

## 자료구조 실습05

Data Structures Lab05

## Lab05 예제(1/2)

◎목표: Doubly Linked List의 구현

◎내용:

☞ 과제

➤ Iterator Class 정의 및 이를 이용한 Doubly Linked List를 구현

◎방법

☞ Doubly Linked list의 메커니즘을 분석하고 sorted linked list의 ADT를 바탕으로 구현

☞ Iterator class의 필요성과 메커니즘을 분석하고 ADT를 바탕으로 구현

☞ Iterator class 란?

➤ 다음의 site를 참고

✓ <http://www.cplusplus.com/reference/std/iterator>

## Lab05 예제(2/2)

### ◎ Iterator Class란?

- ☞ 구조화된 자료에서 원소(data)에 체계적(혹은 순차적)으로 접근
- ☞ List Class의 복잡한 연산을 돕기 위해 전체 List를 순차적으로 접근하는 기능을 별도로 제공
- ☞ List의 멤버변수인 first, last와 NodeType의 data에 접근 가능하도록 friend로 선언되어야 함
- ☞ Iterator를 함수가 아닌 Class로 구현하는 이유
  - iterator를 함수로 구현(ListType class의 ResetCurNode, GetNextItem 등) 하여 사용할 경우 여러 곳에서 동시에 iteration 기능 사용 불가능
  - Iterator Class를 이용해 위의 경우를 해결

## 예제: DoublySortedLinkedList ADT(1/2)

```
template <typename T>
class DoublySortedLinkedList
{
    friend class DoublyIterator<T>;
public:
    DoublySortedLinkedList();           // Default constructor
    ~DoublySortedLinkedList();          // Destructor

    bool IsFull();                      // 리스트가 가득 찼는지 확인
    void MakeEmpty();                   // 리스트를 비움
    int GetLength() const;              // 리스트가 보유하고 있는 item 개수 반환
    void Add(T item);                   // 새로운 레코드 추가
    void Delete (T item);               // 기존 레코드 삭제
    void Replace (T item);              // 기존 레코드 갱신
    int Get(T &item);                   // Primary key를 기준으로 데이터를 검색하고 해당 데이터를 가져옴

private:
    DoublyNodeType<T> *m_pFirst;        // 리스트의 처음 노드를 가리키는 포인터
    DoublyNodeType<T> *m_pLast;        // 리스트의 마지막 노드를 가리키는 포인터
    int m_nLength;                     // 리스트에 저장된 레코드 수
}
```

## 예제: DoublySortedLinkedList ADT(2/2)

```
template <typename T>
struct DoublyNodeType
{
    T data;                // data
    DoublyNodeType *prev;  // Pointer of previous node
    DoublyNodeType *next;  // Pointer of next node
}
```

## 예제: Iterator ADT

```
template <typename T>
class DoublyIterator
{
    friend class DoublySortedLinkedList<T>;
public:
    DoublyIterator(const DoublySortedLinkedList<T> &list): m_List(list), m_pCurPointer(list.m_pFirst)
    {
        // Default constructor
    };
    bool NotNull();
    // list의 현재 원소가 Null이 아닌지 검사
    bool NextNotNull();
    // list의 다음 원소가 Null이 아닌지 검사
    T First();
    // list의 처음 node의 item을 리턴
    T Next();
    // 다음 node로 이동하고 해당 node의 item을 리턴
    DoublyNodeType<T> GetCurrentNode ();
    // 현재 node를 리턴

private:
    const DoublySortedLinkedList<T> &m_List;
    // 사용할 리스트의 참조 변수
    DoublyNodeType<T> *m_pCurPointer;
    // Iterator 변수
};
```

## 예제: Example

// Initialize list to empty state.

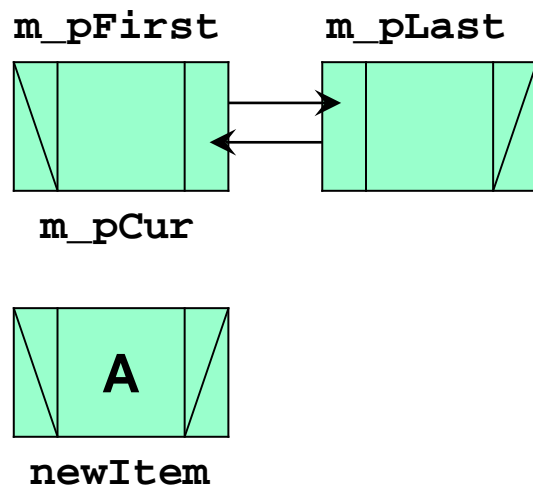
```
template <typename ItemType>
int DoublySortedList<ItemType>::MakeEmpty()
{
    DoublyNodeType<ItemType> *pItem;    // 리스트를 참조하여 삭제를 위한 변수
    DoublyIterator<ItemType> itor(*this); // this 포인터를 이용하여 Iterator 선언

    while(itor.IsNotNull())              // 리스트 비어있지 않으면 참
    {
        pItem = itor.m_pCurPos;         // 현재 curPoint가 가르키는 데이터를 참조
        itor.Next();                     // 리스트에서 curPoint를 다음으로 이동
        delete pItem;
    }

    m_pFirst = m_pLast = NULL;           // 초기화

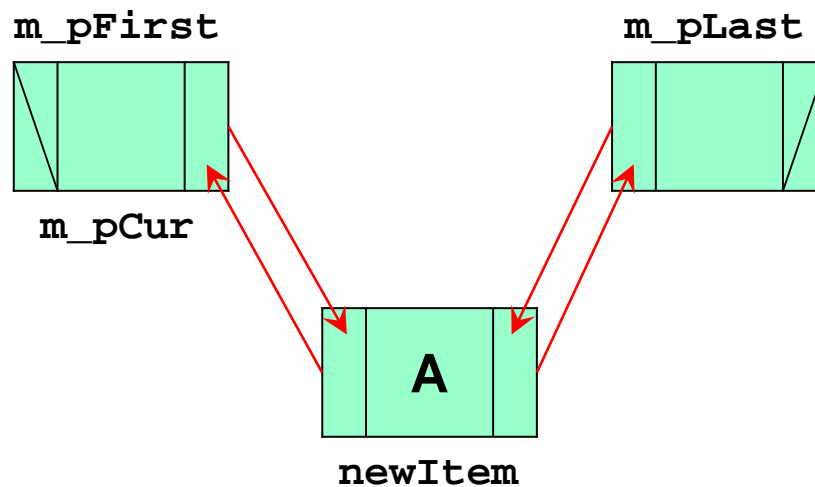
    return 1;
}
```

## Lab05: Reference Add (1/13)

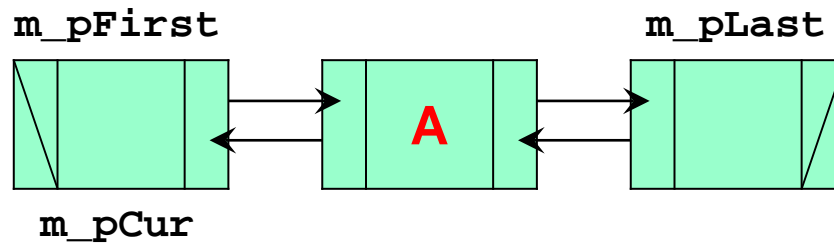




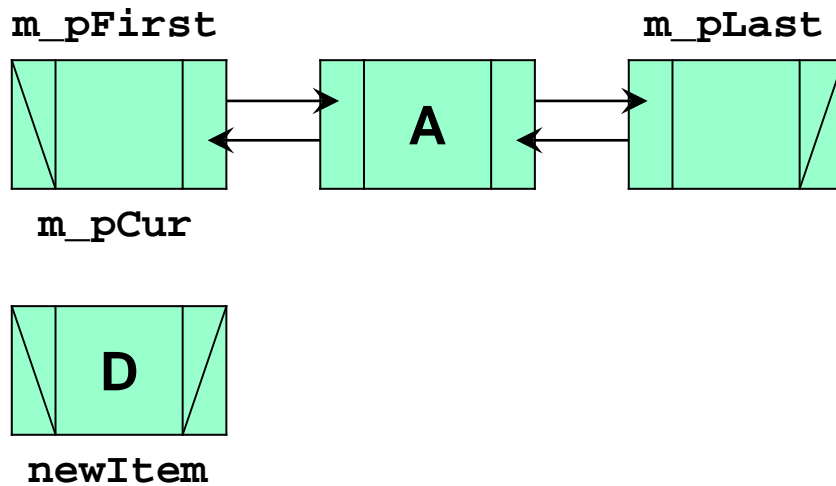
## Lab05: Reference Add (2/13)



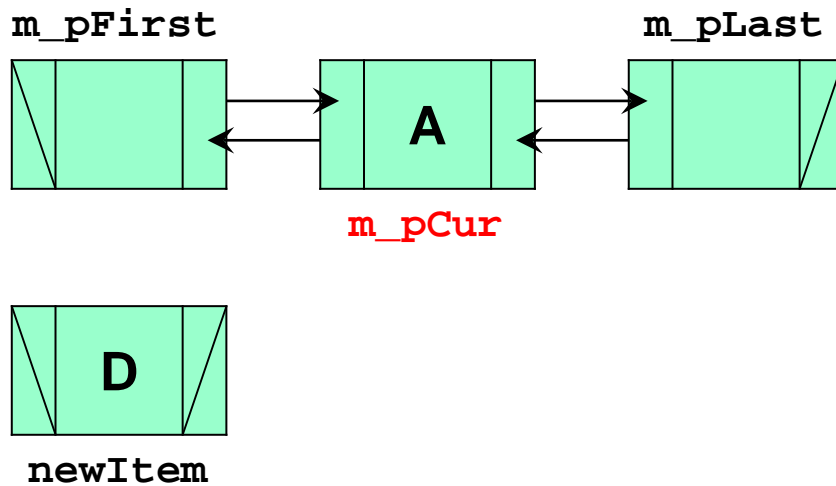
## Lab05: Reference Add (3/13)



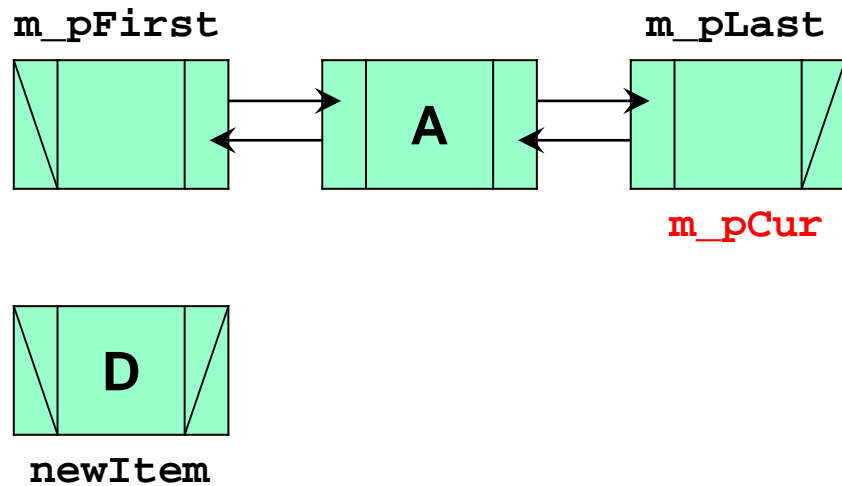
## Lab05: Reference Add (4/13)



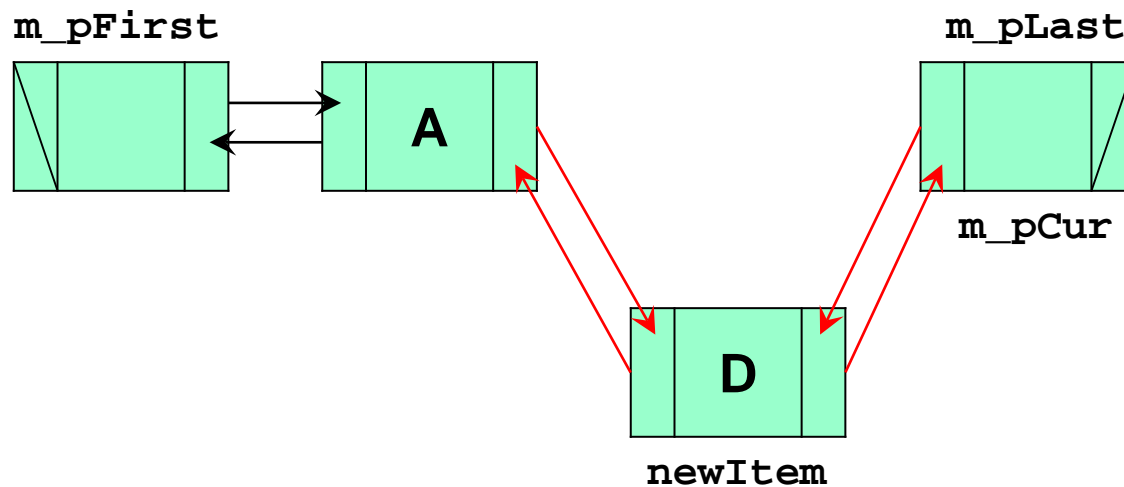
## Lab05: Reference Add (5/13)



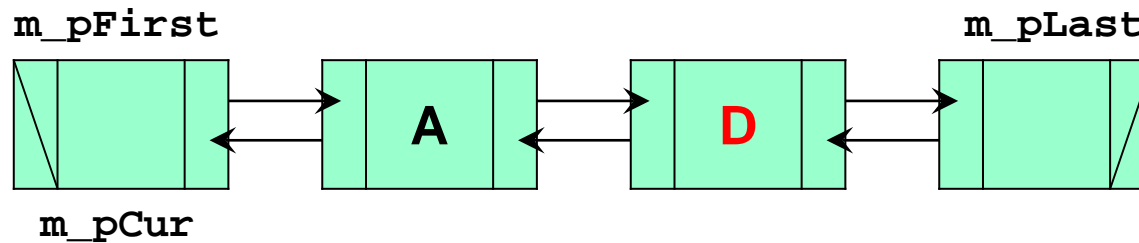
## Lab05: Reference Add (6/13)



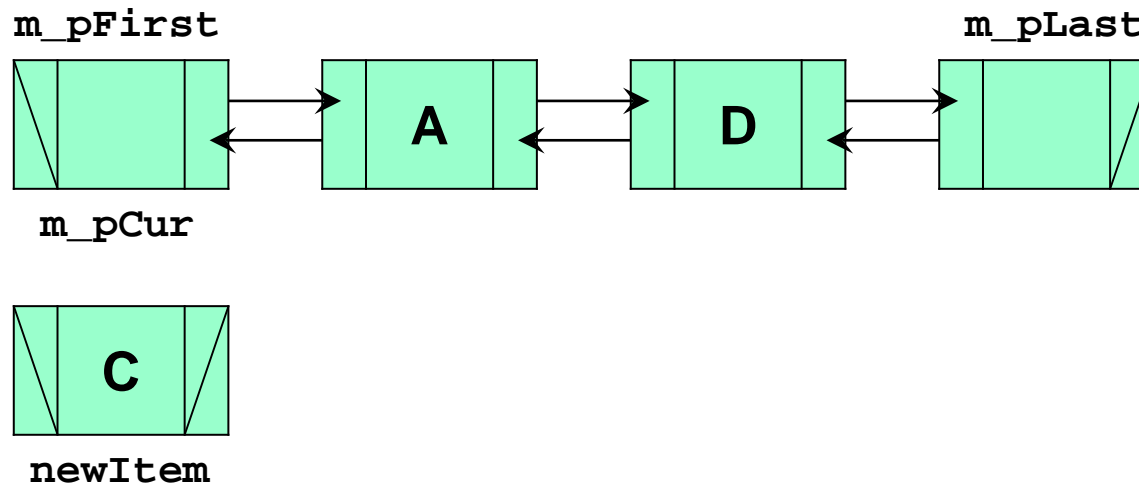
## Lab05: Reference Add (7/13)



## Lab05: Reference Add (8/13)

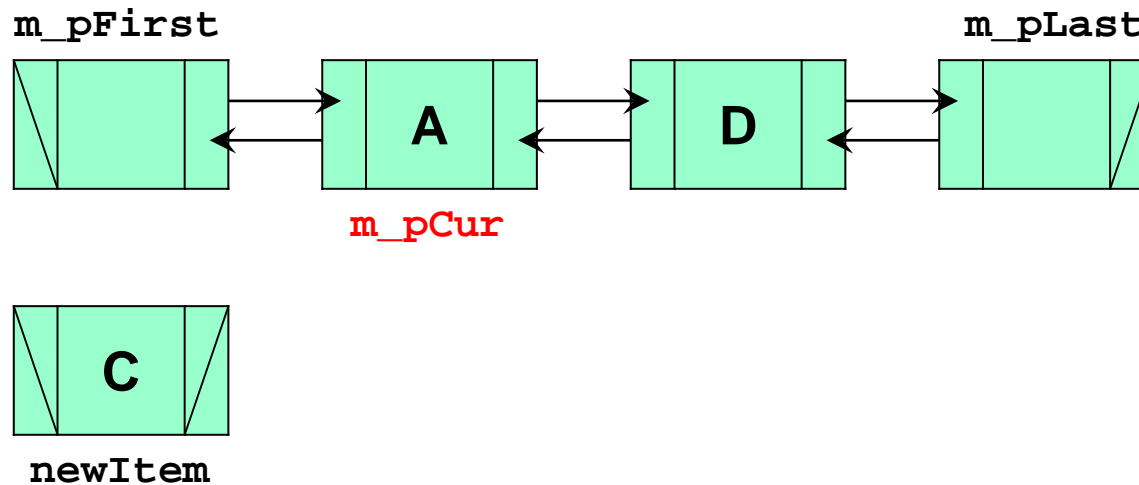


## Lab05: Reference Add (9/13)

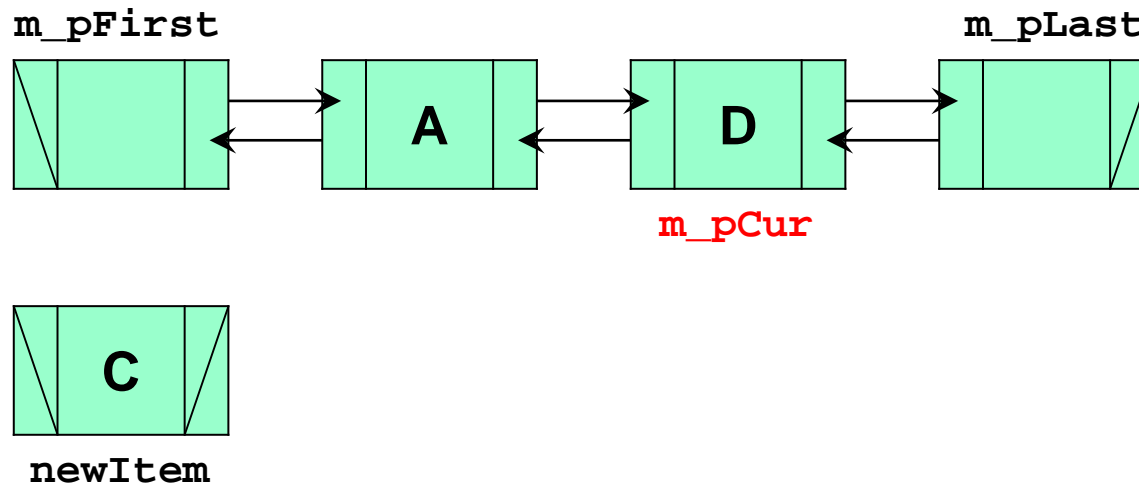




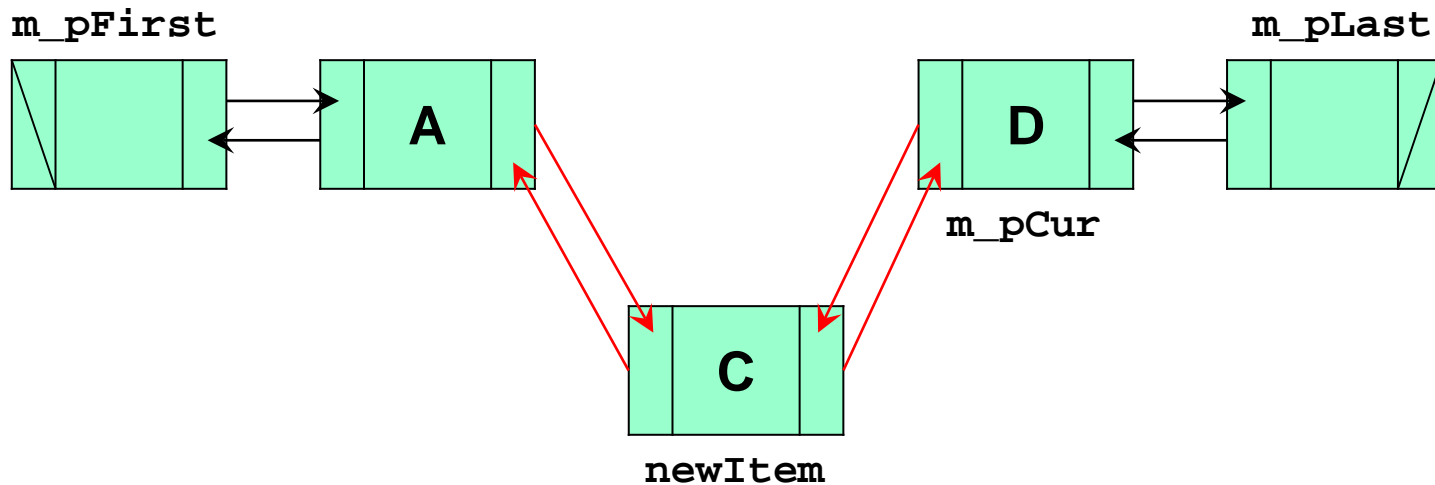
## Lab05: Reference Add (10/13)



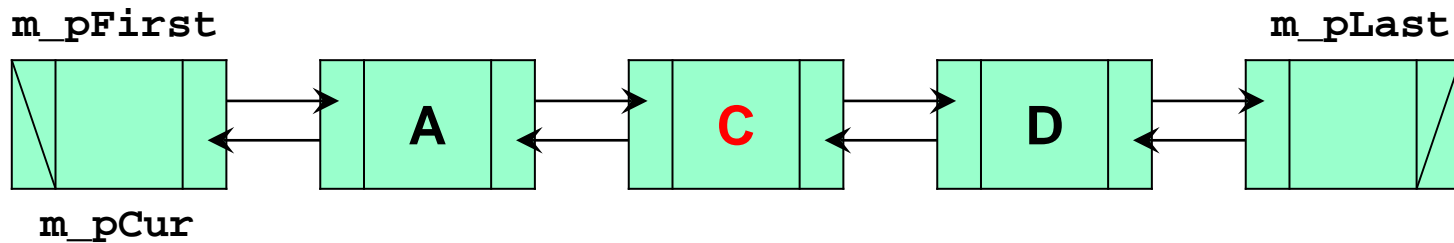
## Lab05: Reference Add (11/13)



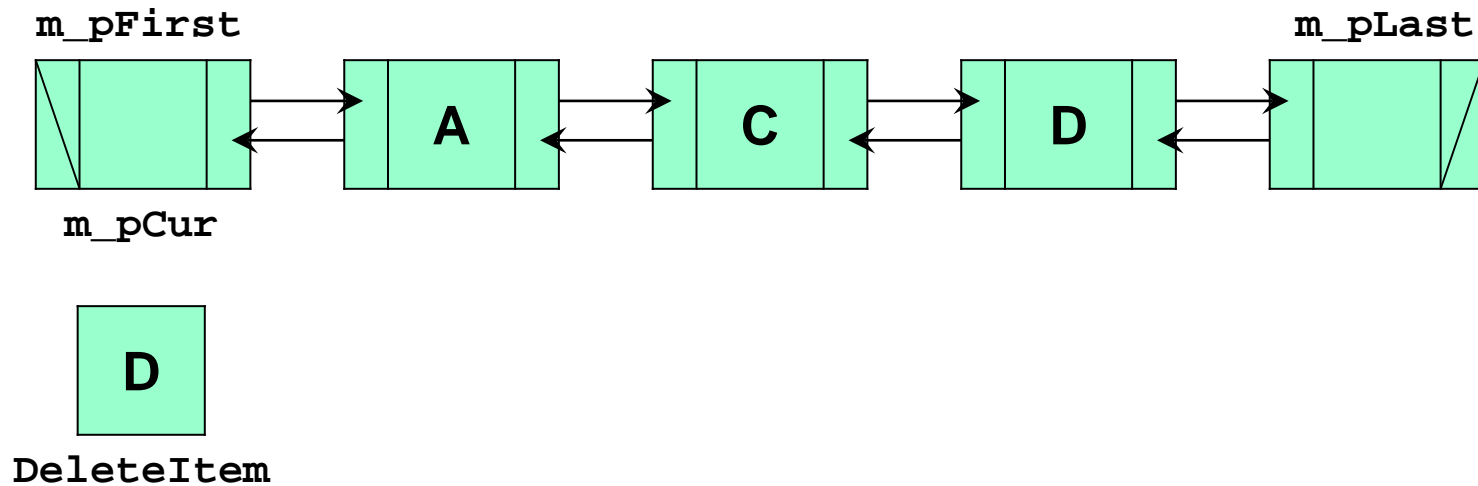
## Lab05: Reference Add (12/13)



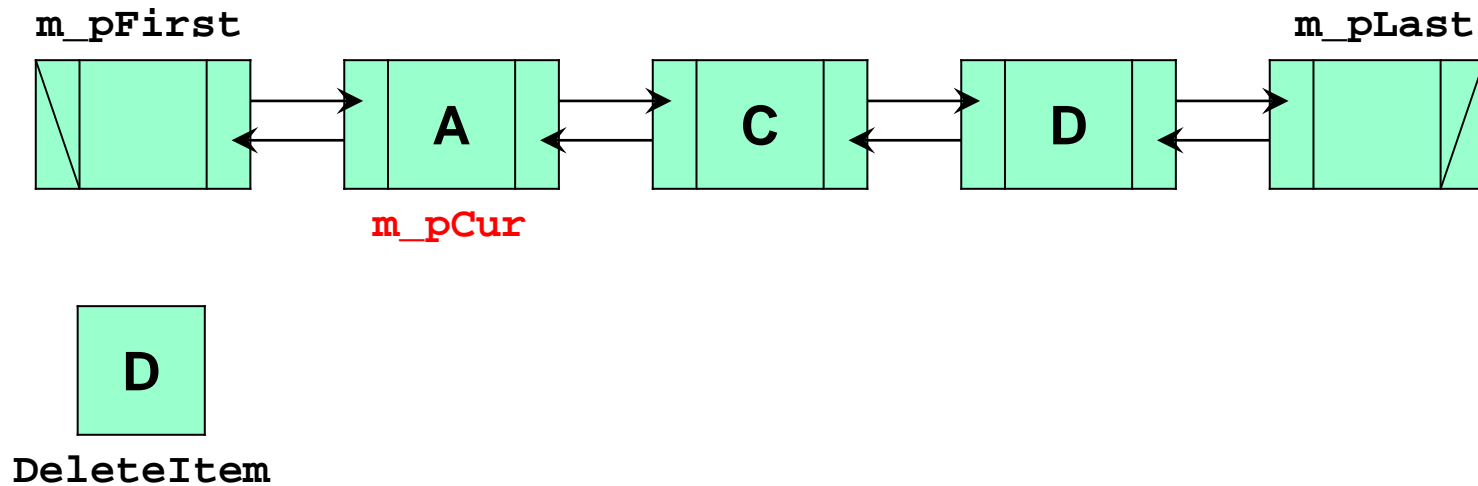
## Lab05: Reference Add (13/13)



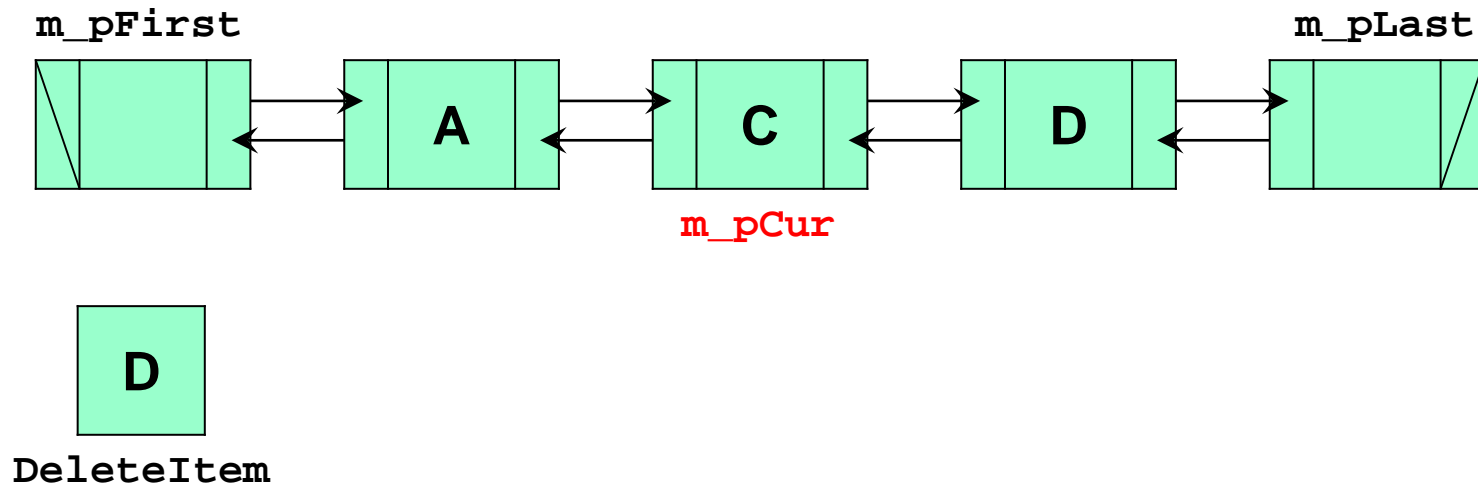
## Lab05: Reference Delete (1/7)



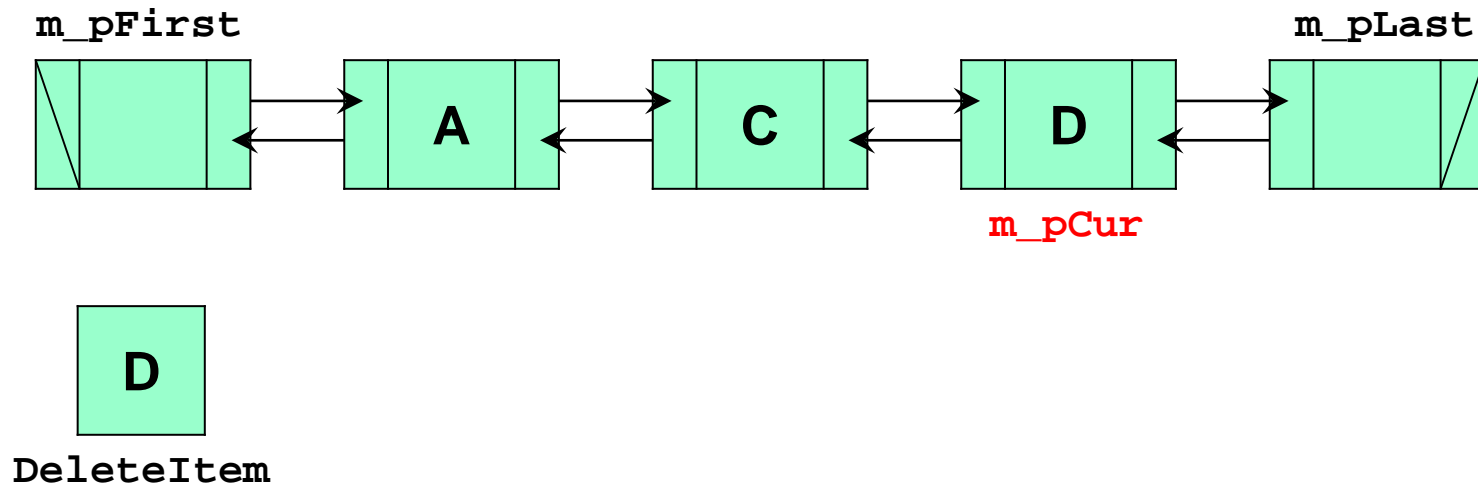
## Lab05: Reference Delete (2/7)



## Lab05: Reference Delete (3/7)

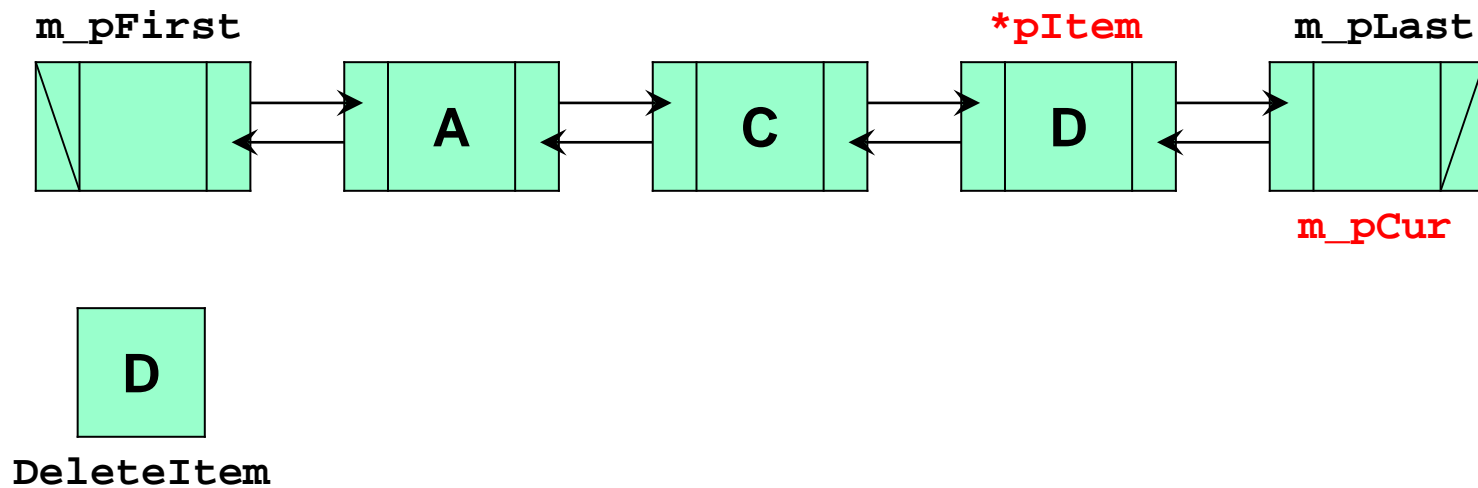


## Lab05: Reference Delete (4/7)

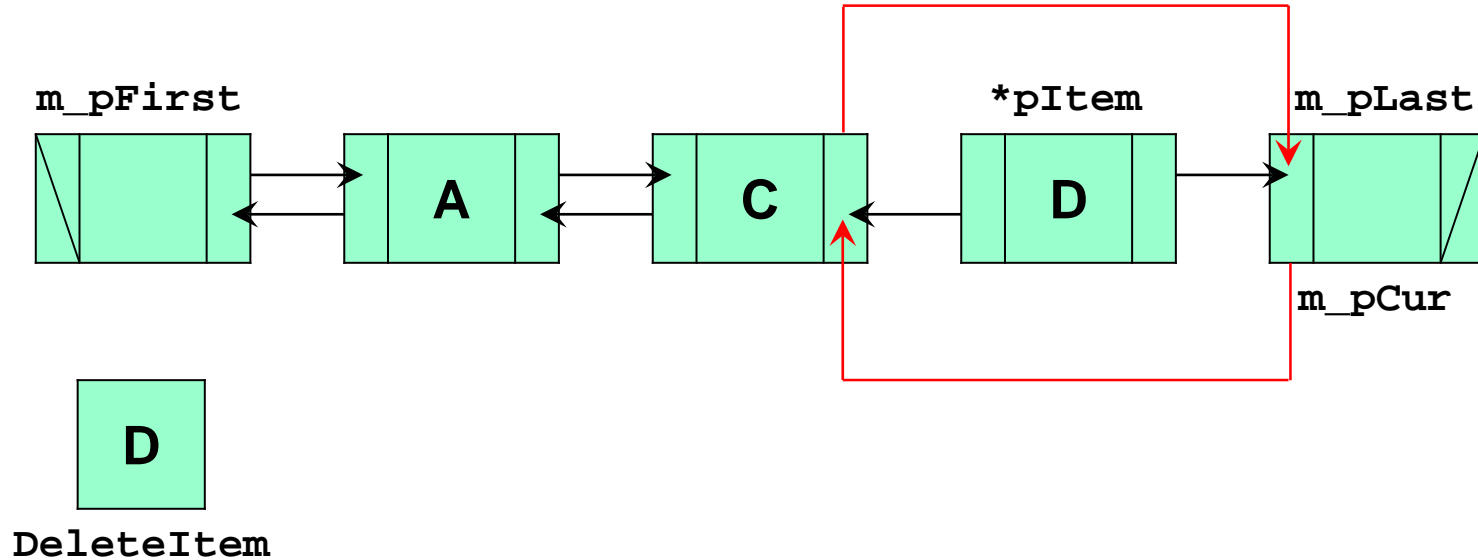




## Lab05: Reference Delete (5/7)



## Lab05: Reference Delete (6/7)



## Lab05: Reference Delete (7/7)

