LABORATORIES 3

Dictionary | AVL Tree

Wojciech Marosek
ID: 295818

1. FILES

--- Dictionary.h
--- main.cpp
--- tests.h

File **Dictionary.h** contains Dictionary class with class Iterator. Moreover, it contains their methods and operators implementations . **Tests.h** keeps tests of class methods(declaration and implementation). In addition, exception handling is provided. **Main.cpp** is using to perform testing methods.

2. DICTIONARY CLASS

```
class Dictionary {

    struct Node{
        Key key;
        Info info;

        Node* left = nullptr;
        Node* right = nullptr;
        Node* parent = nullptr;

        int height = 1;

    };
    Node* root = nullptr;
```

Dictionary class is the most important part of program. It contains the structure called node.
Furthermore, class has pointer to main node (Node* root). Nodes are dynamically allocated.

Private part of methods:

```
/* ==== PRIVATE METHOD OF CLASS ==== */

bool insertHelper(const Key& k,const Info& i, Node* &n);

void destroyHelper(Node* n);
Node* removeHelper(const Key& k, Node* n);

void displayHelper(Node* n, int space);

Node* findMax(Node* n);
Node* findMin(Node* n);
Node* findByKey(Node*& n ,const Key& k);

Node* copyDictionaryHelper(Node *n);

bool compereHelper(Node* lhs, Node* rhs);

void balanceTree(Node *&n);
int getBalance(Node* n);

int getHeight(Node* n);
int updateHeight(Node *&n);

int getSize(Node* n);

bool updateParent(Node *&n);


/* ====     ROTATING METHOD     ==== */

void rRotate(Node* &n);     //Right-Right rotation
void lRotate(Node* &n);     //Left-Left rotation
void lrRotate(Node* &n);    //Left-Right rotation
void rlRotate(Node* &n);    //Right-Left rotation
```

Public part of methods:

```
/* ==== CONSTRUCTORS, OPERATORSS ==== */

Dictionary();
Dictionary(const Dictionary<Key,Info>& otherAVL);
~Dictionary();

Dictionary& operator=(const Dictionary<Key,Info>& other);
bool operator==(const Dictionary &rhs);
bool operator!=(const Dictionary &rhs);

int size();
bool isEmpty() const;

bool findByKey(const Key&);

void insert(const Key &, const Info&);

void update(const Key& old, const Key& newKey, const Info &i);

void display();
void displayInfo(const Key&);
void printInOrder();

void copyDictionary(const Dictionary<Key,Info>& otherAVL);

void destroy();
void removeByKey(const Key&);


/* ====      OTHERS METHODS       ====*/
void randomNodes(int);
```

Inside iterator class of Dicitonary:

```
/* ====      OTHERS METHODS       ====*/
/* ====       Exceptions        ==== */
class InvalidKey {
public:
    void msg(){
        std::cout << "Invalid Key of Dictionary" << std::endl;
    }
};
class InvalidInfo {
public:
    void msg(){
        std::cout << "Invalid Info of Dictionary" << std::endl;
    }
};


    Iterator& operator++();
    Iterator operator++(int);
    Iterator operator+(int r);


    bool operator==(const Iterator &other) const;
    bool operator!=(const Iterator &other) const;

    Key& getKey();
    Info& getInfo();

};
Iterator begin();
Iterator end();
```

Description of unusual method:

- void randomNodes(int number) – the method is used to test program. It creates number of random Node<int,int> with value key generated from range 1 to 100 and then it adds its to the dictionary.

5. Tests

```
//Normal and reverse printing list!
void printingTest();
//Adding by AVL rules and randomly
void insertionTest();
//Deleting by given Key, all AVL tree
void deletingTest();
//Size testing
void sizeTest();
//Assigment operator test
void assigmentTest();
//Update method test
void updateTest();
//Iterator test
void iteratorTest();
```

I created few tests to check proper functional of class methods. **Key, Info** are **integers**, and nodes are generating randomly by method **randNodes.**