcelo da Silva Reis

Monitores PED: Cristiano Gabriel de Souza Campos
Heigon Alafaire Soldera Pires

Monitores PAD: Gabryel Rodrigues Alves Da Silva
Hermes Shimidu

13 de novembro de 2023

Data de entrega: 22/11/2023

Entrega no GitHub Classroom [1]

Informações gerais

Neste laboratório, você será encarregado de desenvolver um sistema de gerenciamento de estoque para uma loja de elementos químicos. Utilizando seus conhecimentos em estruturas de dados, mais especificamente Árvores Binárias de Busca (ABB), você irá organizar e manipular o estoque de elementos baseado em seus números atômicos. Além de implementar funções básicas de compra e venda, seu sistema deverá fornecer recursos para impressão e verificação de elementos no estoque

Observações importantes:

1. Neste laboratório será permitido o uso apenas das bibliotecas `stdio.h`, `stdlib.h`, `stdlib.h` e `string.h`.
2. Para compilar sua solução, utilize o `Makefile`, chamando no terminal o comando `make`.

[1] classroom.github.com/a/E_36Mp9D.

TAD para manipulação da loja de elementos químicos

Neste projeto, você irá construir e manipular um inventário digital para uma loja de elementos químicos, aplicando o conceito de Árvore Binária de Busca (ABB) para organizar os elementos pelo número atômico. A representação de cada elemento químico na loja é feita através de um nó de árvore, que contém o número atômico e referências para os nós filhos esquerdo e direito.

A estrutura de dados para o nó e o ponteiro para o nó são definidos como segue:

- Símbolo do elemento químico
- Número atômico do elemento

Com esta estrutura estabelecida, você implementará as seguintes operações fundamentais para o gerenciamento do estoque:

- `p_no criar_arvore()` - Inicializa uma árvore vazia, retornando um ponteiro para a raiz.
- `void destruir_arvore(p_no raiz)` - Desaloca todos os nós da árvore, iniciando pela raiz, para evitar vazamentos de memória.
- `p_no inserir(p_no raiz, int numero_atomico, char *simbolo)` - Insere um novo elemento na árvore, dado pelo número atômico, e retorna o ponteiro para a raiz atualizada.
- `p_no remover(p_no raiz, char *simbolo)` - Remove um elemento da árvore, especificado pelo nome, e retorna o ponteiro para a raiz atualizada.
- `p_no buscar(p_no raiz, int chave)` - Busca um elemento na árvore pelo número atômico e retorna um ponteiro para o nó correspondente.
- `p_no minimo(p_no raiz)` - Retorna um ponteiro para o nó com o menor número atômico na árvore.
- `p_no maximo(p_no raiz)` - Retorna um ponteiro para o nó com o maior número atômico na árvore.

As funções acima deverão ser implementadas e testadas rigorosamente para assegurar que o gerenciamento do estoque seja feito de forma correta e eficiente. A árvore deve ser capaz de se ajustar dinamicamente às operações de inserção e remoção, mantendo o equilíbrio e a ordem dos elementos pelo número atômico.

Questão 1

Implemente as operações de estoque que permitirão ao usuário comprar e vender elementos químicos, bem como verificar o estado atual do estoque. As operações e seus comandos de entrada correspondentes são:

- **C** Número Atômico Elemento: Insere um novo elemento na árvore, onde Número Atômico é o número atômico do elemento a ser inserido e Elemento é o nome do elemento químico.
- **V** Elemento: Remove um elemento da árvore, indicado pelo nome do elemento químico.
- **E**: Imprime o estoque organizado em ordem crescente de número atômico.
- **I** Número Atômico Elemento: Verifica se possui o item desejado.
- **MAX** : Imprime elemento de maior número atômico na loja
- **MIN** : Imprime elemento de menir número atômico na loja

Exemplo de entrada padrão:

```
C 12 Mg
C 20 Ca
C 48 Cd
I 3 Li
C 3 Li
C 11 Na
C 31 Ga
V Ca
C 49 In
I 12 Mg
V Mg
E
C 14 Si
E
MIN
MAX
```

Saída esperada para o exemplo fornecido:

```
Nao
Sim
Li, Na, Ga, Cd, In
Li, Na, Si, Ga, Cd, In
Li
In
```

Após codificar o cliente e também as operações necessárias na implementação, compile tudo utilizando o `Makefile` e teste executando o seguinte comando:

```
./cliente.bin < teste_Q1.in
```

onde `teste_Q1.in` é um arquivo contendo uma sequência de comandos como a exemplificada acima.