

Exercícios: Árvores

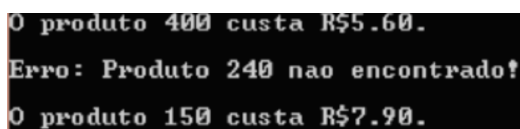
1. Utilizando os conceitos de TAD crie uma estrutura do tipo Árvore Binária de Busca para armazenar produtos de uma loja de conveniência. A ordenação da árvore será pelo número de matrícula dos produtos, que serão guardados na seguinte estrutura:

```
struct Produto{
    int matricula;
    float preco;
};
```

Executando o seguinte código:

```
int adicionar (No** arv, int matricula, float preco);
int preco(No** arv, int matricula);
int main() {
    No* arv = NULL;
    int errorCode;
    errorCode = adicionar(&arv, 100, 5.90);
    errorCode = adicionar (&arv, 200, 8.40);
    errorCode = adicionar (&arv, 250, 2.20);
    errorCode = adicionar (&arv, 500, 3.80);
    errorCode = adicionar (&arv, 400, 5.60);
    errorCode = adicionar (&arv, 150, 7.90);
    errorCode = preco(&arv, 400);
    errorCode = preco(&arv, 240);
    errorCode = preco(&arv, 150);
}
```

A saída será a seguinte:



```
O produto 400 custa R$5.60.
Erro: Produto 240 nao encontrado!
O produto 150 custa R$7.90.
```

- Crie uma *struct* para abrigar um nó da árvore e defina-a como tipo No. Esta *struct* deve conter um produto e os dois ponteiros correspondentes aos nós filhos.
- Escreva a função *adicionar*, que cria e insere um nó na árvore binária, indexando o produto pelo número de sua matrícula.
- Escreva a função *preco*, que busca na árvore binária pelo número de matrícula do produto desejado e imprime na tela seu preço correspondente.
- Escreva uma função que conte o número de nós da árvore.
- Escreva uma função que conte o número de folhas da árvore.
- Escreva uma função que calcula a altura da árvore.

g) Escreva uma função para achar o maior preço.

h) Escreva uma função que receba a matrícula do produto e faça a exclusão do mesmo na árvore.

2. Utilizando Suponha que produtos com códigos 770, 875, 007, 059, 068, 682, 588, 067, 234, 411, 191 e 512 sejam inseridos, nesta ordem, em uma estrutura vazia do tipo Árvore Binária de Busca.

a) Desenhe a árvore resultante (grafo) após a inserção de todos os itens, representando o código do produto dentro de cada nó.

b) Determine em que sequencia esses elementos seriam processados por um algoritmo que execute um percurso em pré-ordem.

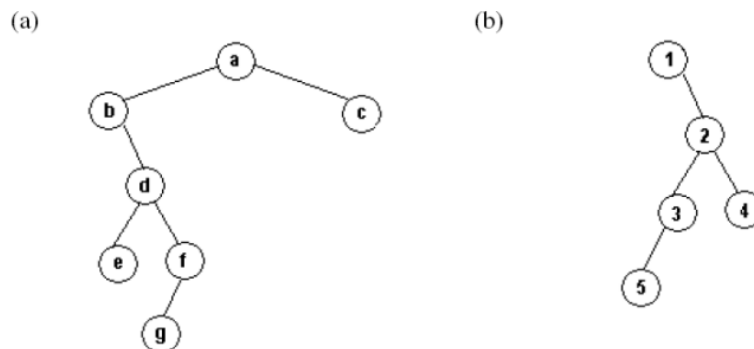
c) Determine em que sequencia esses elementos seriam processados por um algoritmo que execute um percurso em pós-ordem.

d) Determine em que sequencia esses elementos seriam processados por um algoritmo que execute um percurso em ordem simétrica (in-ordem).

3. Defina árvore AVL.

4. Escreva uma função que verifique se uma árvore é AVL.

5. Dada as seguintes árvores binárias abaixo, indique os passos para torná-las uma árvore binária balanceada (AVL).



6) Insira os números 35, 39, 51, 20, 13, 28, 22, 32, 25, 33 (nesta ordem) em uma árvore AVL.

7) Dê um exemplo de inserção de um elemento em uma árvore AVL que cause rearranjo da estrutura da árvore.

8) Dê um exemplo de remoção de um elemento de uma árvore AVL que cause rearranjo da estrutura da árvore.

9) Por que nos damos ao trabalho de procurar trabalhar com árvores binárias balanceadas? Justifique.