Linked List

Learning Objectives

1. Implement the insert at any index function
2. Implement [] and remove functions

Linked List

List.h

```
1  #pragma once
2
3  template <class T>
4  class List {
5    public:
…      /* ... */
28     insertAtFront(const T &data);
29   private:
30     struct ListNode {
31       T data;
32       ListNode * next;
         ListNode(const T & data) :
33        data(data), next(NULL) { }
34     };
35
36   ListNode *head_;
37
38
39
…  };
…
79 #include "List.hpp"
```

List.hpp

```
1
2  template <class T>
3  void List<T>::insertAtFront(const T &data) {
4       ListNode *tmp = new ListNode(data);
5       tmp->next = head_;
6       head_ = tmp;
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22 }
```

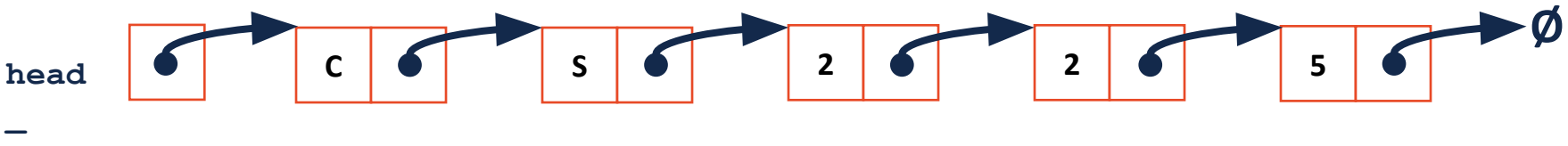
Linked List


```
template <typename T>
void List<T>::insert(const T & data, unsigned index) {




}
```

List.hpp

```
90  template <typename T>
91  void List<T>::insert(const T & data, unsigned index) {
92      ListNode *& tmp = _index(index);
93      ListNode * in = new ListNode (data);
94      in -> next = tmp;
95      tmp = in;
96
97
98
99  }
```

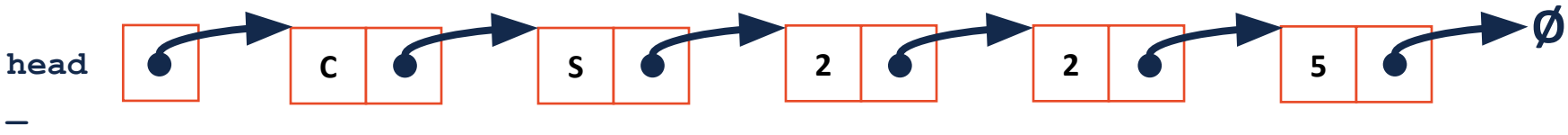
Linked Memory: `operator[]`



List.hpp

```
49  template <typename T>
50  T & List<T>::operator[](unsigned index) {
 …      ListNode *& loc = _index(index);
       return loc->data;




    }
```

Linked Memory: `remove`



List.hpp

```
109 template <typename T>
110 void List<T>::remove(unsigned index) {
 …      ListNode *& prev = _index(index);
        ListNode* tmp = prev;
        prev = prev->next;
        delete tmp;




        }
```