SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

삼성 청년 SW 아카데미

APS 기본

SAMSUNG SW ACADEMY FOR YOUTH

APS(Algorithm Problem Solving)

링크드리스트 (Linked List)

- 연결리스트
- Singly Linked List
- Doubly Linked List

SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

연결리스트

연결 리스트

특성

- 자료의 논리적인 순서와 메모리 상의 물리적인 순서가 일치하지 않고, 개별적으로 위치하고 있는 원소의 주소를 연결하여 하나의 전체적인 자료구조를 이룬다.
- 링크를 통해 원소에 접근하므로, 순차 리스트에서처럼 물리적인 순서를 맞추기 위한 작업이 필요하지 않다.
- 자료구조의 크기를 동적으로 조정할 수 있어, 메모리의 효율적인 사용이 가능하다.

연결 리스트

노드

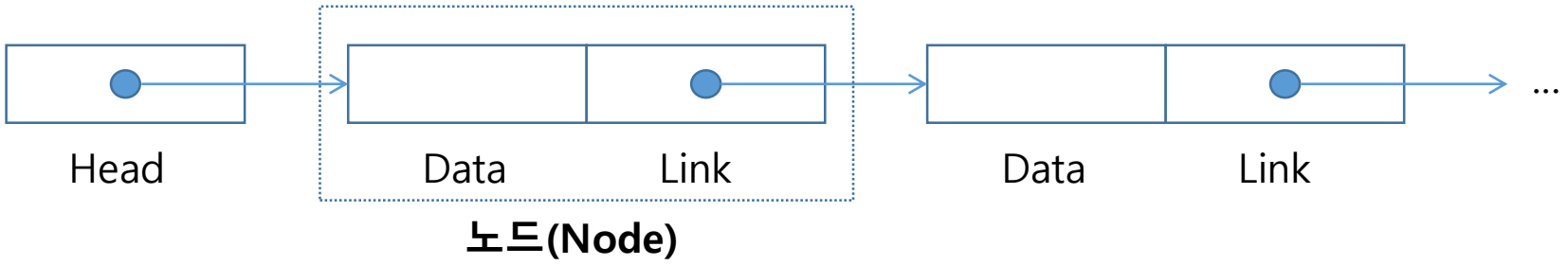
- 연결 리스트에서 하나의 원소에 필요한 데이터를 갖고 있는 자료단위
- 구성 요소

1) 데이터 필드
   - 원소의 값을 저장하는 자료구조
   - 저장할 원소의 종류나 크기에 따라 구조를 정의하여 사용함

2) 링크 필드
   - 다음 노드의 주소를 저장하는 자료구조

헤드

- 리스트의 처음 노드를 가리키는 레퍼런스



SAMSUNG SW ACADEMY FOR YOUTH

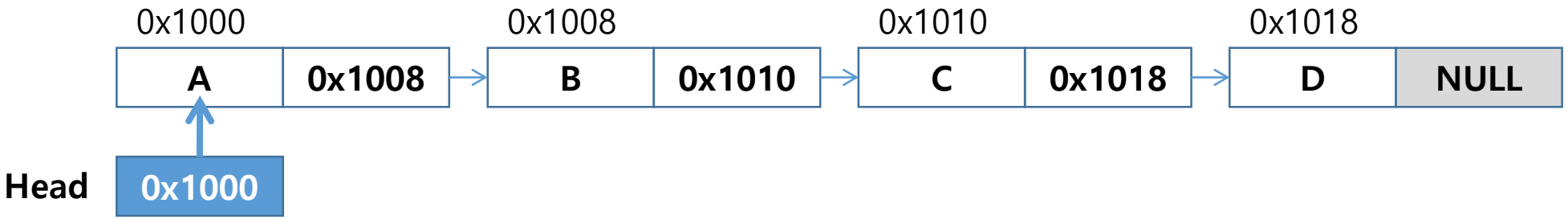
함께가요 미래로!
Enabling People

단순 연결 리스트
(Singly Linked List)

단순 연결 리스트(Singly Linked List)

연결 구조

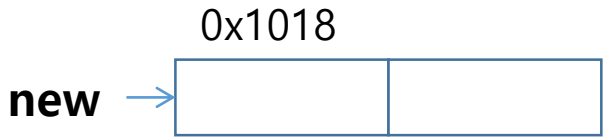
- 노드가 하나의 링크 필드에 의해 다음 노드와 연결되는 구조를 가진다.
- 헤드가 가장 앞의 노드를 가리키고, 링크 필드가 연속적으로 다음 노드를 가리킨다.
- 최종적으로 NULL을 가리키는 노드가 리스트의 가장 마지막 노드이다.



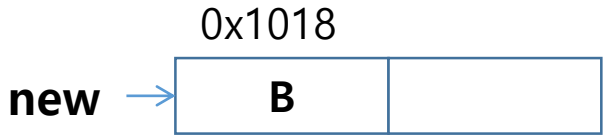
단순 연결 리스트(Singly Linked List)

- **'A', 'C', 'D'를 원소로 갖고 있는 리스트의 두 번째에 'B' 노드를 삽입할 때**

① 메모리를 할당하여 새로운 노드 new 생성



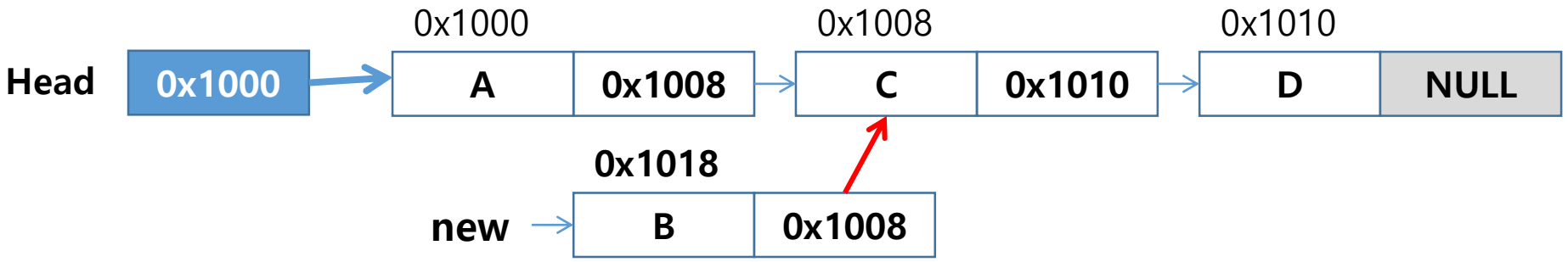
② 새로운 노드 new의 데이터 필드에 'B' 저장



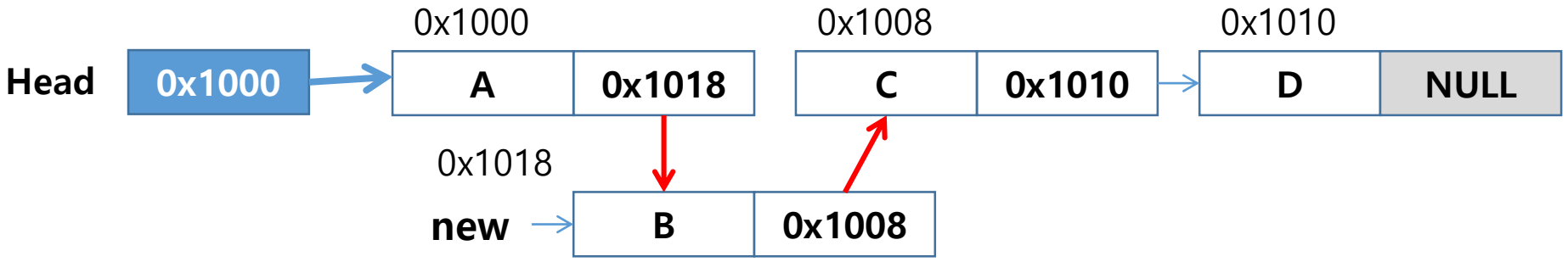
단순 연결 리스트(Singly Linked List)

- **'A', 'C', 'D'를 원소로 갖고 있는 리스트의 두번째에 'B' 노드를 삽입할 때**

③ 삽입될 위치의 바로 앞에 위치한 노드의 링크 필드를 new에 복사



④ new의 주소를 앞 노드의 링크 필드에 저장



단순 연결 리스트(Singly Linked List)

첫 번째 노드로 삽입하는 알고리즘

```
addtoFirst(L, i){          //리스트 포인터 L, 원소 i
    new ← createNode();    //새로운 노드 생성
    new.data = i;          //데이터 필드 작성
    new.link = L;          //링크 필드 작성
    L = new;               //리스트의 처음으로 지정
}
```

단순 연결 리스트(Singly Linked List)

가운데 노드로 삽입하는 알고리즘

- 노드 pre의 다음 위치에 노드 삽입

```
add(L, pre, i)              //리스트 L, 노드 pre, 원소 i
    new ← createNode();     //새로운 노드 생성
    new.data = i;           //데이터 필드 작성
    if(L == NULL) {
        L = new;
        new.link = NULL:
    } else {
        new.link = pre.link;
        pre.link = new;
    }
```

단순 연결 리스트(Singly Linked List)

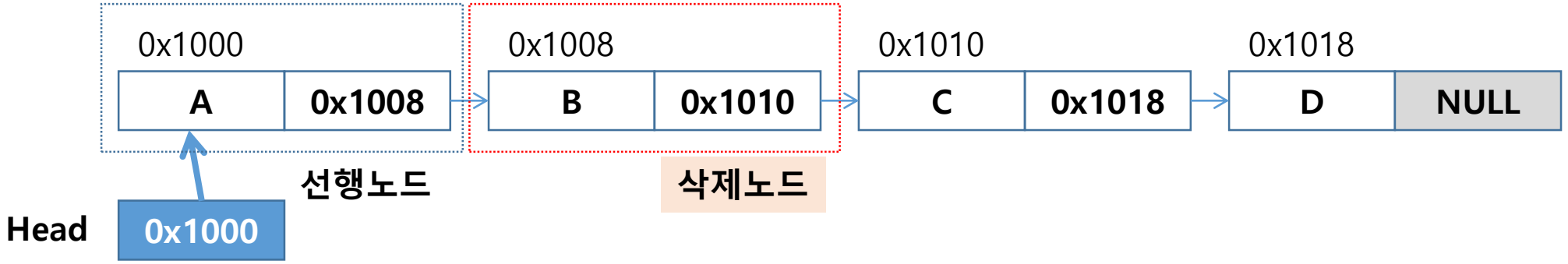
마지막 노드로 삽입하는 알고리즘

```
addtoLast(L, i)           // 리스트 L, 원소 i
    new ← createNode();  // 새로운 노드 생성
    new.data = i;
    new.link = NULL;
    if (L == NULL) {     //빈 리스트일 때, 최초 노드 추가
        L = new;
        return;
    }
    temp = L;                      // 노드 링크 이용하여 리스트 순회
    while (temp.link != NULL) {  // 마지막 노드 찾을 때까지 이동
        temp = temp.link;
    }
    temp.link = new;               // 마지막 노드 추가
```

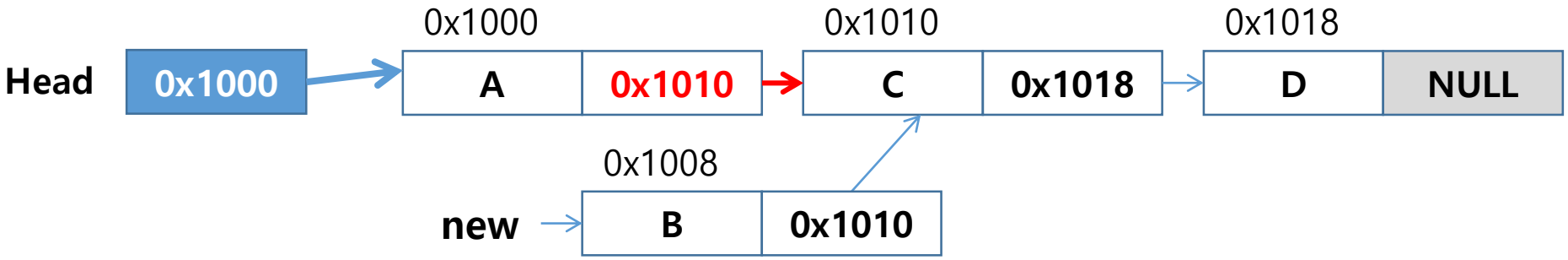
단순 연결 리스트(Singly Linked List)

- **'A', 'B', 'C', 'D'리스트의'B' 노드를 삭제할 때**

① 삭제할 노드의 앞 노드(선행노드) 탐색



② 삭제할 노드의 링크 필드를 선행노드의 링크 필드에 복사



단순 연결 리스트(Singly Linked List)

노드를 삭제하는 알고리즘

- 노드 pre의 다음 위치에 있는 노드 삭제

```
delete(L, pre) {                          // 리스트 L, 노드 pre
    if(L==NULL) error;
    else {
        target = pre.link;                //삭제 노드 지정
        if(target == NULL) return;
        pre.link = target.link;
    }
    freeNode(target)                      //할당된 메모리 반납
}
```

SAMSUNG SW ACADEMY FOR YOUTH

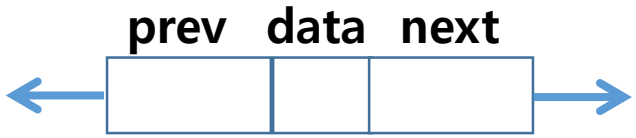
함께가요 미래로!
Enabling People

이중 연결 리스트
(Doubly Linked List)

이중 연결 리스트(Doubly Linked List)

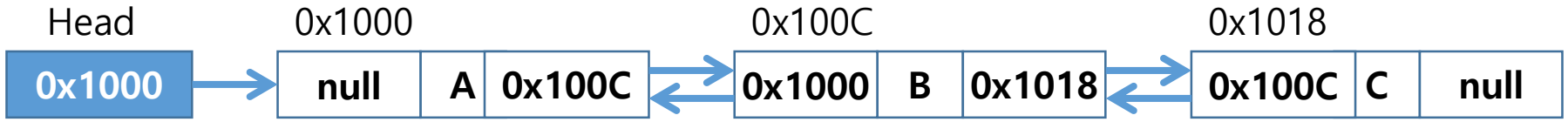
특성

- 양쪽 방향으로 순회할 수 있도록 노드를 연결한 리스트
- 두 개의 링크 필드와 한 개의 데이터 필드로 구성



<이중 연결 리스트의 기본 노드>

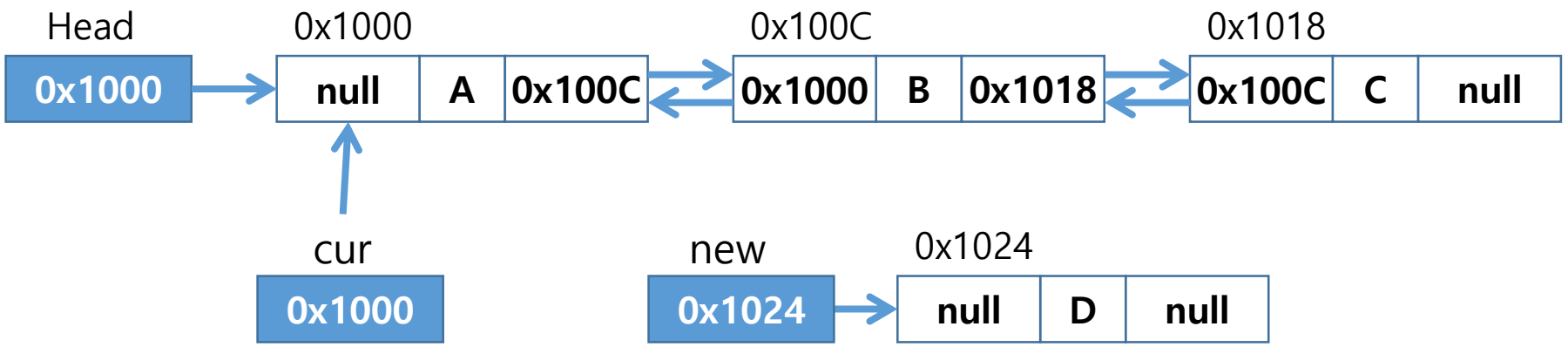
연결 구조



이중 연결 리스트(Doubly Linked List)

- **cur가 가리키는 노드 다음으로 D값을 가진 노드를 삽입하는 과정**

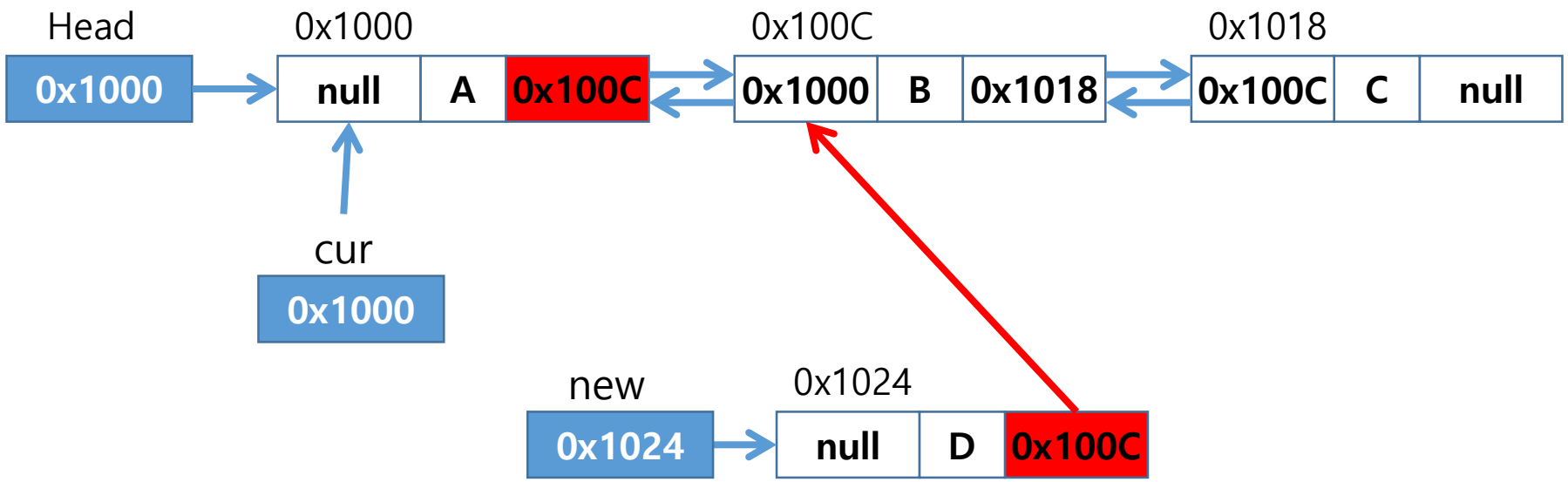
  ① 메모리를 할당하여 새로운 노드 new를 생성하고 데이터 필드에 'D'를 저장한다.



이중 연결 리스트(Doubly Linked List)

- **cur가 가리키는 노드 다음으로 D값을 가진 노드를 삽입하는 과정**

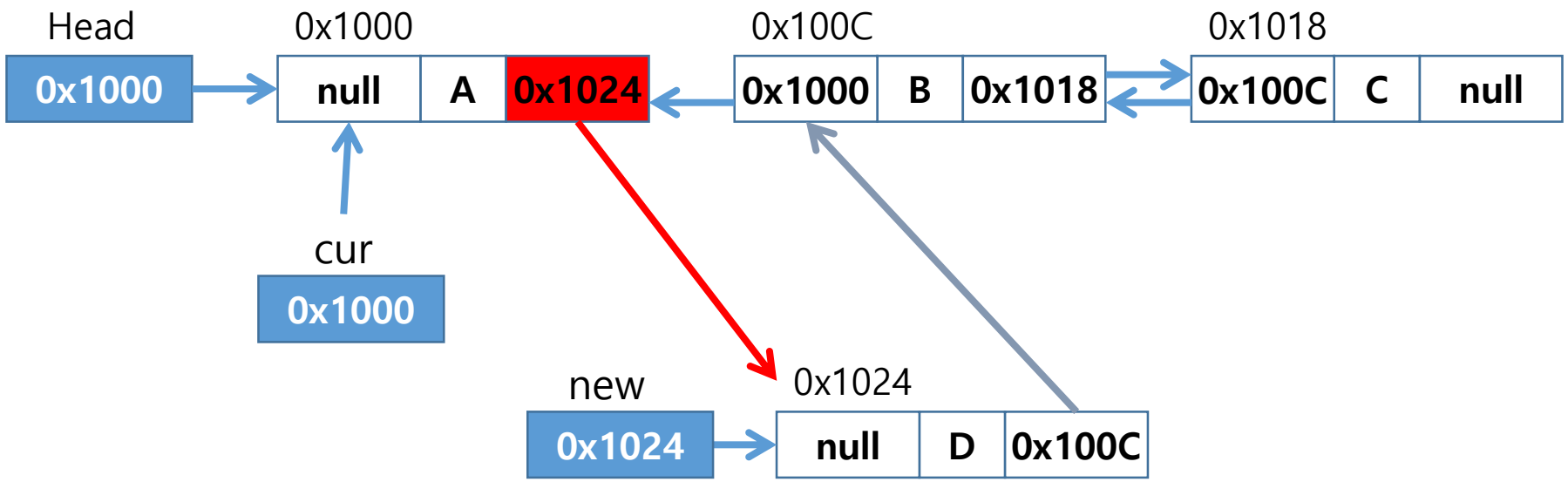
  ② cur의 next를 new의 next에 저장하여 cur의 오른쪽 노드를 삽입할 노드 new의 오른쪽 노드로 연결한다.



이중 연결 리스트(Doubly Linked List)

- **cur가 가리키는 노드 다음으로 D값을 가진 노드를 삽입하는 과정**

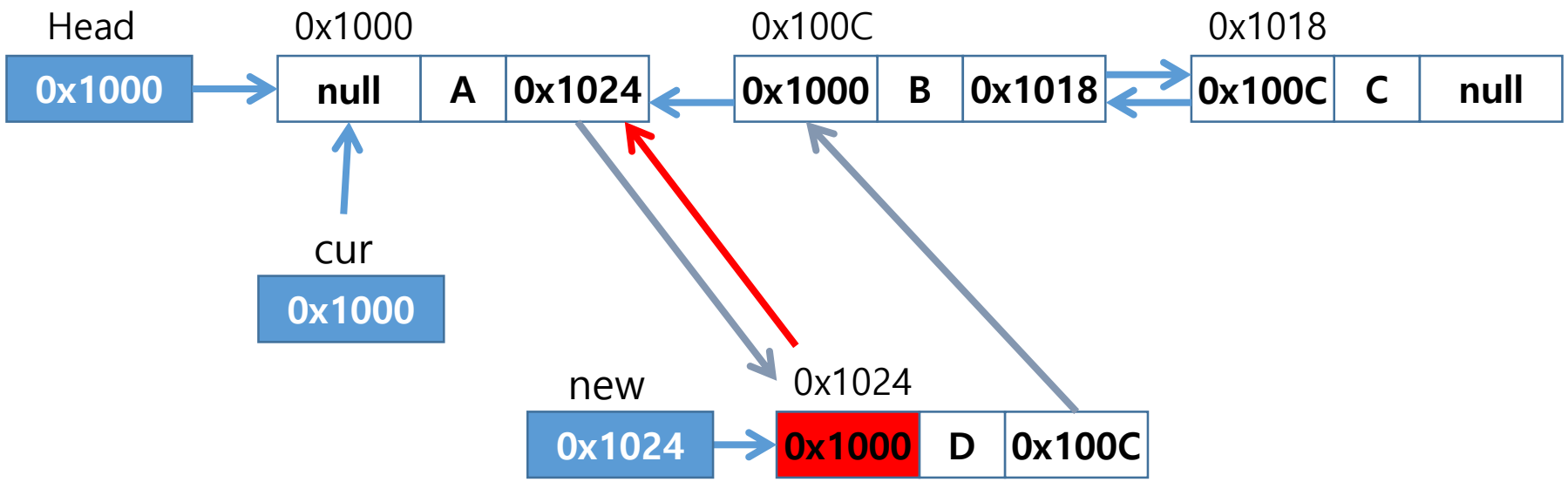
  ③ new의 주소를 cur의 next에 저장하여 노드 new를 cur의 오른쪽 노드로 연결한다.



이중 연결 리스트(Doubly Linked List)

- **cur가 가리키는 노드 다음으로 D값을 가진 노드를 삽입하는 과정**

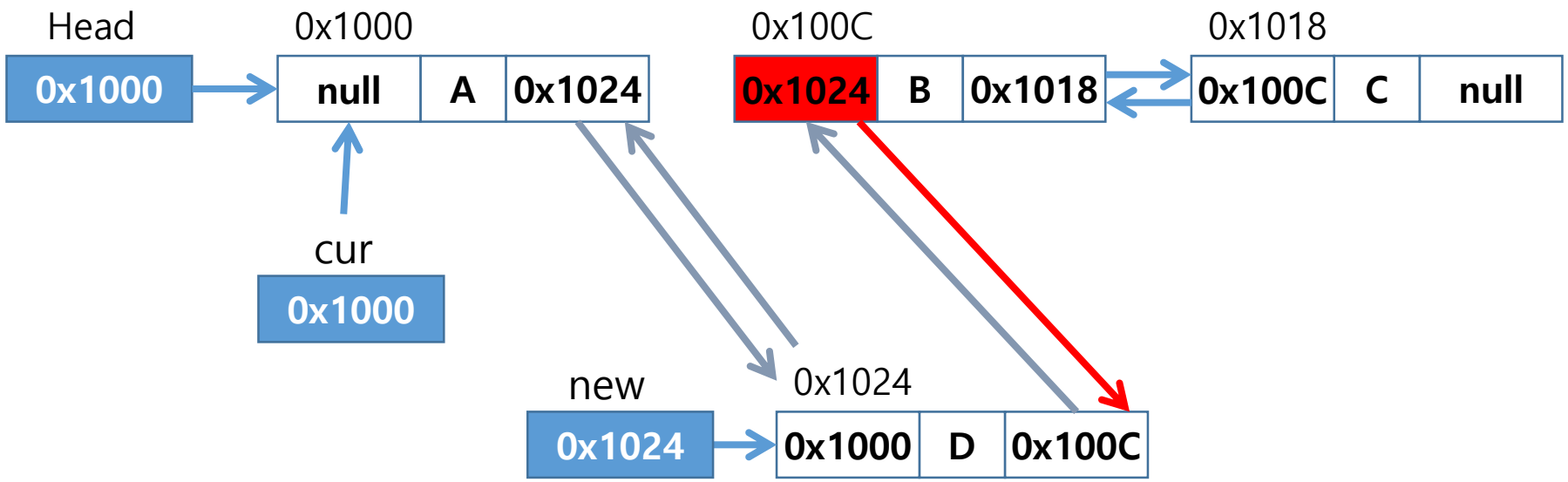
  ④ cur에 있는 링크 값을 new의 prev에 저장하여 cur를 new의 왼쪽노드로 연결한다.



이중 연결 리스트(Doubly Linked List)

- **cur가 가리키는 노드 다음으로 D값을 가진 노드를 삽입하는 과정**

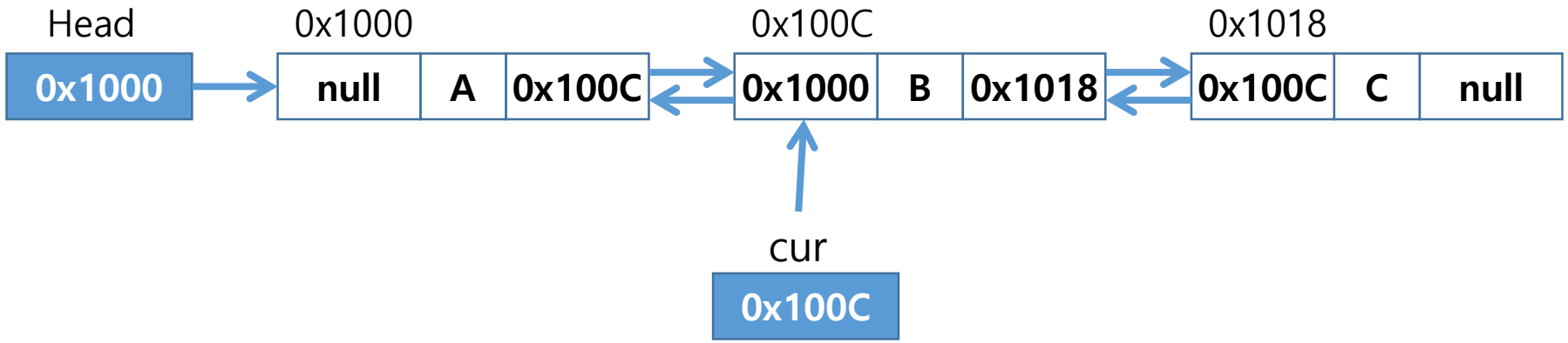
  ⑤ new의 주소를 new의 오른쪽노드의 prev에 저장하여 노드 new의 오른쪽노드의 왼쪽노드로 new를 연결한다.



이중 연결 리스트(Doubly Linked List)

- **cur가 가리키는 노드를 삭제하는 과정**

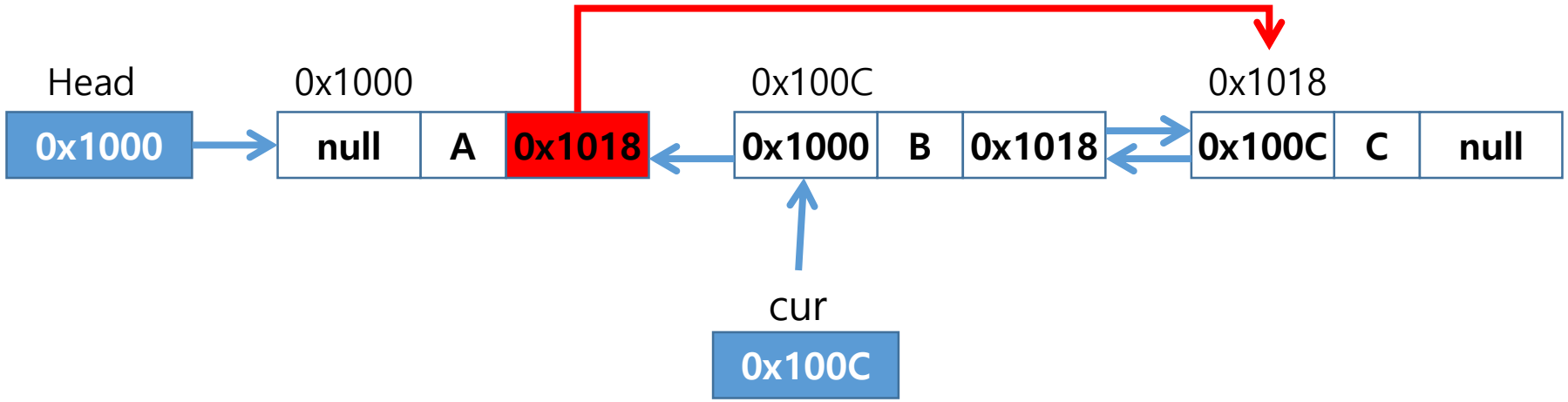
  ① 현재 상태



이중 연결 리스트(Doubly Linked List)

cur가 가리키는 노드를 삭제하는 과정

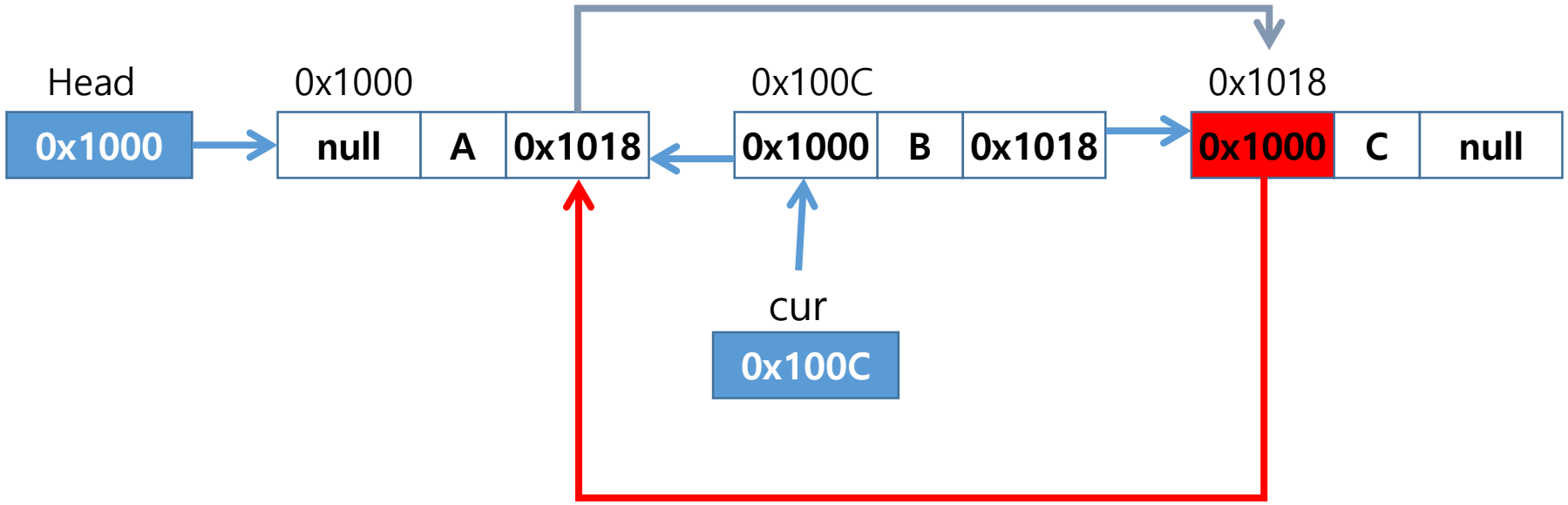
② 삭제할 노드 cur의 오른쪽노드의 주소를 cur의 왼쪽노드의 next에 저장하여 cur의 오른쪽노드를 cur의 왼쪽노드의 오른쪽노드로 연결한다.



이중 연결 리스트(Doubly Linked List)

cur가 가리키는 노드를 삭제하는 과정

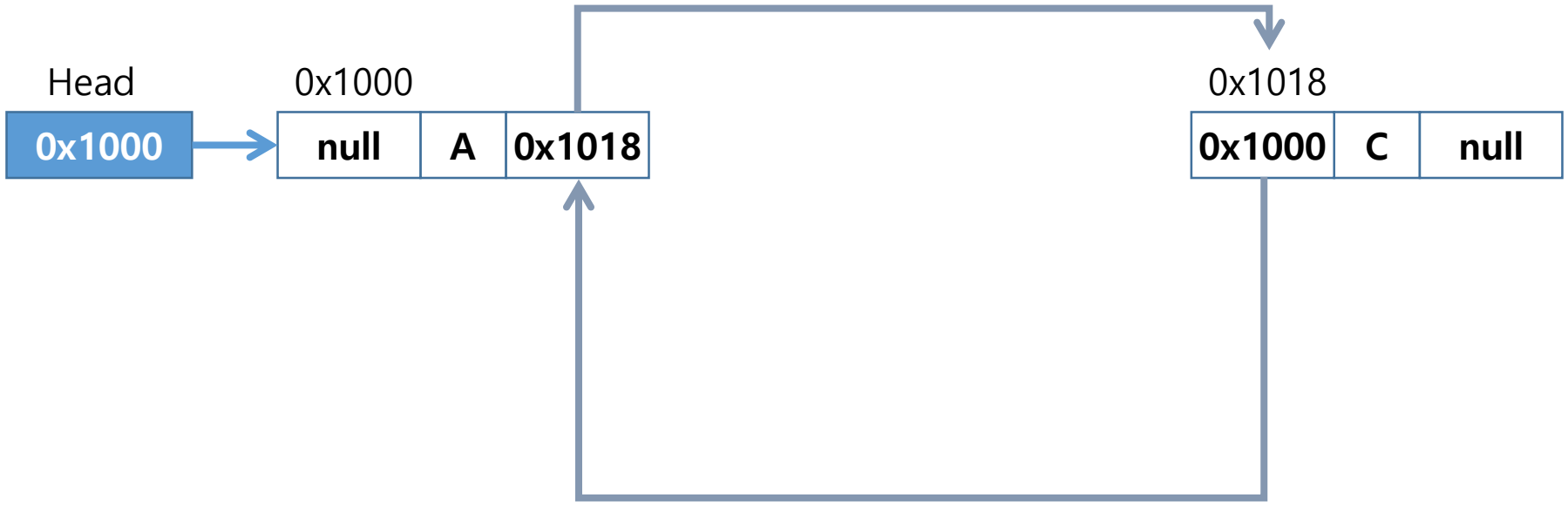
③ 삭제할 노드 cur의 왼쪽노드의 주소를 cur의 오른쪽노드의 prev에 저장하여 cur의 왼쪽노드를 cur의 오른쪽노드의 왼쪽노드로 연결한다.



이중 연결 리스트(Doubly Linked List)

cur가 가리키는 노드를 삭제하는 과정

④ cur가 가리키는 노드에 할당된 메모리를 반환한다.



SAMSUNG SW ACADEMY FOR YOUTH

함께가요 미래로!
Enabling People

다음 방송에서 만나요!

삼성 청년 SW 아카데미