6주차 2차시 이중 연결 리스트

【학습목표】

1. 이중 연결 리스트의 특징을 설명할 수 있다.
2. 이중 연결 리스트의 삽입, 삭제 연산에 대하여 설명할 수 있다.

학습내용1 : 이중 연결 리스트

1. 정의

* 원형 연결 리스트에서 현재 노드의 이전 노드를 접근하려면 전체 리스트를 한 바뀌 순회해야 하는 문제가 있어 이를 해소하기 위해 양쪽 방향으로 순회할 수 있도록 노드를 연결한 리스트

* 이중 연결 리스트 구조
- 두 개의 링크 필드와 한 개의 데이터 필드로 구성
- llink(Left Link) 필드는 왼쪽 노드와 연결하는 포인터
- rlink(Right Link) 필드는 오르쪽 노드와 연결하는 포인터

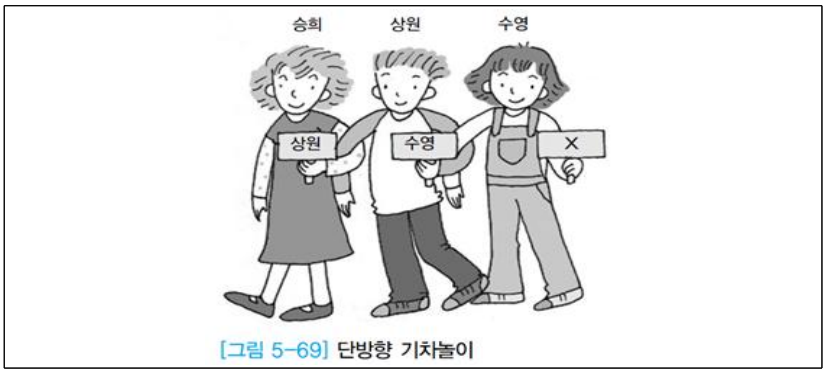
* 이중 연결 리스트 노드의 구조

[그림 5-68] 이중 연결 리스트의 노드 구조

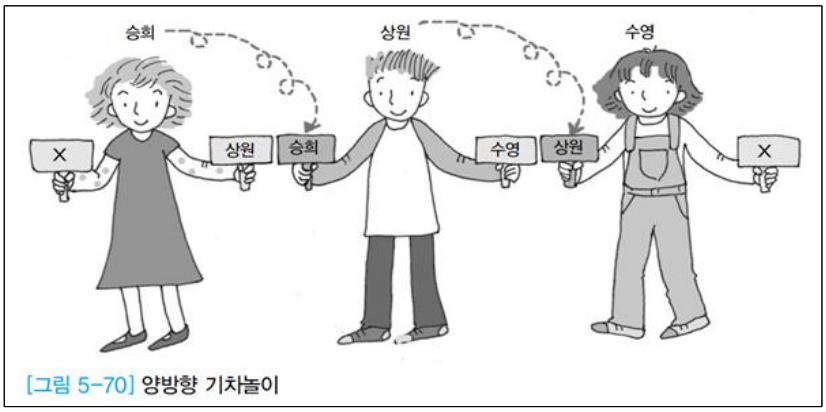
* 이중 연결 리스트 노드의 구조체 정의

```
typedef struct Dnode{
        struct Dnode *llink;
        char data[5];
        struct Dnode *rlink;
}
```

2. 단방향 vs. 양방향 기차



[그림 5-69] 단방향 기차놀이

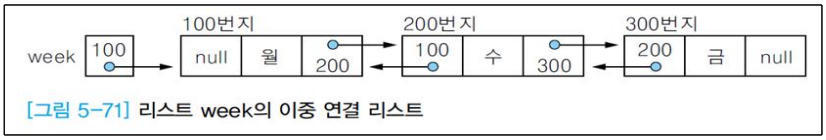


[그림 5-70] 양방향 기차놀이

- 양방향 기차를 이중 연결 리스트라고 가정하면, 아이들은 노드가 되고, 왼손에 있는 이름표는 llink, 오른손에 있는 이름표는 rlink가 된다

3. 이중 연결 리스트 예제

* 리스트 week=(월, 수, 금)의 이중 연결 리스트 구성

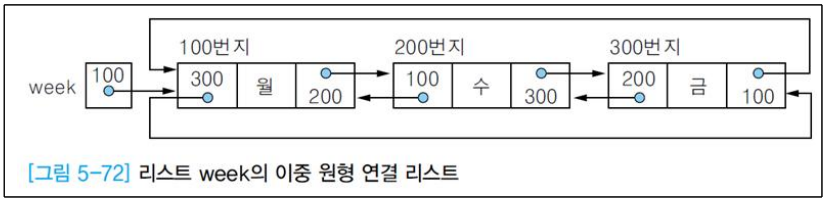


[그림 5-71] 리스트 week의 이중 연결 리스트

- 이중 연결 리스트에서 첫 번째와 마지막 노드가 아닌 임의의 노드 p에 대하여 다음이 성립

p = plink.rlink = prlink.llink

* 원형 이중 연결 리스트

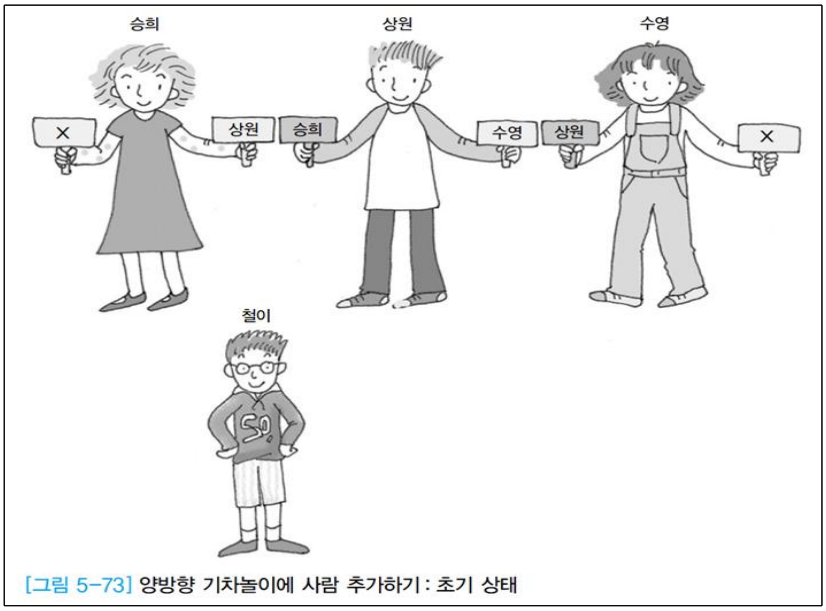


[그림 5-72] 리스트 week의 이중 원형 연결 리스트

- 이중 연결 리스트를 원형으로 구성

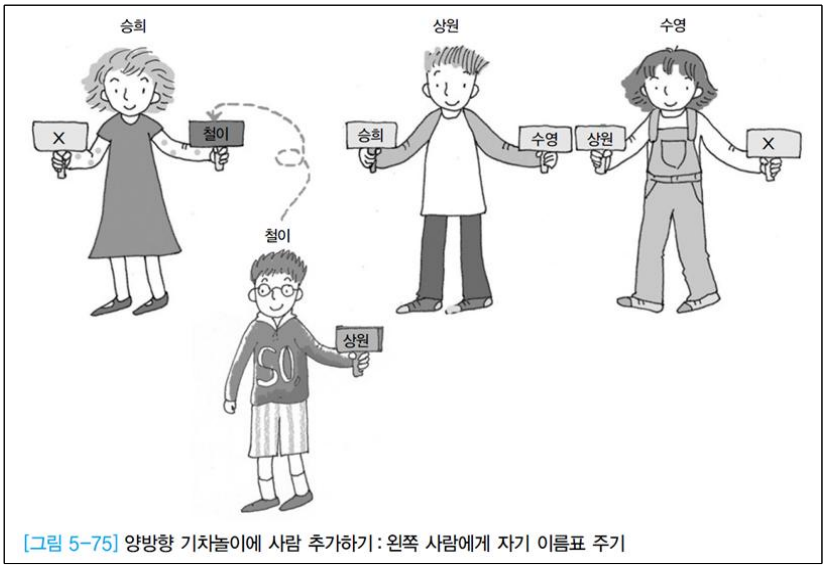
학습내용2 : 이중 연결 리스트의 삽입

1. 양방향 기차 놀이에서 사람 추가하기

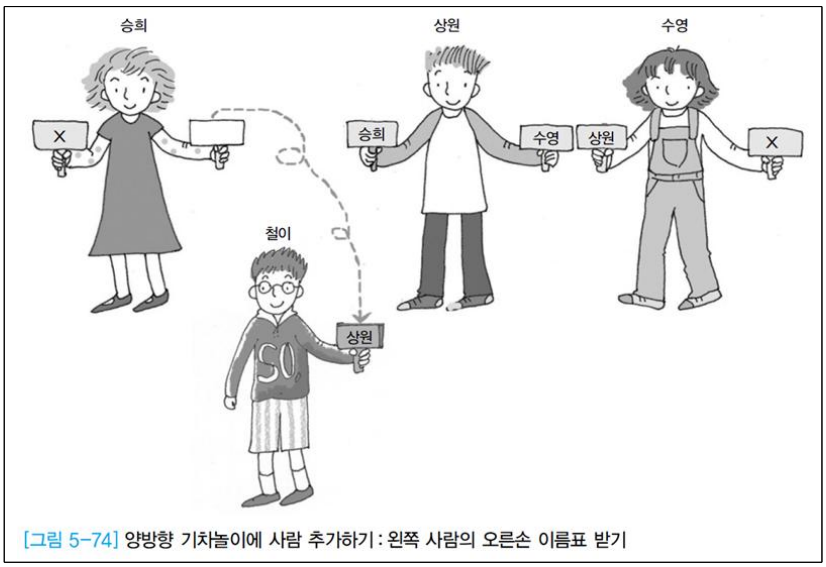


[그림 5-73] 양방향 기차놀이에 사람 추가하기 : 초기 상태

① 철이의 왼쪽 사람이 될 승희의 오른쪽 이름표를 철이의 오른손에 주고 대신 '철이' 이름표를 승희에게 준다.

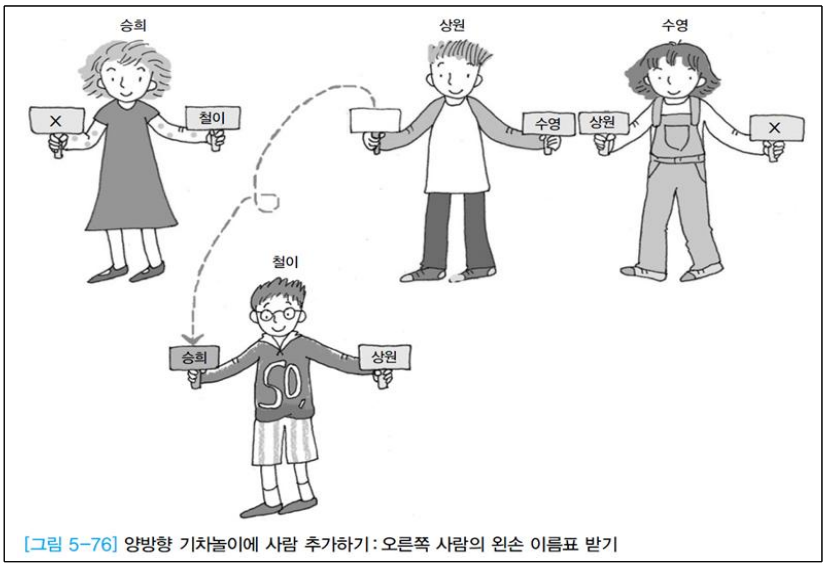


[그림 5-75] 양방향 기차놀이에 사람 추가하기 : 왼쪽 사람에게 자기 이름표 주기



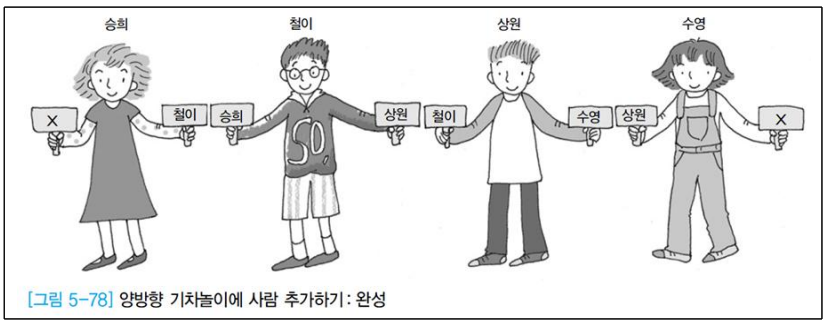
[그림 5-74] 양방향 기차놀이에 사람 추가하기 : 왼쪽 사람의 오른손 이름표 받기

② 철이의 오른쪽이 될 상원의 왼손 이름표를 철이의 왼손에 주고 대신 '철이' 이름표를 상원에게 준다

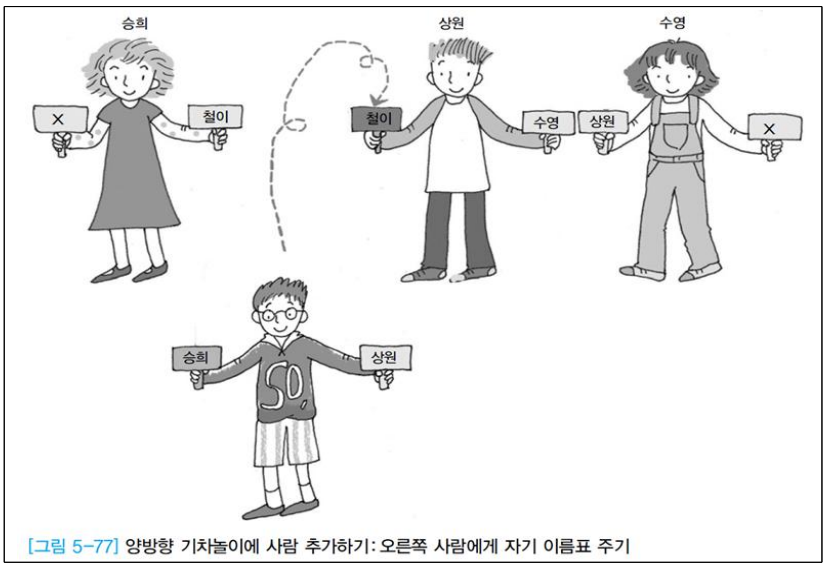


[그림 5-76] 양방향 기차놀이에 사람 추가하기 : 오른쪽 사람의 왼손 이름표 받기

③ 이름표대로 연결하면 양방향 기차가 완성된다



[그림 5-78] 양방향 기차놀이에 사람 추가하기 : 완성



[그림 5-77] 양방향 기차놀이에 사람 추가하기: 오른쪽 사람에게 자기 이름표 주기

2. 이중 연결 리스트에서의 삽입 연산

* 이중 연결 리스트에서의 원소 삽입 연산 과정

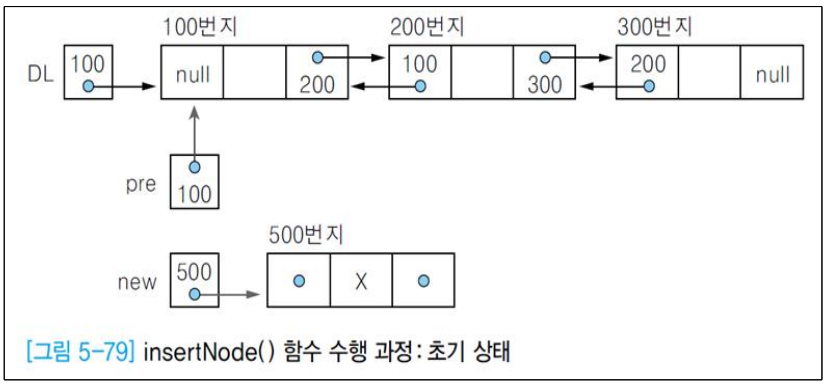
(1) 삽입할 노드를 가져온다
(2) 새 노드의 데이터 필드에 값을 저장한다
(3) 새 노드의 왼쪽 노드의 오른쪽 링크(rlink)를 새 노드의 오른쪽 링크에 저장한다
(4) 그리고 왼쪽 노드의 오른쪽 링크(rlink)에 새 노드의 주소를 지정한다
(5) 새 노드의 오른쪽 노드의 왼쪽 링크(llink)를 새 노드의 왼쪽 링크에 저장한다
(6) 그리고 새 노드의 왼쪽 링크(llink)에 새 노드의 주소를 저장한다

* 이중 연결 리스트의 원소 삽입 알고리즘

알고리즘 5-11 이중 연결 리스트의 원소 삽입 알고리즘

```
insertNode(DL, pre, x)
        new ← getNode();
        new.data ← x;
        new.rlink ← pre.rlink;      // ❶
        pre.rlink ← new;            // ❷
        new.llink ← pre;            // ❸
        new.rlink.llink ← new;      // ❹
end insertNode()
```

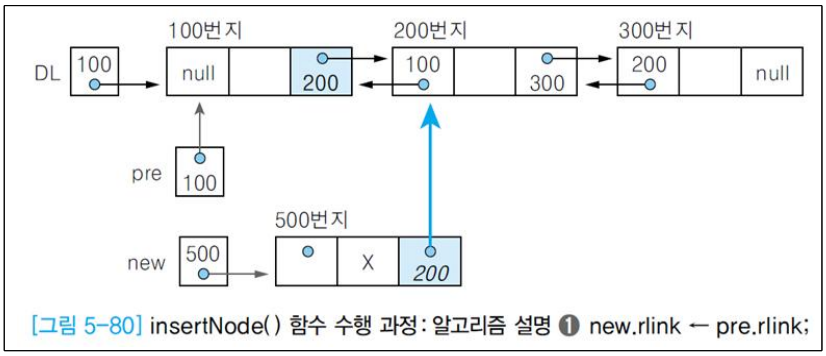
- 이중 연결 리스트 DL에서 포인터 pre가 가리키는 노드의 다음 노드로 노드 new를 삽입



[그림 5-79] insertNode( ) 함수 수행 과정 : 초기 상태

❶ new.rlink ← pre.rlink;

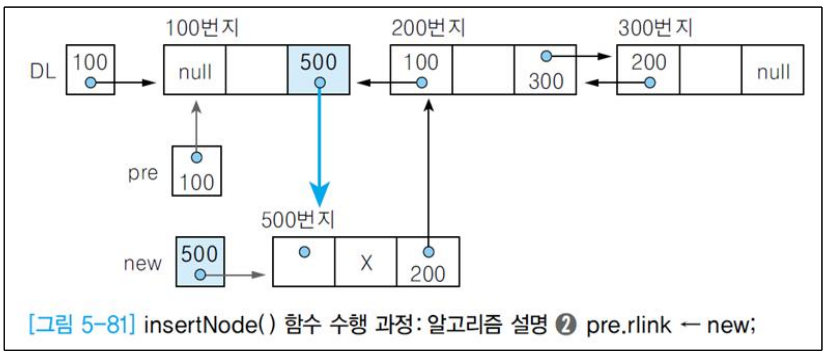
- 노드 pre의 rlink를 노드 new의 rlink에 저장하여 노드 pre의 오른쪽 노드를 삽입할 노드 new의 오른쪽 노드로 연결



[그림 5-80] insertNode( ) 함수 수행 과정 : 알고리즘 설명 ❶ new.rlink ← pre.rlink;

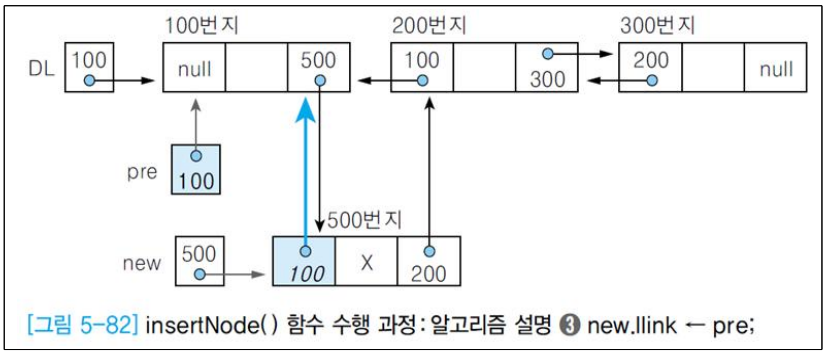
❷ pre.link ← new;

- 새 노드 new의 주소를 노드 pre의 rlink에 저장하고 노드 new를 노드 pre의 오른쪽 노드가 되도록 연결한다



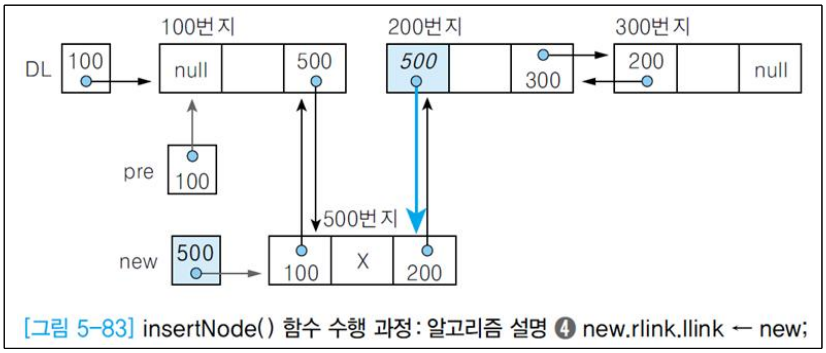
[그림 5-81] insertNode( ) 함수 수행 과정 : 알고리즘 설명 ❷ pre.rlink ← new;

❸ new.llink ← pre;
- 포인터 pre의 값을 삽입할 노드 new의 llink에 저장하여 노드 pre를 노드 new의 왼쪽 노드가 되도록 연결한다



[그림 5-82] insertNode( ) 함수 수행 과정 : 알고리즘 설명 ❸ new.llink ← pre;

❹ new.rlink.llink ← new;
- 포인터 new의 값을 노드 new의 오른쪽 노드(new.rlink)의 llink에 저장하여 노드 new의 오른쪽 노드의 왼쪽 노드로 노드 new를 연결한다

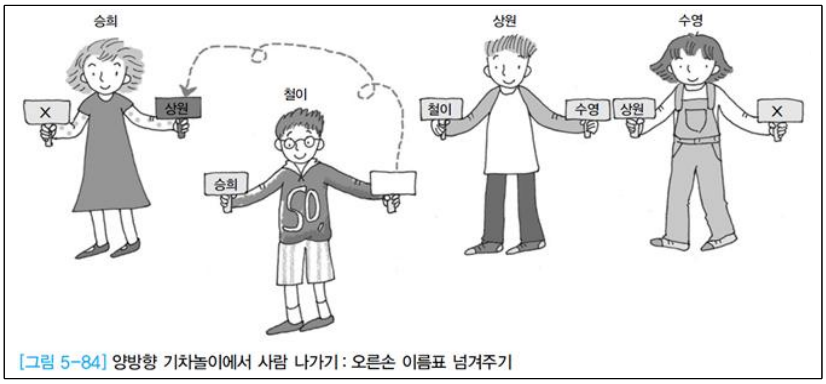


[그림 5-83] insertNode( ) 함수 수행 과정 : 알고리즘 설명 ❹ new.rlink.llink ← new;

학습내용3 : 이중 연결 리스트에서의 삭제 연산

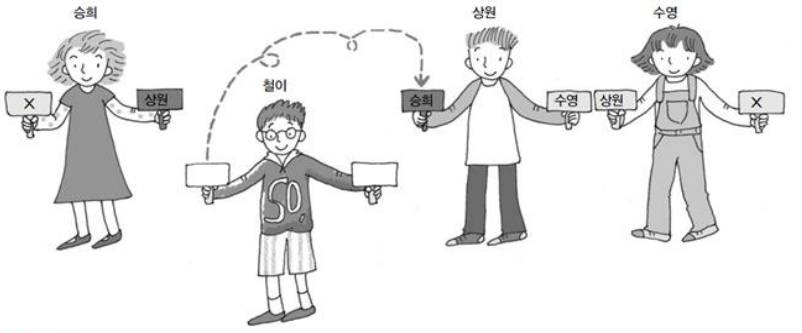
1. 양방향 기차 놀이에서 사람 나가기

① 철이의 오른손에 있는 사람의 이름표를 철이의 왼쪽에 있는 승희의 오른손에 넘겨준다



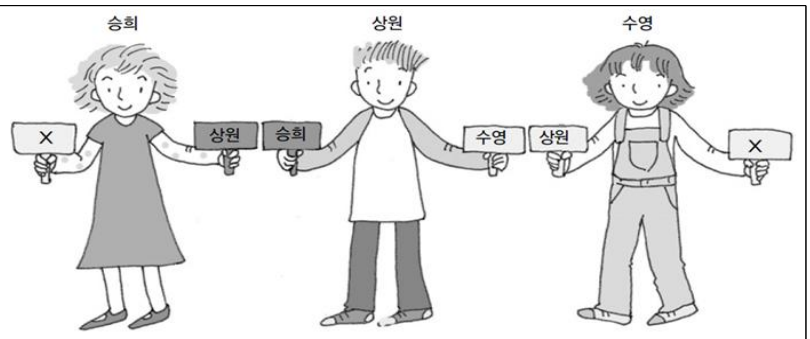
[그림 5-84] 양방향 기차놀이에서 사람 나가기 : 오른손 이름표 넘겨주기

② 철이의 왼손에 있는 이름표를 철이의 오른쪽에 있는 상원의 왼손에 넘겨준다



[그림 5-85] 양방향 기차놀이에서 사람 나가기: 왼손 이름표 넘겨주기

③ 이름표대로 연결하면 철이가 나가고 없는 양방향 기차가 된다



[그림 5-86] 양방향 기차놀이에서 사람 나가기 : 완성

2. 이중 연결 리스트에서의 삭제 연산

* 이중 연결 리스트에서의 원소 삭제 연산 과정

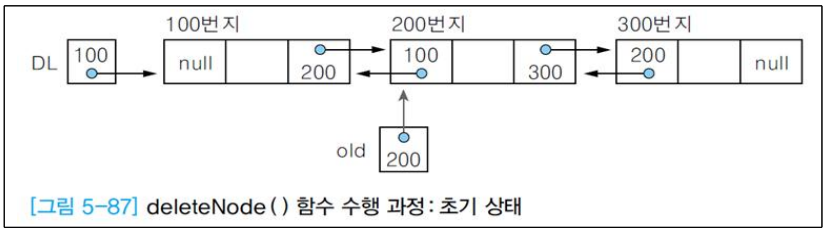
(1) 삭제할 노드의 오른쪽 노드의 주소를 삭제할 노드의 왼쪽 노드(old.llink)의 오른쪽 링크(rlink)에 저장한다
(2) 삭제할 노드의 왼쪽 노드의 주소(old.llink)를 삭제할 노드의 오른쪽 노드(old.rlink)의 왼쪽 링크(llink)에 저장한다
(3) 삭제한 노드를 자유 공간 리스트에 반환한다

* 이중 연결 리스트의 원소 삭제 알고리즘

알고리즘 5-12 이중 연결 리스트의 원소 삭제 알고리즘

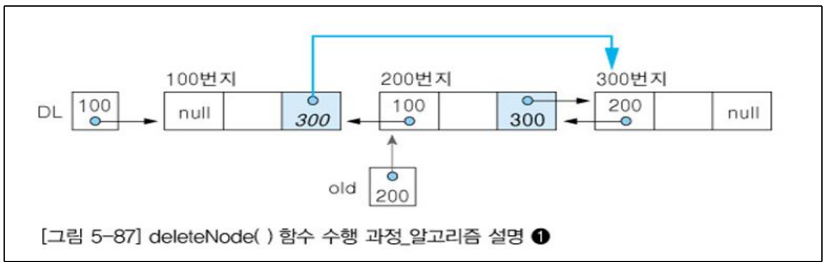
```
deleteNode(DL, old)
    old.llink.rlink ← old.rlink;        // ❶
    old.rlink.llink ← old.llink;        // ❷
    returnNode(old);                    // ❸
end deleteNode()
```

- 이중 연결 리스트 CL에서 포인터 old가 가리키는 노드를 삭제하는 알고리즘



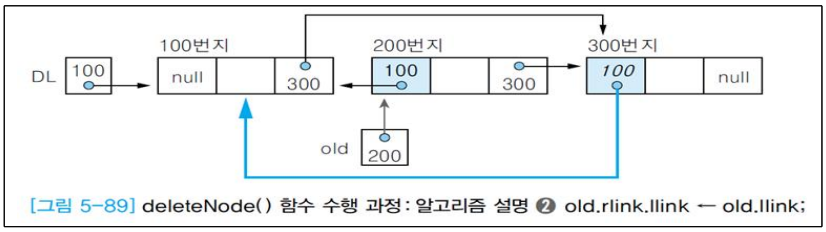
[그림 5-87] deleteNode( ) 함수 수행 과정 : 초기 상태

❶ old.llink.rlink ← old.rlink;
- 삭제할 노드 old의 오른쪽 노드의 주소(oldrlink)를 노드 old의 왼쪽 노드의 rlink에 저장하여 노드 old의 오른쪽 노드를 노드 old의 왼쪽 노드의 오른쪽 노드가 되도록 연결한다



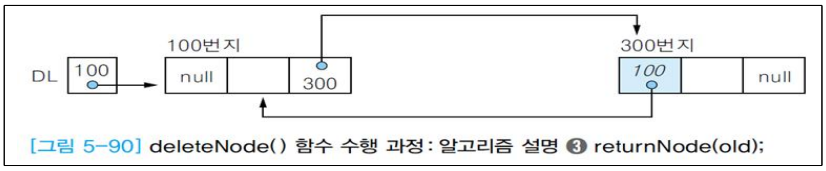
[그림 5-87] deleteNode( ) 함수 수행 과정_알고리즘 설명 ❶

❷ old.rlink.llink ← old.llink;
- 삭제할 노드 old의 왼쪽 주소(llink)를 노드 old의 오른쪽 노드의 llink에 저장하여 노드 old의 왼쪽 노드를 노드 old의 오른쪽 노드의 왼쪽 노드가 되도록 연결한다



[그림 5-89] deleteNode( ) 함수 수행 과정 : 알고리즘 설명 ❷ old.rlink.llink ← old.llink;

❸ rerurnNode(old);
- 삭제된 노드 old를 자유 공간 리스트로 반환한다



[그림 5-90] deleteNode( ) 함수 수행 과정 : 알고리즘 설명 ❸ returnNode(old);

【학습정리】

1. 리스트를 양쪽 방향으로 순회할 수 있도록 두 개의 링크 필드를 사용하여 양방향으로 노드를 연결한 리스트를 이중 연결 리스트(Doubly Linked List)라 한다.