땅울림

02

스택 Stack

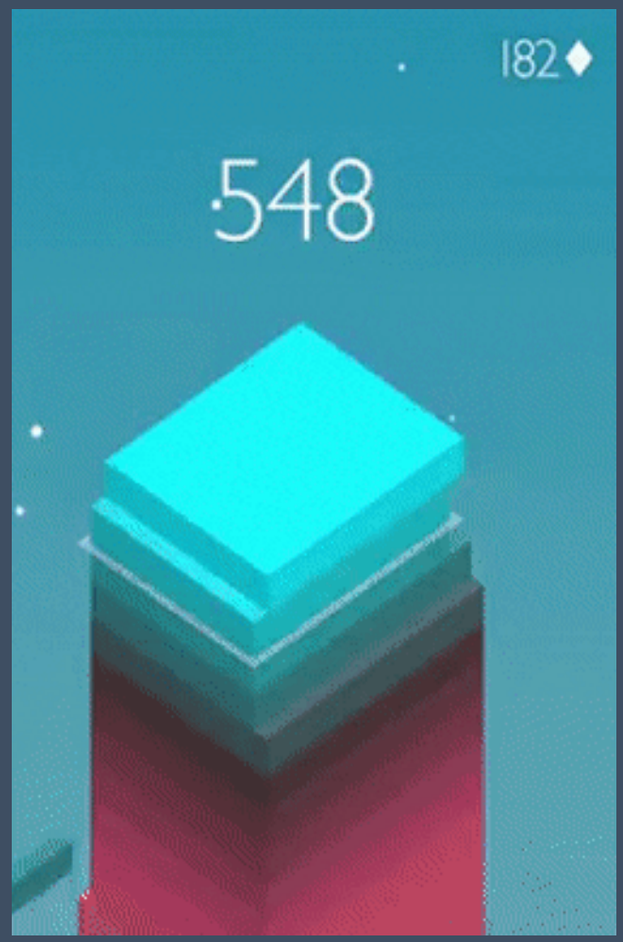
자료구조 스터디 2주차

- 스택 개념설명
- 스택 구현방법
- 유형 설명

Last In First Out (LIFO 후입선출)
나중에 들어간 자료가 먼저 나온다.

한 쪽 끝에서만 자료를 넣고 뺄 수 있는 자료구조
데이터가 들어오고 나가는 방향이 하나이며 같다.

데이터를 쌓아올리는 방식
(책 쌓는거랑 비슷)



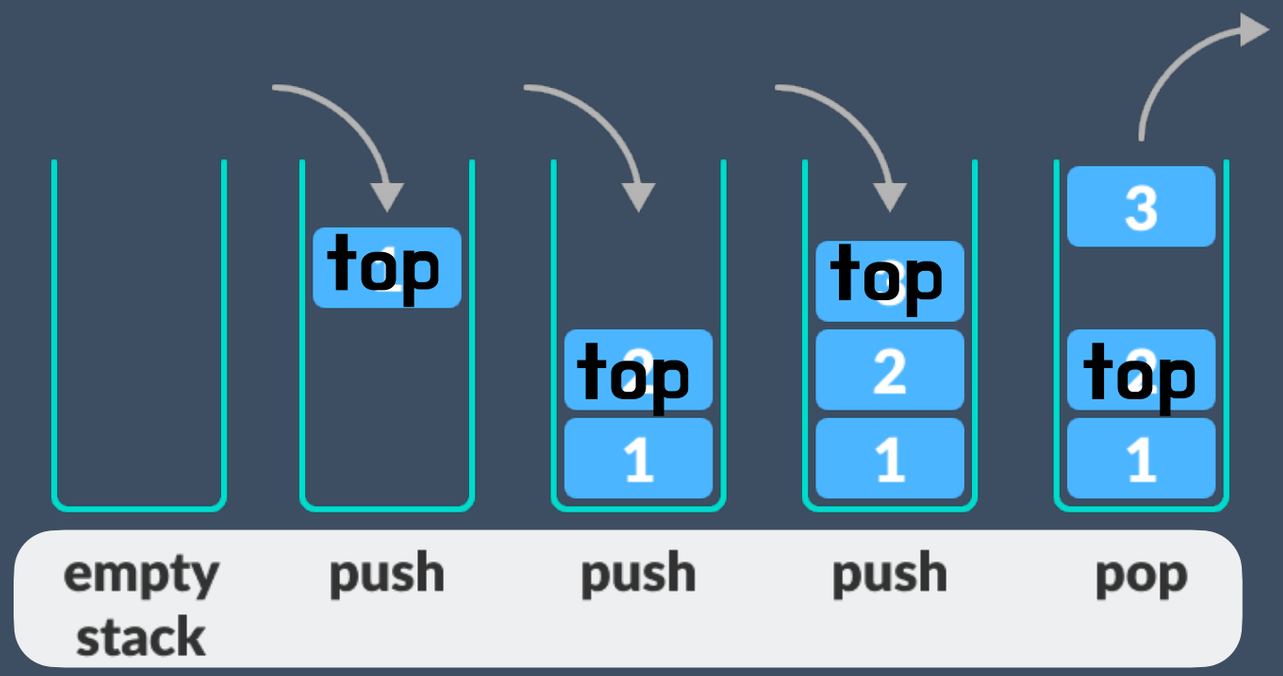
—

스택
Stack

스택의 탑이 뭐야

스택의 탑(Top of Stack)은 현재 가장 최근에 스택에 추가된 요소를 가리키는 포인터를 말합니다. 스택은 후입선출(LIFO) 구조를 가지기 때문에, 탑은 가장 먼저 제거될 요소를 가리키게 됩니다. 새로운 요소가 스택에 추가될 때마다 탑은 업데이트되며, 요소가 제거될 때마다 그 아래에 있는 다음 요소가 새로운 탑이 됩니다. 스택을 구현할 때 탑은 매우 중요한 개념이며, 삽입, 삭제, 탐색 등의 연산을 수행할 때 사용됩니다.

스택
Stack



top에서만 자료를 넣고(push) 뺄 수(pop) 있는 자료구조

1 기본연산

" PUSH (삽입) "

스택의 **맨 위**에 데이터를
추가하는 연산

기본연산 2

" POP (삭제) "

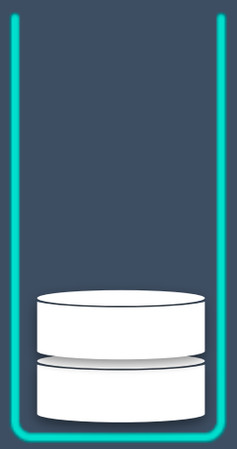
스택의 **맨 위**에 있는 데이터를
삭제하는 연산

—

스택
Stack

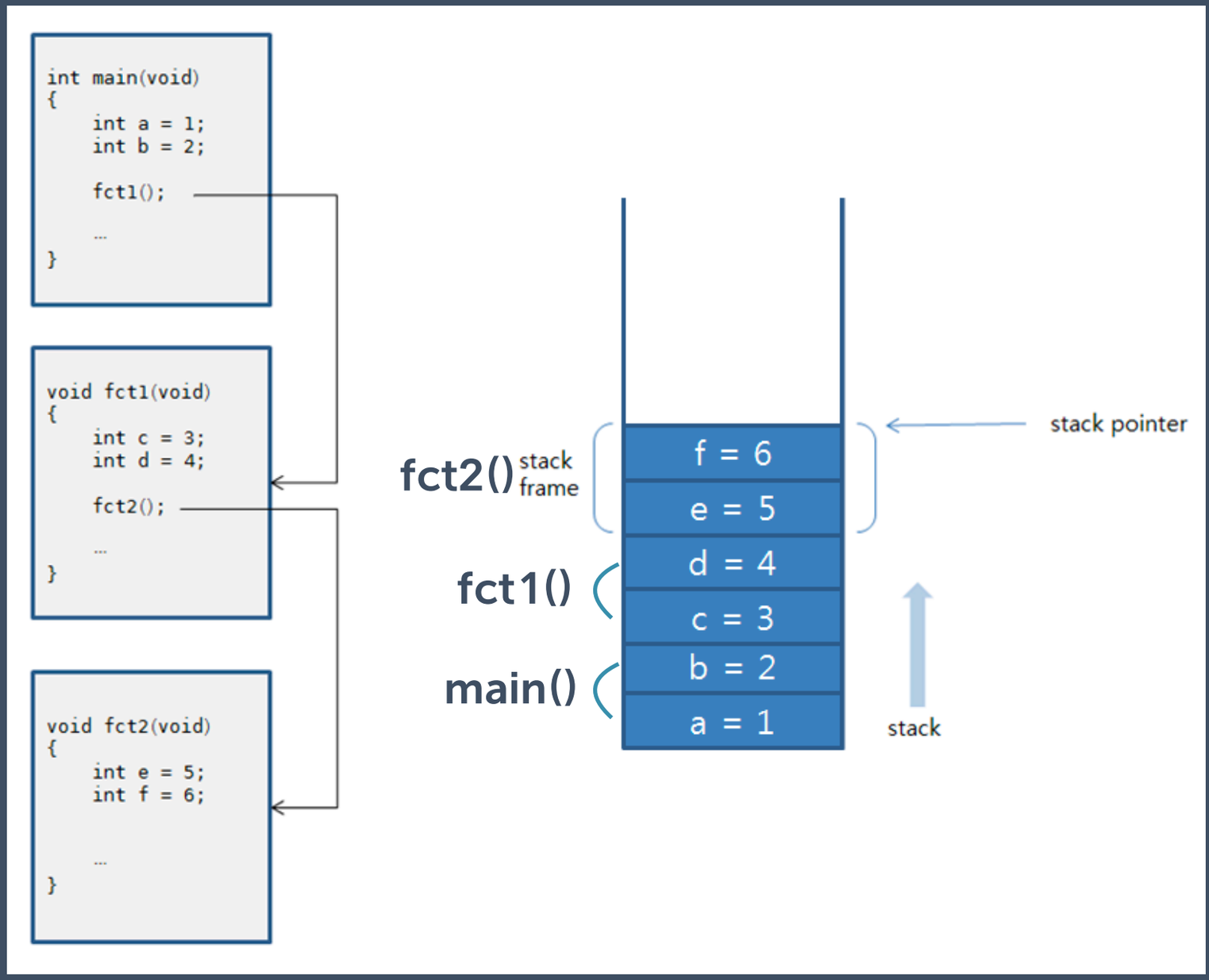
PUSH

POP



스택 활용

- 함수 호출의 재귀적 구현
- 수식의 후위 표기법 계산
- 웹 브라우저의 뒤로 가기 버튼



—

스택
Stack

스택 구현

1. 배열

2. 링크드 리스트

스택 구현

push()
: 스택에 삽입

pop()
: 스택에서 빼기

top()
: 스택 맨 위의 수 return

size()
: 스택 크기 return

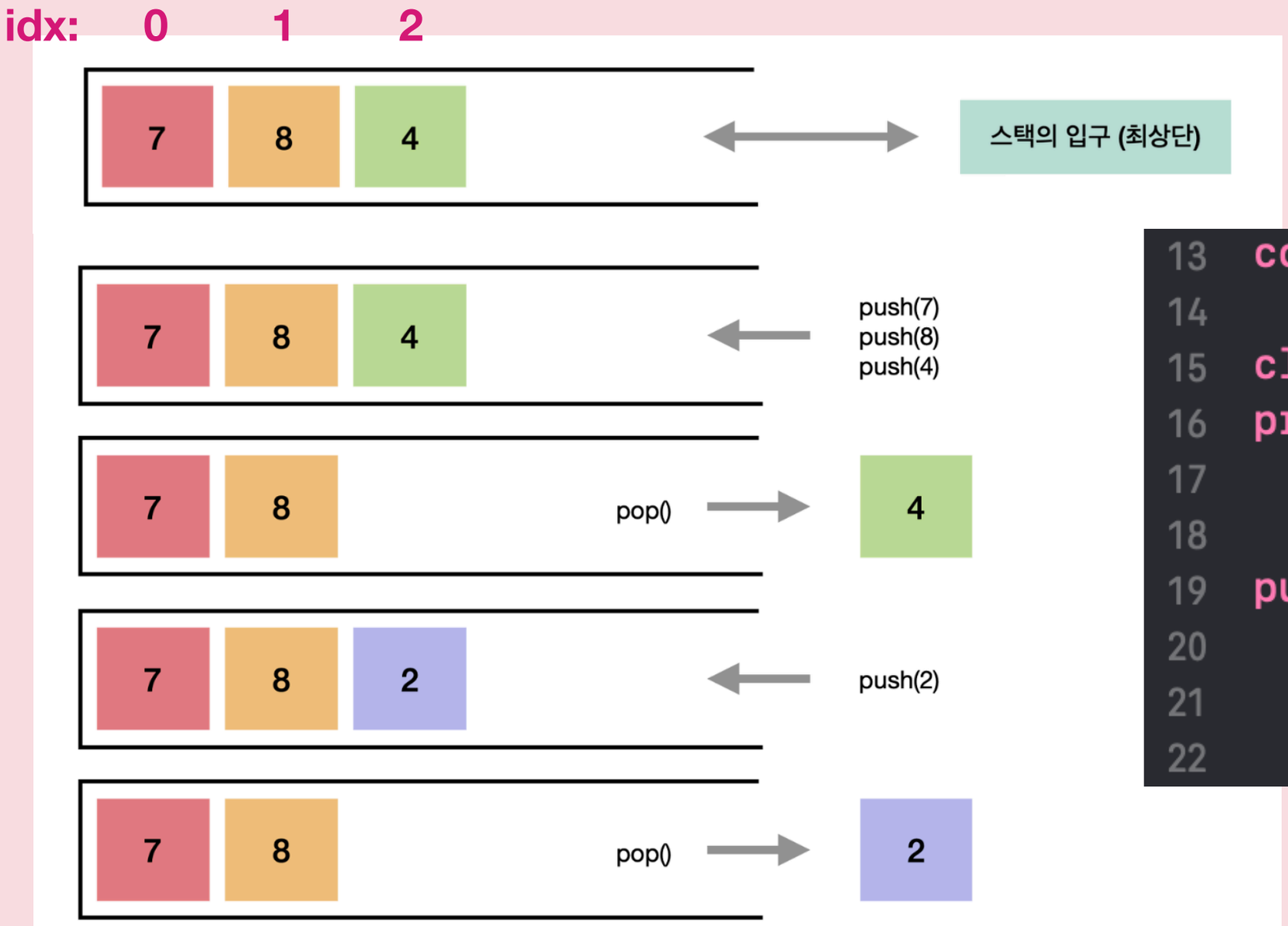
empty()
: 비었으면 T, 아니면 F

배열을
사용한 스택

유형 1-1

배열을 사용한 스택

top
: arr[size-1]




```
13  const int MAX_SIZE = 10;
14
15  class Stack {
16  private:
17      int arr[MAX_SIZE];
18      int size;
19  public:
20      Stack() {
21          size = 0;
22      }
```

유형 1-1

배열을 사용한 스택

TOP

if (비어있으면) {
    // 문제 조건 따라 에러처리
} else {
    arr[size-1] 리턴 또는 출력
}

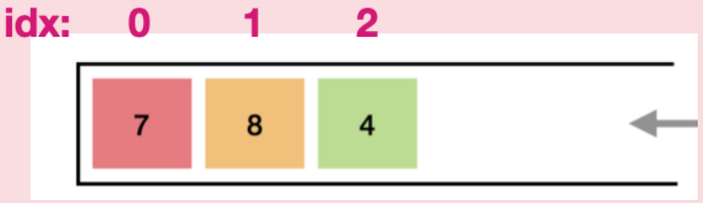
EMPTY

bool 타입

size 0인지 확인

```
void top() {
    if (empty()) {
        cout << "-1\n";
    } else {
        cout << arr[size-1] << "\n";
    }
}
```

```
23      bool empty() {
24          return size==0;
25      }
```



size = 3
top은 arr[2]

배열을 사용한 스택

PUSH

```
if (사이즈 == 최대사이즈) {
    // 문제 조건 따라 에러처리
} else {
    arr[size] = x
    size++
}
```

```
33      void push(int x) {
34          if (size==t) {
35              cout << "FULL\n";
36          } else {
37              arr[size] = x;
38              size++;
39          }
40      }
```

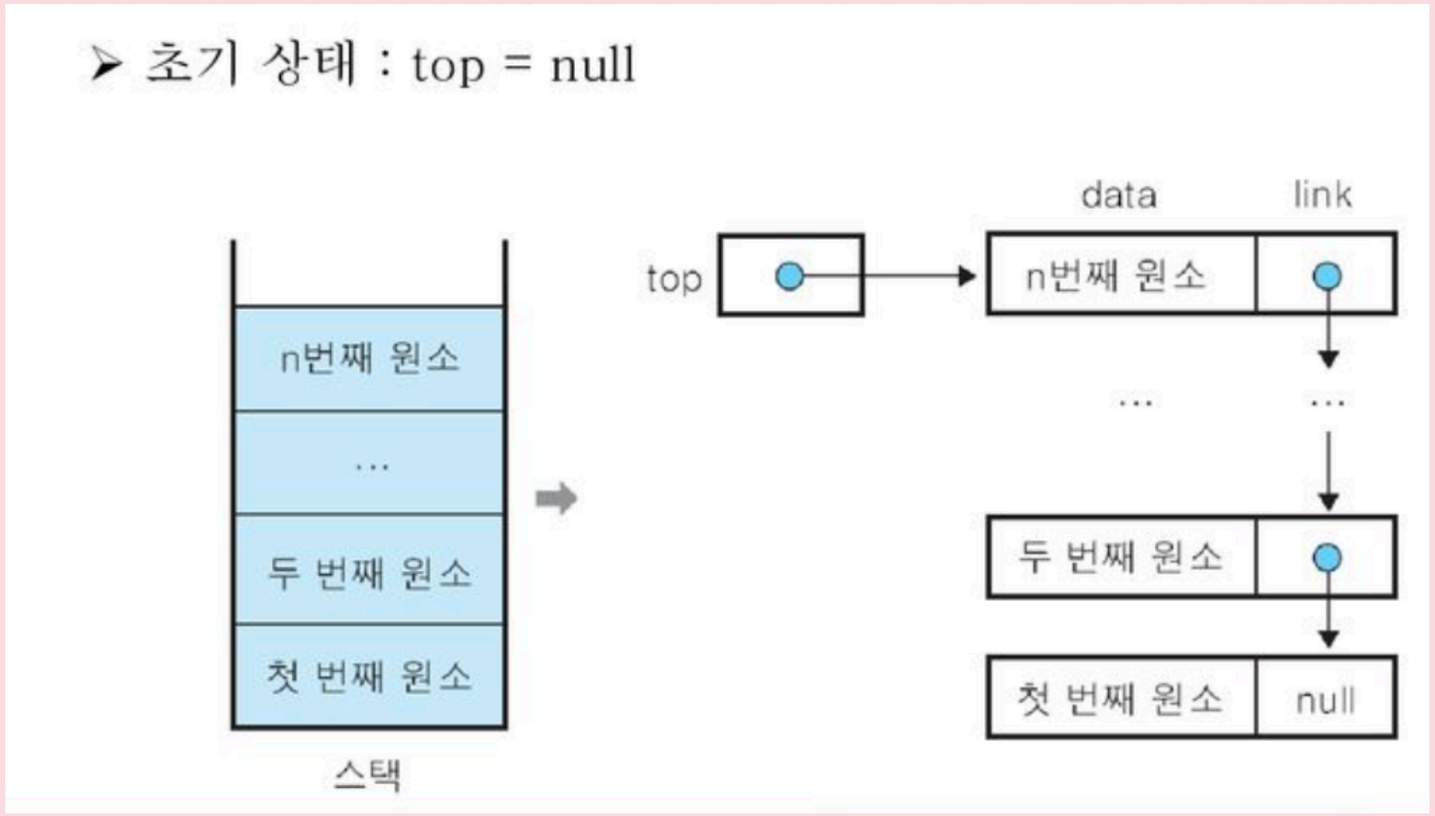
*순서 주의

POP

```
if (비어있으면) {
    // 문제 조건 따라 에러처리
} else {
    top 함수 실행 또는 출력
    size--
}
```

```
41      void pop() {
42          if (empty()) {
43              cout << "-1\n";
44          } else {
45              top();
46              size--;
47          }
48      }
```

링크드 리스트를 사용한 스택



1. 노드구현
2. 링리구현
3. 링리 함수 구현(append, delete)
4. 스택 구현
4. 스택 함수 구현(top, push, pop)

유형 1-2

링리를 사용한 스택

1. 노드 구현

```
11  class Node {
12  private:
13      int value;
14      Node *next;
15
16  public:
17      Node() {
18          value = 0;
19          next = nullptr;
20      }
21      friend class LinkedList;
22      friend class Stack;
23  };
```

2. 링크드 리스트 구현

```
25  class LinkedList {
26  private:
27      Node *head;
28      Node *tail;
29      int size;
30
31  public:
32      LinkedList() {
33          head = nullptr;
34          tail = nullptr;
35          size = 0;
36      }
37
38      bool empty() {
39          return size==0;
40      }
78      friend class Stack;
79  };
```

3. 링리 함수 구현

링리를 사용한 스택

```
42      // 스택이라 끝에 추가하는 경우만 존재
43      void append(int x) {
44          Node *appendNode = new Node;
45          appendNode->value = x;
46
47          if (empty()) {
48              head = tail = appendNode;
49          } else {
50              tail->next = appendNode;
51              tail = appendNode;
52          }
53          size++;
54      }
```

PUSH

노드 생성 후 값 할당

비어있는 경우
head와 tail 모두 새로운 노드로

비어있지 않은 경우
tail의 next를 새로운 노드로
tail 업데이트

size증가

3. 링리 함수 구현

링리를 사용한 스택

```
56      // 스택이라 끝을 삭제하는 경우만 존재
57      int deleteNode() {
58          int deletedValue = tail->value;
59
60          // 한 개만 있는 경우
61          if (size == 1) {
62              delete head;
63              head = tail = nullptr;
64          } else {
65              Node *curNode = head;
66              while(curNode->next != tail) { // tail의 바로 이전 노드 탐색
67                  curNode = curNode->next;
68              }
69              delete curNode->next;
70              curNode->next = nullptr;
71              tail = curNode;
72          }
73          size--;
74
75          return deletedValue;
76      }
```

링리를 사용한 스택

4. 스택 구현

```
81  class Stack{
82  private:
83      LinkedList ll;
84      int size;
85
86  public:
87      Stack() {
88          ll = LinkedList();
89          size = 0;
90      }
91      bool empty() { // 비어있으면 1 출력
92          return size==0;
93      }
```

5. 스택 함수 구현

링리를 사용한 스택

TOP

```
cout << ll.tail->value << "\n";
```

if (비어있는 경우) {
    // 문제 조건따라 에러처리
}
tail의 value값 출력 또는 리턴

PUSH

```
ll.append(x);
size++;
```

링리의 append함수 실행
사이즈 증가

POP

```
cout << ll.deleteNode() << "\n";
size--;
```

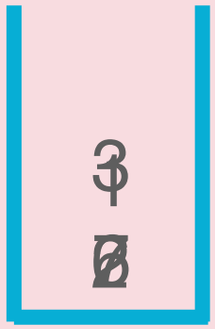
if (비어있는 경우) {
    // 문제 조건따라 에러처리
}
링리의 delete함수 실행
사이즈 감소

스택 응용

1. 후위표기식 계산
2. 중위표기 -> 후위표기
3. 괄호문 체크

응용 1 : 후위 표기식 계산

23*1+



6+1=7
2*3=6

후위표기식은 이미 우선순위에 맞게 순서가 짜여져있음
앞에 나오는 연산자부터 단순히 계산해주기

Solve)

입력받는 수식 한글자씩 순회하면서 (1~3)반복

1. 피연산자 만나면 스택에 push
2. 연산자 만나면 스택에서 피연산자 두개씩 꺼내서 계산
3. 계산해서 나온 결과 다시 스택에 push

유형 2

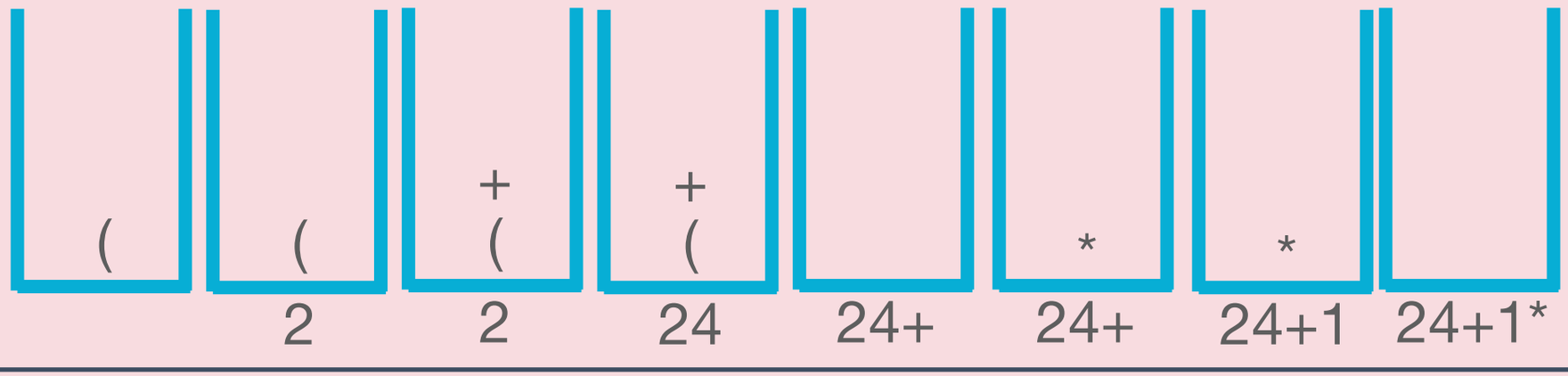
응용 2 : 후위 표기식 변환

괄호 안에 있는 연산자를 가장 먼저 출력
우선순위가 더 높은 연산자가 먼저 출력되어야함
피연산자는 바로 출력

Solve)

1. '('는 스택에 push
2. ')'가 나오면 스택에서 '('가 나올 때까지 pop하여 출력
3. '('는 출력하지 않고 버림
4. 연산자 만나면 스택에서 그 연산자보다 낮은 연산자 나올 때 까지 pop하여 출력하고 자신을 push
5. 피연산자는 그냥 출력
6. 모든 과정 끝나면 스택 내 모든 연산자 출력

(2+4)*1



24+1*

유형 3

응용 3 : 괄호

괄호로 이루어진 문자열이 올바른 문자열인지 확인하는 문제

여는 괄호 '('와 닫는 괄호')'의 수는 같아야 한다.
여는 괄호 '('보다 닫는 괄호')'가 앞에 나오면 안된다.

Solve)

여는 괄호가 나오면 스택에 push

닫는 괄호가 나오면 스택에서 여는 괄호 하나를 뺌
* 닫는 괄호 나왔는데 스택이 비어있으면 에러

문자열의 끝까지 검사했을 때 스택이 비어있으면 정상
* 끝까지 검사했는데 스택에 괄호 남아있으면 에러

백준으로 갑시다!

—

스택
Stack