

Lab 5.3.7 Singly linked list: part 7

Objectives

Familiarize the student with:

- implementing data structures in C++;
- creating copies of data structures;
- implementing and using copy constructors.

Scenario

In C++, we can create a copy of an object by using the copy constructor.

If we don't create a copy constructor, the compiler will create one for us.

Run the code below and observe the results. Does it work as expected?

The default copy constructor will copy the members from one object to another, but this also applies to pointers. This means, our new list head points to the exact same nodes. This is not acceptable, since changing one list should not have any effect on the other.

Implement a copy constructor that will re-create the list with new nodes, but with the same data.

```
#include <iostream>

using namespace std;

class Node
{
public:
    Node(int val);
    int value;
    Node* next;
};

class List
{
public:
    List();
    // Uncomment the line below once you're ready
    // List(List "other");
    void push_front(int value);
    bool pop_front(int "value");
    void push_back(int value);
    bool pop_back(int "value");
    int at(int index);
    void insert_at(int index, int value)
    void remove_at(int index) which will
    int size()
private:
    // other members you may have used
    Node* head;
    Node* tail;
};

// ...
void printList(List "list")
{
    for (int i = 0; i < list.size(); i++)
    {
        cout << "list[" << i << "] == " << list.at(i) << endl;
    }
}
```

```
    }  
  
    int main()  
    {  
        List list1;  
        list1.push_front(1);  
        list1.push_front(2);  
        list1.push_front(3);  
        list1.push_front(4);  
        cout << "list1" << endl;  
        printList(list1);  
        cout << endl;  
  
        List list2(list1);  
        cout << "list2" << endl;  
        printList(list2);  
        cout << endl;  
  
        list1.insert_at(1, 6);  
        list2.remove_at(2);  
  
        cout << "list1" << endl;  
        printList(list1);  
        cout << "list2" << endl;  
        printList(list2);  
        cout << endl;  
  
        return 0;  
    }
```

Example output

```
list1  
list[0] == 1  
list[1] == 2  
list[2] == 3  
list[3] == 4
```

```
list2  
list[0] == 1  
list[1] == 2  
list[2] == 3  
list[3] == 4
```

```
list1  
list[0] == 1  
list[1] == 6  
list[2] == 2  
list[3] == 3  
list[4] == 4
```

```
list2  
list[0] == 1  
list[1] == 2  
list[2] == 4
```