



CS 225

Data Structures

Sept. 16 – Templates and Linked Memory

G Carl Evans

animalShelter.cpp

```
5 class AnimalShelter {  
6     public:  
7         Animal & adopt();  
...     // ...  
};
```

animalShelter.cpp

```
1 class Animal {
2     public:
3         virtual void speak() {           }
4 };
5
6 class Dog : public Animal {
7     public:
8         void speak() {                   }
9 };
10
11 class Cat : public Animal {
12     public:
13         void speak() {                   }
14 };
```



Abstract Class:

[Requirement]:

[Syntax]:

[As a result]:

virtual-dtor.cpp

```
4 class Cube {
5     public:
6         ~Cube() { std::cout << "~Cube() invoked."
7                 << std::endl; }
8
9     class RubikCube : public Cube {
10        public:
11            ~RubikCube() { std::cout << "~RubikCube() invoked."
12                          << std::endl; }
13    };
14
15    std::cout << "Non-virtual dtor:" << std::endl;
16    Cube *ptr = new RubikCube();
17    delete ptr;
```

virtual-dtor.cpp

```
15 class CubeV {
16     public:
17         virtual ~CubeV() { std::cout << "~CubeV() invoked."
18                             << std::endl; }
19 };
20
21 class RubikCubeV : public CubeV {
22     public:
23         ~RubikCubeV() { std::cout << "~RubikCubeV() invoked."
24                             << std::endl; }
25 };
26
27
28
29
30
31 std::cout << "Virtual dtor:" << std::endl;
32 CubeV *ptrV = new RubikCubeV();
33 delete ptrV;
```

```
waf@siebl-2215-02:/mnt/c/Users/waf/Desktop/cs225/_lecture/09-linkedMemory$ make
clang++ virtual-dtor.cpp -lm -o virtual-dtor
waf@siebl-2215-02:/mnt/c/Users/waf/Desktop/cs225/_lecture/09-linkedMemory$ ./virtual-dtor
Non-virtual dtor:
~Cube() invoked.
Virtual dtor:
~RubikCubeV() invoked.
~CubeV() invoked.
waf@siebl-2215-02:/mnt/c/Users/waf/Desktop/cs225/_lecture/09-linkedMemory$
```

MP: Extra Credit

The most successful MP is an MP done early!

Unless otherwise specified in the MP, we will award +1 extra credit point per day **for completing Part 1** before the due date (*up to +7 points*):

Example for MP2:

+7 points: Complete by **Monday**, Sept. 16 (11:59pm)

+6 points: Complete by **Tuesday**, Sept. 17 (11:59pm)

+5 points: Complete by **Wednesday**, Sept. 18 (11:59pm)

+4 points: Complete by **Thursday**, Sept. 19 (11:59pm)

+3 points: Complete by **Friday**, Sept. 20 (11:59pm)

+2 points: Complete by **Saturday**, Sept. 21 (11:59pm)

+1 points: Complete by **Sunday**, Sept. 22 (11:59pm)

MP2 Due Date: Monday, Sept. 23



Abstract Data Type



List ADT



What types of “stuff” do we want in our list?

--	--	--	--	--	--	--	--

--	--	--	--	--	--	--	--

--	--	--	--	--	--	--	--



Templates

template1.cpp

```
1  
2  
3 T maximum(T a, T b) {  
4     T result;  
5     result = (a > b) ? a : b;  
6     return result;  
7 }
```

List.h

```
1 #pragma once
2
3
4 class List {
5     public:
6
7
8
9
10
11
12
13
14     private:
15
16
17
18 };
19
20
21
22
```

List.hpp

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
```



List Implementations

1.

2.

Linked Memory



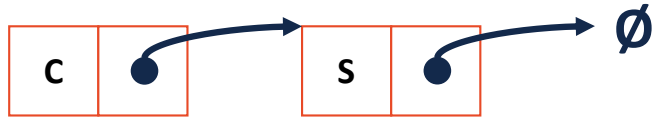
List.h

```
28 class ListNode {
29     T & data;
30     ListNode * next;
31     ListNode(T & data) : data(data), next(NULL) { }
32 };
```

Linked Memory



Linked Memory



List.h

```
1 #pragma once
2
3 template <class T>
4 class List {
5     public:
6     /* ... */
7
8     private:
9     class ListNode {
10         T & data;
11         ListNode * next;
12         ListNode(T & data) :
13             data(data), next(NULL) { }
14     };
15 };
16
17
18
19
20
21
22
```

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 }
```



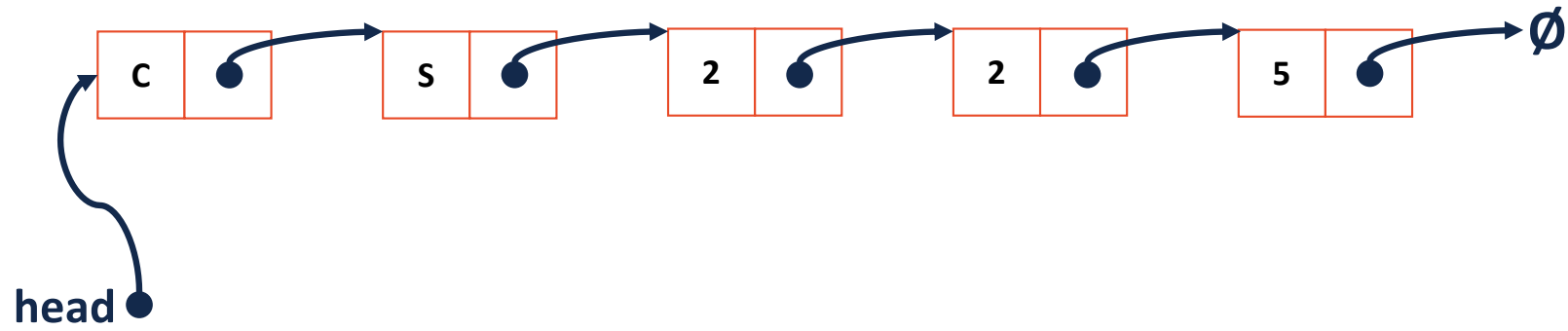
Running Time of Linked List `insertAtFront`

List.cpp

80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```
14 void List<T>::printReverse()  
    const {  
15  
16  
17  
18  
19  
20  
21  
22 }
```

Linked Memory





Running Time of Linked List `printReverse`

List.cpp

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

List.cpp

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