# 자료구조 실습04

Data Structures Lab04

# Lab04 예제(1/2)

◎ 목표: Sorted Linked List 설계 및 구현

◎ 내용:

- ☞ NodeType를 이용한 Generic Sorted Linked List class 정의 및 구현
- ☞ Template을 이용한 Generic data type과 포인터을 이용한 Sorted Linked List Application 구현

◎ 방법

- ☞ 주어진 solution 코드를 분석하여 Sorted Linked List Application 구현
- ☞ 구현한 List를 template을 이용한 Generic class로 변환

# 예제: SortedLinkedList ADT(1/2)

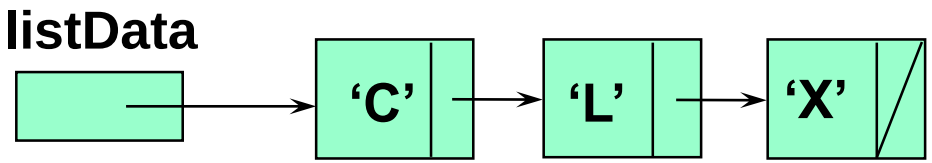
```
template <typename T>
class LinkedList
{
public:
      LinkedList();                  // Constructor
     ~LinkedList();                  // Destructor

      void MakeEmpty();              // List를 비움..
      int GetLength() const;         // 리스트가 보유하고 있는 item 개수 반환
      int Add(T item);               // 새로운 레코드를 리스트에 삽입.
      int Get(T &item);              // Primary key를 기준으로 데이터를 검색하고 해당 데이터를 가져옴
      void ResetList();              // 레코드 포인터 초기화
      void GetNextItem(T &item); // Current Pointer 가 다음 node 를 가리키도록 이동 후 해당 레코드를 가져옴.

 private:
      NodeType<T> *m_pList;          // 리스트 포인터
      NodeType<T> *m_pCurPointer;    // current pointer
      int m_nLength;                 // 리스트에 저장된 레코드 수
};
```

# 예제: SortedLinkedList ADT(2/2)

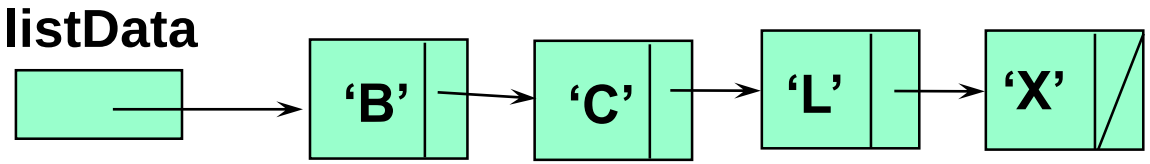
◎ Declaration use “ struct ‘NodeType’ ”

```
template <typename T>
struct NodeType
{
    T info;                 // 노드에서 관리할 레코드
    NodeType *next;         // 다음 노드를 가리키는 포인터
};
```

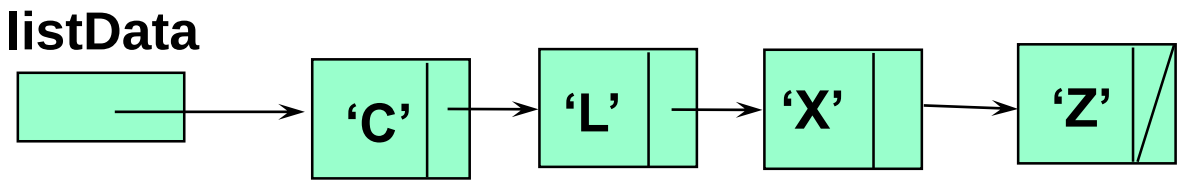
# SortedList::Add



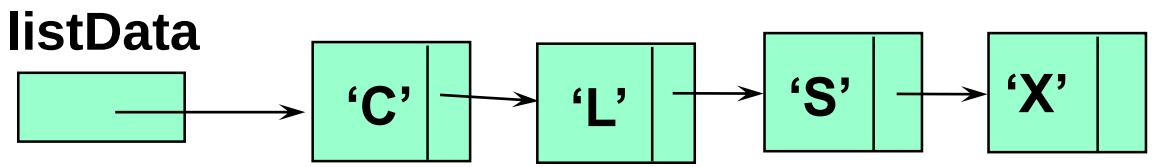

Original List




앞에 추가: listData가 수정됨

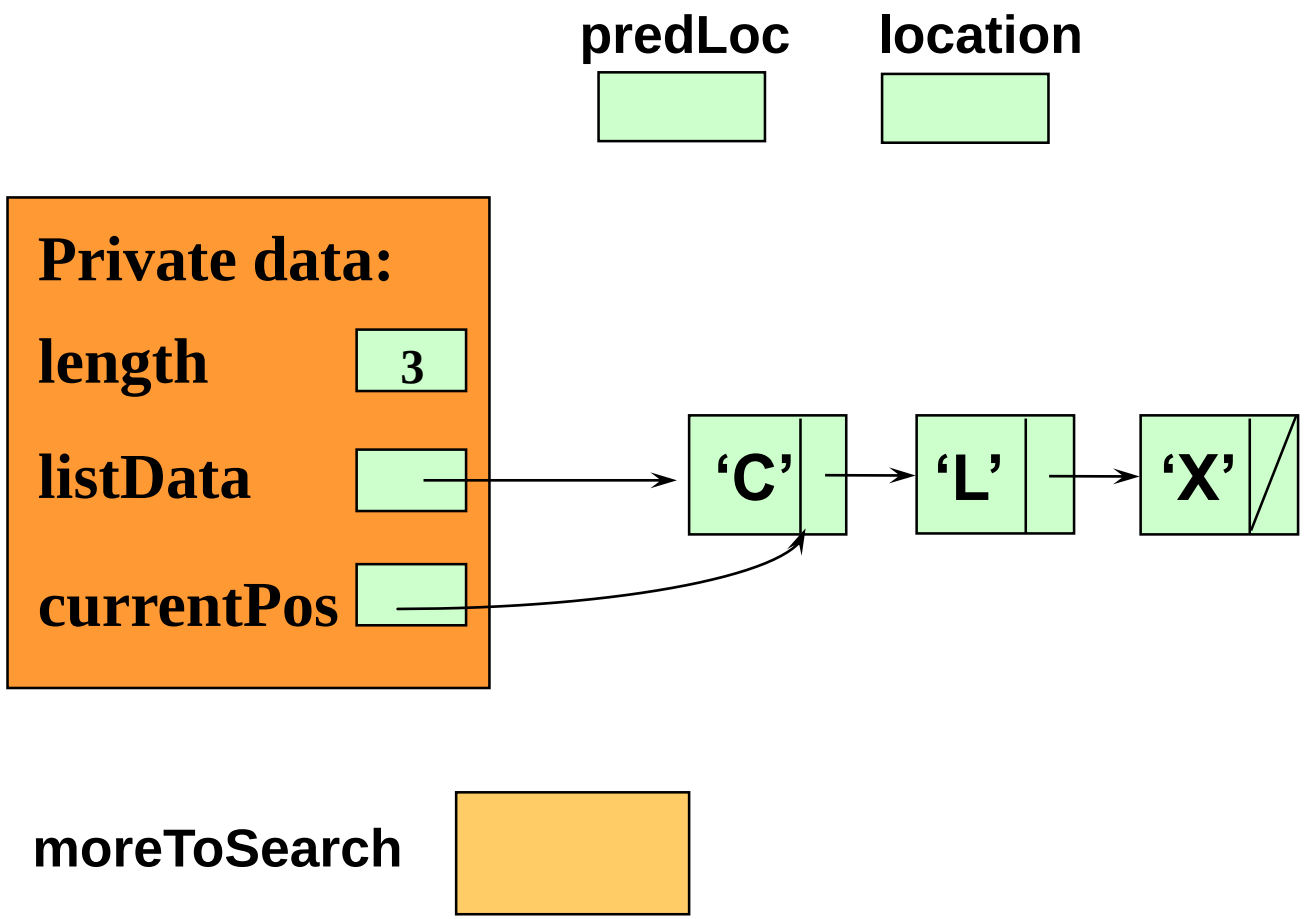



마지막에 추가: X노드와 C노드의 next 포인터가 수정됨

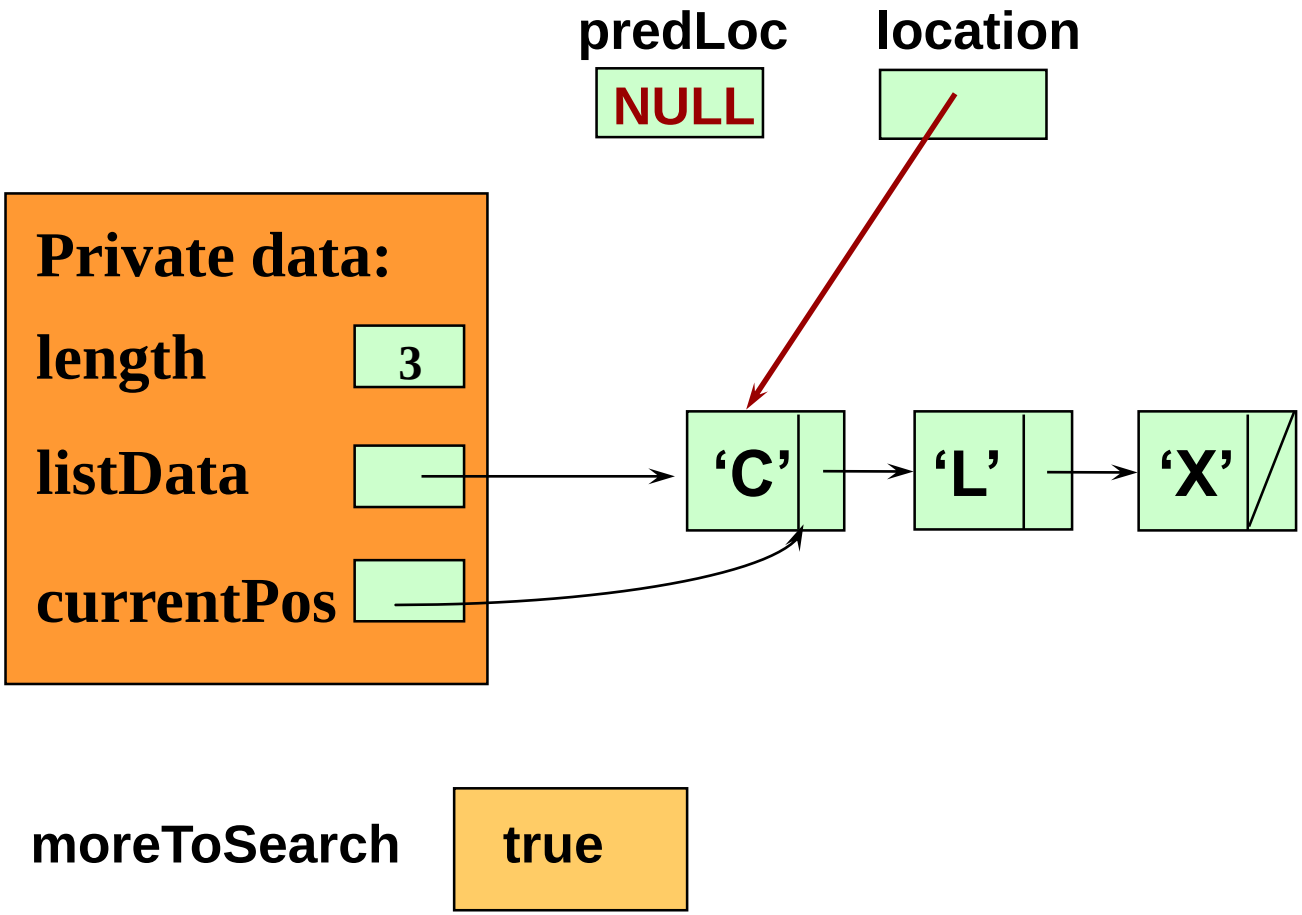



중간에 추가:현재와 이전 노드 포인터가 필요함

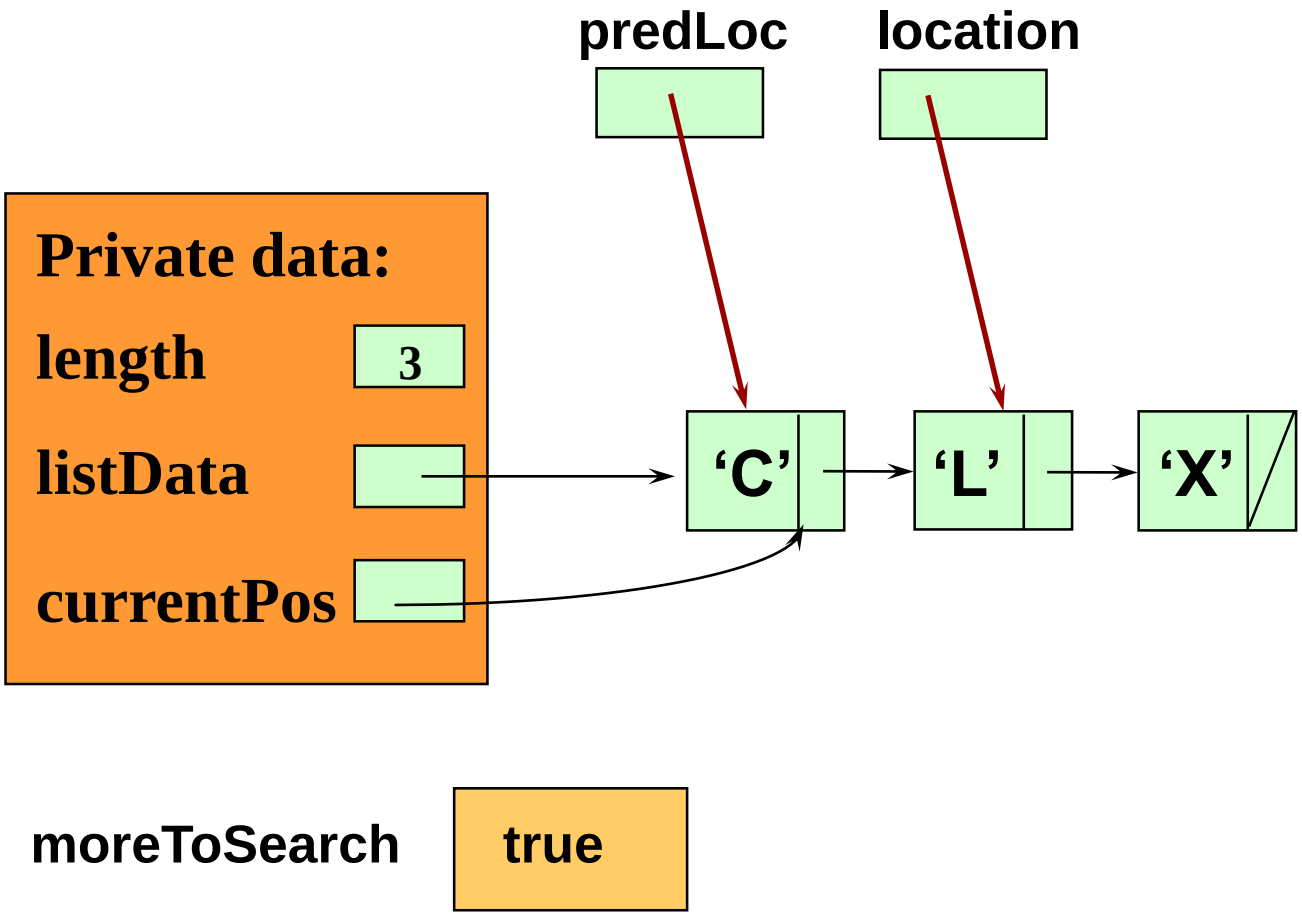
# Inserting 'S' into a Sorted List



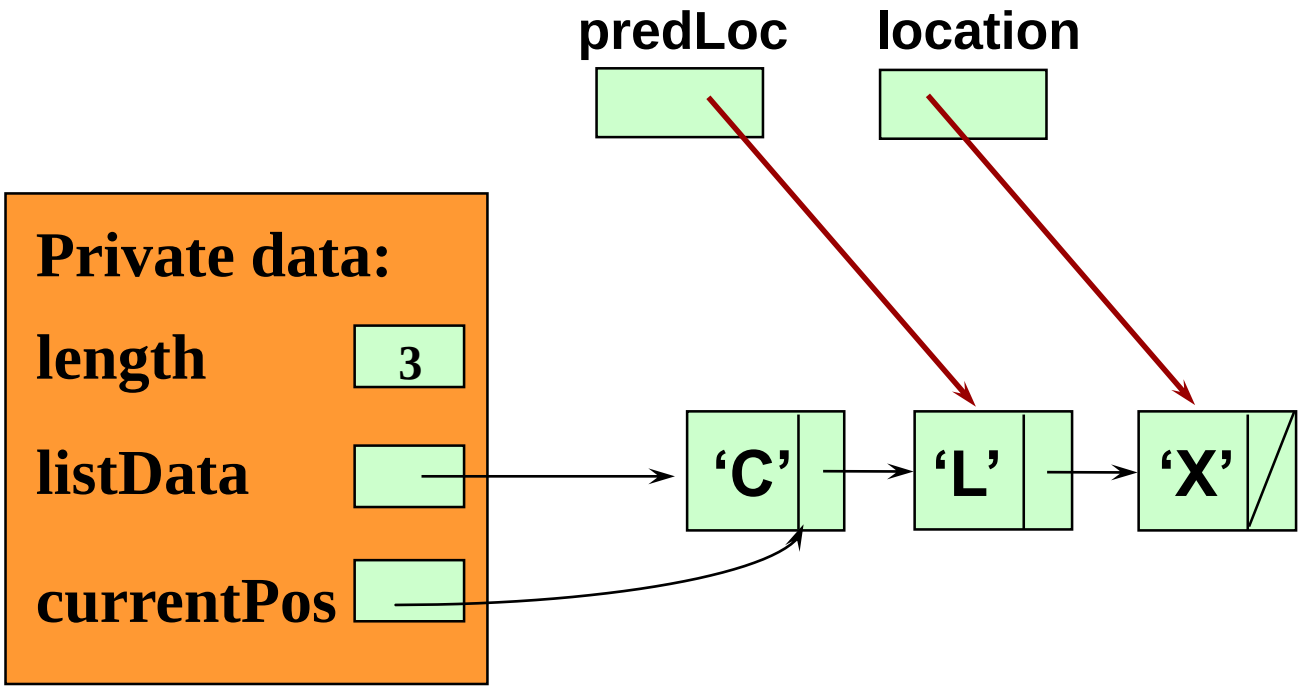
# Finding proper position for 'S'



# Finding proper position for 'S'



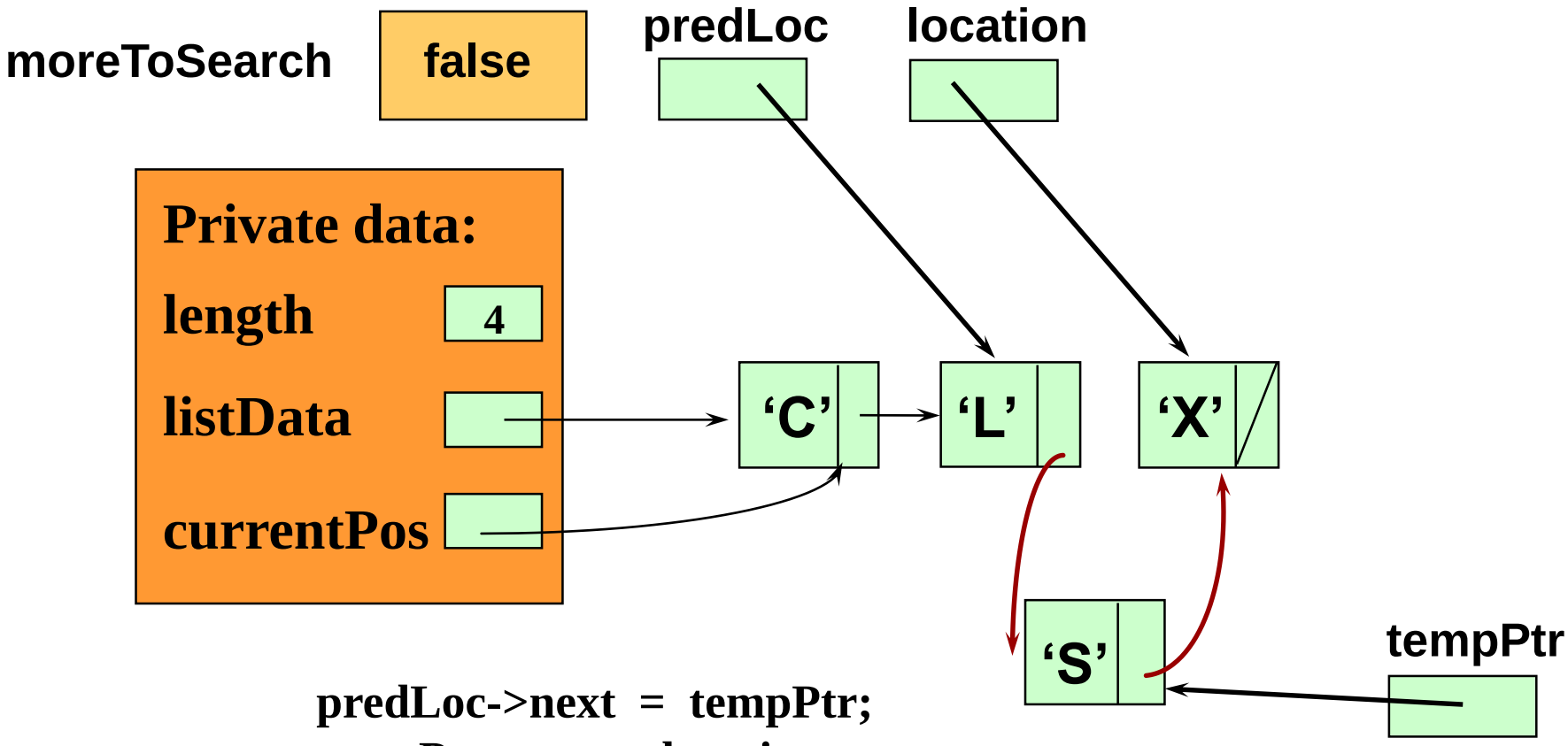
# Finding proper position for 'S'




moreToSearch
But S<X

true

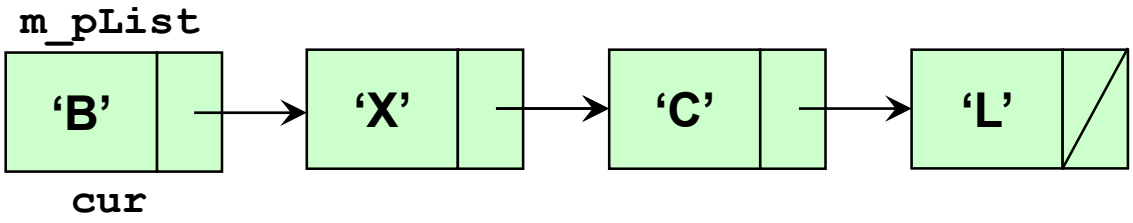
# Inserting 'S' into proper position



# Lab04: Reference (2/2)

DeleteItem ‘C’

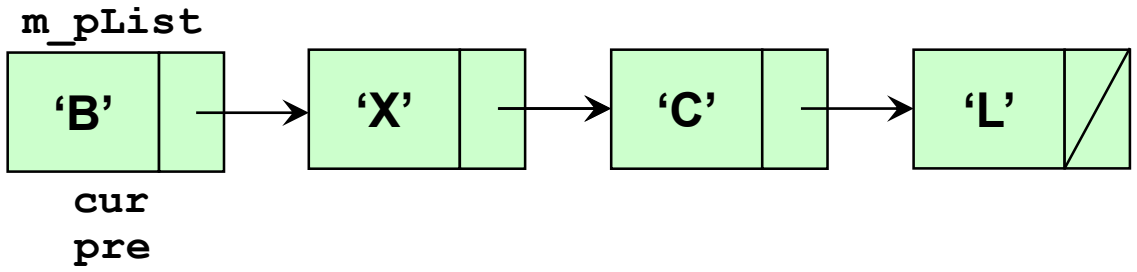
```
cur = m_List;
while(cur != null)
        if(cur->info != DeleteItem->info)
                pre = cur;
                cur = cur->next;
        else
                pre->next = cur->next;
                delete cur;
```



# Lab04: Reference (2/2)

DeleteItem ‘C’

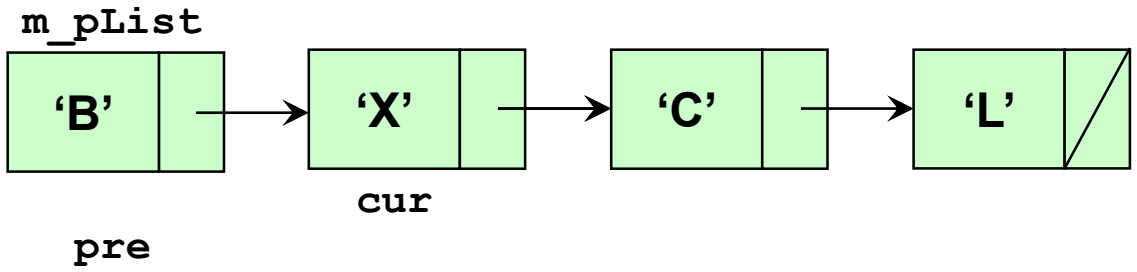
```
        cur = m_List;
        while(cur != null)
                if(cur->info != DeleteItem->info)
                        pre = cur;
                        cur = cur->next;
                else
                        pre->next = cur->next;
                        delete cur;
```



# Lab04: Reference (2/2)

DeleteItem ‘C’

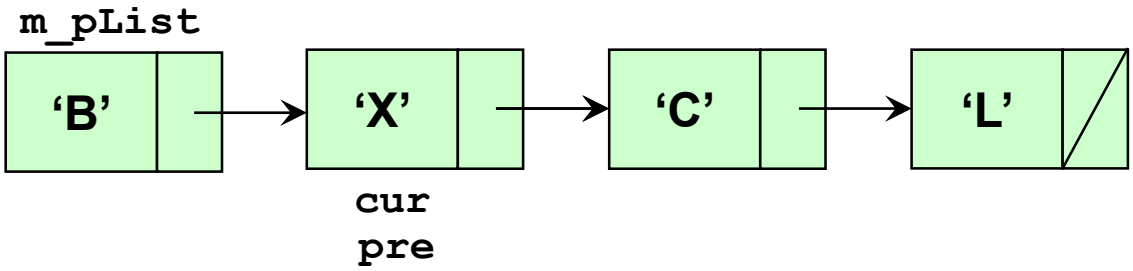
```
        cur = m_List;
        while(cur != null)
                if(cur->info != DeleteItem->info)
                        pre = cur;
                        cur = cur->next;
                else
                        pre->next = cur->next;
                        delete cur;
```



# Lab04: Reference (2/2)

DeleteItem ‘C’

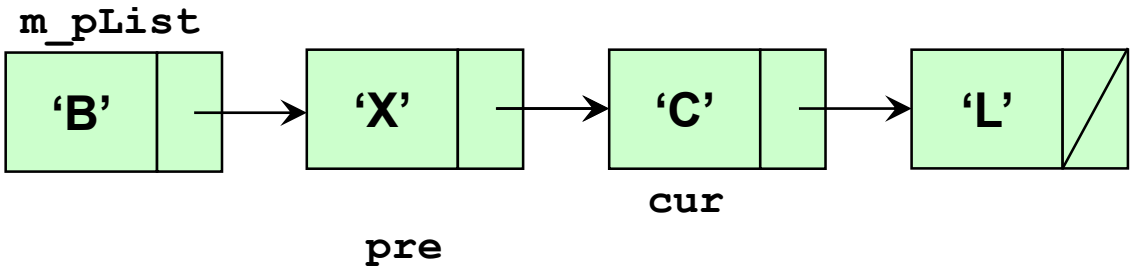
```
        cur = m_List;
        while(cur != null)
                if(cur->info != DeleteItem->info)
                        pre = cur;
                        cur = cur->next;
                else
                        pre->next = cur->next;
                        delete cur;
```



# Lab04: Reference (2/2)

DeleteItem ‘C’

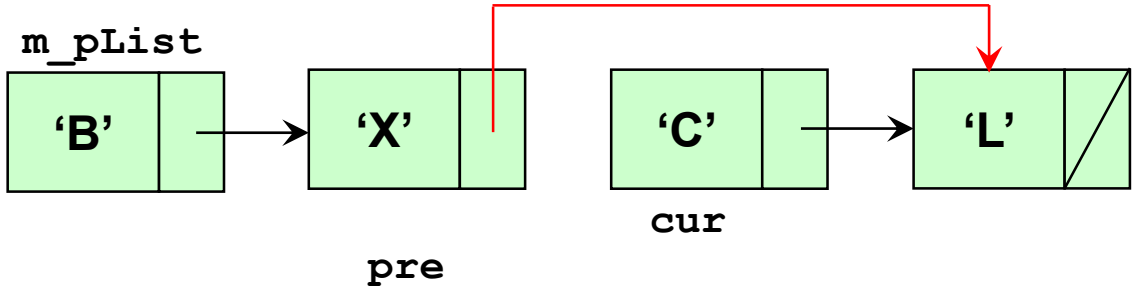
```
        cur = m_List;
        while(cur != null)
                if(cur->info != DeleteItem->info)
                        pre = cur;
                        cur = cur->next;
                else
                        pre->next = cur->next;
                        delete cur;
```



# Lab04: Reference (2/2)

DeleteItem `'C'`

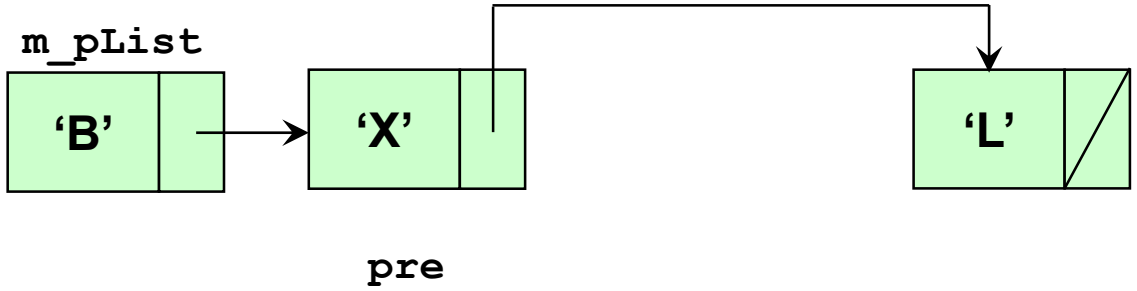
```
        cur = m_List;
        while(cur != null)
                if(cur->info != DeleteItem->info)
                        pre = cur;
                        cur = cur->next;
                else
                        pre->next = cur->next;
                        delete cur;
```



# Lab04: Reference (2/2)

DeleteItem ‘C’

```
cur = m_List;
while(cur != null)
        if(cur->info != DeleteItem->info)
                pre = cur;
                cur = cur->next;
        else
                pre->next = cur->next;
                delete cur;
```



```
void  SortedType :: InsertItem ( ItemType  item )
{    bool  moreToSearch ;
     NodeType<ItemType>*  location, predLoc, tempPtr ;
     // Allocate new node and store new item
     tempPtr = new  NodeType<ItemType> ;
     tempPtr->info = item ;
     // Empty list. Add the first item
     if (listData==NULL)        {
            listData=tempPtr;    tempPtr->next=NULL;     length++;
       }
     else   {
          // Find the position of new item
          location = predLoc=listData ;
         moreToSearch = ( location !=NULL ) ;
         while ( moreToSearch  )
         {
               switch ( item.ComparedTo( location->info ) )   {
                     case   LESS        : // insert at the front of the current node
                                          moreToSearch = false ;
                                  break ;
```

```
                    case  GREATER  :   predLoc=location;
                                location=location->next;
                                moreToSearch = ( location !=NULL ) ;
                                break ;
                          // case EQUAL: error message
                    }
              }
              if (predLoc==location) { // add to front
                   tempPtr->next = listData;     listData=tempPtr;
              }
              else {       // add between  two nodes or to the end(location==NULL)
                    predLoc->next = tempPtr;
                    tempPtr->next=location;
              }
              length++ ;
         }
    }
```

# 예제: console

◎ List를 테스트할 driver는 다음과 같이 작성함

```
--- ID – Command -----
        1 : Insert Item
        2 : Delete Item
        3 : Replace Item
        4 : Retrieve Item
        5 : Display all Item
        0 : Quit

Choose a Command -->
```

# 실습 : Singer List구현

- 예제3에서 구현한 배열을 이용한 Sorted List를 이용하여 masterList를 정의
  - masterList: 학생들의 자세한 신상기록을 ID 순으로 저장
  - 이 부분은 예제 3 실습과 동일
- 실습 4 예제에서 구현한 sorted singly list를 이용하여 컴퓨터공학과 학생 리스트를 생성하고 다음 작업을 수행
  - 학과 리스트(classList)에는 소속 학생의 학번만 저장
  - 학과 소속학생 이름을 다음과 같이 출력하는 기능구현
    - 학과 리스트를 차례대로 방문하면서 학번을 읽어서 이것을 이용하여 masterList에서 자세한 정보를 검색하여 화면에 출력
- 힌트: masterList와 classList를 application class의 멤버변수로 추가하고 위 기능을 멤버함수로 추가