Módulo 1 - Estruturas básicas - Lista #2

Data de entrega: 2 de abril de 2017

Acompanham esta lista os arquivos **mod1_lista2.h** e **mod1_lista2.cpp**, contendo as declarações iniciais. Modifique estes arquivos para implementar as questões pedidas e envie-os de volta zipados com nome no padrão <numero_matricula>.zip por email para **profs-eda@tecgraf.puc-rio.br**, com o assunto **[EDA] Lista 2**. Atenção: Crie um arquivo contendo a função main do seu programa para testar suas implementações, mas envie SOMENTE os arquivos e as classes solicitadas.

---

1. Implemente as funções da classe Abb, que representa uma árvore binária de busca. Cada nó da árvore é representado por uma struct que contém o valor do inteiro e os ponteiros para os elementos up (pai), left (filho esquerdo) e right (filho direito).

```
#pragma once

#include <string>

struct Node
{
    int _key;
    Node* _up;
    Node* _left;
    Node* _right;
};

class Abb
{
public:
    //Cria uma arvore vazia
    Abb();

    //Cria uma arvore com um primeiro elemento
    Abb(int key);

    //Cria uma arvore a partir de outra
```

```
    Abb(const Abb& orig);

    //Destroi a arvore
    ~Abb();

    //Insere um elemento na arvore
    void insert(int key);

    //Remove o elemento key da arvore, retornando
    //true se ele existia e false caso contrario.
    bool remove(int key);

    //Exibe os elementos da arvore utilizando
    //o padrao <...>
    void show(const std::string& title);

    //Exibe os elementos da arvore em ordem
    //utilizando o padrao <...>
    void order(const std::string& title);

    //Retorna o menor elemento da arvore
    int min();

    /**** Funcoes para percorrer a arvore: ****/

    //Posiciona o cursor no primeiro elemento
    bool first();

    //Posiciona o cursor no ultimo elemento
    bool last();

    //Atualiza o no cursor para o proximo elemento
    bool next();

    //Atualiza o no cursor para o elemento anterior
    bool prev();

    //Exibe o valor do no cursor atual
    int value();

    /*****************************************/

private:
    //Busca o no que contem a chave 'key'
    Node* search(int key);
```

```
    //Busca recursivamente o elemento 'key'
    //a partir de 'node'
    Node* searchrec(Node* node, int key);

    //Deleta recursivamente a partir
    //do no passado como argumento
    void deleterec(Node * root);

    //Insere recursivamente o elemento 'key'
    //a partir de 'node'
    Node* insertrec(Node* a, int key);

    //Funcao recursiva para exibir a arvore
    void showrec(Node * a);

    //Funcao recursiva para exibir a arvore
    //em ordem
    void orderrec(Node * a);

    //Remove no simples (que tem no
    //maximo uma subarvore)
    void removesimplenode(Node* node);

private:
    Node* _root;
    Node* _cursor;

};
```