Doubly-linked ring

Borys Bondos

1. ***Task***

The aim of the exercise was to create doubly linked ring which is able to store data of arbitrary type (templates). Every node has got key and info and both of them are not unique.

We had to write implementation of method produce() which is able to create new ring based on two source rings.

2. ***Design of the class***

In private section of the class I created structure which keeps all data of specified node. Since we are dealing with doubly linked ring, each node has got pointer to next and previous element of the ring. I defined also pointer first which is an element from which we are starting indexing (first has got index 0).

In public section of the class I created all necessary methods for adding, printing, removing, constructors and destructor. I defined also iterator with it s methods to give access to private data such as information about first node.

3. Description of the methods

Constructors and destructor:

- Ring();
- ~Ring();
- Ring(const Ring &source); - copy constructor

Adding methods:

- void add_node_end(key ID, info val); -adds element to the ring. New element is added at the “end” of the ring.
- void add_node_between(key after, key before, key ID, info VAL); - Adds element on specified position between two existing nodes with given keys.
- void add_node_on_position(int position,key ID, info val); - Adds element on place with given index (first element – index 0) negative indexes are allowed because it is doubly linked ring. We are looking for negative index by going counter-clockwise thru data structure.
- void push_front(key ID, info val); - Adds node at the beginning • void push_back(key ID, info val); -Adds node at the end

Removing methods:

- void remove_node_key(key ID); - Removes node with given key
- void remove_from_position(int position); - Removes node with given index (negative indexes are also allowed)
- void remove_ring(); -Removes whole ring
- void pop_front(); -Removes node from the beginning
- void pop_back(); -Removes node from the end

Printing method:

- void print_ring();

Overloaded operator:

- Ring& operator=(const Ring & source);

Methods of iterator:

- Iterator begin()const - Returns first element
- Iterator end()const -Returns last element

Methods of class iterator:

- Iterator() - constructor
- Iterator(node *source) - copy constructor
- const Iterator &operator = (const Iterator &copyIter)
- bool operator == (const Iterator &compare)const
- Iterator operator ++ ()
- Iterator operator ++ (int)
- Iterator operator -- ()
- Iterator operator -- (int)
- key getKey()
- info getInfo()
- bool isnull()
- void print_iterator()
- bool operator != (const Iterator &compare)const

4. Concept of the function produce()

Function produce is an external function which creates new Ring. It has got access to read data and traverse thru whole data structure thanks to public methods of class Ring and Iterator.

Because of the fact that those methods give access only to read data, our source data is safe and encapsulation is preserved.

5. *Testing*

Testing scenarios cover proper and improper usage of every function and method.

Description of every test scenario is shown during execution of the program.

Testing is divided into parts:

- Function produce test
- Adding functions test
- Removing function test
- Operator test
- Copy constructor test