Trabalho Prático 3 - Gerenciando Produtos do Estoque

Data de entrega: domingo, 18 Ago 2024, 10:37
Número máximo de arquivos: 1
Tipo de trabalho: Trabalho individual

Universidade Federal de Minas Gerais

Departamento de Ciência da Computação

Programação e Desenvolvimento de Software I

Trabalho Prático 3 - Gerenciando Produtos do Estoque

Você trabalha no departamento de TI de uma empresa de varejo que vende produtos de diversos departamentos distintos, como eletrônicos, comidas, itens de higiene e outros. Nos últimos anos a companhia passou por crises e problemas de orçamento, então a chefia deseja implementar processos mais eficientes dentro da organização para reduzir os gastos e aumentar os lucros. Neste trabalho prático, a ser realizado por meio de código escrito na linguagem C, você deverá implementar as alterações propostas pelos executivos com o objetivo de treinar, por meio da aplicação prática, os conceitos de apontadores, alocação dinâmica e recursividade, além de conceitos cumulativos praticados anteriormente. **Leia este arquivo até o final para não perder informações importantes!**

Sua área é responsável pelo sistema que gerencia o estoque da loja, que é um dos alvos de mudança dos diretores. Eles desejam que o estoque não seja mais implementado como um vetor, e sim como uma **Árvore Binária de Busca (ABB)**. Uma ABB é uma estrutura com as seguintes características:

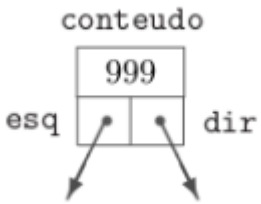
- formada por estruturas chamadas **nódulos** ou **nós**;
- cada nó possui uma informação (que pode ser um inteiro, um char ou mesmo outra estrutura) e dois ponteiros para outros dois nós: um nó esquerdo e um nó direito;
- **não há** informações duplicadas;
- ao inserir uma nova informação, deve-se comparar seu valor com os valores existentes na árvore. Se o novo valor for **menor** que o valor existente, a nova informação deve ser posicionada na **esquerda**. Se o novo valor for **maior** que o valor existente, a nova informação deve ser posicionada na **direita**;

- os nós apontados por um nó são chamados de **filhos**, e o nó que aponta é o nó **pai**. Com exceção do primeiro nó da árvore, que não possui pai, todos os nós possuem exatamente um pai e **podem ter, no máximo, 2 filhos**;
- você deve inserir uma informação nova somente em um ponteiro vazio (nulo) da árvore. Enquanto um nódulo tiver alguma informação, você deve continuar comparando o valor da nova informação com o valor do nódulo atual para definir se deve caminhar para a esquerda ou direita da árvore.

Observe as seguintes imagens ilustrando a implementação de um nódulo.

```
typedef struct reg {
   int  conteudo; // conteúdo
   noh *esq;
   noh *dir;
} noh; // nó
```

Implementação da estrutura de um nó.



Representação visual de um nó.

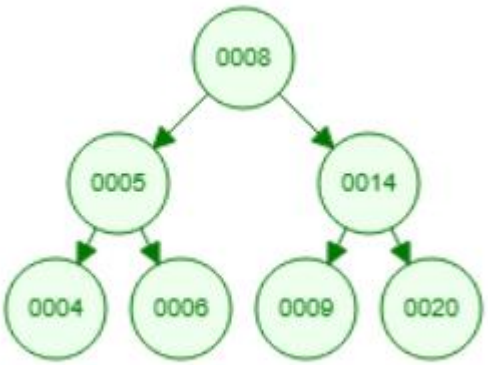
Vamos inserir alguns valores em uma árvore para exemplificar. Temos uma árvore vazia e inicialmente colocamos o valor 8:

Vamos agora inserir o valor 14:

O valor 14 é maior do que 8, então caminha para a direita da árvore. Como o filho da direita do 8 é vazio, pode ser colocado nessa posição. Vamos agora inserir o valor 5:

5 é menor do que 8, então caminhamos para a esquerda. O filho da esquerda do 8 ainda está vazio, então podemos inserir nessa posição. Adicionaremos o valor 9:

9 é maior do que 8, então vamos para a direita. O filho direito de 8 não é vazio, então precisamos comparar com o valor existente nessa posição, que é o 14. 9 é menor do que 14, então caminhamos para a esquerda. O filho esquerdo de 14 é vazio, então podemos inserir. Vamos inserir agora os valores 4, 6 e 20, nessa ordem, para finalizar nossa árvore:



Novas inserções seguirão as mesmas regras que vimos até agora, caminhando pela árvore até encontrar uma posição vazia e decidindo entre ir para direita ou esquerda a depender do novo valor ser maior ou menor que o atual.

Para a loja que você trabalha, a informação será um **Produto**, contendo um identificador único (inteiro positivo), um nome (uma única palavra com no máximo 50 caracteres), um departamento (uma única palavra com no máximo 50 caracteres) e um preço (valor não-negativo de ponto flutuante). As comparações para inserção na árvore devem ser realizadas com base no **identificador** do produto.

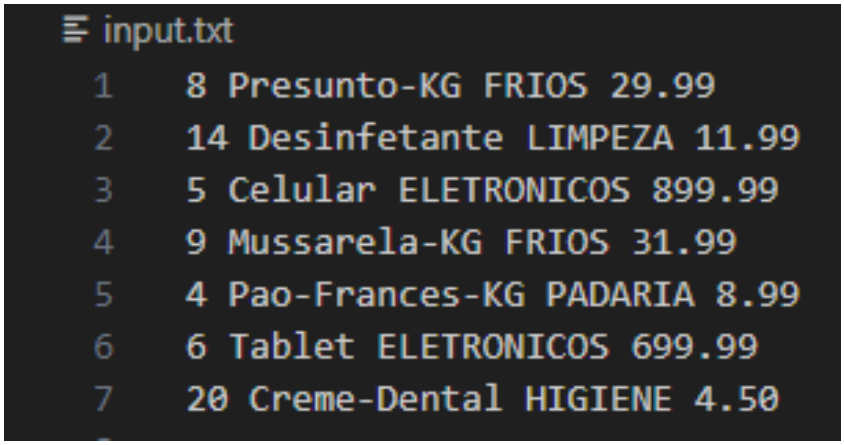
Você também deve implementar algumas operações para atuar sobre o estoque, que serão descritas de forma mais detalhada a seguir:

1. Procurar produto por ID
2. Procurar produtos de um departamento
3. Inserir um novo produto
4. Filtrar produtos por preço

Seu programa sempre será executado passando um argumentos pela linha de comando, que será um arquivo TXT com as informações sobre os produtos já existentes no estoque (ou seja, o arquivo de entrada). Exemplo de execução:

./tp2 input.txt

O arquivo de entrada possui um produto por linha, com as informações dispostas na seguinte ordem: identificador, nome, departamento e preço. O arquivo termina com EOF (end of file). Segue o exemplo de um arquivo de entrada:



Após abrir o arquivo de entrada, ler e armazenar os dados, seu programa deve imprimir o seguinte menu:

1 - Procurar por ID

2 - Procurar por Departamento

3 - Inserir Produto

4 - Filtrar Produtos por Preço

5 - Sair

Ele deve então ler e tratar a opção fornecida pelo usuário. Caso seja escolhida uma das operações entre a 1 e a 4, seu programa deve **aguardar uma nova instrução** após finalizar a operação atual. O programa encerra-se **somente** após o usuário escolher a operação 5.

Operação Escolhida	**Entrada Fornecida**	**Saída Esperada**
1	ENTRADA PADRÃO inteiro identificador do produto desejado	SAÍDA PADRÃO (Terminal): Para cada produto no caminho até encontrar o produto desejado: (Departamento) Produto - R$ preço.00

		Se encontrou o produto: imprime na mesma formatação acima. Se não encontrar, imprimir: Produto nao encontrado!
2	ENTRADA PADRÃO departamento desejado	SAÍDA PADRÃO: Para cada produto do departamento, ordenados por ordem crescente de ID: (Departamento) Produto - R$ preço.00 Se não encontrar nenhum, imprimir: Departamento vazio!
3	ENTRADA PADRÃO: id nome departamento preço	nenhuma, o produto deve ser inserido corretamente
4	orçamento (valor de ponto flutuante)	Para cada produto abaixo ou igual ao orçamento, ordenados por ordem crescente de preço: (Departamento) Produto - R$ preço.00 Se não encontrar nenhum, imprimir: Sem resultados para o filtro!
5	nenhuma.	nenhuma. O programa encerra.

Informações Importantes:

- **identificadores** e **preços** não serão repetidos;
- preços devem ser impressos com duas casas decimais;
- a chefia deseja atualizar os modelos iterativos para algo inovador, então **todas** as partes de inserção/busca na Árvore Binária devem ser feitas **recursivamente**;
- na operação 1 (busca por identificador), os produtos devem ser impressos na ordem em que se caminha pela árvore;
- na operação 2 (busca por departamento), os produtos devem ser impressos em **ordem crescente de identificador**;
- na operação 3 (filtro por preço), os produtos devem ser impressos em **ordem crescente de preço**;
- as operações de ordenação **podem** ser feitas de forma iterativa;
- a chefia comunicou que os sistemas atuais consomem muita memória e que este recurso é escasso. Assim, nas operações que podem envolver vetores, (como a 2 e a 3), você **não pode** criar um vetor enorme com posições sobrando. Ao invés disso, você deve alocá-lo dinamicamente e realocá-lo conforme necessário.

Exemplo de entradas e saídas esperadas para uma execução do programa. Considere como entrada a imagem do arquivo input.txt exibida mais acima.

./tp3 input.txt

ENTRADA PADRÃO	SAÍDA PADRÃO
2 FRIOS 1 25 2 PADARIA 4	1 - Procurar por ID 2 - Procurar por Departamento 3 - Inserir Produto 4 - Filtrar por Preco 5 - Sair (FRIOS) Presunto-KG - R$ 29.99 (FRIOS) Mussarela-KG - R$ 31.99

10.00	(FRIOS) Presunto-KG - R$ 29.99
2	(LIMPEZA) Desinfetante - R$ 11.99
DECORACAO	(HIGIENE) Creme-Dental - R$ 4.50
3	Produto nao encontrado!
2 Salame FRIOS 7.99	(PADARIA) Pao-Frances-KG - R$ 8.99
3	(HIGIENE) Creme-Dental - R$ 4.50
25 Apresuntado-KG FRIOS 24.99	(PADARIA) Pao-Frances-KG - R$ 8.99
2	Departamento vazio!
ELETRONICOS	(ELETRONICOS) Celular - R$ 899.99
1	(ELETRONICOS) Tablet - R$ 699.99
25	(FRIOS) Presunto-KG - R$ 29.99
1	(LIMPEZA) Desinfetante - R$ 11.99
20	(HIGIENE) Creme-Dental - R$ 4.50
4	(FRIOS) Apresuntado-KG - R$ 24.99
50.00	(FRIOS) Presunto-KG - R$ 29.99
2	(LIMPEZA) Desinfetante - R$ 11.99
FRIOS	(HIGIENE) Creme-Dental - R$ 4.50
5	(HIGIENE) Creme-Dental - R$ 4.50
	(FRIOS) Salame - R$ 7.99
	(PADARIA) Pao-Frances-KG - R$ 8.99
	(LIMPEZA) Desinfetante - R$ 11.99
	(FRIOS) Apresuntado-KG - R$ 24.99
	(FRIOS) Presunto-KG - R$ 29.99

	(FRIOS) Mussarela-KG - R$ 31.99 (FRIOS) Salame - R$ 7.99 (FRIOS) Presunto-KG - R$ 29.99 (FRIOS) Mussarela-KG - R$ 31.99 (FRIOS) Apresuntado-KG - R$ 24.99

Avisos Importantes após o TP1/TP2:

- imprima na saída padrão ou no arquivo de saída **exatamente** o que está especificado no enunciado. Evite impressões do tipo "Digite sua escolha", visto que parte da correção é automática e ela pode identificar tais mensagens como sendo parte da sua resposta, resultando em avaliação incorreta;
- você **não precisa** tratar possíveis entradas incorretas, a não ser que esteja especificado no enunciado que isso deve ser feito. Se o enunciado diz que pode ser escolhida uma opção entre 1 e 7, não é necessário tratar o envio de valores abaixo de 1 ou acima de 7;
- nós apoiamos que os estudantes busquem deixar o programa mais interativo para o usuário e incentivamos que eles tentem adicionar mensagens informativas e tratamento de dados, mas **façam isso fora do ambiente de avaliação**, como um exercício extra, para que a avaliação não seja prejudicada.