Escreva uma implementação de dicionário baseada em árvore binária de busca.

Você deverá definir um tipo Dicio<TC,TV> para representar dicionários cujas chaves têm tipo TC e os valores tipo TV. Esse tipo deve possuir um tipo interno Dicio<TC,TV>::Iterador que será usado nas operações abaixo, que devem ser implementadas como "métodos" (funções-membro):

a) Construtor: o construtor deve deixar o dicionário no estado vazio.

b) Iterador inserir (TC c, TV v): insere a chave "c" e o valor "v", retornando um iterador referente a esse par (na prática, esse iterador encapsulará um ponteiro para o nó correspondente na árvore).

c) Iterador procurar (TC c): procura a chave "c" no dicionário, retornando o iterador correspondente (que pode "apontar" para o "fim", caso a chave não esteja presente).

d) void remover (Iterador i): remove do dicionário a chave e o valor correspondentes ao iterador recebido. Não faz nada se o iterador aponta para o fim.

e) Destrutor: desaloca a árvore utilizada para representar o dicionário.

f) Iterador inicio (): retorna um iterador apontando para o nó de menor chave. OBSERVAÇÃO: procure implementar de forma que esta função execute em tempo O(1) no pior caso.

g) Iterador fim (): retorna um iterador "apontando para o fim" (algo que indique não aponte para nenhum dos elementos do dicionário, e que também possa ser utilizado para indicar chave não presente numa busca, etc).

O tipo Iterador deverá possuir pelo menos as seguintes operações:

a) void operator++ (): passa ao próximo elemento (menor dos elementos que são maiores que o atual), ou então ao "fim".

b) bool operator!= (const Iterador &i): informa se o iterador atual é diferente do iterador "i".

c) TV operator* (): retorna o valor do par chave-valor relativo ao iterador.