# Lab 5.3.6 Singly linked list: part 6

## Objectives

Familiarize the student with:

- implementing data structures in C++;
- traversing data structures;
- accessing data stored in data structures.

## Scenario

For the last part of adding, removing and accessing the data in our list, we'll provide you with a way to do all of these actions by indexing the elements in the list.

To enable this, we'll add three new methods to the List class:

- at(int index) which will return the value of the element with index [index];
- insert_at(int index, int value) which will insert the value at the desired index;
- remove_at(int index) which will remove the value at the desired index (without returning it).

```
#include <iostream>

using namespace std;

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

class List
{
public:
  List();
  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 list;
  //
  list.push_front(1);
  list.push_front(2);
  list.push_front(3);
  list.push_front(4);
  printList(list);
  cout << endl;

  list.remove_at(2);
  printList(list);
  cout << endl;

  list.insert_at(1, 6);
  printList(list);

  return 0;
}
```

## Example output

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

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

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