CMP1054 - Estrutura de Dados I
6ª Lista de Exercícios - Árvores Binárias de Busca
Max Gontijo de Oliveira

- Todos os métodos e funções criados nas questões deverão ser testadas em um programa principal (`main`).
- Caso haja necessidade, crie parâmetros adicionais para os métodos ou funções recursivos além dos explicitamente solicitados nas questões.
- Quando não estiver explícito na questão, o aluno poderá escolher entre criar um algoritmo iterativo ou recursivo.
- Em todas as questões, considere as classes `Node` e `Arvore` conforme visto em sala de aula.

```
class Node {
   public:
      CHAVE x;
      TIPO valor;
      Node *esq;
      Node *dir;
};

class Arvore {
        private:
                Node *raiz;

                void inserir_rec(Node*, CHAVE, TIPO);
                void imprimir_rec2(Node*);
                void imprimir_invertido_rec(Node*);

        public:
                Arvore() : raiz(NULL){} // Inicializa inicio e fim com NULL
                void inserir(CHAVE, TIPO);
                void inserir_rec(CHAVE, TIPO);
                TIPO buscar(CHAVE);
                void imprimir();
                void imprimir_invertido();
                TIPO proximo(CHAVE);
                TIPO anterior(CHAVE);
                int quantidade_elementos();
                int altura();
                TIPO remover(CHAVE);

                // Outros métodos
};
```

- Na classe `Node` acima, `TIPO` se refere ao tipo de dado que a árvore deverá armazenar (`int`, `char`, `float`) enquanto `CHAVE` se refere ao tipo de dado do valor que será utilizado para montar a árvore (geralmente, `int`, mas pode ser qualquer tipo que possa ser usado em comparações). Todavia, NÃO é obrigatório o uso de `template`. Assim, se o aluno preferir, pode criar uma classe `Node` e uma classe `Arvore` utilizando apenas o tipo `int` para `CHAVE` e `VALOR`.

1. Implemente os seguintes métodos da classe `Arvore`:
   (a) `inserir`: deve inserir um elemento na árvore por meio de um algoritmo iterativo.
   (b) `inserir_rec`: deve inserir um elemento na árvore por meio de um algoritmo recursivo. Crie e utilize um método adicional para auxiliar nessa tarefa (`inserir_rec2`).
   (c) `buscar`: deve buscar na árvore o elemento cujo valor da chave seja igual ao passado por parâmetro. Caso não encontre, o método deverá retornar `NULL` (ou lançar uma exceção).
   (d) `imprimir`: deve imprimir todos os elementos da árvore em ordem crescente pela chave por meio de um algoritmo recursivo. Crie e utilize um método adicional para auxiliar nessa tarefa (`imprimir_rec`).
   (e) `imprimir_invertido`: deve imprimir todos os elementos da árvore em ordem decrescente pela chave por meio de um algoritmo recursivo. Crie e utilize um método adicional para auxiliar nessa tarefa (`imprimir_invertido_rec`).
   (f) `proximo`: deve retornar o valor do elemento cuja chave seja imediatamente posterior à chave passada por parâmetro. Caso a chave passada por parâmetro seja a maior da árvore e, por consequência, não tenha elemento posterior, o método deve retornar `NULL`;
   (g) `anterior`: deve retornar o valor do elemento cuja chave seja imediatamente anterior à chave passada por parâmetro. Caso a chave passada por parâmetro seja a maior da árvore e, por consequência, não tenha elemento posterior, o método deve retornar `NULL`;

(h) `quantidade_elementos`: deve contar todos os elementos da árvore e retornar a quantidade total. Dica: utilize um algoritmo recursivo parecido com o de imprimir os elementos.

(i) `altura`: deve retornar a altura da árvore (quantidade de níveis). Dica: utilize um algoritmo recursivo.

(j) `remover`: deve encontrar e remover o elemento cujo valor da chave seja igual ao passado por parâmetro. Note que é necessário desalocar o `Node` do elemento encontrado e, quando removido, a árvore deverá manter as restrições da estrutura. O método deverá ainda retornar o valor do elemento removido. Caso não encontre, o método deverá retornar `NULL` (ou lançar uma exceção).

2. Questão para testar sua árvore: crie um PROGRAMA que instancie uma classe `Arvore` para armazenar números inteiros (essa classe deve ter os métodos criados na questão 1) e, em seguida, execute os seguintes passos:

   1. *Programa lê do teclado um número inteiro OP.*
      (a) *Se $OP = 1$:*
         i. *programa deverá ler um inteiro $X > 0$*
         ii. *se $X > 0$, o programa deverá inserir o valor X na árvore (X será usado como chave e valor) usando o método iterativo (`inserir`).*
         iii. *programa ficará lendo valores e inserindo enquanto $X > 0$.*
      (b) *Se $OP = 2$:*
         i. *programa deverá ler um inteiro $X > 0$*
         ii. *se $X > 0$, o programa deverá inserir o valor X na árvore (X será usado como chave e valor) usando o método recursivo (`inserir_rec`).*
         iii. *programa ficará lendo valores e inserindo enquanto $X > 0$.*
      (c) *Se $OP = 3$ o programa deverá ler um inteiro $X > 0$ (chave), buscar o elemento na árvore e exibir na tela o valor consultado (`buscar`). Caso não encontre, informar ao usuário.*
      (d) *Se $OP = 4$ o programa deverá exibir na tela todos os elementos da árvore em ordem crescente (`imprimir`).*
      (e) *Se $OP = 5$ o programa deverá exibir na tela todos os elementos da árvore em ordem decrescente (`imprimir_invertido`).*
      (f) *Se $OP = 6$ o programa deverá ler um inteiro $X > 0$ (chave), buscar e imprimir na tela o valor do elemento cuja chave seja imediatamente posterior ao elemento de chave X (`proximo`). Caso não encontre o elemento de chave X, informar ao usuário. Informe também, caso encontre mas o elemento não tenha posterior.*
      (g) *Se $OP = 7$ o programa deverá ler um inteiro $X > 0$ (chave), buscar e imprimir na tela o valor do elemento cuja chave seja imediatamente anterior ao elemento de chave X (`anterior`). Caso não encontre o elemento de chave X, informar ao usuário. Informe também, caso encontre mas o elemento não tenha anterior.*
      (h) *Se $OP = 8$ o programa deverá imprimir na tela a quantidade de elementos da árvore (`quantidade_elementos`).*
      (i) *Se $OP = 9$ o programa deverá imprimir na tela a altura da árvore (`altura`).*
      (j) *Se $OP = 10$ o programa deverá ler um inteiro $X > 0$ (chave), buscar o elemento na árvore, remover da árvore e exibir na tela o valor do elemento removido (`remover`). Caso não encontre, informar ao usuário.*
      (k) *Se $OP = 11$ o programa deverá exibir na tela a árvore de forma gráfica. Pode ser em modo texto ou utilizando alguma biblioteca de desenho, como QT ou GTK.*[1]
      (l) *Se $OP = 0$ o programa deve finalizar.*
      (m) *Se OP for qualquer outro valor, deverá ser ignorado.*
   2. *Após a execução da operação determinada por OP, se $OP = 0$, então o programa deverá finalizar; caso contrário, o programa voltará a execução no passo 1.*

[1] O professor vai disponibilizar o código fonte para a realização desse desenho em modo texto e utilizando QT, caso o aluno não queira gastar tempo com essa atividade.