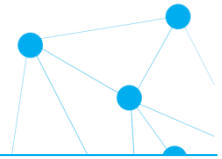
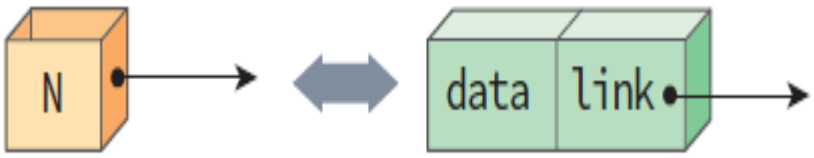
데이터구조 4장

04. 연결리스트

연결리스트

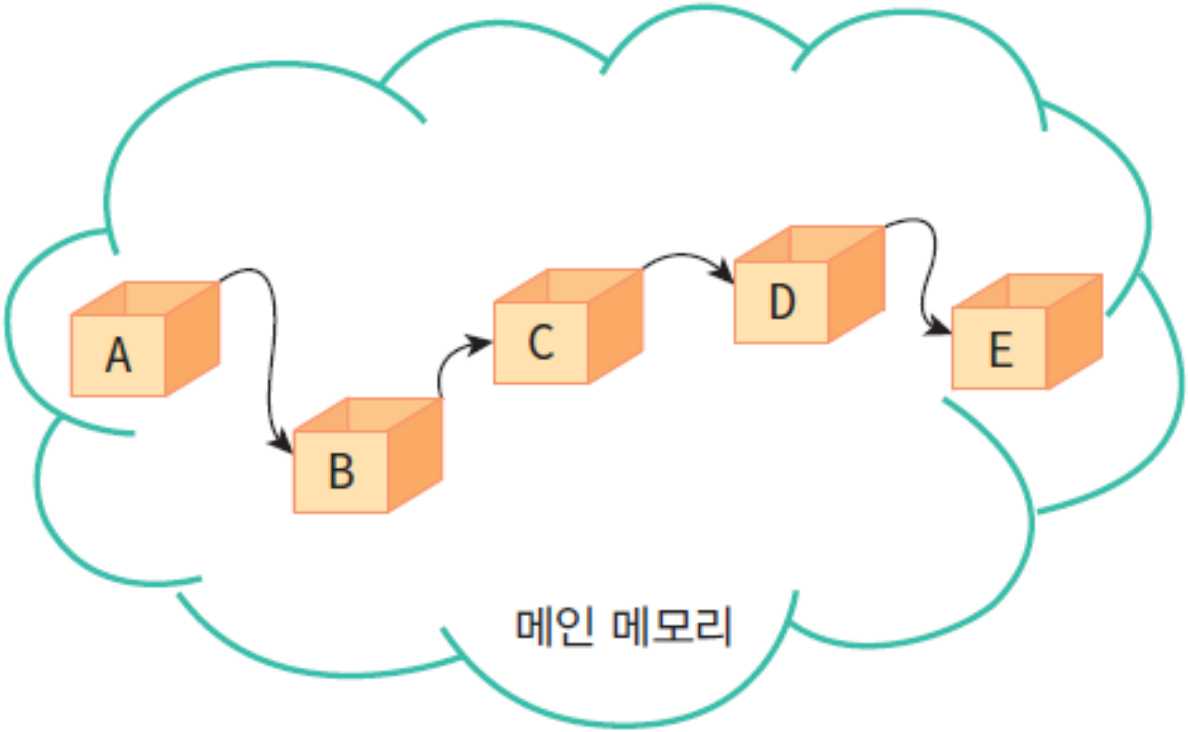


- 리스트의 항목들을 노드(node)라고 하는 곳에 분산하여 저장
- 노드 = 데이터 필드 + 링크 필드
  - 데이터 필드 : 리스트의 원소, 즉 데이타 값을 저장하는 곳
  - 링크 필드 : 다른 노드의 주소값을 저장하는 장소 (포인터)

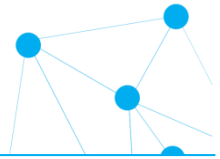


노드(node)

연결리스트(linked-list,링크드 리스트)



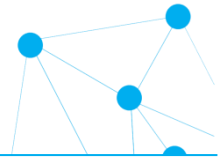
연결리스트



- 연결리스트 구현의 장단점(어떤 자료구조인지 상관없이 연결리스트로 구현시의 장단점이기도 함)

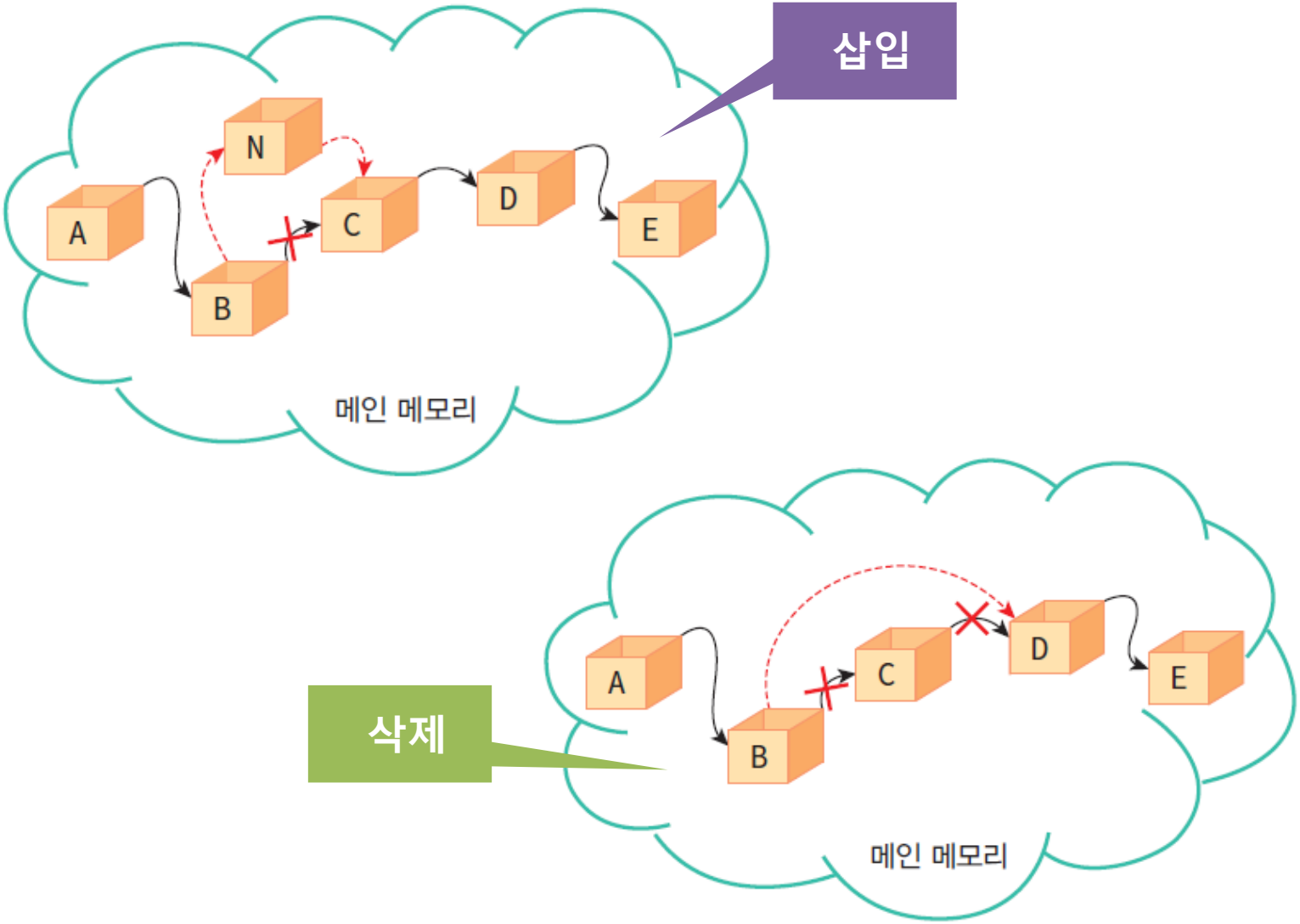
- 연결리스트 구현의 장점
  - 노드의 삽입, 삭제 작업이 용이하다.
  - 기억공간이 연속적으로 놓여 있지 않아도 저장이 가능하다.
  - 데이터를 저장할 공간이 필요할 때마다 동적으로 공간을 만들어 추가 가능
  - 트리를 표현하기에 적합하다.

연결리스트

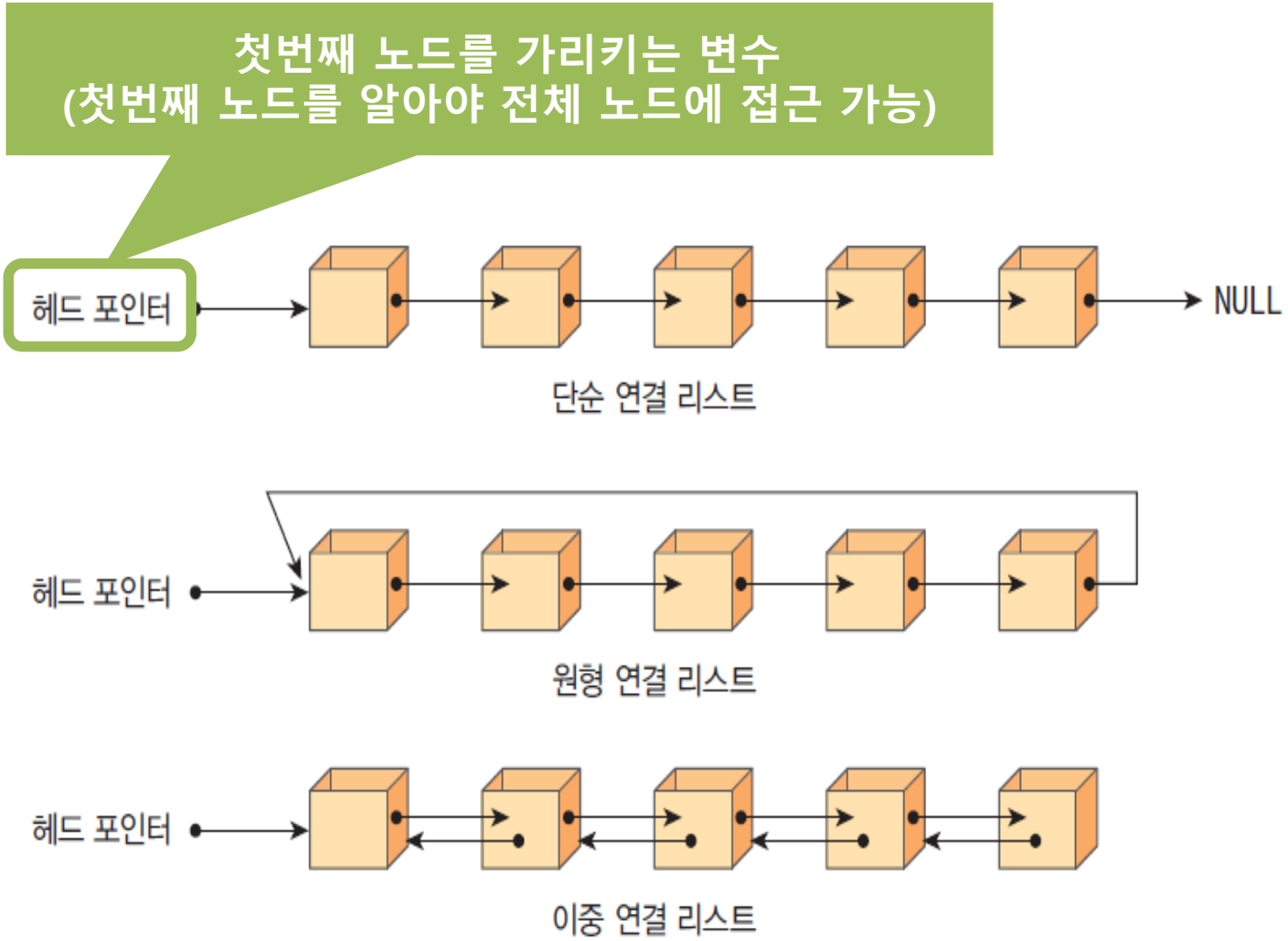


- 연결리스트 구현의 단점
  - 구현이 어렵고 오류 발생 확률이 높다.
  - 연결을 위한 링크 부분이 필요하기 때문에 기억공간의 이용 효율이 좋지않다.(메모리 공간을 많이 사용한다.)
  - 접근속도가 느리다.(순차적으로 앞에서부터 차례대로 접근해야 한다.)

연결리스트

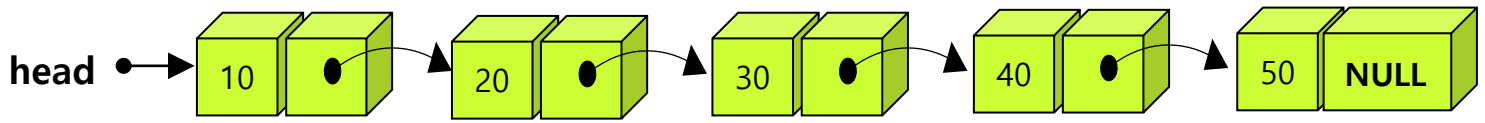


연결리스트의 종류


첫번째 노드를 가리키는 변수
(첫번째 노드를 알아야 전체 노드에 접근 가능)
헤드 포인터
NULL
단순 연결 리스트
헤드 포인터
원형 연결 리스트
헤드 포인터
이중 연결 리스트

단순 연결리스트를 이용한 리스트의 구현

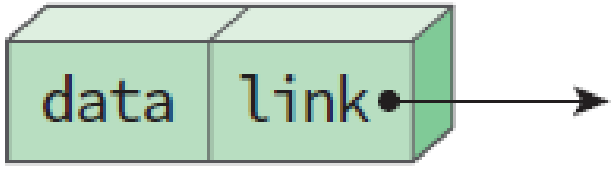
- 단순 연결리스트(simply linked list)의 사용
  - 하나의 방향으로만 연결되어 있는 연결리스트
  - 체인(chain)이라고도 함
  - 하나의 링크 필드를 이용하여 연결
  - 마지막 노드의 링크 값은 NULL



노드의 정의

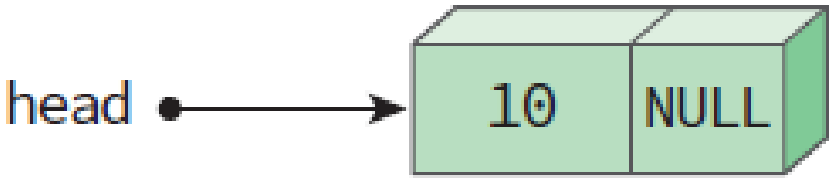
page 188 소스 참고

```
typedef int element;

typedef  struct  ListNode { // 노드 타입을 구조체로 정의한다.
        element data;
        struct  ListNode  *link;
} ListNode;
```

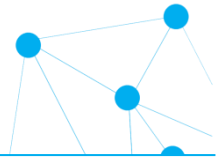


리스트의 생성(1번째 노드 생성)

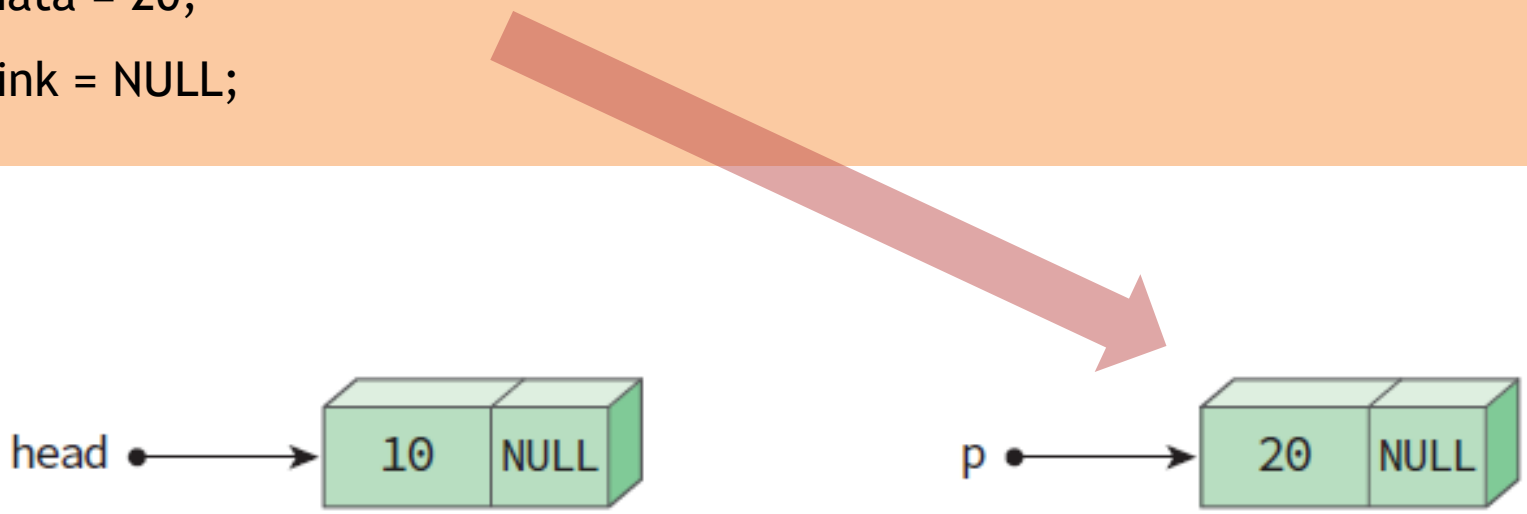
```
ListNode  *head  =  NULL; //초기

head = (ListNode *) malloc (sizeof(ListNode));

head->data = 10;
head->link = NULL;
```



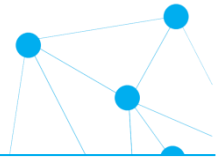
2번째 노드 생성



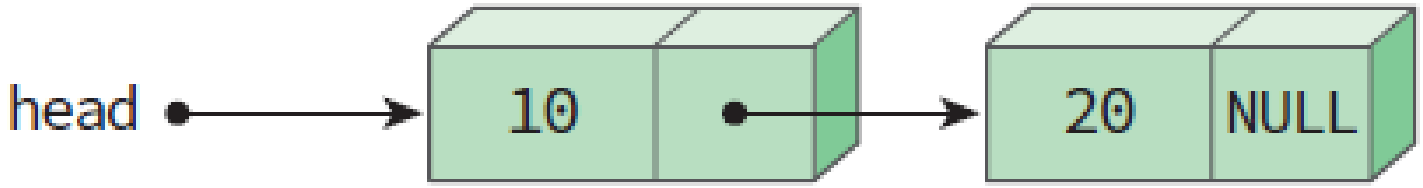
```
ListNode *p;
p = (ListNode *)malloc(sizeof(ListNode));
p->data = 20;
p->link = NULL;
```



노드의 연결(연결리스트)

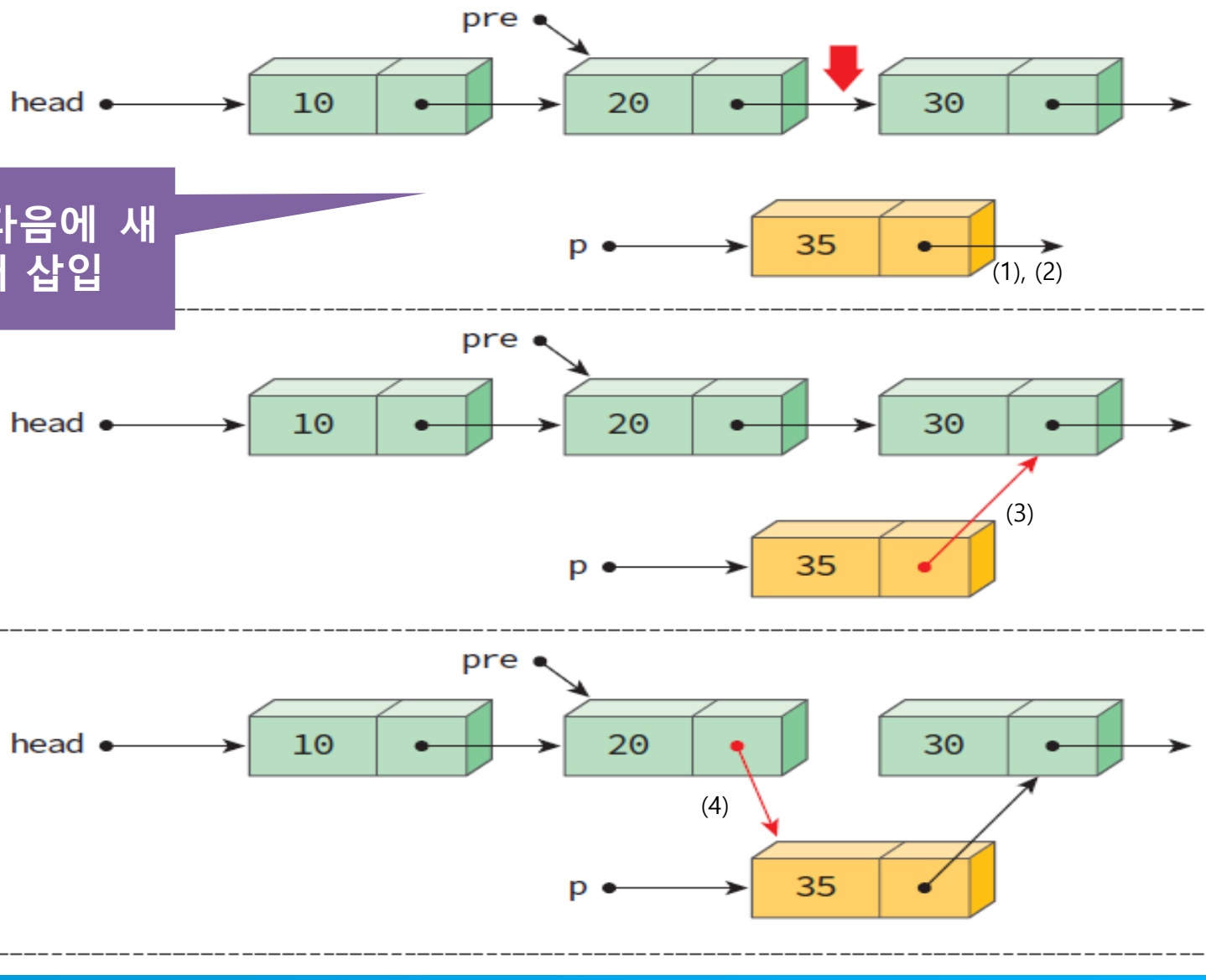


```
head->link = p;
```



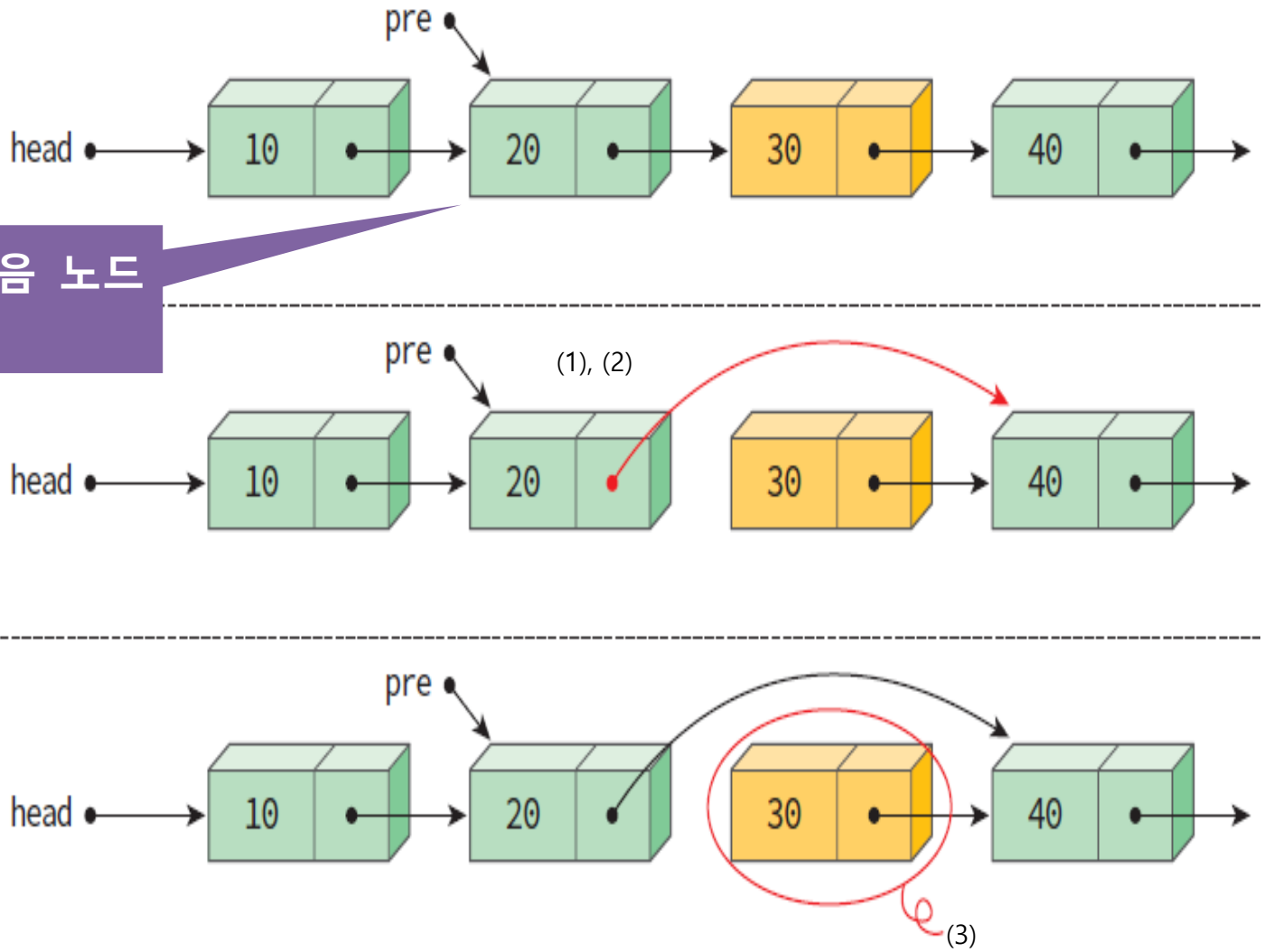
단순 연결리스트의 삽입연산

노드 pre 다음에 새로운 데이터 삽입



단순 연결리스트의 삭제연산

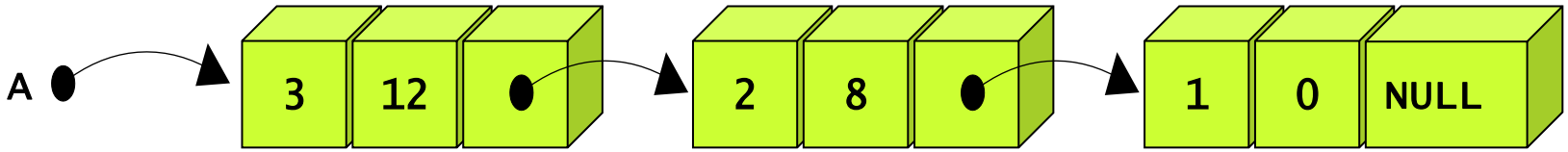
노드 pre 다음 노드 삭제

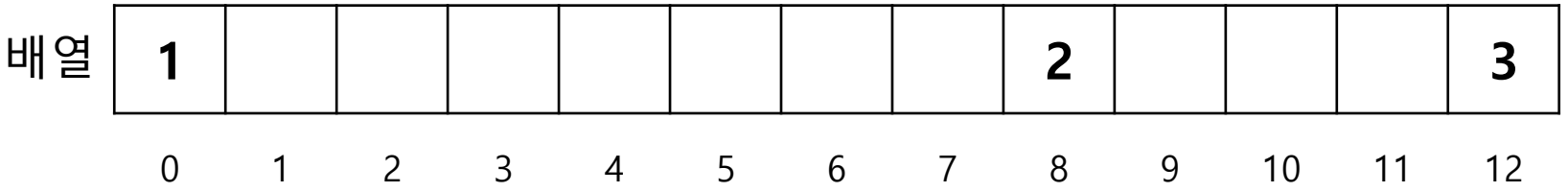


단순 연결리스트의 응용 : 다항식

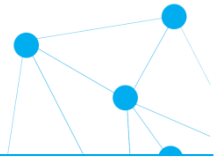
- 다항식을 컴퓨터로 처리하기 위한 자료구조
  - 다항식의 덧셈, 뺄셈…

- 하나의 다항식을 하나의 연결리스트로 표현
  - **$A = 3x^{12} + 2x^{8} + 1$**

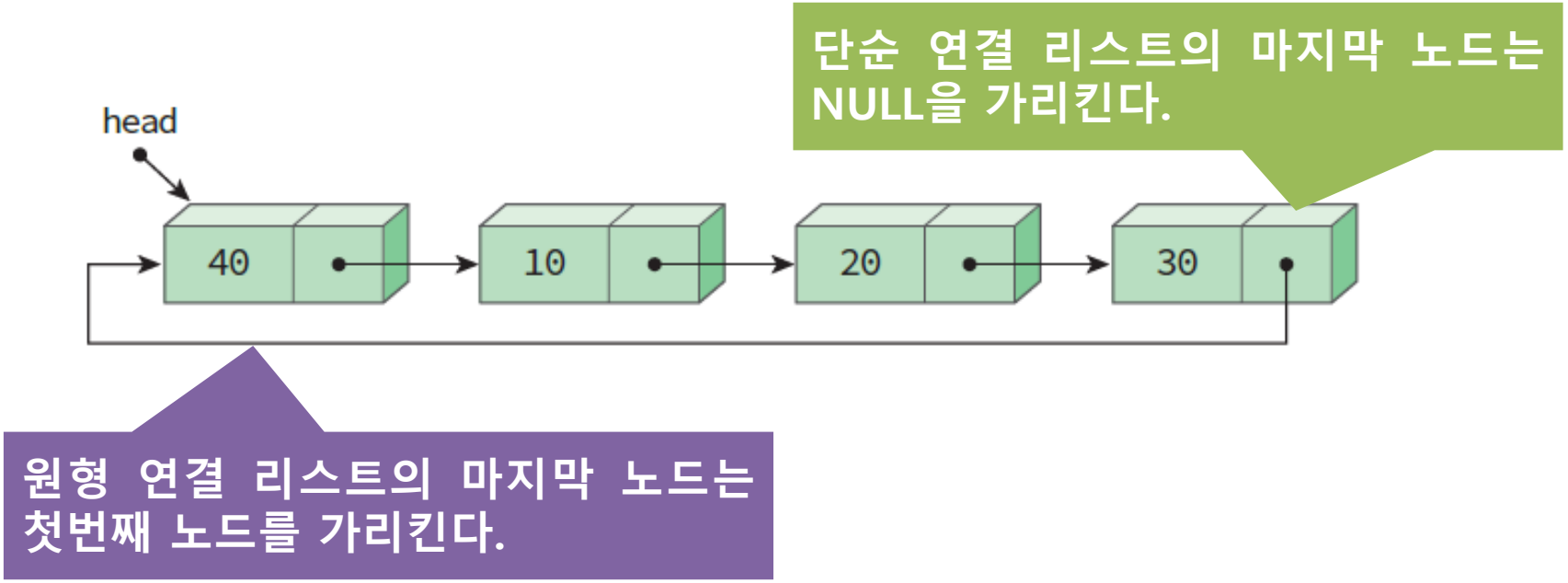




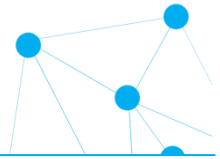
원형 연결리스트



- 원형 연결 리스트(Circular Linked List)의 사용
  - 마지막 노드의 링크가 첫 번째 노드를 가리키는 리스트
  - 장점
    - 한 노드에서 다른 모든 노드로의 접근이 가능
    - 노드의 삽입,삭제가 단순 연결리스트에 비해 용이



이중 연결리스트



- 이중 연결리스트(Double Linked List)의 사용
  - 하나의 노드가 선행 노드와 후속 노드에 대한 두 개의 링크를 가지는 리스트
  - 단순 연결리스트는 선행 노드를 알기 어려운 반면 이중 연결리스트는 링크가 양방향이라 양방향 검색이 가능
  - 공간을 많이 차지하고 코드가 복잡

데이터를 가지지 않고 단지 삽입, 삭제 코드를 간단하게 할 목적으로 만들어진 노드

원형 이중 연결 리스트