CS 225

Data Structures

August 31 – Linked List Implementation

G Carl Evans

Linked Lists

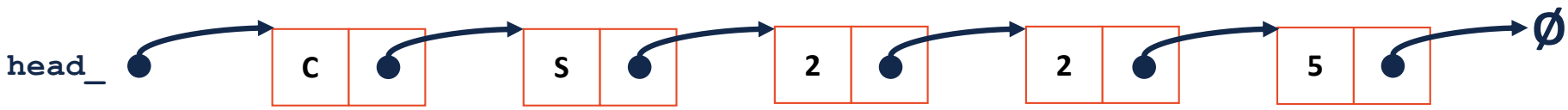
List.h

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

List.hpp

```
24 template <typename T>
25 T &List<T>::operator[](unsigned index) {
26         return _index(index)->data;
27 }
```

Linked Memory

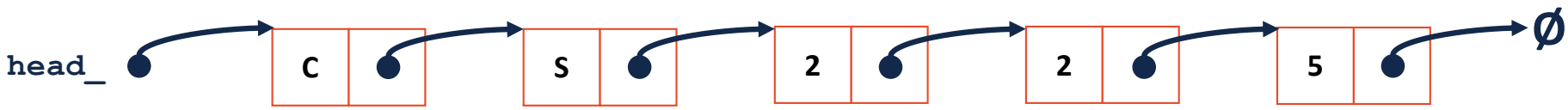


List.hpp

```
33 ListNode *& List<T>::_index(unsigned index) const {
34         return _index(index, head_)
35
36 }
```

```
38 ListNode *& List<T>::_index(unsigned index, ListNode *&head) const {
39         if(index == 0)
40         {
41                 return head;
42         } else {
43                 return _inded(index - 1, head->next);
44         }
45 }
```

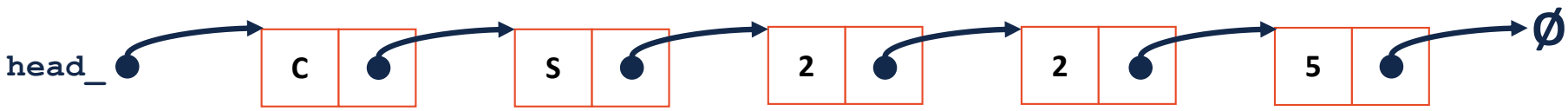
Linked Memory



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;
  }
}
```

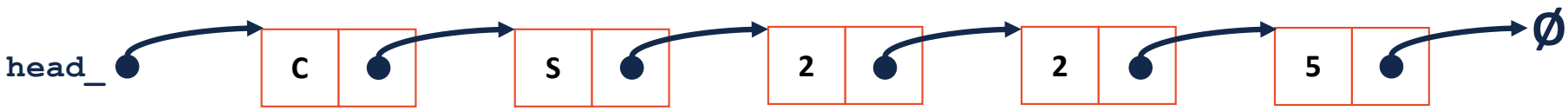
Linked Memory



List.hpp

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

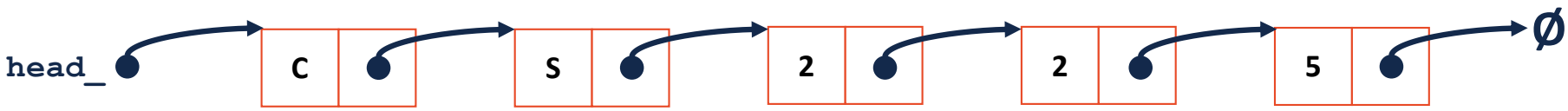
Linked Memory



List.hpp

```
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.hpp

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

Linked Memory Runtimes