

DataStructure Seminar Chapter.5

STACK

W I S O F T

박 대 원

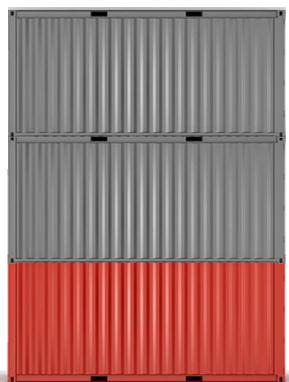
I 스택(Stack)

스택의 사전적 의미

동사 : 쌓다 명사 : 더미

자료구조에서의 스택

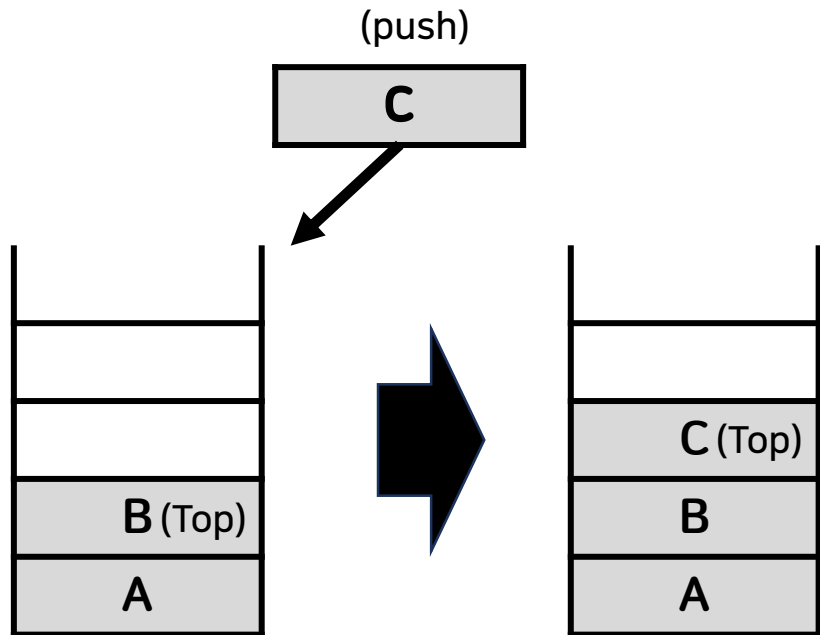
LIFO(Last—In—First—Out, 리포) : 가장 나중에 들어간 자료가 가장 먼저 나온다. = 후입선출
가장 먼저 추가한 자료가 가장 나중에 가져올 수 있다고 해서 FILO, 선입후출 이라고도 한다.



컨테이너 더미에서 하나를 꺼낸다면 당연히 맨 위에 있는 컨테이너이다.
위에 있는 컨테이너부터 꺼내지 않으면 아래에 있는 컨테이너에는 접근조차 못하기 때문이다.

이처럼 스택은 실제 현실 세계를 정확하게 표현하기 위해서 사용한다.
또한, 후입선출의 특성은 다양한 알고리즘에서 필수적인 요소이다.

| 스택의 자료 추가하기



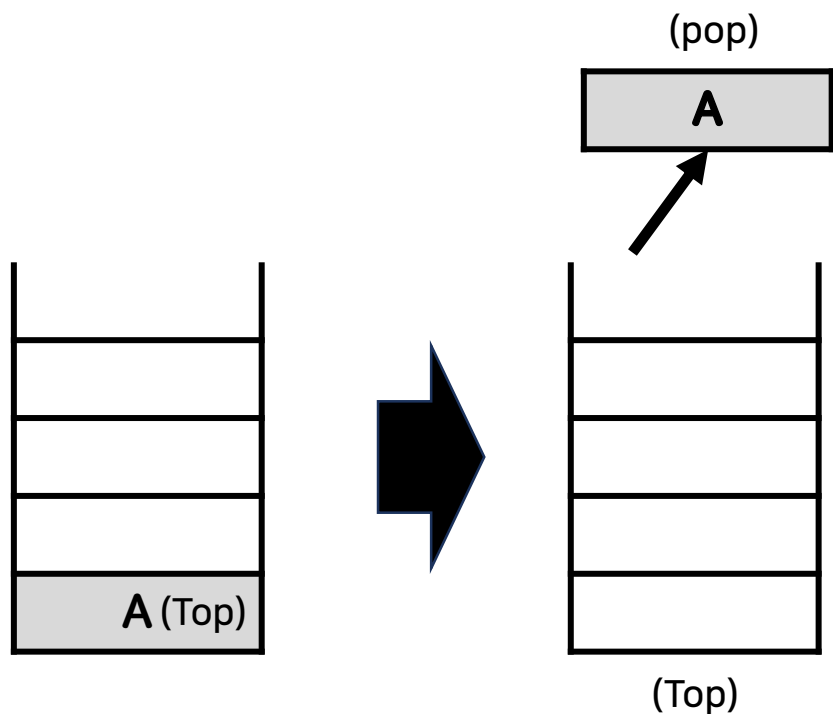
푸시(push) 연산 : 새로운 자료를 스택에 추가하는 과정.
이 연산은 스택의 맨 위에서만 수행된다.

탑(Top) : 스택에서 은 항상 최신 자료를 카리키는 끝이나 맨 위.
푸시 연산을 구현하면서 탑을 어떻게 변경할지 주의.

스택의 크기 : 스택이 저장할 수 있는 최대 자료의 개수.

넘침(overflow) :
스택의 크기를 초과해 새로운 자료를 추가하지 못하는 현상

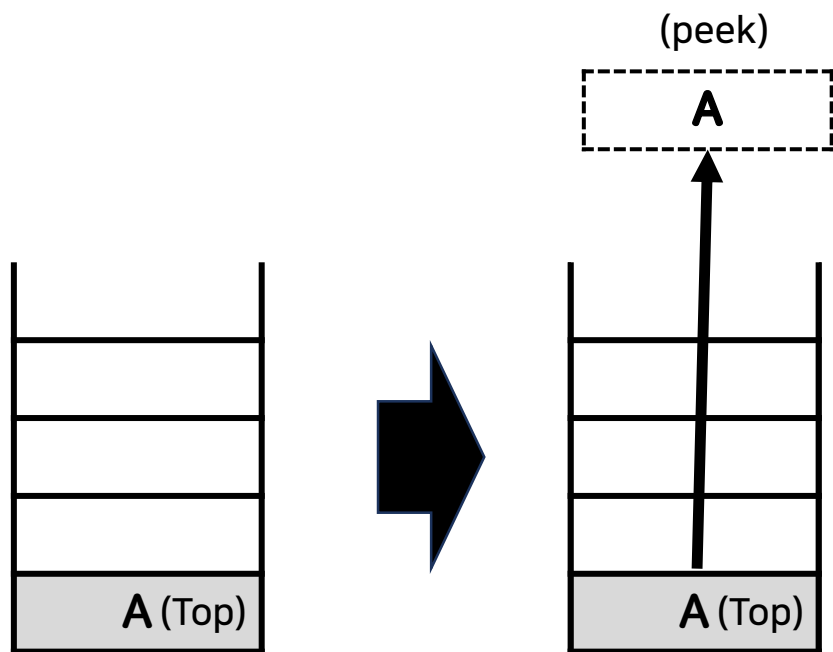
| 스택의 자료 가져오기



팝(pop)연산 : 스택에서 자료를 가져오는 연산.
스택에서는 자료를 제거한 뒤 가져온다.
푸시 연산과 마찬가지로 스택의 맨 위에서만 수행.

부족(underflow) : 아무 자료가 없는 빈(empty) 스택에서
팝 연산을 수행할 경우 제거할 자료가 없어서
아무 자료도 반환하지 못하는 현상.

| 스택의 자료 가져오기



피크(peek)연산 : 팝 연산과 마찬가지로 스택에서 자료를 가져오는 연산.
하지만, 기존 스택에서 자료를 제거하지는 않음.
탑(Top)을 이용하여 스택의 맨 위 자료를 반환

I 스택의 추상 자료형

스택의 구현에 필요한 기본 연산을 정리한 추상 자료형

이름		입력	출력	설명
스택 생성	createStack()	스택의 크기 n	스택	빈 스택을 생성
스택 삭제	deleteStack()	스택	N/A	스택의 메모리를 해제
자료 추가 가능 여부 판단	isFull()	스택	True/False	스택에 푸시를 수행할 수 있는지를 반환 단, 배열 스택인 경우
빈 스택인지 여부 판단	isEmpty()	스택	True/False	빈 스택인지를 반환
푸시	push()	스택	성공/실패 여부	스택의 맨 위에 새로운 자료를 추가
팝	pop()	자료	자료	스택의 맨 위에 있는 자료를 제거한 뒤 이를 반환
피크	peek()	스택	자료	스택의 맨 위에 있는 자료를 반환 (제거하지 않음)

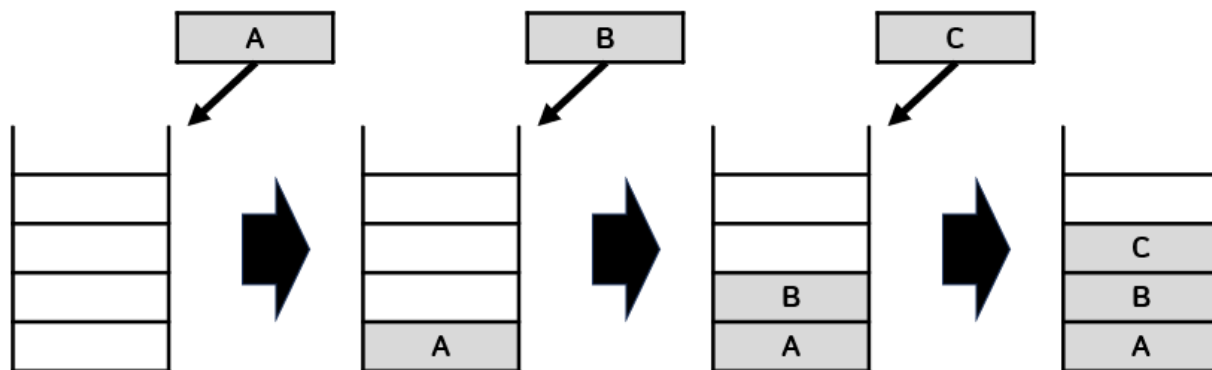
| 스택의 적용 1. 역순인 문자열 만들기

'ABC'의 문자열

'ABC'의 역순인 문자열 'CBA'

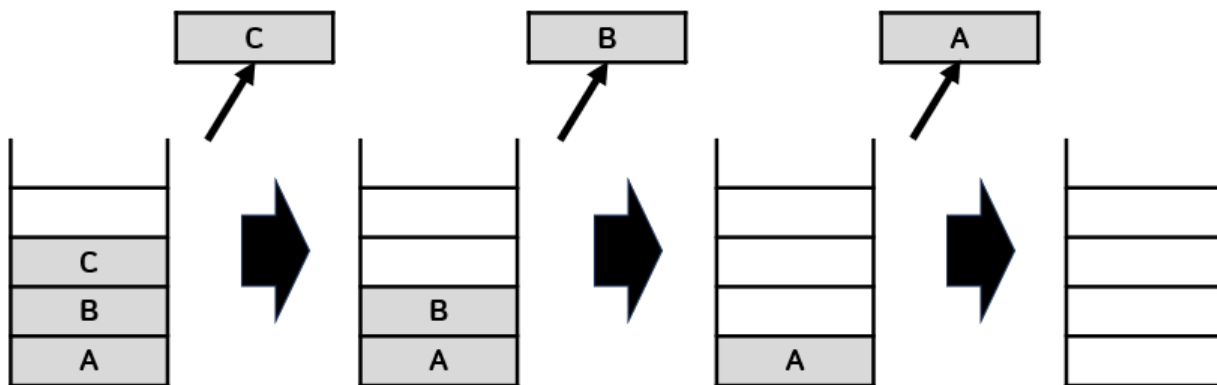
$A \rightarrow B \rightarrow C$

$C \rightarrow B \rightarrow A$



Step-A. 푸시하기

입력 문자열의 문자를 순서대로
모두 스택에 푸시.



Step-B. 팝하기

빈 스택이 될 때까지 스택에서 문자를
팝하기.

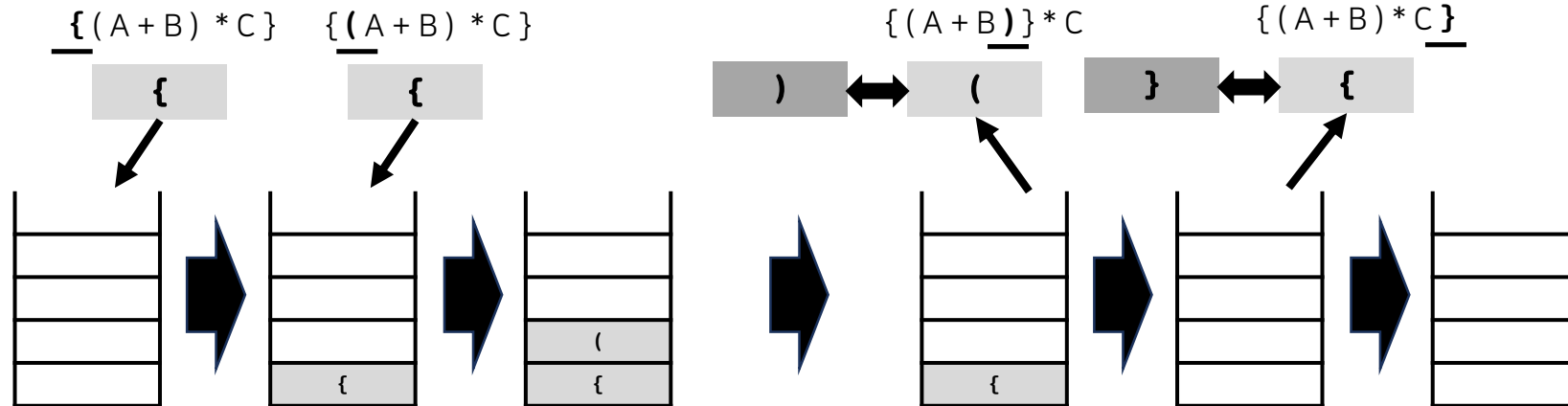
| 스택의 적용 2. 수식에서 괄호 검사하기

입력받은 수식에서 괄호의 쌍이 맞는지 검사.

여는 괄호를 만나면 → 푸시

닫는 괄호를 만나면 → 팝해서 괄호의 종류를 조사

올바른 예
$(A + B) * C$
$\{(A + B) * C\}$
$\{(A + B)\} * C$





I 스택의 적용 2. 수식에서 괄호 검사하기 : 의사코드

```
checkBracketMatching(expression) {  
    result <- 성공  
    while (expression이 끝이 아닌 경우 && result != 오류) {  
        symbol <- expression의 다음 글자  
        switch(symbol) {  
            case '(' case '[': case '{' :  
                symbol을 스택에 푸시  
                break;  
            case ')' case ']' case '}' :  
                if (스택이 비어 있는 상태) {  
                    result <- 오류  
                }  
                else {  
                    checkSybol <- 스택에서 팝  
                    if (symbol과 checkSybol이 쌍이 맞지 않는 경우) {  
                        result <- 오류  
                    }  
                }  
                break;  
        }  
        if (스택이 비어 있지 않다면) {  
            result <- 오류  
        }  
    }  
    return result  
}
```

| 스택의 적용 3. 후위 표기법으로 수식 계산하기

$A + B * C$

① 중위 표기법

1단계 : $B * C$

2단계 : $A + (B * C)$

$A B C * +$

② 후위 표기법

1단계 : **A** B C * + : 피연산자 A를 읽음

1단계 : A **B** C * + : 피연산자 B를 읽음

1단계 : A B **C** * + : 피연산자 C를 읽음

4단계 : A B C ***** + : 연산자 * 앞의 두 피연산자 B와 C를 곱한다. 즉, $B * C$

5단계 : A B C * **+** : 연산자 + 앞의 두 피연산자 A와 $(B * C)$ 를 더한다. 즉, $A + (B * C)$

| 스택의 적용 3. 후위 표기법으로 수식 계산하기

$A - (B + C) * D$

① 중위 표기법

1단계 : $B + C$

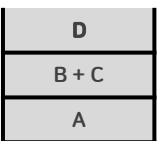
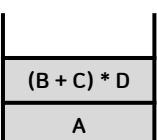
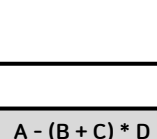
2단계 : $(B + C) * D$

3단계 : $A - (B + C) * D$

$A B C + D * -$

② 후위 표기법

단계		처리	스택의 상태
1	$A B C + D * -$	피연산자 A를 스택에 푸시	
2	$A B C + D * -$	피연산자 B를 스택에 푸시	
3	$A B C + D * -$	피연산자 C를 스택에 푸시	

단계		처리	스택의 상태
4	A B C + D * -	연산자 + 처리를 위해 피연산자 두 개(C, B)를 스택에서 팝	
		계산 결과인 B + C를 스택에 푸시	
5	A B C + D * -	피연산자 D 를 스택에 푸시	
6	A B C + D * -	연산자 * 처리를 위해 피연산자 두 개 D와 B+C를 스택에서 팝	
		계산 결과인 (B + C) * D를 스택에 푸시	
7	A B C + D * -	연산자 - 처리를 위해 피연산자 두 개 A와 (B + C) * D를 스택에서 팝	
		계산 결과인 A - (B + C) * D를 스택에 푸시	

| 스택의 적용 3. 후위 표기법으로 수식 계산하기 : 의사 코드

후위 표기법으로 표현된 수식의 연산에 적용되는 규칙

1. 피연산자를 만나면 스택에 푸시한다.
2. 연산자를 만나면 연산에 필요한 개수만큼(보통 2개) 피연산자를 스택에서 팝한다.
3. 계산 결과는 다시 스택에 푸시한다.

```
calcExpression (expression) {  
    while(토큰 in expression) {                                //토큰 = 피연산자와 연산자  
        if(토큰 == 피연산자) {  
            토큰을 스택에 푸시  
        }  
        else if (토큰 == 연산자) {  
            피연산자2 <- 스택에서 팝  
            피연산자1 <- 스택에서 팝  
            결과 <- 피연산자1 연산자 피연산자2  
            결과를 스택에 푸시  
        }  
    }  
    result <- 스택에서 팝  
    return result  
}
```

| 스택의 적용 4. 중위 표기 수식을 후위 표기 수식으로 변환하기

$A * (B + C) \rightarrow A B C + *$

//실제로 이전의 함수를 이용하려면 이처럼 변환이 필요하다.

단계	처리	출력 내용	스택의 상태
1	$A * (B + C)$ 피연산자 A를 출력	A	
2	$A * (B + C)$ 연산자 * 를 스택에 푸시	A	
3	$A * (B + C)$ 연산자 (를 스택에 푸시	A	
4	$A * (B + C)$ 피연산자 B를 출력	A B	

단계		처리	출력 내용	스택의 상태
5	$A * (B + C)$	연산자 + 를 스택에 푸시	A B	<div><div>+</div><div>(</div><div>*</div></div>
6	$A * (B + \mathbf{C})$	피연산자 C를 출력	A B C	<div><div>+</div><div>(</div><div>*</div></div>

연산자	High <- 우선순위 -> Low			
스택 내부)	* /	+ -	(
스택 외부) (* /	+ -	

단계		처리	출력 내용	스택의 상태
7	$A * (B + C)$	연산자 (를 만날 때까지 팝	A B C +	<div><div></div><div></div><div>*</div></div>
6	< 종료 >	스택에서 남은 토큰들 팝	A B C + *	<div><div></div><div></div><div></div></div>

I 스택의 적용 4. 중위 표기 수식을 후위 표기 수식으로 변환하기

중위 표기법을 후위 표기법으로 변환하는 규칙

1. 피연산자를 만나면 바로 출력한다.
2. 연산자를 만나면 일단 스택에 저장한다.
3. 단, 스택에 저장중인 연산자 중에서 우선순위가 높은 연산자는 팝하여 출력한다.
4. (주의) 스택의 내부와 외부에서의 연산자 우선순위는 다르다.
5. 닫는 괄호 연산자) 를 만나면 스택에서 여는 괄호 연산자 (를 만날 때까지 스택에 저장된 연산자들을 모두 팝하여 이를 출력한다.

감사합니다