CS 225

Data Structures

Feb. 6 – List Implementation

Wade Fagen-Ulmschneider, Craig Zilles

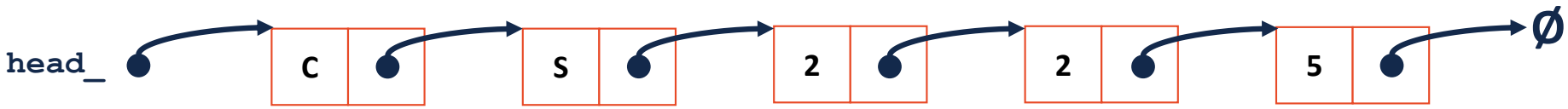
List.h

```
 1  #pragma once
 2
 3  template <typename T>
 4  class List {
 5    public:
…     /* ... */
19
20    private:
21      class ListNode {
22        public:
23          T & data;
24          ListNode * next;
25          ListNode(T & data) :
            data(data), next(NULL) { }
26      };
27
28      ListNode *head_;
…
        };
```

List.hpp

```
 9  #include "List.h"
…
14  template <typename T>
15  void List::insertAtFront(const T& t) {
16
17
18
19
20
21
22  }
```

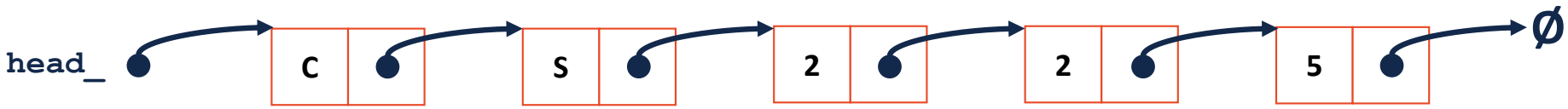
Linked Memory



List.hpp

```
57 template <typename T>
58 typename List<T>::ListNode *&
     List<T>::_index(unsigned index) {
59
60
61 }
62
63
64
65
```

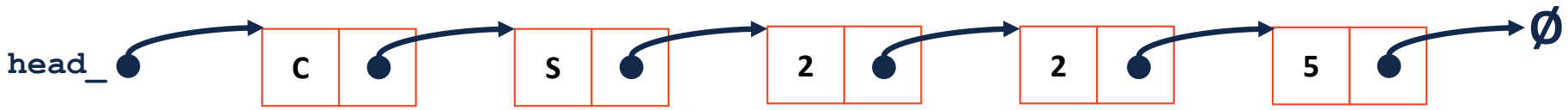
Linked Memory



List.hpp

```
// Iterative Solution:
template <typename T>
typename List<T>::ListNode *& List<T>::_index(unsigned index) {
  if (index == 0) { return head; }
  else {
    ListNode *thru = head;
    for (unsigned i = 0; i < index - 1; i++) {
       thru = thru->next;
    }
    return thru->next;
  }
}
```

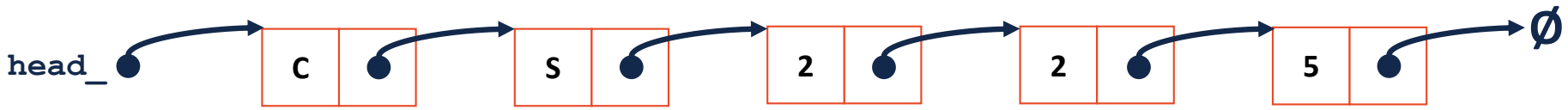
Linked Memory



List.cpp

```
48 template <typename T>
49 T & List<T>::operator[](unsigned index) {
50
51
52
53
54
55
56
57
58 }
```

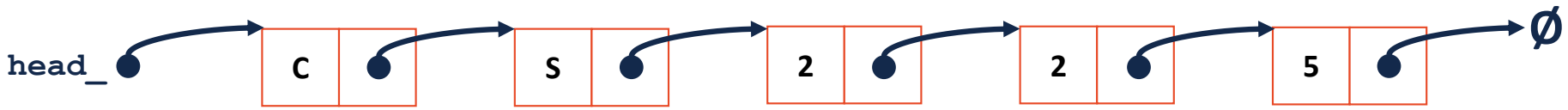
Linked Memory



List.cpp

```
90  template <typename T>
91  void List<T>::insert(const T & t, unsigned index) {
92
93
94
95
96
97
98
99  }
```

Linked Memory



List.cpp

```
103 template <typename T>
104 T & List<T>::remove(unsigned index) {
105
106
107
108
109
110
111
112
113
114
115
116 }
117
```

Linked Memory