EADS Laboratory 3
Dictionary AVL TREE template
Report

Kacper Kamieniarz
Computer Science
Faculty of Electronics and Information Technology
Book number: 293065

Intro

The project consists of 2 header files: “Dictionary.h” containing the data structure AVL tree and “iterator.h” containing the iterator.
Implementation of all methods and functions is in the header files.

Template parameters

The template parameters are **Key** and **Info**.
Key parameter is used for identification of a single element of the sequence, **Info** is simply the data contained in the element.

Class Dictionary

Private part

```
typedef struct node
  {
     Key key;
     Info info
     int bf;
     int height;
     node *left;
     node* right;
  }node;
  node *root;
```

Structure node contains **Key** value, **Info** value and pointer to **next** and **previous** node.
Root is node pointer to the root node.
bf is the balance factor of each node.
height is the number of elements in the subtree of given node.

Constructors

Dictionary class contains default constructor with initializer list as well as copy constructor.

Operators

DLR class contains assignment and move assignment operator.

Constant functions

Dictionary class contains constant functions which are checking the contents of the tree, for instance isEmpty() or belongs() which returns true if element with given Key and Info exists.

Modifying methods

Dictionary contains methods for inserting new element to the tree, removing element with given key, and clear method which erases all elements. All of them are recursive.

Access elements

DLR contains public methods for accessing some private parts of data. getMax() returns element with highest ID in the tree, getMin() returns the lowest ID.

Display methods

There are 2 methods of displaying the tree, one of them is InOrder which prints all the elements ID's and Data in order. The other one is display2D which visualises the tree in 2 dimensions.

Display methods

There are 4 rotation methods which rotate the elements of the tree around given node.
it's standard left, right, leftRight and rightLeft rotation.

Balancing

The balance() method checks whether the tree is imbalance and if it is not it makes necessary rotations. There are 4 cases of depending on the balance factor of the nodes.

Class iterator

Iterator class contains an iterator for the tree. There are ++ pre increment and post increment operators defined along with assignment and boolean operators defined. There is also a findParent function which returns the ancestor node of a node with given ID.