CS 225

Data Structures

August 29 – Lists and Linked Memory

G Carl Evans

mp_stickers

- Releasing later today

List Implementations

1.

2.

Linked Memory

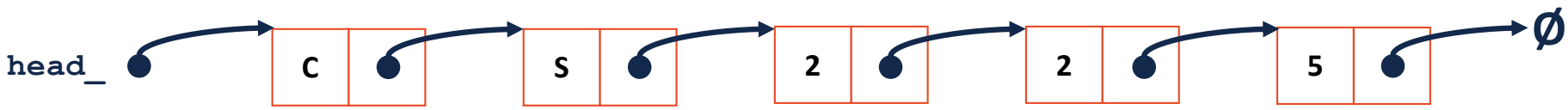
List.h

```
 1  #pragma once
 2
 3  template <class T>
 4  class List {
 5    public:
…      /* ... */
28    private:
29
30
31
32
33
34
35
36
37
38
39
40
41
    };
```

List.hpp

```
 1  #include "List.h"
 2
 3  template <class T>
 4  void List<T>::insertAtFront(const T& t) {
 5
 6
 7
 8
 9
10
11
12
13
14
15
16
17
18
19
20
21
22  }
```

Linked Memory

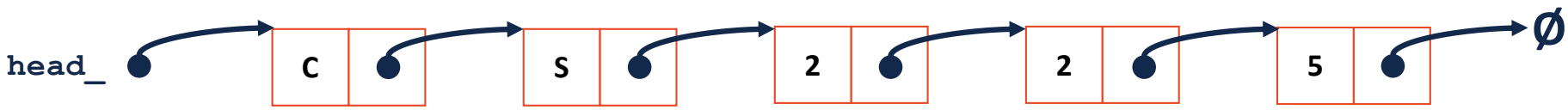


Running Time of Linked List `insertAtFront`

List.hpp

```
24 template <typename T>
25 T &List<T>::operator[](unsigned index) {
26
27
28
29
30
31 }
```

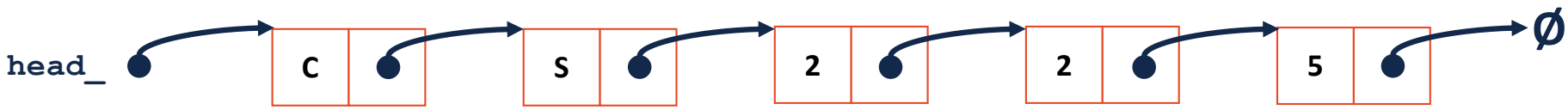
Linked Memory



List.hpp

```
33 ListNode *& List<T>::_index(unsigned index) const {
34
35
36
37
38
39
40 }
```

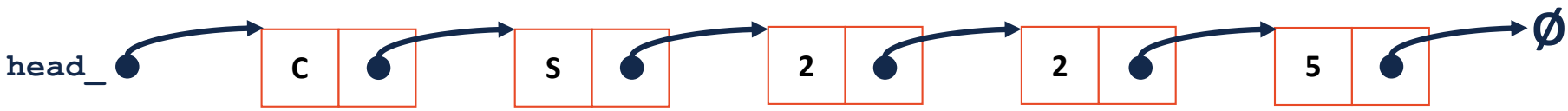
Linked Memory



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;
  }
}
```

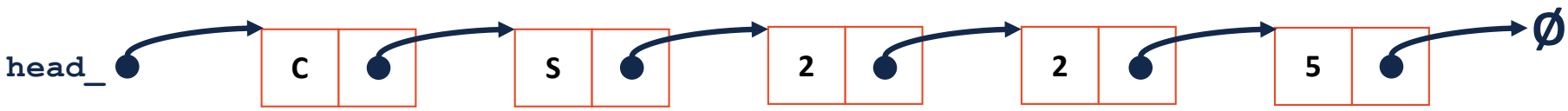
Linked Memory



List.hpp

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

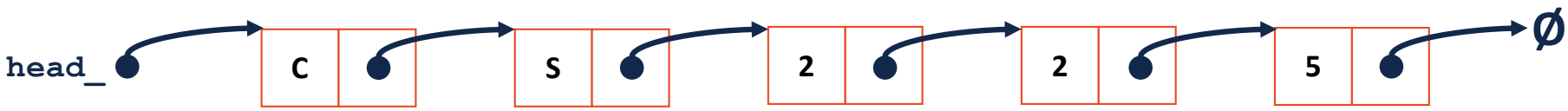
Linked Memory



List.hpp

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

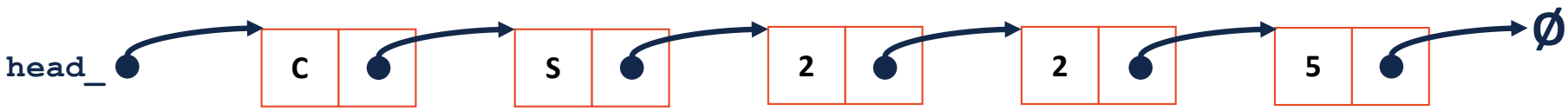
Linked Memory



List.hpp

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

Linked Memory



Linked Memory Runtimes