

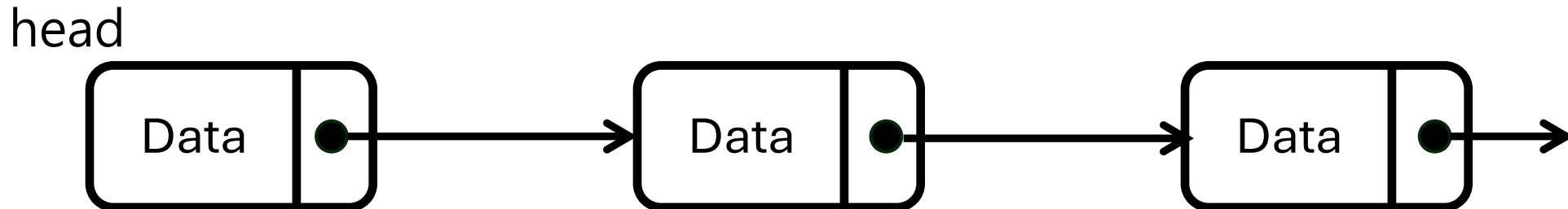
Linked List

즐겁고 알찬 자료구조 튜터링

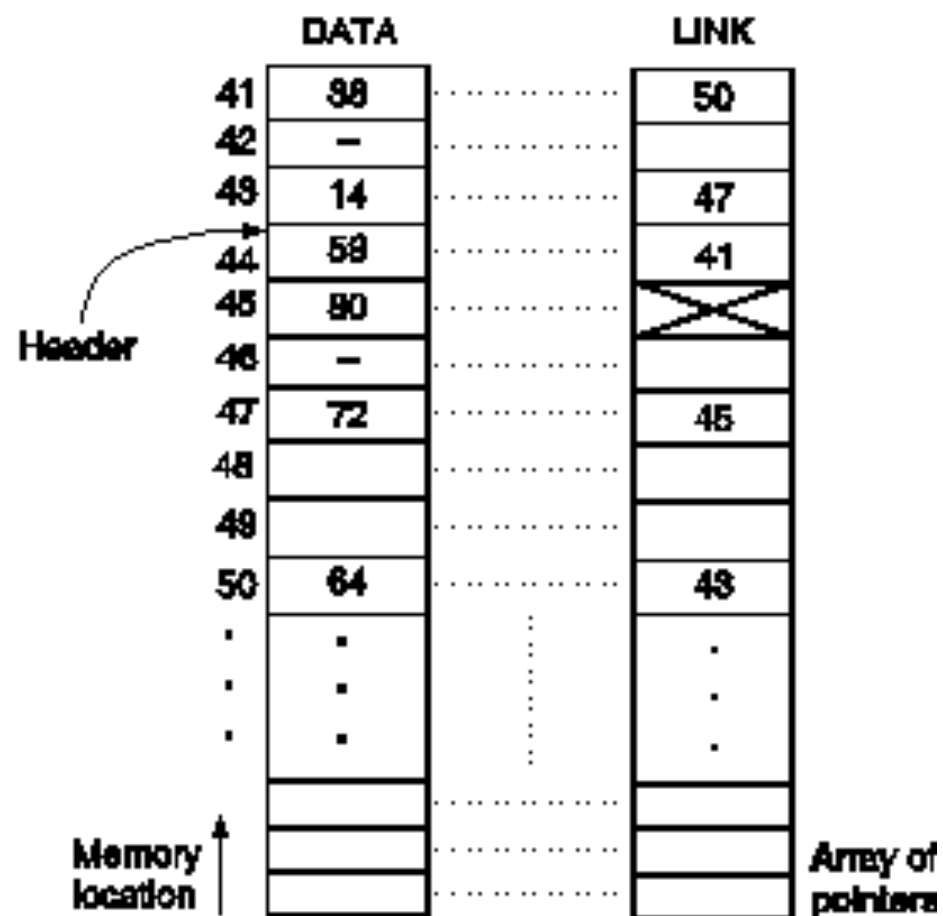
Singly Linked List

Linked List

- 연속된 원소들이 일정한 거리만큼 떨어져서 저장된 **순차 리스트**
- **노드**들이 서로 **연결**되어 표현



Linked List 메모리 할당



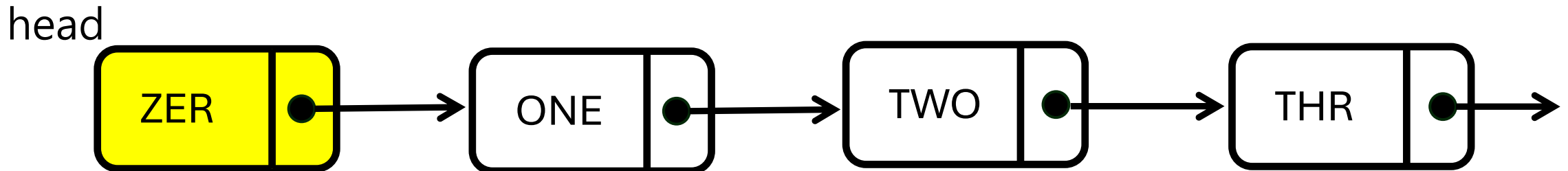
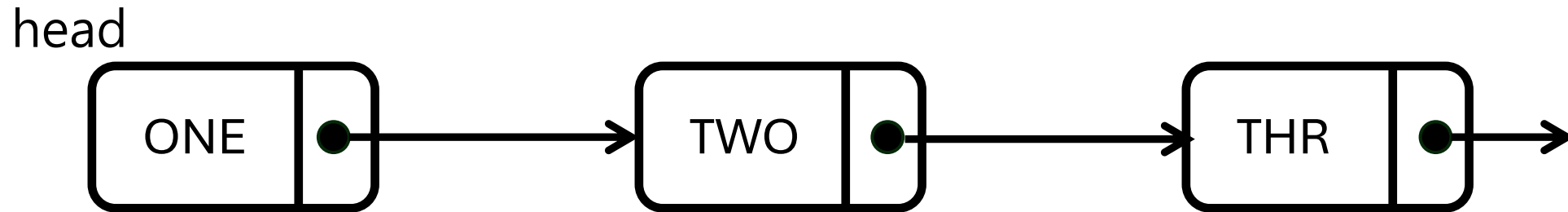
Singly Linked List class

```
1 class SinglyLinkedList;
2 class Node {
3 private:
4     int data;
5     Node* next;
6
7 public:
8     Node(int data) {
9         this->data = data;
10        this->next = nullptr;
11    }
12
13    friend class SinglyLinkedList;
14 };
```

```
1 class SinglyLinkedList {
2 private:
3     Node* head;
4
5 public:
6     SinglyLinkedList() {
7         head = nullptr;
8     }
9
10    bool empty() {
11        return head == nullptr;
12    }
13 }
```

Insert in Singly Linked List

- 새로운 노드를 할당하고, head에 삽입



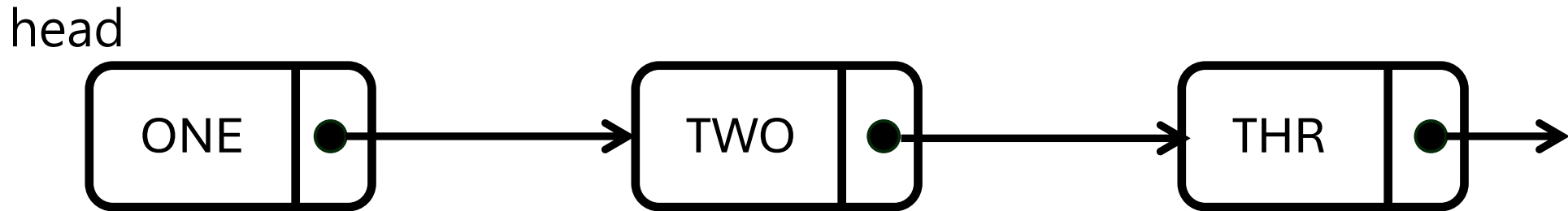
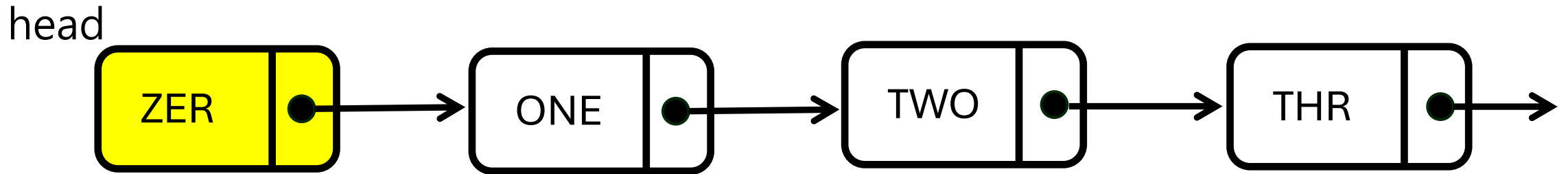
Insert in Singly Linked List




```
1 void add_front(int data) {  
2     Node* new_node = new Node(data);  
3     new_node->next = head;  
4     head = new_node;  
5 }
```

Remove in Singly Linked List

- head에 있는 노드를 제거



Remove in Singly Linked List



```
1 void remove_front() {  
2     if (empty()) {  
3         return;  
4     }  
5     Node* tmp = head;  
6     head = head->next;  
7     delete tmp;  
8 }
```

Singly Linked List의 장단점

- 장점

- Linked List중 구현이 가장 쉽다.

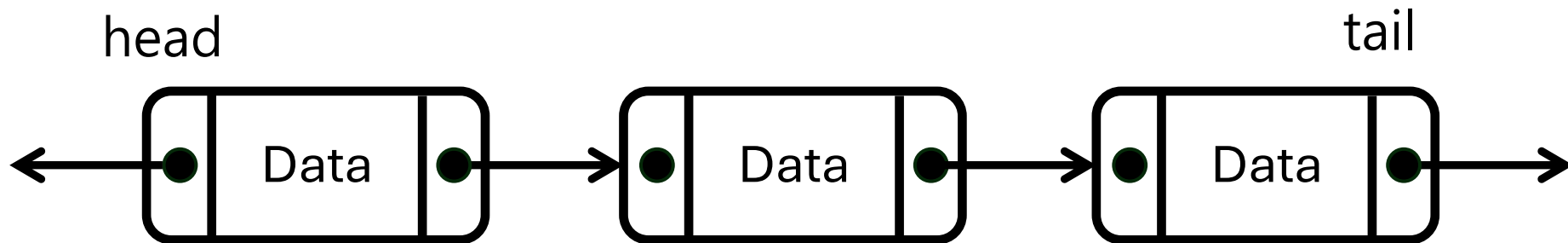
- 단점

- 처음 원소 접근은 head에서만 가능하다
- 역방향 순회가 불가능하다.

Doubly Linked List

Doubly Linked List

- Singly Linked List에서는 역방향 순회가 불가능하고, tail에 있는 원소 제거가 쉽지 않음
- 노드가 다음 노드 뿐만 아니라 이전의 노드도 가리킴



Doubly Linked List class


```
1 class DoublyLinkedList;
2 class Node {
3 private:
4     int data;
5     Node *next;
6     Node *prev;
7
8 public:
9     Node(int data) {
10         this->data = data;
11         this->next = nullptr;
12         this->prev = nullptr;
13     }
14
15     friend class DoublyLinkedList;
16 };
```

```
1 class DoublyLinkedList {
2 private:
3     Node *head;
4     Node *tail;
5
6 public:
7     DoublyLinkedList() {
8         head = nullptr;
9         tail = nullptr;
10    }
11
12    bool empty() {
13        return head == nullptr;
14    }
15 }
```

Insert in Linked List class



```
1 void add_front(int data) {
2     Node *new_node = new Node(data);
3     if (empty()) {
4         head = new_node;
5         tail = new_node;
6     } else {
7         new_node->next = head;
8         head->prev = new_node;
9         head = new_node;
10    }
11 }
```



```
1 void add_back(int data) {
2     Node *new_node = new Node(data);
3     if (empty()) {
4         head = new_node;
5         tail = new_node;
6     }
7     else {
8         new_node->prev = tail;
9         tail->next = new_node;
10        tail = new_node;
11    }
12 }
```

Remove in Linked List class



```
1 void remove_front() {
2     if (empty()) {
3         return;
4     }
5
6     Node *tmp = head;
7     head = head->next;
8     if (head != nullptr) {
9         head->prev = nullptr;
10    }
11    else {
12        tail = nullptr;
13    }
14
15    delete tmp;
16 }
```



```
1 void remove_back() {
2     if (empty()) {
3         return;
4     }
5
6     Node *tmp = tail;
7     tail = tail->prev;
8     if (tail != nullptr) {
9         tail->next = nullptr;
10    }
11    else {
12        head = nullptr;
13    }
14
15    delete tmp;
16 }
```

Doubly Linked List의 장단점

- 장점

- 노드의 이동이 순방향, 역방향으로 자유롭다.
- 원소의 삭제가 singly linked list 보다 쉽다.

- 단점

- 구현이 singly linked list보다 어렵다.