EADS

LAB 2 Project Report

Doubly Linked List Ring

Keshav Dandeva
Student ID : 302333

Introduction

This project report is about implementing circular doubly linked list (also called Ring) and performing various operations on it. A class of Iterator is created within the class of the Ring for convenience in moving through the list and performing most of the functions without having to use pointers in class member functions. An external function of shuffle is created to make a new ring from existing rings with mixed elements.

BiRing Class

The BiRing class contains various public member functions for different utilities and a class of Iterator as well for iterating through the list. There is also a private structure Node for the creation of doubly linked list. The template used for Sequence class is *template<typename Key, typename Info>* where Key is like the id (need not be unique) and Info is the data stored.

- struct Node
  It contains four members :
  Key key; -> This is the key for the data in the node.
  Info info; -> This is the data stored in the node.
  *Node *next;* -> This is the pointer for the next node. It is used to travel forward through the list.
  *Node *prev;* -> This is the pointer for the previous node. It is used to travel backward through the list.
  It also has a pointer **element* that points to the first element of the list.

- BiRing()
  It is a constructor of the class which creates a new node and assign it to element. Then initialises the element to be the first element of the list.

- ~BiRing()
  It is a destructor which deletes all the elements of the list when it goes out of scope.

- BiRing(const BiRing<Key, Info> &copyBiRing)
  It is a copy constructor which copies the elements from the given list to the newly created list.

- BiRing<Key, Info> &operator=(const BiRing<Key, Info> &BiRing)
  This function is the overloading of the assignment operator. It copies the values of the list at right of the operator "=" to the list at the left.

- BiRing<Key, Info> &operator+(const BiRing <Key, Info> &ring)
  This function is the overloading of the addition operator i.e. "+" . It adds the elements of the lists provided and forms a new list with the sum of elements from both the given lists.

- int length() const

This function returns the length of the list i.e. the number of elements present in the list.

- Iterator beginning() const ;
  This function returns the iterator to the beginning of the list.

- Iterator end() const ;
  This function returns the iterator to the end of the list.

- Iterator push(const Key &key, const Info &info)
  This function adds the provided key and info to the list.

- Iterator deleteFirst()
  This function deletes the first element of the list and returns iterator to the first element.

- Iterator deleteLast()
  This function deletes the last element of the list and returns iterator to the last element.

- void deleteAll()
  This function deletes the whole list.

- friend ostream &operator<<(ostream &os, const BiRing <Key, Info> &BiRing);
  This is a friend function for overloading the "<<" operator to print the list.

Iterator Class

The iterator class is defined in public of the BiRing class. It contains BiRing class as friend class and has various public member functions for different utilities. It also has a private member of a pointer to the Node structure.

- Iterator()
  This is a default constructor of the class and it initialises the target node pointer to NULL.

- Iterator(const Iterator &copy)
  This is a copy constructor of the class and it copies the iterator provided to the newly created.

- Iterator(Node *copy)
  This is a function that provides the target node to the iterator.

- Key &getKey()
  This function returns the key from the target node.

- Info &getInfo()
  This function returns the info from the target node.

- bool operator!=(const Iterator &compare) const

This function return true or false whether the target node is same as the node being compared.

- bool operator==(const Iterator &compare);
  This function return true or false after checking if target node is same as the node being compared.

- Iterator &operator+(const int add)
  This function moves the iterator to the next element as many times as provided in parameter.

- Iterator &operator-(const int substract);
  This function moves the iterator to the previous element as many times as provided in parameter.

- Iterator &operator++()
  This function moves the iterator to the next element. It is the overloading of the pre-increment operator function.

- Iterator &operator--()
  This function moves the iterator to the previous element. It is the overloading of the pre-decrement operator function.

- Iterator operator++(int)
  This function moves the iterator to the next element. It is the overloading of the post-increment operator function. The int parameter is dummy parameter to differentiate between pre-increment and post-increment operator functions.

- Iterator operator--(int)
  This function moves the iterator to the previous element. It is the overloading of the post-decrement operator function. The int parameter is dummy parameter to differentiate between pre-decrement and post-decrement operator functions.

- Iterator &operator=(const Iterator &equal)
  This function assigns the target node the same value as the provided node's.

- Iterator &operator+=(const int add)
  This function moves the iterator to the next number of elements added after the += operator. By default, it moves to the next element.

- Iterator operator-=(const int substract)
  This function moves the iterator to the previous number of elements added after the -= operator. By default, it moves to the previous element.

- Iterator &add(const Key &key, const Info &info)
  This function adds the provided key and info to a new node and adds the node to the list at the current position and returns iterator at the new node.

- Iterator &removeCurrent()
  This function deletes the element the iterator is currently at.

Shuffle Function

- BiRing<Key, Info> shuffle(const BiRing<Key, Info>& first, BiRing<Key, Info>& second, int nbfirst, int nbsecond, int reps)
  This is a global function that returns a new ring with the elements from two rings provided as parameter. The number of elements in the returned ring is equal to the elements taken from first list + elements taken from the second list multiplied by the number of repetitions asked for.

Testing

1. BiRing Class

All the functions and their all possible behaviour is tested for this class :

First , an object of the BiRing class is created i.e. a doubly linked ring and elements are added to the list using push function.

```
BiRing<int, string> ring1;
ring1.push(1, "Horse");
```

Then the length of the ring and the ring is printed using cout and << operator.

```
cout << "Ring 1 : " << ring1 << endl;
cout << "Ring 1 length is: " << ring1.length() << endl;
```

Then the assignment operator is tested by creating a new list ring2 and assigning ring1 to it.

```
BiRing<int, string> ring2 = ring1;
```

Then deleteLast(), deleteFirst() and deleteAll() functions are tested by deleting the elements in ring1.

Then, the beginning function is tested by creating a iterator and moving it to the beginning of ring2.

```
BiRing<int, string>::iterator iterator = ring2.beginning();
```

Then, the add function is tested with all the post and pre increment and decrement functions.

Then, a new ring is created by dynamically allocating memory to it and also using copy constructor.

*BiRing<int, string> *ring5 = new BiRing<int, string>(ring2);*

At last, the removeCurrent() function is tested for ring2.

iterator.removeCurrent();

2. Shuffle Function

This function is tested by creating two new rings and adding elements to them. Then, both the lists are printed to reassure the values in them.

cout << "Ring 6: " << ring6 << endl << endl;

cout << "Ring 7: " << ring7 << endl << endl;

Then, a new ring is created and shuffle function is called to return the shuffled list in the new ring. The new ring contains 6 elements from the first ring and 6 elements from the second ring with repetition of 2 times. Varipus scenarios for shuffle function are tested as well.

BiRing<int, string> ring4;

ring4 = shuffle(ring6, ring7, 6, 6, 2);

cout << "Ring 4: " << ring4 << endl;