CS 225 Spring 2019 :: TA Lecture Notes
2/4 LIST ADT

By Wenjie

- **Abstract Class**
  - Requirement: one or more pure virtual functions
  - Syntax: nothing - no abstract keyword in cpp
  - As a result: cannot create an instance of an abstract class

- **Virtual Destructor**
  - All destructors in base classes must be virtual
  - Destructors will call the base classes destructors

virtual-dtor.cpp

```
1   class Cube {
2     public:
3       ~Cube() { std::cout << "~Cube() invoked." <<
4   std::endl; }
5   };
6   class RubikCube : public Cube {
…     public:
2       ~RubikCube() { std::cout << "~RubikCube()
8   invoked." << std::endl; }
2   };
9   class CubeV {
3     public:
0       virtual ~CubeV() { std::cout << "~CubeV()
3   invoked." << std::endl; }
1   };
3   class RubikCubeV : public CubeV {
2     public:
3       ~RubikCubeV() { std::cout << "~RubikCubeV()
3   invoked." << std::endl; }
3   };
4   int main() {
3       std::cout << "Non-virtual dtor:" <<
5   std::endl;
3       Cube *ptr = new RubikCube();
6       delete ptr;
        std::cout << "Virtual dtor:" << std::endl;
        CubeV *ptrV = new RubikCubeV();
```

CS 225 Spring 2019 :: TA Lecture Notes
2/4 LIST ADT

By Wenjie

```
    delete ptrV;

    return 0;
}
```

- In this case we have rubikcube dtor invoked first then cube dtor is invoked

- **Abstract Data Type (ADT)**
  - English definition / the basic operations of a data structure
  - ADT describes functionality but not implementation details

List ADT	Definition of Functionality
Create the empty list	Creates and empty list.
Add data to the list	Store data.
Get data from the list	Access data.
Remove data from the list	Remove data.
Check if a list is empty/size	How much data is in the list.

- **Templates: a dynamic data type**
  - Using "Template <typename T>" so that we do not need to write same function for various types
  - Template type are checked at compile time
    - maximum(3, 5): T = int
    - maximum("world", "hello"): T = string
    - maximum(cube(7), cube(42)) - but this may not complied since no > op defined
  - We can use other replace for T but using T is universally standard way

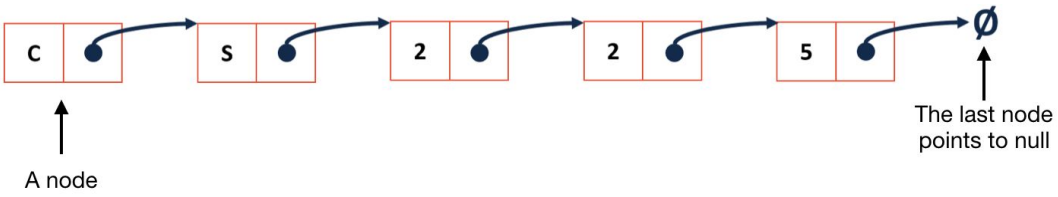
```
1 template <typename T>
2 T maximum(T a, T b) {
3     T result;
4     result = (a > b) ? a : b;
5     return result;
6 }
```

CS 225 Spring 2019 :: TA Lecture Notes
2/4 LIST ADT

By Wenjie

- There are two basic implementations of the list:
  - **Array** - sequential block of items.
  - **Linked Memory** - linked list.

- Linked Memory



- A node has two member variables:
  - A list node pointer that points to the next block (ListNode * next).
  - The data stored in the block (T & data).
- A ListNode is available within List class, it is like a helper class. *Never* return a ListNode object/pointer outside of List, always return the data.

List.h

```
1   #ifndef LIST_H
2   #define LIST_H
3
4   template <typename T>
5   class List {
6        public:
…             /* ... */
2        private:
8            class ListNode {
2                T & data;
9                ListNode * next;
3                ListNode(T & data) : data(data),
0   next(NULL) { }
3            };
1   };
3
2   #endif
```