

## Lab 5.3.2 Singly linked list: part 2

### Objectives

Familiarize the student with:

- implementing data structures in C++;
- preventing memory leaks and deallocating acquired resources.

### Scenario

Let's continue working on our list.

We're currently making one serious error: we're not cleaning up after ourselves and we're leaking memory.

Add a destructor to your List class. The destructor should delete all the Nodes in the list, leaving it empty.

To help you make sure that you've deleted all the elements, we've added some text output to the Node constructor and destructor.

```
#include <iostream>

using namespace std;

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

Node::Node(int val) : value(val), next(nullptr)
{
    cout << "+Node" << endl;
}

Node::~~Node()
{
    cout << "-Node" << endl;
}

class List
{
public:
    List();
    void push_front(int value);
    bool pop_front(int "value");
private:
    Node* head;
};

// ...

int main()
{
    List list;
    //
    list.push_front(1);
    list.push_front(2);
    list.push_front(3);
    list.push_front(4);

    return 0;
}
```

### Example input

### Example output

```
+Node
+Node
+Node
+Node
-Node
-Node
-Node
-Node
```