자료 구조
(리스트)

https://youtu.be/wXSr4PWM4T0

송경주

목차

리스트

- 리스트란?

하나 이상의 데이터가 순서대로 나열된 형태

-공간적 순서

-데이터 값에 의한 순서

- 리스트 구현

1. 배열
2. 포인터

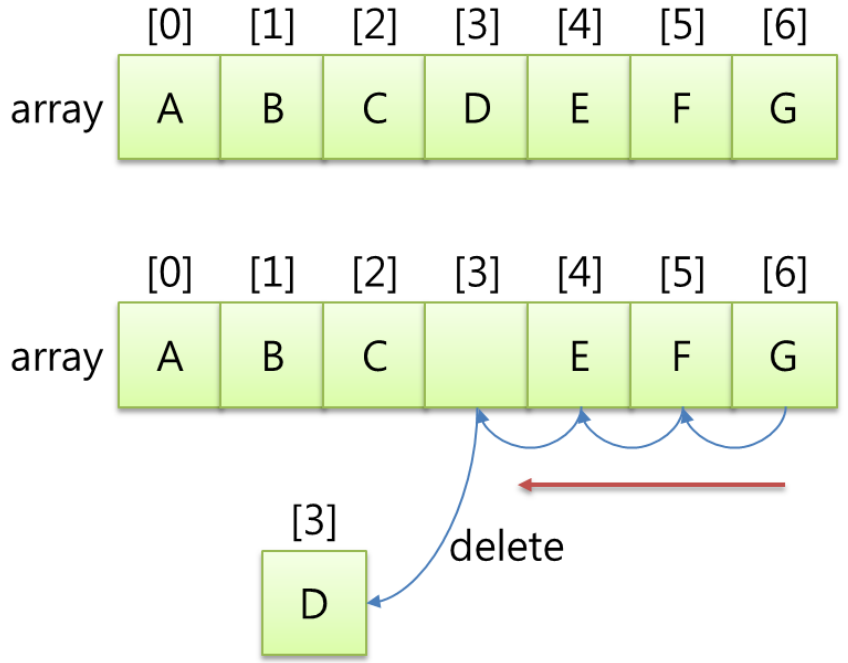
배열로 구성된 리스트

-자료 구조 중 가장 간단함.

-인덱스 번호를 통해 접근.

-크기가 고정됨.

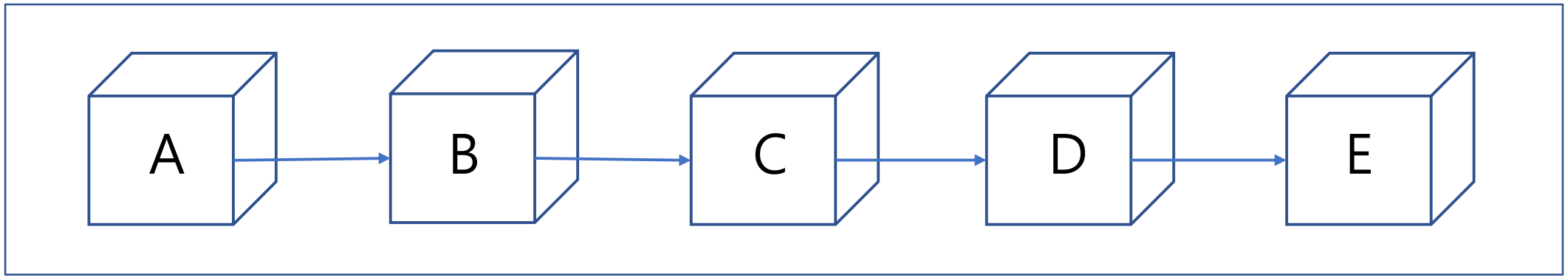
-삽입과 삭제 시에 상당한 오버헤드. → **"연결 리스트"**



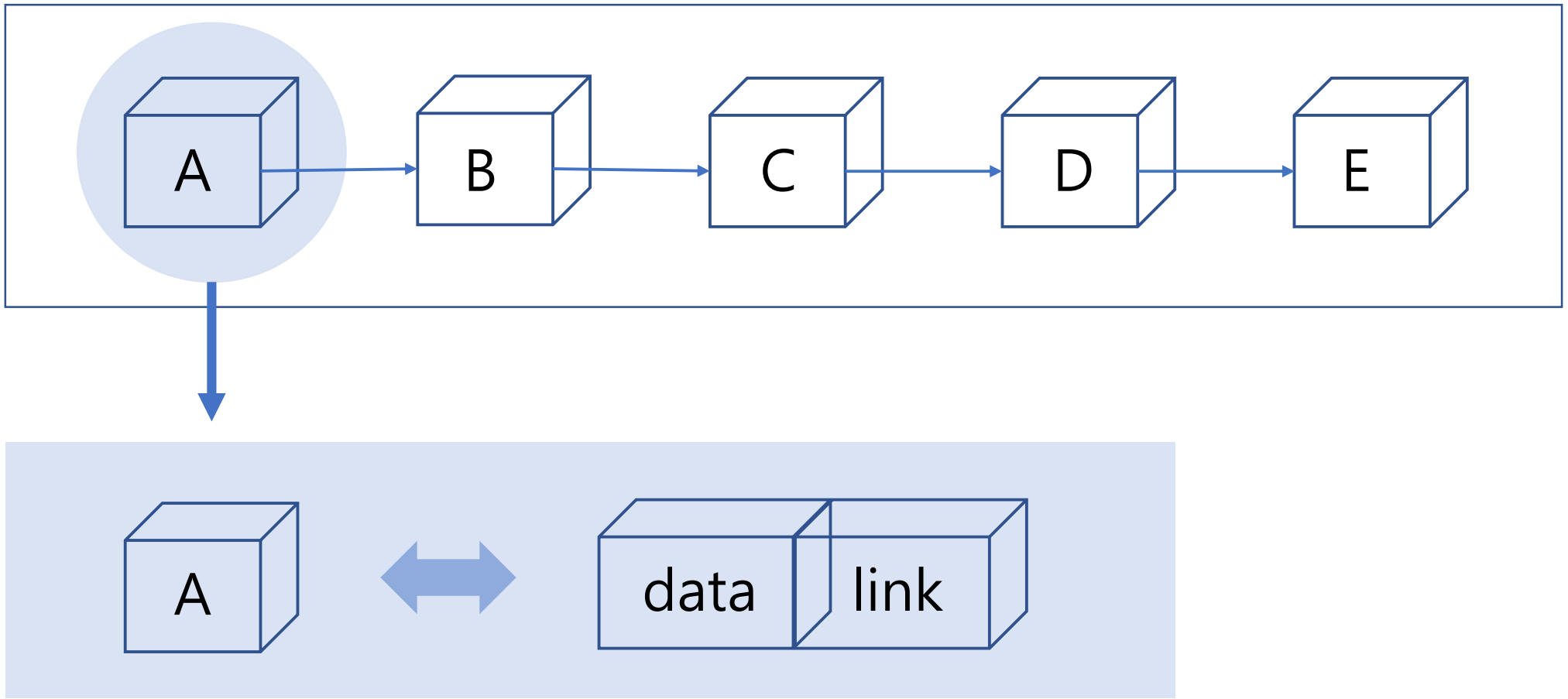
연결 리스트란?

-물리적으로 흩어져 있는 자료들을 서로 연결하여 하나로 묶는 방법.
-헤드 포인터 를 통해 첫번째 노드를 가리킴.

1. 단순 연결 리스트
2. 원형 연결 리스트
3. 이중 연결 리스트



연결 리스트란?

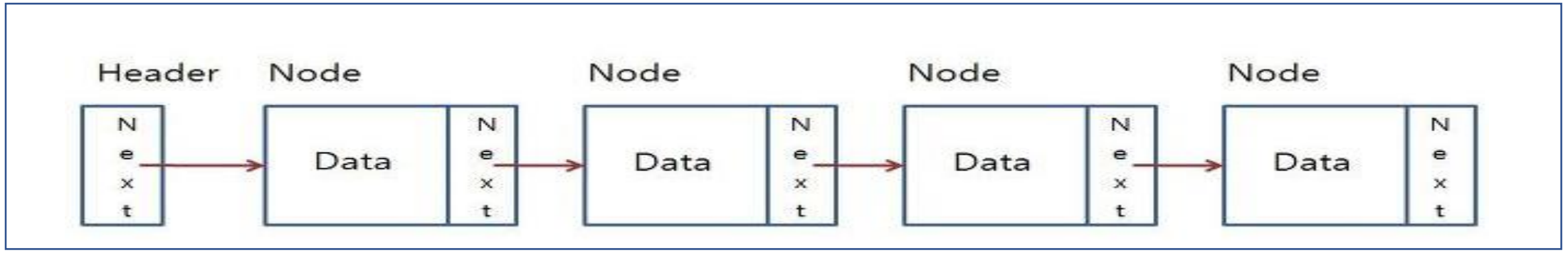


1. 단순 연결 리스트

-하나의 방향으로만 연결, 각 노드들은 하나의 링크 필드를 가짐.
(마지막 노드의 링크 필드 값은 NULL)

[장점]
① 구조가 간단함.

[단점]
① 노드의 데이터를 찾거나 추가, 제거 하기 위해서는 처음 head 노드에서 부터 탐색 해야함.



1. 단순 연결 리스트 - 삽입연산

```
void addItNode(headNode* H, listNode* prevNode, int x) {
    listNode* newNode;
    newNode = (listNode*)malloc(sizeof(listNode));
    newNode -> data = x;
    newNode -> link = NULL;

    newNode -> link = prevNode -> link;
    prevNode -> link = newNode;
    return;
}
```

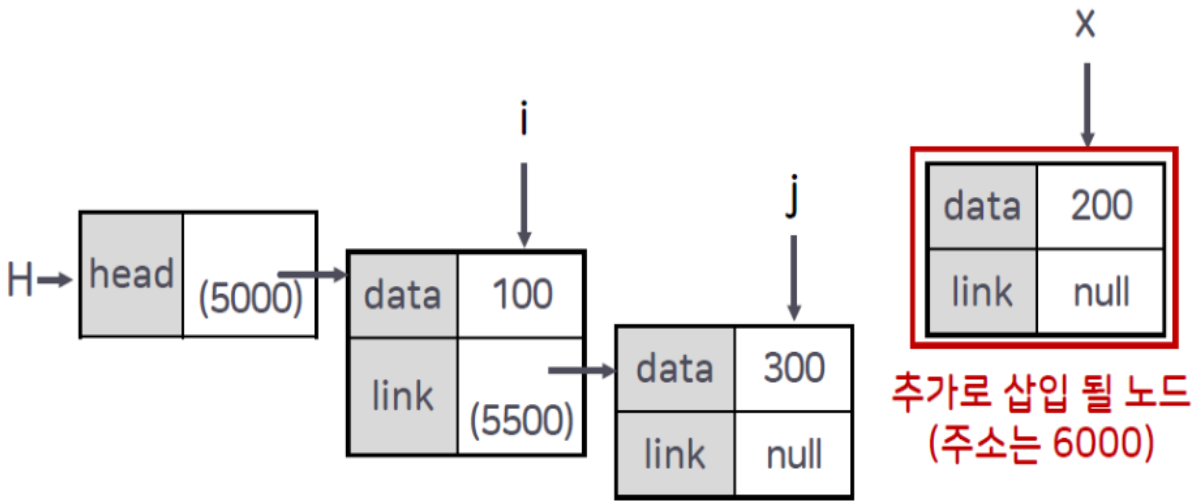
동적할당을 통해 새로운 노드 생성

새로운 노드의 링크에 삽입 위치 앞 노드의 링크를 복사함.

앞 노드의 링크가 새로운 노드를 가리킴

1. 단순 연결 리스트 - 삽입연산

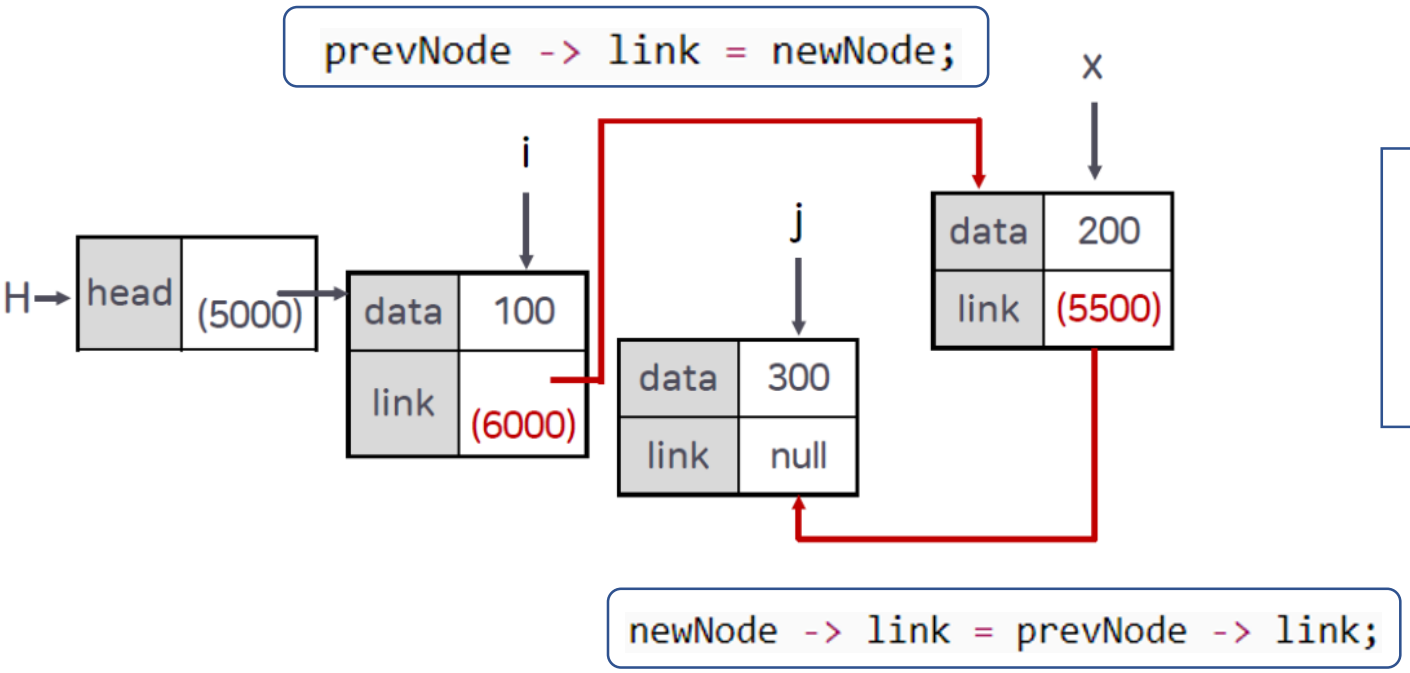
[new node x를 i와 j 노드 사이에 삽입하는 경우]



노드를 삽입하고자 하는 위치를 지정한 후,
i노드의 link를 새로운 노드의 link로 복제하고
i노드의 link가 새로운 노드를 가리킨다.

1. 단순 연결 리스트 - 삽입연산

[new node x를 i와 j 노드 사이에 삽입하는 경우]



i 노드의 link 가 x를 가리키고
X 노드의 link는 j 노드를 가리킨다

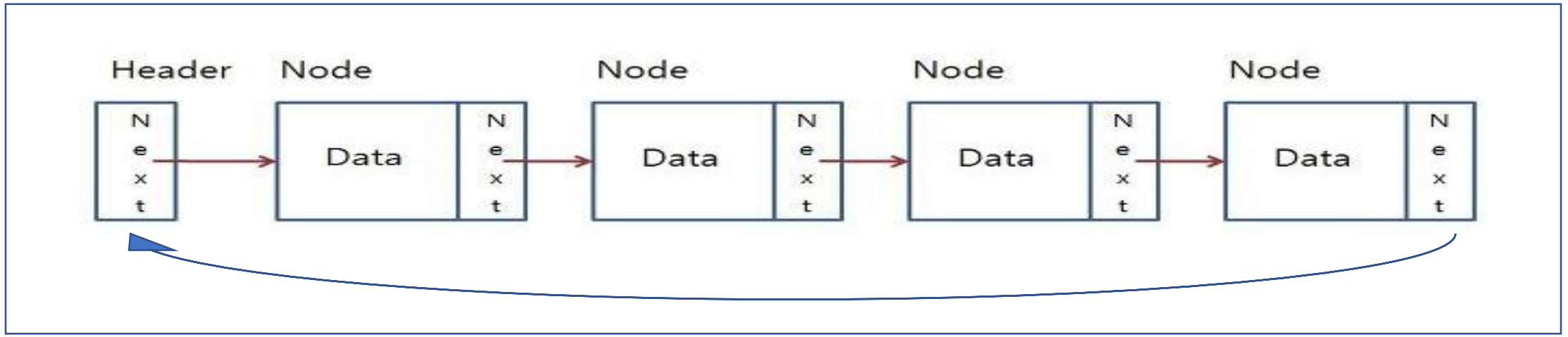
2. 원형 연결 리스트

-리스트의 마지막 노드의 링크가 첫 번째 노드를 가리키는 리스트.

(즉, 마지막 노드의 링크 필드가 NULL이 아닌 첫 번째 노드 주소)

[장점]

① 특정 한 노드에서 모든 노드에 접근 가능



2. 원형 연결 리스트 - 삽입연산

[원형리스트 가장 끝에 노드 삽입]

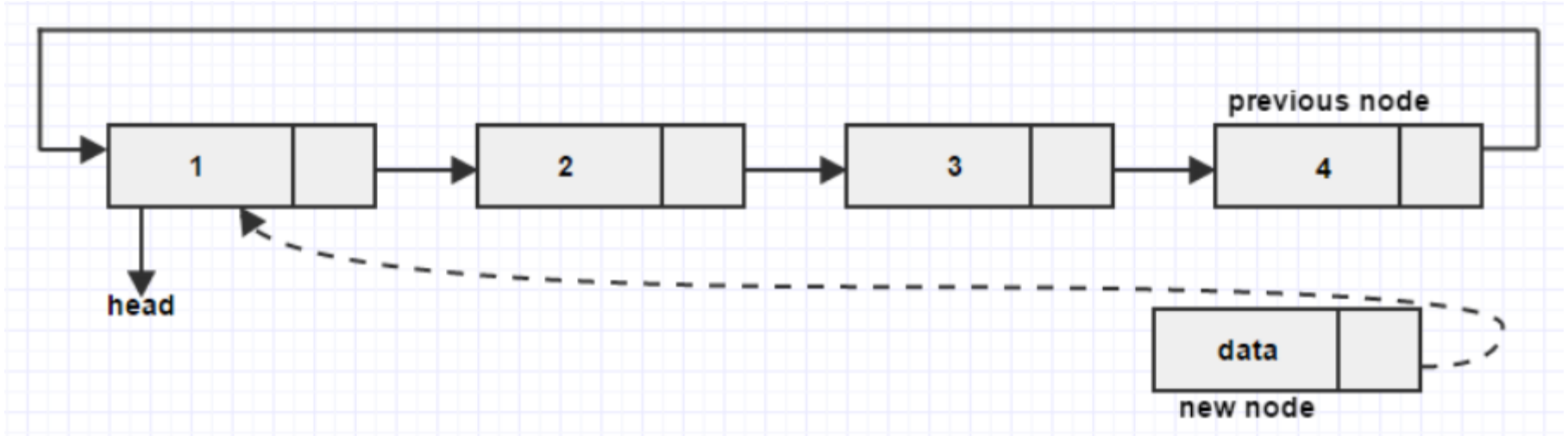
```
struct CLIstNode current = *head;
struct CListNode *newNode = (struct node*)(malloc(sizeof(struct CListNode)));
if (!newNode)
{
        printf("Memory Error");
        return;
}
newNode->data = data;
while (current->next != *head)
        current = current->next;
newNode->next = newNode;
if (*head == NULL)
        *head = newNode;
else
{
        newNode->next = *head;
        current->next = newNode;
}
```

다음 노드가 head 인 노드, 즉 마지막 노드를 찾아 새로운 노드 삽입 위치를 고름.

Head가 없을 경우, 리스트가 비어 있으므로 새로운 노드가 head가 됨.

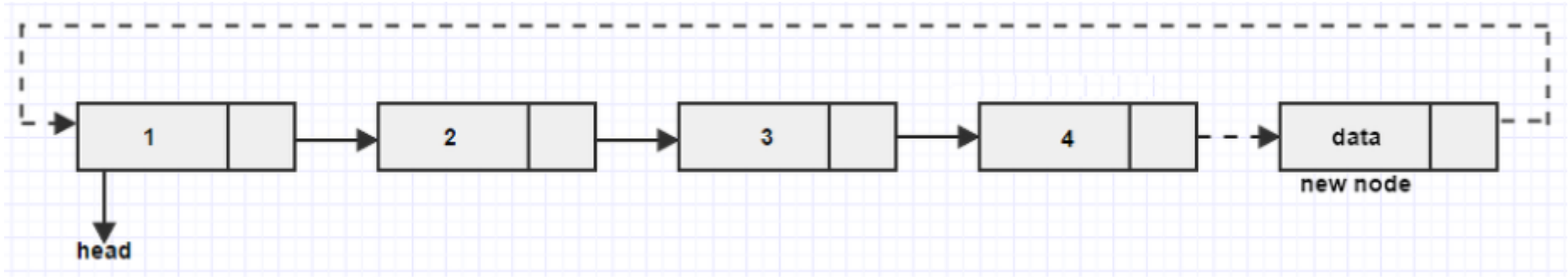
Head가 있을 경우, 새로운 노드의 next 링크가 head를 가리키도록 함.

2. 원형 연결 리스트 - 삽입연산



새로 삽입 할 노드 link가 head를 가리키도록 하고 head
이전의 노드가 새로운 노드를 가리키도록 한다.

2. 원형 연결 리스트 - 삽입연산



원형 연결 리스트는 원형으로 연결되어 있기 때문에
head 위치만 바꿔주면 새로운 노드가 마지막 노드가 된다.

3. 이중 연결 리스트

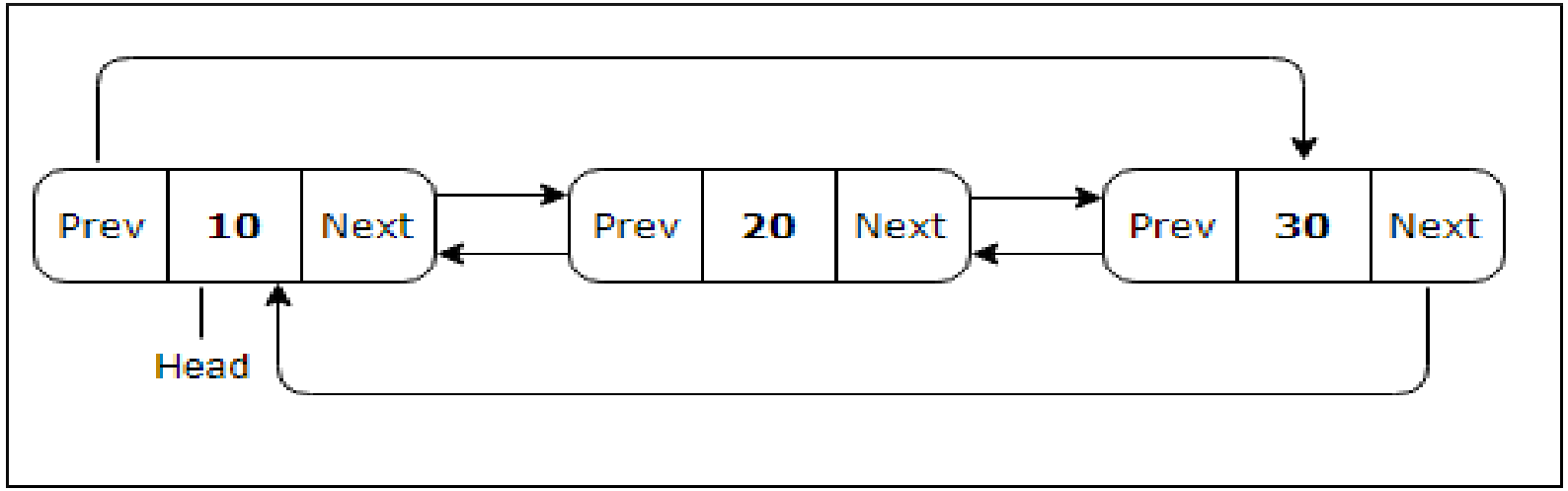
-하나의 노드가 선행 노드와 후속 노드에 대한 두 개의 링크를 가지는 리스트. (헤드 노드를 추가해서 많이 사용함.)

[장점]

① 양방향으로 자유롭게 움직일 수 있다.

[단점]

① 공간을 많이 차지하고 코드가 복잡해짐.



3. 이중 연결 리스트 - 삽입연산

[node를 중간에 삽입]

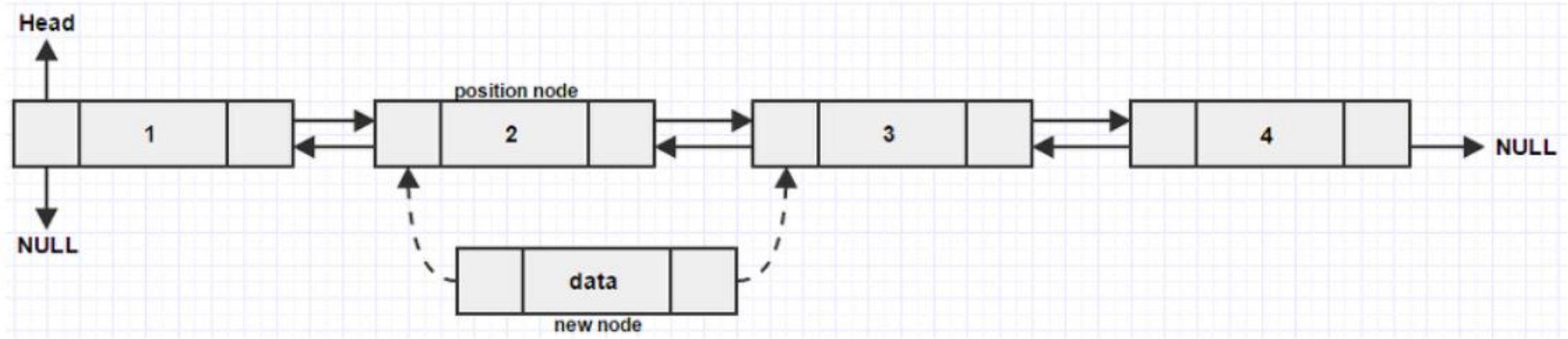
```
else // 리스트 중간에 삽입
{
        newNode->next = temp->next;
        newNode->prev = temp;
        temp->next->prev = newNode;
        temp->next = newNode;
}
return;
```

새로운 노드의 next 포인터에 앞 노드의 next 포인터를 복사하고
새로운 노드의 previous 포인터가 앞 노드를 가리키도록 한다.

앞 노드의 next 링크가 가리키고 있던 노드 즉, 뒤 노드의
previous 포인터가 새로운 노드를 가리키도록 한다.
앞 노드의 next 링크는 새로운 노드를 가리키도록 한다.

3. 이중 연결 리스트 - 삽입연산

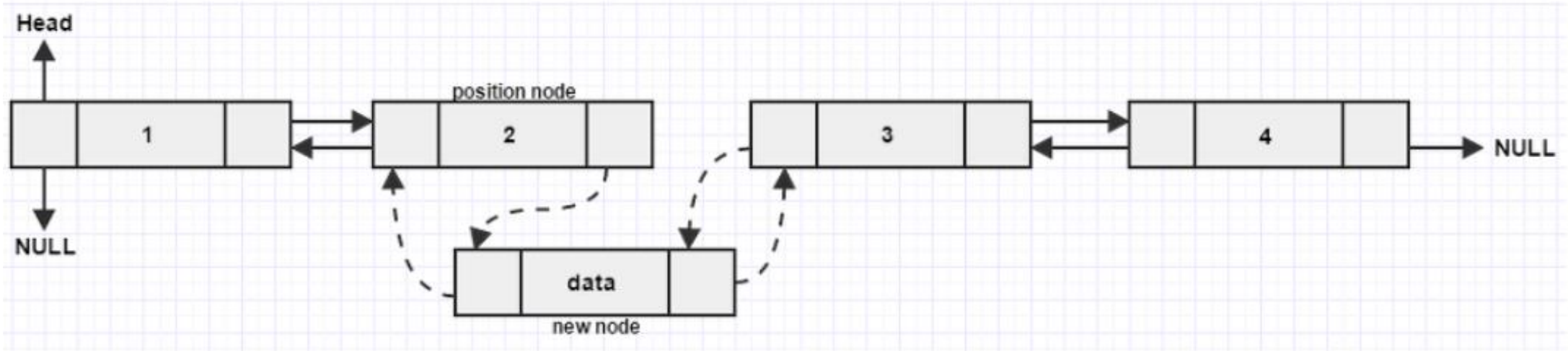
[node를 중간에 삽입]



새로운 노드의 previous 포인터에 뒤 노드의 previous 를 복사하고 새로운 노드의 next 포인터에 앞 노드의 next 포인터를 복사한다.

3. 이중 연결 리스트 - 삽입연산

[node를 중간에 삽입]



뒤 노드의 previous 가 새로운 노드를 가리키고
앞 노드의 next가 새로운 노드를 가리키도록 한다.

감사합니다 :-)