LABORATORIES 2

Doubly Circular Linked List

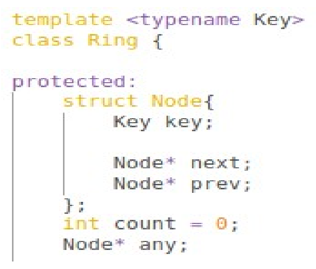
Wojciech Marosek
ID: 295818

1. FILES

-| include
--- DLR.hpp
-| src
--- main.cpp
--- tests.cpp

File DLR**.hpp** contains Ring class with class Iterator and their methods and operators implementations . In addition,the file contains implementation of function **split**. **Tests.cpp** keeps tests of class methods(declaration and implementation) . **Main.cpp** is using to perform testing methods.

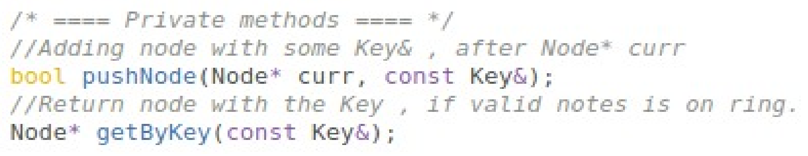
2. RING CLASS



```
template <typename Key>
class Ring {

protected:
    struct Node{
        Key key;

        Node* next;
        Node* prev;
    };
    int count = 0;
    Node* any;
```

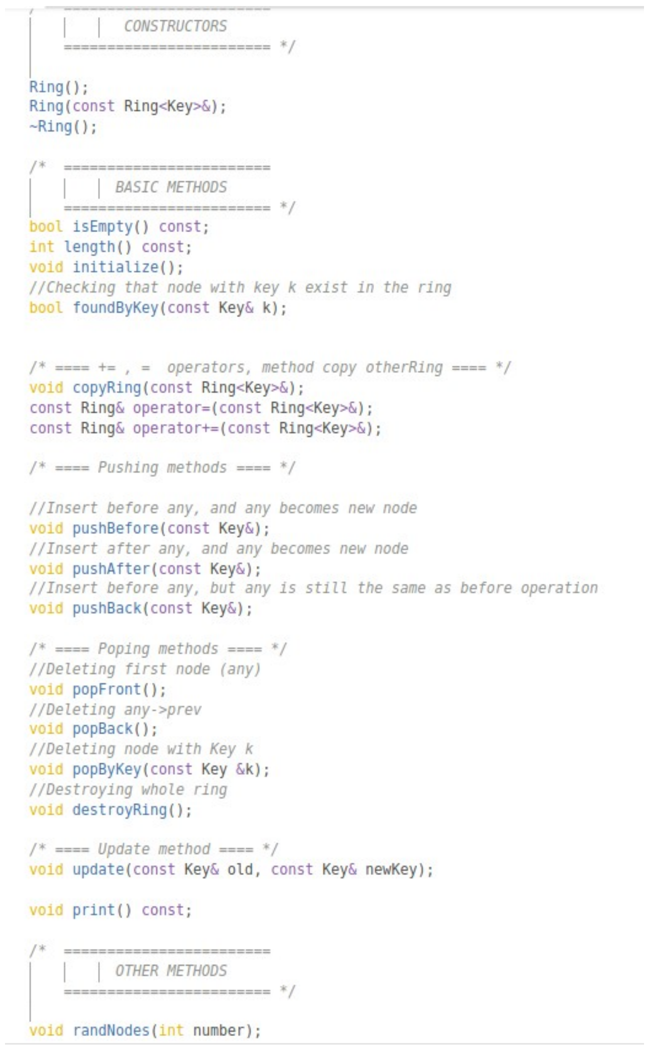
Ring class is the most important part of program. It contains the structure called node.
Furthermore, class has number of nodes (int count) , pointer to any node (Node* any). Nodes are dynamically allocated.

Below, we can see the methods, constructors and operators of sequence:
Private part of methods:

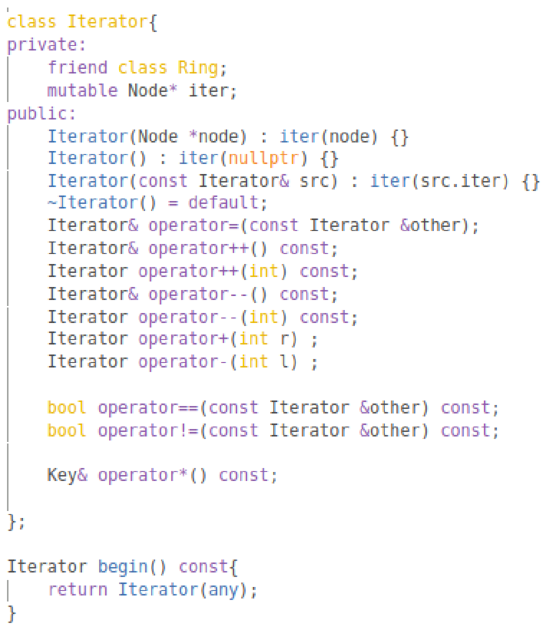


```
/* ==== Private methods ==== */
//Adding node with some Key& , after Node* curr
bool pushNode(Node* curr, const Key&);
//Return node with the Key , if valid notes is on ring.
Node* getByKey(const Key&);
```

Public part of methods:


```
    |   CONSTRUCTORS
    ======================= */

Ring();
Ring(const Ring<Key>&);
~Ring();

/*  =======================
    |   BASIC METHODS
    ======================= */
bool isEmpty() const;
int length() const;
void initialize();
//Checking that node with key k exist in the ring
bool foundByKey(const Key& k);


/* ==== += , =  operators, method copy otherRing ==== */
void copyRing(const Ring<Key>&);
const Ring& operator=(const Ring<Key>&);
const Ring& operator+=(const Ring<Key>&);

/* ==== Pushing methods ==== */

//Insert before any, and any becomes new node
void pushBefore(const Key&);
//Insert after any, and any becomes new node
void pushAfter(const Key&);
//Insert before any, but any is still the same as before operation
void pushBack(const Key&);

/* ==== Poping methods ==== */
//Deleting first node (any)
void popFront();
//Deleting any->prev
void popBack();
//Deleting node with Key k
void popByKey(const Key &k);
//Destroying whole ring
void destroyRing();

/* ==== Update method ==== */
void update(const Key& old, const Key& newKey);

void print() const;

/*  =======================
    |   OTHER METHODS
    ======================= */

void randNodes(int number);
```

Inside iterator class of Ring:

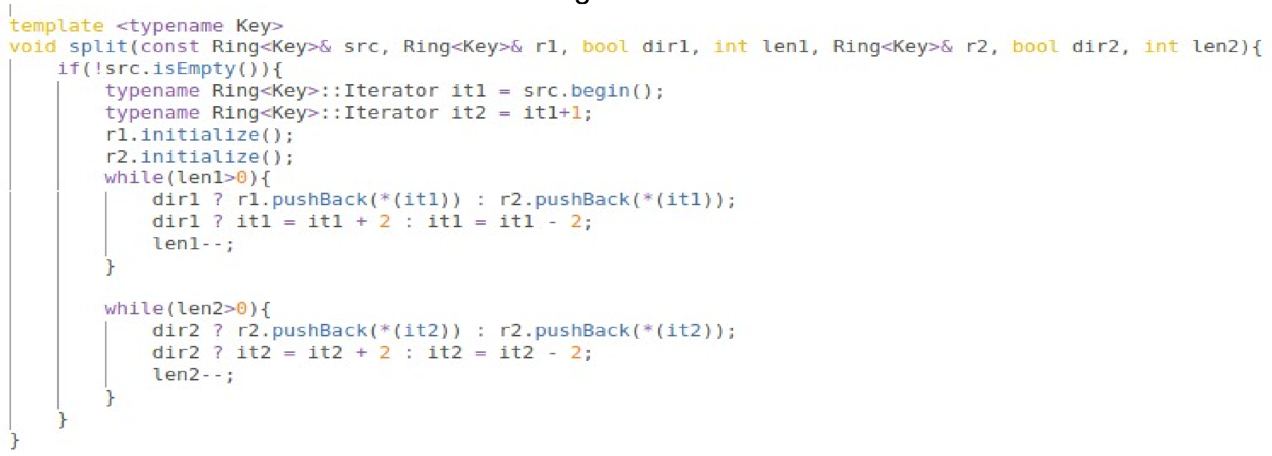

```
class Iterator{
private:
    friend class Ring;
    mutable Node* iter;
public:
    Iterator(Node *node) : iter(node) {}
    Iterator() : iter(nullptr) {}
    Iterator(const Iterator& src) : iter(src.iter) {}
    ~Iterator() = default;
    Iterator& operator=(const Iterator &other);
    Iterator& operator++() const;
    Iterator operator++(int) const;
    Iterator& operator--() const;
    Iterator operator--(int) const;
    Iterator operator+(int r) ;
    Iterator operator-(int l) ;

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

    Key& operator*() const;

};

Iterator begin() const{
    return Iterator(any);
}
```

Description of unusual method:

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

4. Split function

The function is contained in DLR.hpp. It takes three rings as parameters. Additional it takes two boolean variables and two integers.

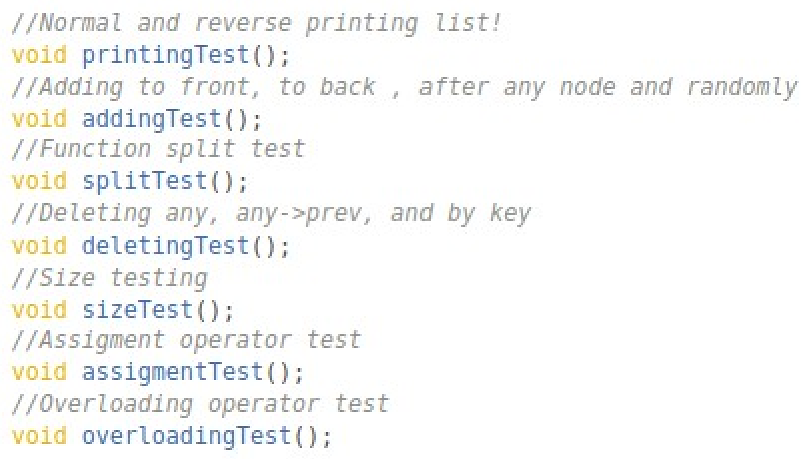


```
template <typename Key>
void split(const Ring<Key>& src, Ring<Key>& r1, bool dir1, int len1, Ring<Key>& r2, bool dir2, int len2){
    if(!src.isEmpty()){
        typename Ring<Key>::Iterator it1 = src.begin();
        typename Ring<Key>::Iterator it2 = it1+1;
        r1.initialize();
        r2.initialize();
        while(len1>0){
            dir1 ? r1.pushBack(*(it1)) : r2.pushBack(*(it1));
            dir1 ? it1 = it1 + 2 : it1 = it1 - 2;
            len1--;
        }

        while(len2>0){
            dir2 ? r2.pushBack(*(it2)) : r2.pushBack(*(it2));
            dir2 ? it2 = it2 + 2 : it2 = it2 - 2;
            len2--;
        }
    }
}
```

src, r1, r2 – ring
dir1,dir2 – direction: clockwise(true) or counter-clockwise(false)
len1, len2 – number of nodes for each sequence

Split function builds two rings which are a combination of source ring (src).

5. Tests



```
//Normal and reverse printing list!
void printingTest();
//Adding to front, to back , after any node and randomly
void addingTest();
//Function split test
void splitTest();
//Deleting any, any->prev, and by key
void deletingTest();
//Size testing
void sizeTest();
//Assigment operator test
void assigmentTest();
//Overloading operator test
void overloadingTest();
```

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