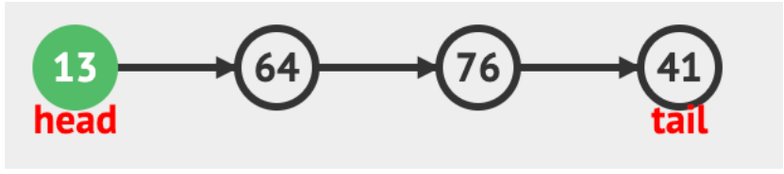
# 땅울림 자료구조 스터디

## Linked List (LL)
## 연결 리스트

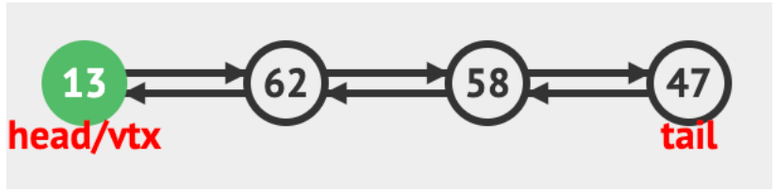
22.03.09

# Linked List

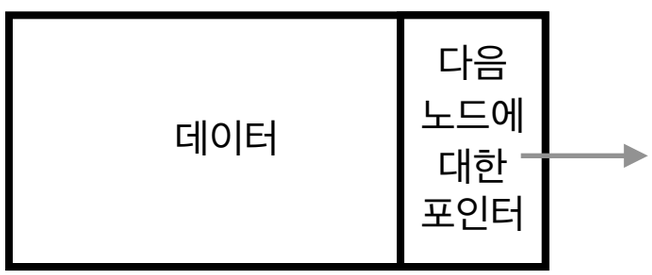
## 단일 Linked List

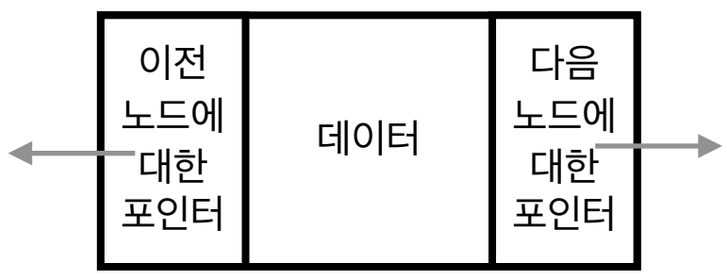



## 이중 Linked List




# Node



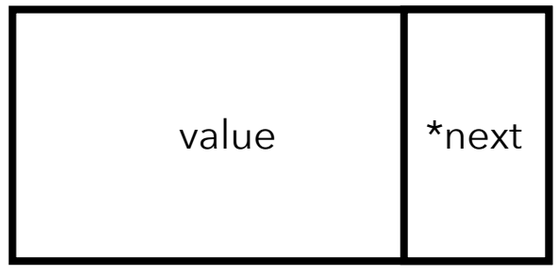



# 단일 연결 리스트

22년도 2주차 2번문제
21년도 2주차 1번문제
21년도 2주차 3번문제

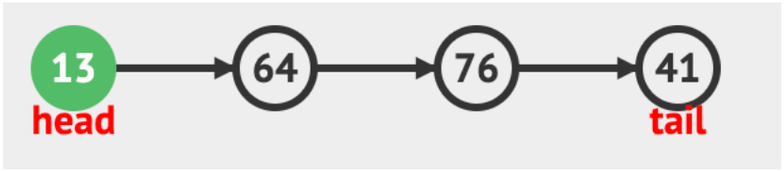
# 노드 구현

```
11  class Node {
12  private:
13      int value;
14      Node *next;
15
16  public:
17      Node() {
18          this->value = 0;
19          this->next = nullptr;
20      }
21
22      ~Node() {
23          this->value = 0;
24          this->next = nullptr;
25      }
26
27      friend class LinkedList;
28  };
```



# 단일 연결 리스트 구현

```
30  class LinkedList {
31  private:
32      Node *head;
33      Node *tail;
34      int size;
35
36  public:
37      LinkedList() {
38          this->head = nullptr;
39          this->tail = nullptr;
40          this->size = 0;
41      }
42
43      ~LinkedList() {
44          this->head = nullptr;
45          this->tail = nullptr;
46          this->size = 0;
47      }
48  }
```




size = 4

# 문제 1

자연수를 저장하는 단일 연결 리스트를 생성하고, 다음의 명령어들을 처리하는 프로그램을 작성하시오.
명령어는 다음과 같이 총 5기지이다.

- **Print()**: 리스트를 순회하며 각 node에 저장된 자연수를 출력하는 함수이다. 단, 리스트가 비어 있을 경우 "empty"를 출력한다.
- **Append($X$)** : 자연수 $X$가 저장된 node를 리스트의 가장 뒤에 삽입하고 Print 함수를 수행하는 함수이다. (단, $X$는 $1 \leq X \leq 10{,}000$)
- **Delete($i$)** : 리스트의 순서를 나타내는 정수 index $i$를 입력 받고 해당하는 index의 node를 삭제하면서 node에 저장된 자연수를 반환하는 함수이다. 삭제 후 반환된 값을 출력한다. 단, 리스트가 비어 있거나 index $i$가 리스트 크기보다 같거나 크면 -1을 반환하고 출력한다. (단, $i$는 $0 \leq i \leq 10{,}000$인 정수)
- **Insert($i, X$)** : 리스트의 순서를 나타내는 정수 index $i$에 자연수 $X$값을 저장하는 새로운 node를 삽입하는 함수이다. 삽입 후 Print함수를 수행한다. 단 $i$가 리스트의 크기보다 클 경우 "Index Error"을 출력한다. (단 $i$는 $0 \leq i \leq 10{,}000$인 정수, $X$는 $1 \leq X \leq 10{,}000$인 자연수)
- **Sum()** : 리스트의 node들에 저장된 자연수들의 합을 출력하는 함수이다. 단, 리스트가 비어 있을 경우 0을 출력한다.
- **Min()** : 리스트의 node들에 저장된 자연수 중에서 최솟값을 출력하는 함수이다. 단, 리스트가 비어 있을 경우 "empty"를 출력한다.

자연수 저장 **단일 연결 리스트** 생성

**Print() :**
순회하며 출력
비어있으면 empty 출력

**Delete(i) :**
해당하는 인덱스의 노드 삭제 후 삭제된 값 출력
비어있거나 인덱스 값 유효하지 않으면 -1 출력

**Append(X) :**
가장 뒤에 노드 삽입 후 Print 수행

**Insert(i, X) :**
index에 노드 삽입 후 print 수행
인덱스 값 유효하지 않으면 Index Error 출력

**Sum() :**
노드 값 합 출력
비어있으면 0 출력

**Min() :**
최소 노드 값 출력
비어있을 경우 empty 출력

자연수 저장 **단일 연결 리스트** 생성

**Print() :**
순회하며 출력
비어있으면 empty 출력

순회

**Append(X) :**
가장 뒤에 노드 삽입 후 Print 수행

추가

**Sum() :**
노드 값 합 출력
비어있으면 0 출력

순회

**Delete(i) :**
해당하는 인덱스의 노드 삭제 후 삭제된 값 출력
비어있거나 인덱스 값 유효하지 않으면 -1 출력

삭제

**Insert(i, X) :**
index에 노드 삽입 후 print 수행
인덱스 값 유효하지 않으면 Index Error 출력

추가

**Min() :**
최소 노드 값 출력
비어있을 경우 empty 출력

순회

```
49          bool empty() {
50              return (this->size == 0);
51          }
```

사이즈 변수의 값이 0인지 확인해서 비어있는지 체크

# 순회

Print() :
순회하며 출력
비어있으면 empty 출력

```
53      void printNode() {
54          if (empty()) {
55              cout << "empty";
56              return;
57          }
58
59          Node *curNode = this->head;
60          while (curNode != nullptr) {
61              cout << curNode->value << " ";
62              curNode = curNode->next;
63          }
64      }
```

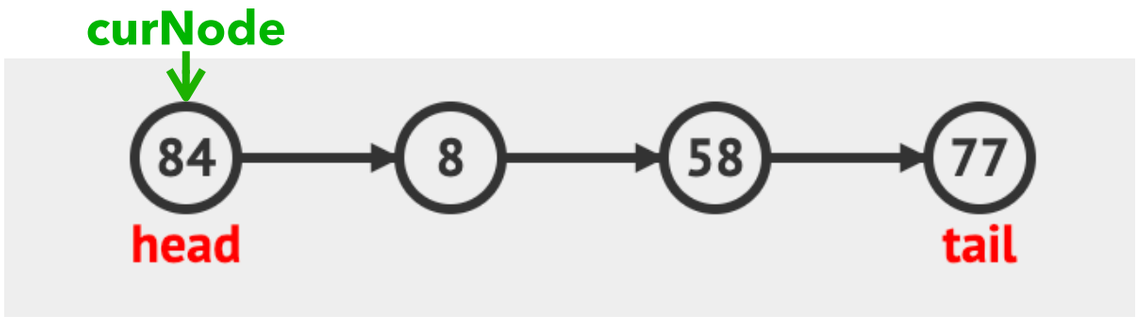
비어있으면 empty 출력 후 리턴

---

head를 가리키는 노드 생성

---

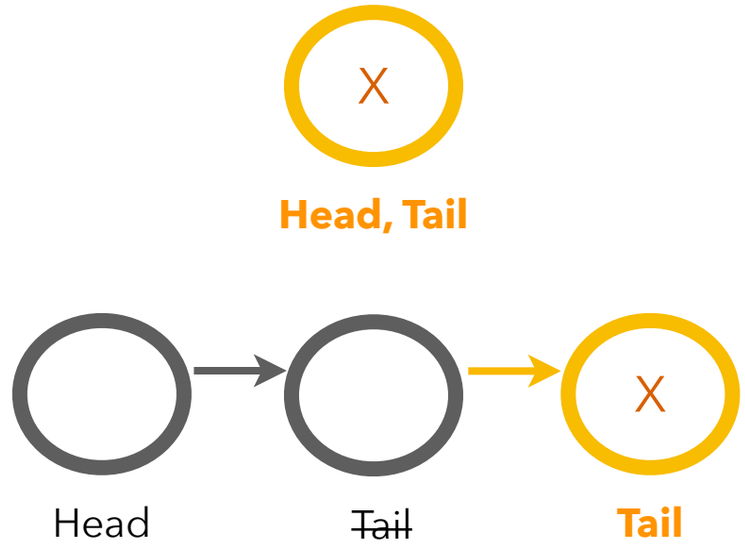
노드의 값 출력
다음 노드로 이동
(노드가 더이상 없을 때까지 반복)




```
84    8    58    77
```

**추가**

Append(X) :
가장 뒤에 노드 삽입 후 Print 수행




```
66      void append(int x) {
67          Node *appendNode = new Node;
68          appendNode->value = x;
69
70          if (empty()) {
71              this->head = appendNode;
72              this->tail = appendNode;
73          } else {
74              this->tail->next = appendNode;
75              this->tail = appendNode;
76          }
77
78          this->size++;
79          print();
80      }
```

새로운 노드 생성
새로운 노드에 x값 할당

---

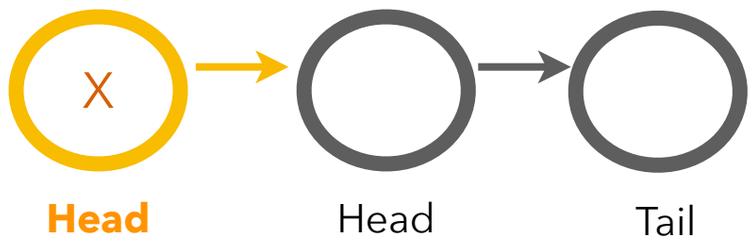
생성한 노드를 리스트에 연결
empty -> head로 지정
생성한 노드를 tail로 지정
no empty -> tail의 next로 지정
생성한 노드를 tail로 지정

---

사이즈 추가
프린트 함수 실행

**추가**

Insert(i, X) :
index에 노드 삽입 후 print 수행
인덱스 값 유효하지 않으면 Index Error 출력




```
113     void insertNode(int index, int x) {
114         if (index > this->size) {
115             cout << "Index Error";
116             return;
117         }
118 
119         if (index == 0) {
120             Node *insertNode = new Node;
121             insertNode->value = x;
122             insertNode->next = this->head;
123             this->head = insertNode;
124             printNode();
125         } else if (index == this->size) {
126             appendNode(x);
127         } else {
128             Node *insertNode = new Node;
129             insertNode->value = x;
130 
131             Node *curNode = this->head;
132             for (int i=1; i<index; i++) {
133                 curNode = curNode->next;
```

인덱스 유효성 검사

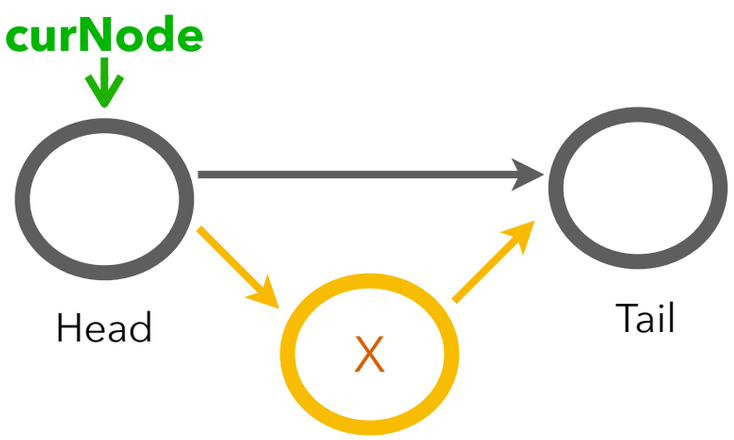
---

(head에 삽입하는 경우)
새 노드 생성 후 값 할당
기존 head와 연결해주기
head 업데이트해주기
print함수 실행

---

(tail에 삽입하는 경우)
appendNode() 함수 실행

```
119         if (index == 0) {
120             Node *insertNode = new Node;
121             insertNode->value = x;
122             insertNode->next = this->head;
123             this->head = insertNode;
124             printNode();
125         } else if (index == this->size) {
126             appendNode(x);
127         } else {
128             Node *insertNode = new Node;
129             insertNode->value = x;
130 
131             Node *curNode = this->head;
132             for (int i=1; i<index; i++) {
133                 curNode = curNode->next;
134             }
135             insertNode->next = curNode->next;
136             curNode->next = insertNode;
137             printNode();
138         }
139         this->size++;
140     }
```




(중간에 삽입하는 경우)
새로운 노드 생성 후 값 할당하기

원하는 인덱스 전까지 이동하기

리스트와 새 노드 연결하기

---

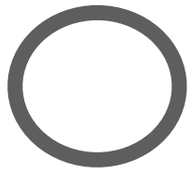
프린트 함수 실행

---

사이즈 증가(까먹지 말기)

# 삭제

Delete(i) :
해당하는 인덱스의 노드 삭제 후 삭제된 값 출력
비어있거나 인덱스 값 유효하지 않으면 -1 출력



Head, Tail

```
int deleteNode(int index) {
    if (index >= this->size) {
        return -1;
    }

    Node *deleteNode = this->head;

    if (index == 0) {
        if (this->size == 1) {
            this->head = nullptr;
            this->tail = nullptr;
        } else {
            this->head = deleteNode->next;
        }
    } else {
```

인덱스 유효성 검사

---

deleteNode라는 노드 생성 후
head로 임시 초기화

---

(Head 삭제하는 경우(index=0))

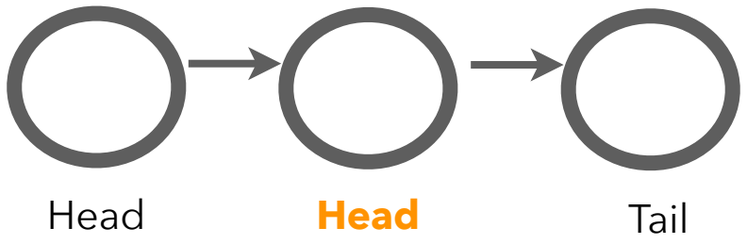
size가 1인 경우
head랑 tail 둘 다 nullptr로

---

size가 2 이상인 경우
head를 head의 next로 설정

# 삭제

Delete(i) :
해당하는 인덱스의 노드 삭제 후 삭제된 값 출력
비어있거나 인덱스 값 유효하지 않으면 -1 출력




```
int deleteNode(int index) {
    if (index >= this->size) {
        return -1;
    }

    Node *deleteNode = this->head;

    if (index == 0) {
        if (this->size == 1) {
            this->head = nullptr;
            this->tail = nullptr;
        } else {
            this->head = deleteNode->next;
        }
    } else {
```

인덱스 유효성 검사

---

deleteNode라는 노드 생성 후
head로 임시 초기화

---

(Head 삭제하는 경우(index=0))

size가 1인 경우
head랑 tail 둘 다 nullptr로

---

size가 2 이상인 경우
head를 head의 next로 설정

```
    } else {
        Node *prevNode = this->head;
        for (int i=0; i<index; i++) {
            prevNode = deleteNode;
            deleteNode = deleteNode->next;
        }
        prevNode->next = deleteNode->next;
        if(deleteNode == this->tail) {
            this->tail = prevNode;
        }
    }

    int deleteValue = deleteNode->value;
    delete deleteNode;
    this->size--;
    return deleteValue;
}
```

(Head가 아닌 거 삭제하는 경우)

(i-1번째 노드 찾기)->prevNode

head에서 i만큼 이동해서
이전 노드(prevNode)와
삭제할 노드(deleteNode) 설정

---

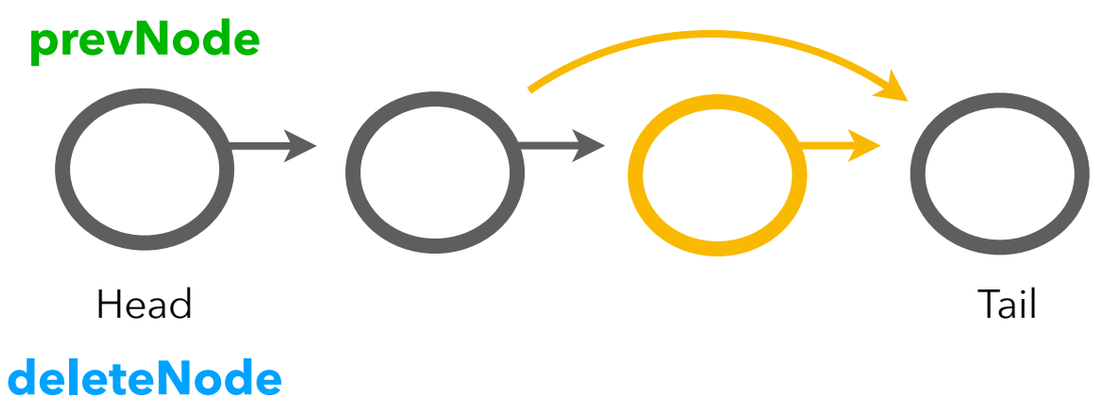
이전노드의 next를
삭제할 노드의 next로 연결해주기

---

(tail을 삭제할 경우)
이전노드를 tail로 설정해주기

---

deleteValue변수에 삭제값 저장
메모리 반환
사이즈 감소
삭제된 값 리턴



# 순회

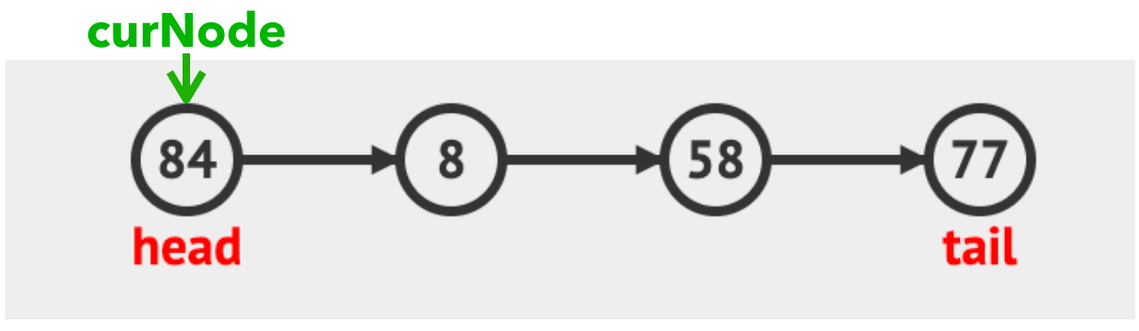
Sum() :
노드 값 합 출력
비어있으면 0 출력

```
160     int sumNode() {
161         int sumValue = 0;
162         Node *curNode = this->head;
163 
164         while (curNode != nullptr) {
165             sumValue += curNode->value;
166             curNode = curNode->next;
167         }
168 
169         return sumValue;
170     }
```

sum을 저장할 변수 0 초기화
임시 노드 변수 head 가리키게 초기화

순회하면서
sum변수에 값을 계속 더해주기

sum변수값 리턴




| sum | 1940 |
|---|---|

# 순회

Min() :
최소 노드 값 출력
비어있을 경우 empty 출력

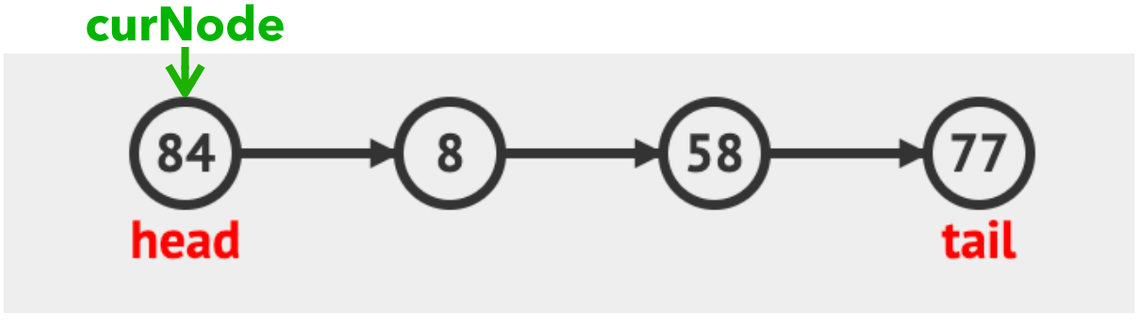
```
142     void minNode() {
143         if(empty()) {
144             cout << "empty";
145             return;
146         }
147
148         int minValue = 10000;
149         Node *curNode = this->head;
150
151         while(curNode != nullptr) {
152             if(curNode->value < minValue) {
153                 minValue = curNode->value;
154             }
155             curNode = curNode->next;
156         }
157         cout << minValue;
158     }
```

비어있을 경우 empty 출력

최소값 저장 변수 큰 값으로 초기화
임시 노드 변수 head로 초기화

순회하면서
minValue변수 업데이트

출력




| min | 8 8 4 |
|---|---|

# 이중 연결 리스트

21년도 2주차 2번문제
21년도 2주차 4번문제

# 문제 2

자연수를 저장하는 이중 연결 리스트를 생성하고, 다음의 명령어들을 처리하는 프로그램을 작성하시오.
명령어는 다음과 같이 총 5가지이다.

- **Print()** : header에서 trailer 방향으로 리스트를 순회하며 각 node에 저장된 자연수를 출력하는 함수이다. 단, 리스트가 비어 있을 경우 "empty"를 출력한다.
- **Append($X$)** : 자연수 $X$가 저장된 node를 리스트의 가장 뒤에 삽입하고 Print 함수를 수행하는 함수이다. (단, $1 \leq X \leq 10{,}000$)
- **Delete($i$)** : 리스트의 순서를 나타내는 정수 index $i$를 입력 받고 해당하는 index의 node를 삭제하면서 node에 저장된 자연수를 반환하는 함수이다. 삭제 후 반환된 값을 출력한다. 단, 리스트가 비어 있거나 index $i$가 리스트 크기보다 같거나 크면 -1을 반환하고 출력한다. (단, $i$는 $0 \leq i \leq 10{,}000$인 정수)
- **Print_reverse()** : trailer에서 header 방향으로 리스트를 순회하며 각 node에 저장된 자연수를 출력하는 함수이다. 단, 리스트가 비어 있을 경우 "empty"를 출력한다.
- **Sum()** : 리스트의 node들에 저장된 자연수들의 합을 출력하는 함수이다. 단, 리스트가 비어 있을 경우 0을 출력한다.
- **Max()** : 리스트의 node들에 저장된 자연수들 중에서 최댓값을 출력하는 함수이다. 단, 리스트가 비어 있을 경우 "empty"를 출력한다.

자연수 저장 **이중 연결 리스트** 생성

**Append(X) :**
가장 뒤에 노드 삽입 후 Print 수행

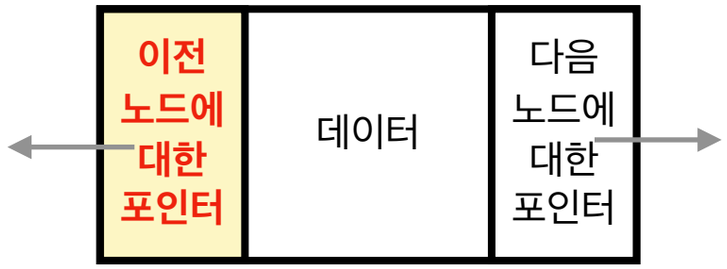
**Delete(i) :**
해당하는 인덱스의 노드 삭제 후 삭제된 값 출력
비어있거나 인덱스 값 유효하지 않으면 -1 출력

**Print() :**
header에서 trailer 방향으로 순회하며 출력
비어있으면 empty 출력

**Print_reverse() :**
trailer에서 header 방향으로 순회하며 출력
비어있으면 empty 출력

**Sum() :**
노드 값 합 출력
비어있으면 0 출력

**Max() :**
최대 노드 값 출력
비어있을 경우 empty 출력




```
class Node {
private:
    int value;
    Node *next;
    Node *prev; // 추가됨

public:
    Node() {
        this->value = 0;
        this->next = nullptr;
        this->prev = nullptr; // 추가됨
    }

    ~Node() {
        this->value = 0;
        this->next = nullptr;
        this->prev = nullptr; // 추가됨
    }

    friend class LinkedList;
};
```

```
27  class LinkedList {
28  private:
29      Node *head;
30      Node *tail;
31      int size;
32
33  public:
34      LinkedList() {
35          this->head = nullptr;
36          this->tail = nullptr;
37          this->size = 0;
38      }
39
40      ~LinkedList() {
41          this->head = nullptr;
42          this->tail = nullptr;
43          this->size = 0;
44      }
45  };
```

Print_reverse() :
trailer에서 header 방향으로 순회하며 출력
비어있으면 empty 출력

기존 정방향 순회 출력 코드

```
53      void printNode() {
54          if (empty()) {
55              cout << "empty";
56              return;
57          }
58
59          Node *curNode = this->head;
60          while (curNode != nullptr) {
61              cout << curNode->value << " ";
62              curNode = curNode->next;
63          }
64      }
```

코드 작성해 보세요

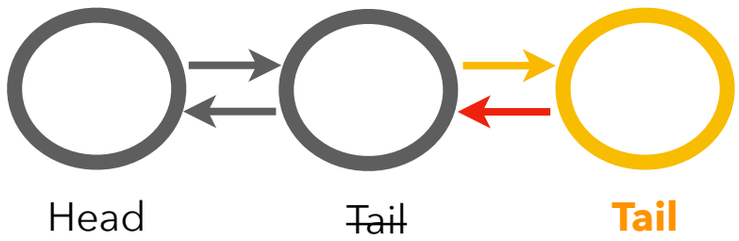
```
void printReverseNode() {
    if (empty()) {
        cout << "empty";
        return;
    }

    Node *curNode = this->tail;
    while (curNode != nullptr) {
        cout << curNode->value << " ";
        curNode = curNode->prev;
    }
}
```

Append(X) :
가장 뒤에 노드 삽입 후 Print 수행

## 기존 단일 연결리스트 코드

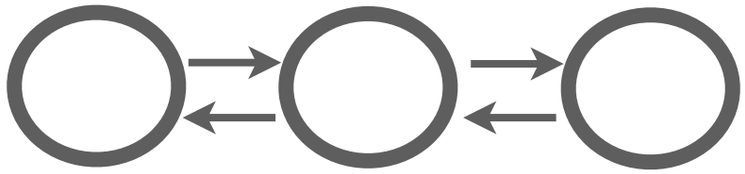
```
65
66      void appendNode(int x) {
67          Node *appendNode = new Node;
68          appendNode->value = x;
69
70          if (empty()) {
71              this->head = appendNode;
72          } else {
73              this->tail->next = appendNode;
74              appendNode->prev = this->tail;
75          }
76          this->tail = appendNode;
77          this->size++;
78          printNode();
79      }
80
```




새로 추가된 노드의 prev에
기존 tail 노드 연결하는 코드 추가

코드 작성해 보세요

# 코드 작성해 보세요




```
int deleteNode(int index) {
    if (index >= this->size) {
        return -1;
    }

    Node *deleteNode = this->head;

    if (index == 0) {
        if (this->size == 1) {
            this->head = nullptr;
            this->tail = nullptr;
        } else {
            this->head = deleteNode->next;
            deleteNode->next->prev = nullptr; //
    } else {
```

(Head 삭제하는 경우(index=0))

size가 2 이상인 경우

삭제 노드->next->prev를
다시 설정해주기

```
    } else {
        Node *prevNode = this->head;
        for (int i=0; i<index; i++) {
            prevNode = deleteNode;
            deleteNode = deleteNode->next;
        }
        prevNode->next = deleteNode->next;
        if(deleteNode == this->tail) {
            this->tail = prevNode;
        } else {   // 코드 작성   }
    }

    int deleteValue = deleteNode->value;
    delete deleteNode;
    this->size--;
    return deleteValue;
}
```

(Head가 아닌 거 삭제하는 경우)

(i-1번째 노드 찾기)->prevNode

head에서 i만큼 이동해서
이전 노드(prevNode)와
삭제할 노드(deleteNode) 설정

---

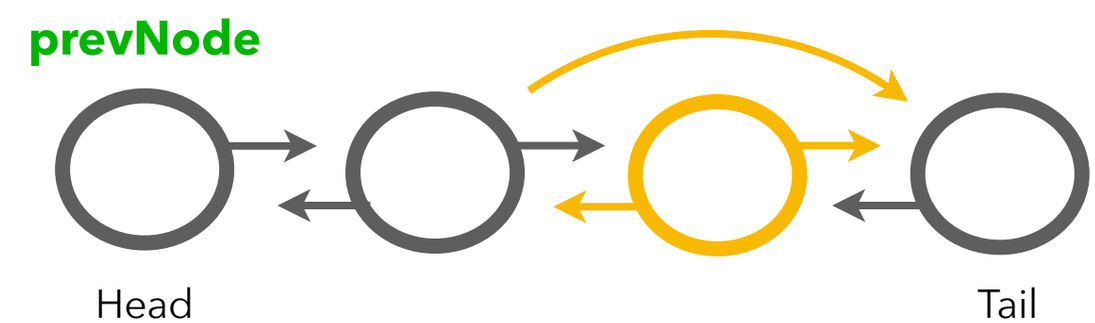
이전노드의 next를
삭제할 노드의 next로 연결해주기

---

(tail을 삭제할 경우)
이전노드를 tail로 설정해주기

---

deleteValue변수에 삭제값 저장
메모리 반환
사이즈 감소
삭제된 값 리턴



```
    } else {
        Node *prevNode = this->head;
        for (int i=0; i<index; i++) {
            prevNode = deleteNode;
            deleteNode = deleteNode->next;
        }
        prevNode->next = deleteNode->next;
        if(deleteNode == this->tail) {
            this->tail = prevNode;
        } else { // 추가됨
            deleteNode->next->prev = prevNode;
        }
    }

    int deleteValue = deleteNode->value;
    delete deleteNode;
    this->size--;
    return deleteValue;
}
```

(Head가 아닌 거 삭제하는 경우)

(i-1번째 노드 찾기)->prevNode

head에서 i만큼 이동해서
이전 노드(prevNode)와
삭제할 노드(deleteNode) 설정

---

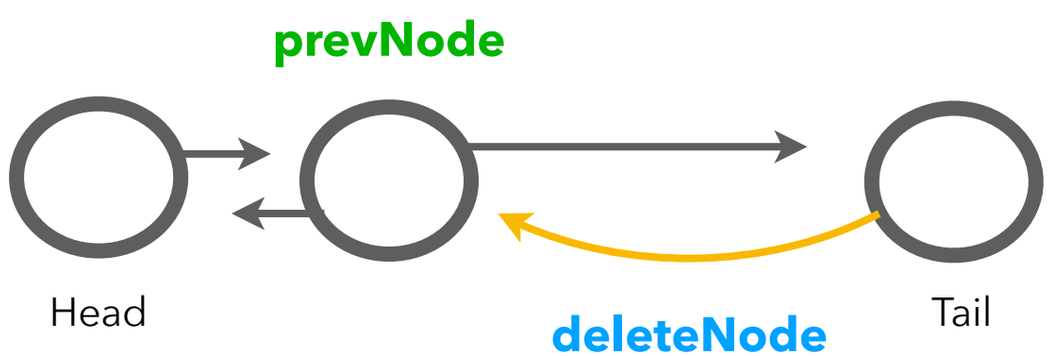
이전노드의 next를
삭제할 노드의 next로 연결해주기

---

(tail을 삭제할 경우)
이전노드를 tail로 설정해주기

---

deleteValue변수에 삭제값 저장
메모리 반환
사이즈 감소
삭제된 값 리턴



· 순회 ·

· 추가 ·

· 삭제 ·

# 순회(출력)

head를 가리키는 순회용 노드 생성

while (순회용 노드 != 널포인터) {
    다음 노드로 넘어가기
}

```
Node *curNode = this->head;
while (curNode != nullptr) {
    curNode = curNode->next;
}
```

추가

삭제

출력

# 추가

```
if (head에 추가) {
    새로운 노드 생성 후 값 할당
    새로운 노드->next = head
} else if (tail에 추가) {
    새로운 노드 생성 후 값 할당
    if (비어있던 경우) {
        head = tail = 새로운 노드
    } else {
        tail->next = 새로운 노드
        새로운 노드->prev = tail        // 이중연결리스트
        tail = 새로운 노드
    }
} else {
    새로운 노드 생성 후 값 할당
    head를 가리키는 순회용 노드 생성
    for문으로 순회용 노드를 원하는 인덱스까지 이동
    새로운 노드->next = 순회용 노드->next
    순회용 노드->next = 새로운 노드
}
size 1 증가
```

삭제

출력

추가

* 이중연결리스트

# 삭제

head를 가리키는 임시 삭제할 노드 생성
if (head 삭제) {
    if (size가 1이면) {
        head = tail = nullptr
    } else {
        head = 삭제할 노드->next
        삭제할노드->next->prev = nullptr
    }
} else {
    head를 가리키는 임시 prev노드 생성
    for문 사용
        원하는 인덱스까지 삭제할 노드와 prev노드 순회
    prev노드->next = 삭제할 노드->next
    if (tail 삭제) {
        tail = prev노드
    } else {
        삭제할노드->next->prev = prev 노드
    }
}
삭제값 = 삭제할 노드->value
delete 삭제할 노드
size 1 감소

* 런타임 에러 주요 원인

배열 인덱스 잘못 참조했을 때

arr[100]이라는 배열을 선언 하고 arr[-1]이나 arr[101]을 참조하거나 하는 경우
(-> for문에서 변수 초기값이나 범위 잘못 설정하면 이렇게 됨)

* 런타임 에러 원인 알려주는 사이트

https://ideone.com/

* 연결 리스트 시뮬 사이트

https://visualgo.net/en/list