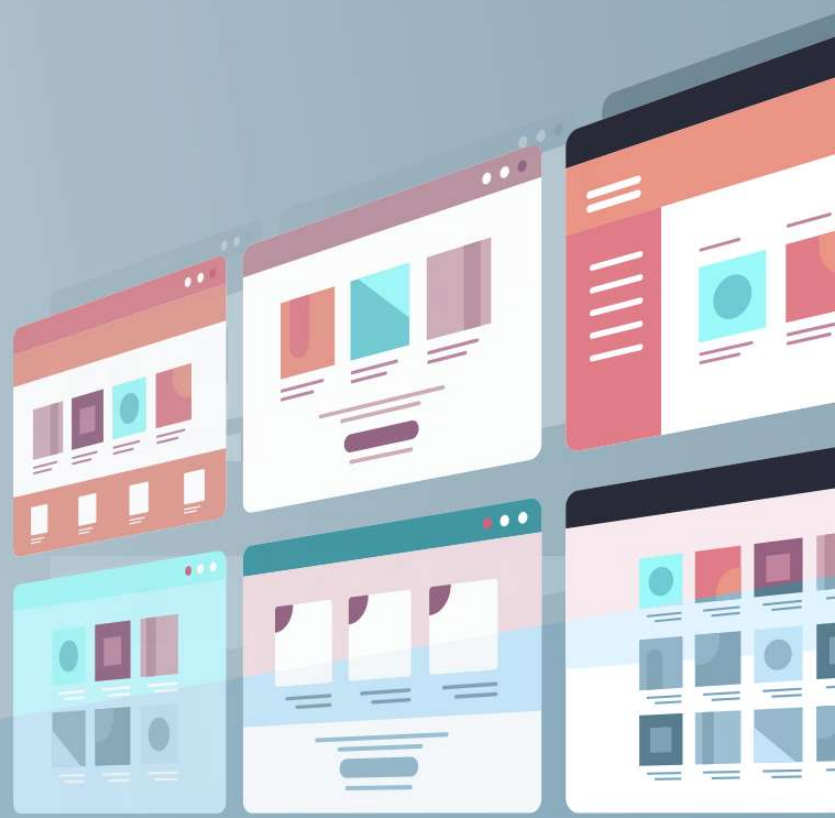
03

자료구조

스택

컴퓨터과학과 정광식 교수

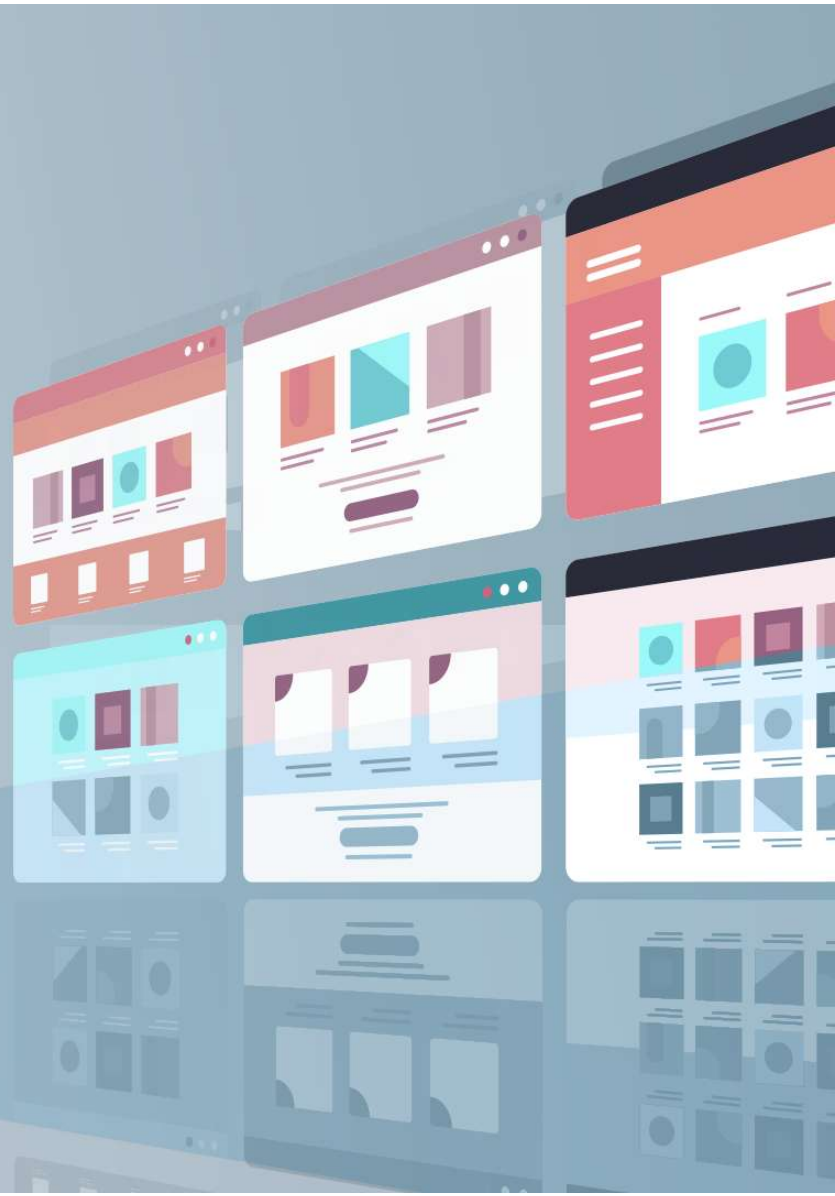


KOREA NATIONAL OPEN UNIVERSITY

학습목차

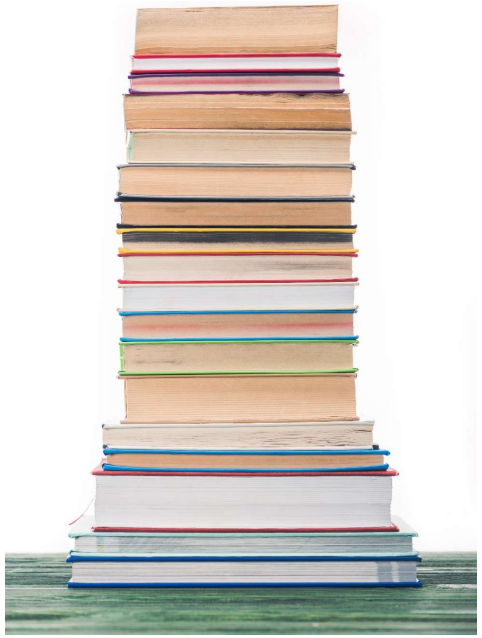
01

스택의 개념



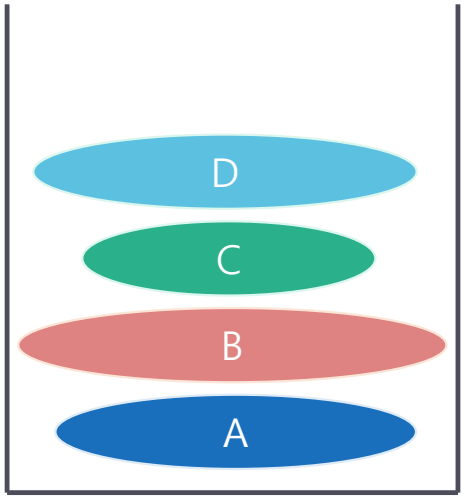
1. 스택의 개념

스택의 정의



1. 스택의 개념

스택의 정의



1. 스택의 개념

스택의 정의

- 객체와 그 객체가 저장되는 순서를 기억하는 방법에 관한 자료구조
  - ✓ 가정 먼저 입력된 자료가 가장 나중에 출력되는 관계를 표현함
  - ✓ 관계를 표현하기 위해서 연산이 필요하며, 객체에 대한 정의와 연산이 모여서 순서가 기억되는 스택의 추상 자료형이 완성됨

1. 스택의 개념

스택의 정의

- 0개 이상의 원소를 갖는 유한 순서 리스트
- push(add)와 pop(delete)연산이 한곳에서 발생되는 자료구조

02

스택의 추상 자료형

2. 스택의 추상자료형

스택의 추상자료형

- 스택 객체: 0개 이상의 원소를 갖는 유한 순서 리스트

1. 스택의 추상자료형

CreateStack 연산

- 연산: stack ∈ Stack, item ∈ element,
  maxStackSize ∈ positive integer인 모든 stack, item,
  maxStackSize에 대하여 다음과 같은 연산이 정의된다
  (stack은 0개 이상의 원소를 갖는 스택, item은 스택에 삽입되는 원소,
  maxStackSize는 스택의 최대 크기를 정의하는 정수).

- Stack CreateStack(maxStackSize) ::=
  스택의 크기가 maxStackSize인 빈 스택을 생성하고 반환한다;

2. 스택의 추상자료형

Push 연산

- Stack Push(stack, item) ::=

  if (StackIsFull(stack))

  then { 'stackFull'을 출력한다; }

  else { 스택의 가장 위에 item을 삽입하고, 스택을 반환한다; }

2. 스택의 추상자료형

Pop 연산

- Element Pop(stack) ::=

  if (StackIsEmpty(stack))

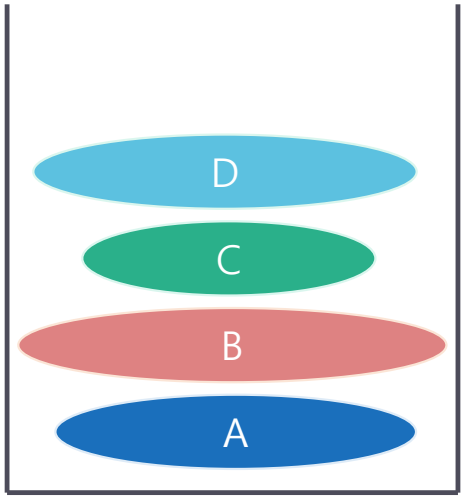
  then { 'stackEmpty'을 출력한다; }

  else { 스택의 가장 위에 있는 원소(element)를

  삭제하고 반환한다; }

1. 스택의 개념

스택의 정의



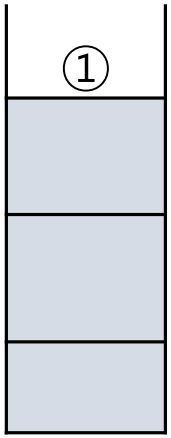
2. 스택의 추상자료형

Pop/Push 연산의 실행

① CreateStack(3);
② Push(stack, 'S');
③ Push(stack, 'T');
④ Pop(stack);
⑤ Push(stack, 'R');
⑥ Push(stack, 'P');
⑦ Push(stack, 'Q');
⑧ Pop(stack);

2. 스택의 추상자료형

- Pop/Push 연산의 실행

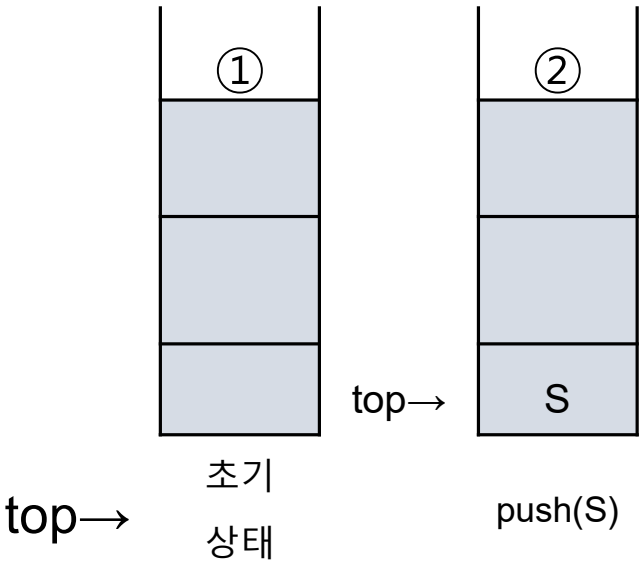


초기 상태

① CreateStack(3);
② Push(stack, 'S');
③ Push(stack, 'T');
④ Pop(stack);
⑤ Push(stack, 'R');
⑥ Push(stack, 'P');
⑦ Push(stack, 'Q');
⑧ Pop(stack);

2. 스택의 추상자료형

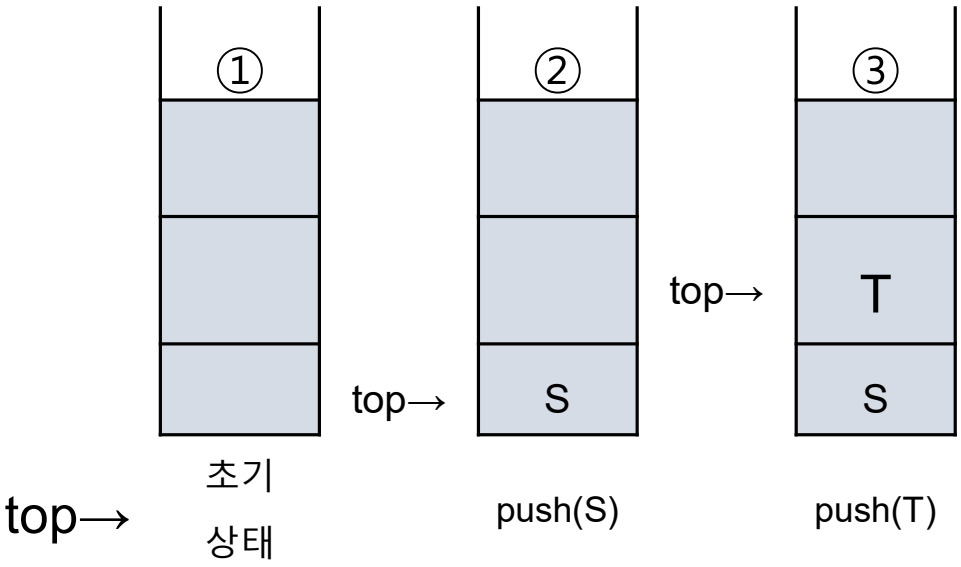
- Pop/Push 연산의 실행



① CreateStack(3);
② Push(stack, 'S');
③ Push(stack, 'T');
④ Pop(stack);
⑤ Push(stack, 'R');
⑥ Push(stack, 'P');
⑦ Push(stack, 'Q');
⑧ Pop(stack);

2. 스택의 추상자료형

- Pop/Push 연산의 실행

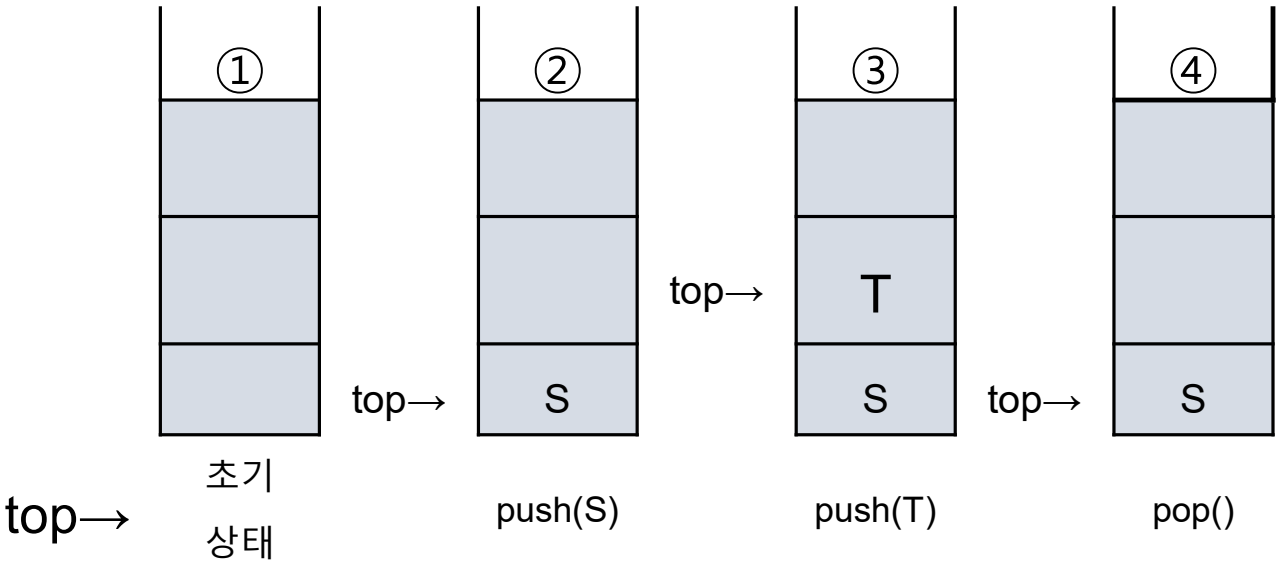


① CreateStack(3);
② Push(stack, 'S');
③ Push(stack, 'T');
④ Pop(stack);
⑤ Push(stack, 'R');
⑥ Push(stack, 'P');
⑦ Push(stack, 'Q');
⑧ Pop(stack);

2. 스택의 추상자료형

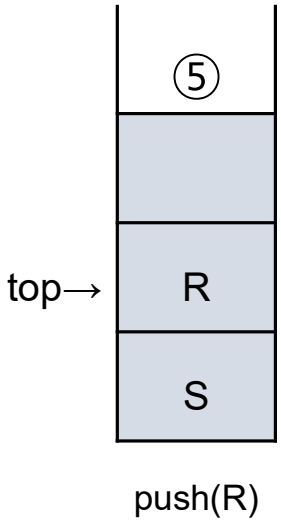
- Pop/Push 연산의 실행

① CreateStack(3);
② Push(stack, 'S');
③ Push(stack, 'T');
④ Pop(stack);
⑤ Push(stack, 'R');
⑥ Push(stack, 'P');
⑦ Push(stack, 'Q');
⑧ Pop(stack);



2. 스택의 추상자료형

Pop/Push 연산의 실행

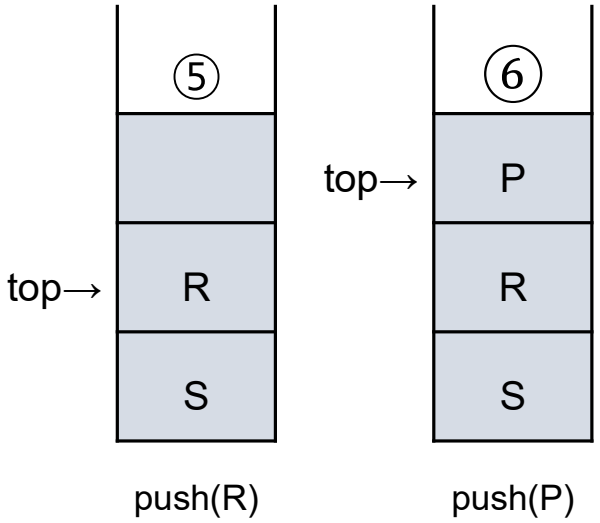


push(R)

① CreateStack(3);
② Push(stack, 'S');
③ Push(stack, 'T');
④ Pop(stack);
⑤ Push(stack, 'R');
⑥ Push(stack, 'P');
⑦ Push(stack, 'Q');
⑧ Pop(stack);

2. 스택의 추상자료형

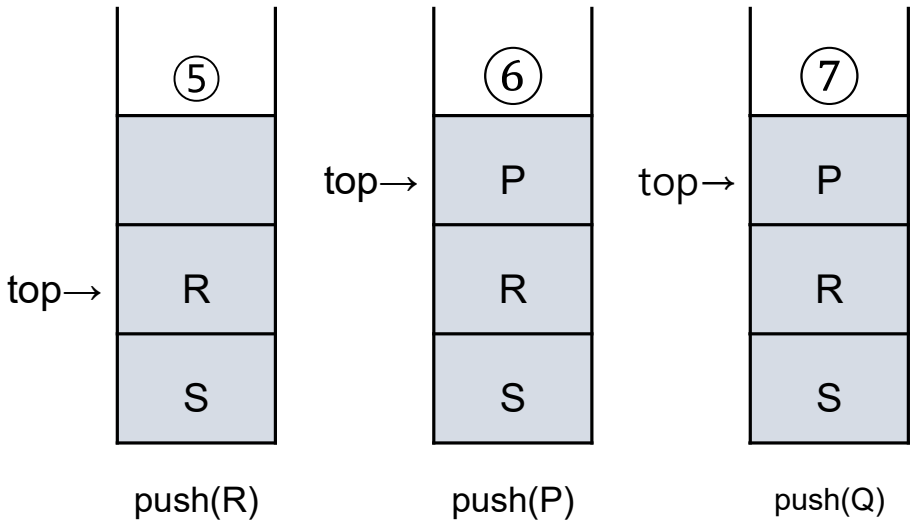
Pop/Push 연산의 실행



① CreateStack(3);
② Push(stack, 'S');
③ Push(stack, 'T');
④ Pop(stack);
⑤ Push(stack, 'R');
⑥ Push(stack, 'P');
⑦ Push(stack, 'Q');
⑧ Pop(stack);

2. 스택의 추상자료형

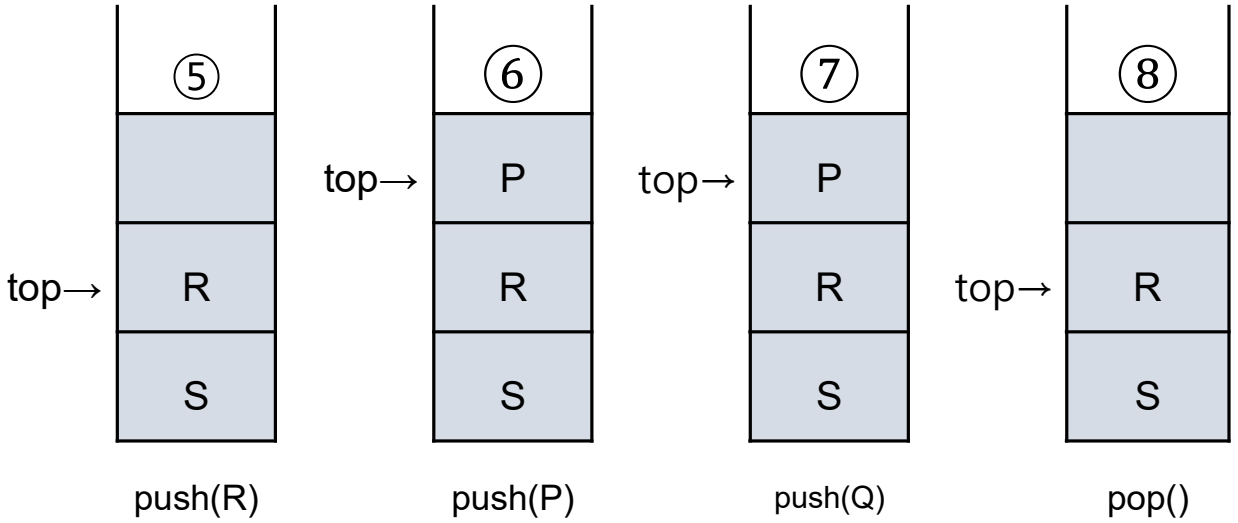
Pop/Push 연산의 실행



① CreateStack(3);
② Push(stack, 'S');
③ Push(stack, 'T');
④ Pop(stack);
⑤ Push(stack, 'R');
⑥ Push(stack, 'P');
⑦ Push(stack, 'Q');
⑧ Pop(stack);

2. 스택의 추상자료형

Pop/Push 연산의 실행



① CreateStack(3);
② Push(stack, 'S');
③ Push(stack, 'T');
④ Pop(stack);
⑤ Push(stack, 'R');
⑥ Push(stack, 'P');
⑦ Push(stack, 'Q');
⑧ Pop(stack);

2. 스택의 추상자료형

StackIsFull·StackIsEmpty연산

- Boolean StackIsFull(stack, maxStackSize) ::=

  if((stack의 elements의 개수) == maxStackSize)

  then { 'TRUE' 값을 반환한다; }

  else { 'FALSE' 값을 반환한다; }

- Boolean StackIsEmpty(stack) ::=

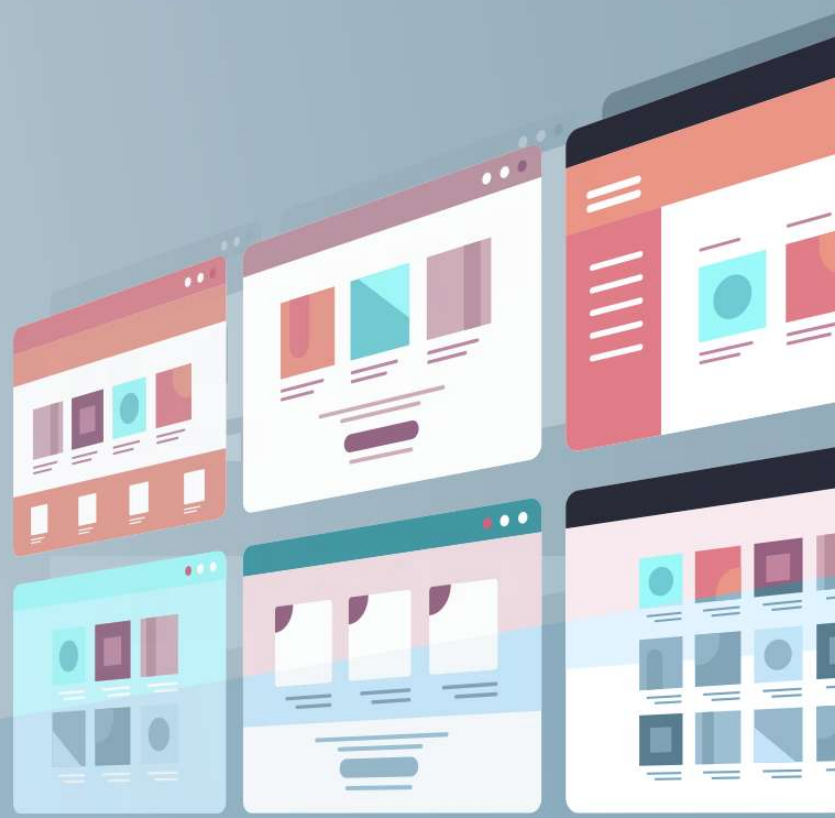
  if(stack == CreateStack(maxStackSize))

  then { 'TRUE' 값을 반환한다; }

  else { 'FALSE' 값을 반환한다; }

03

스택의 응용



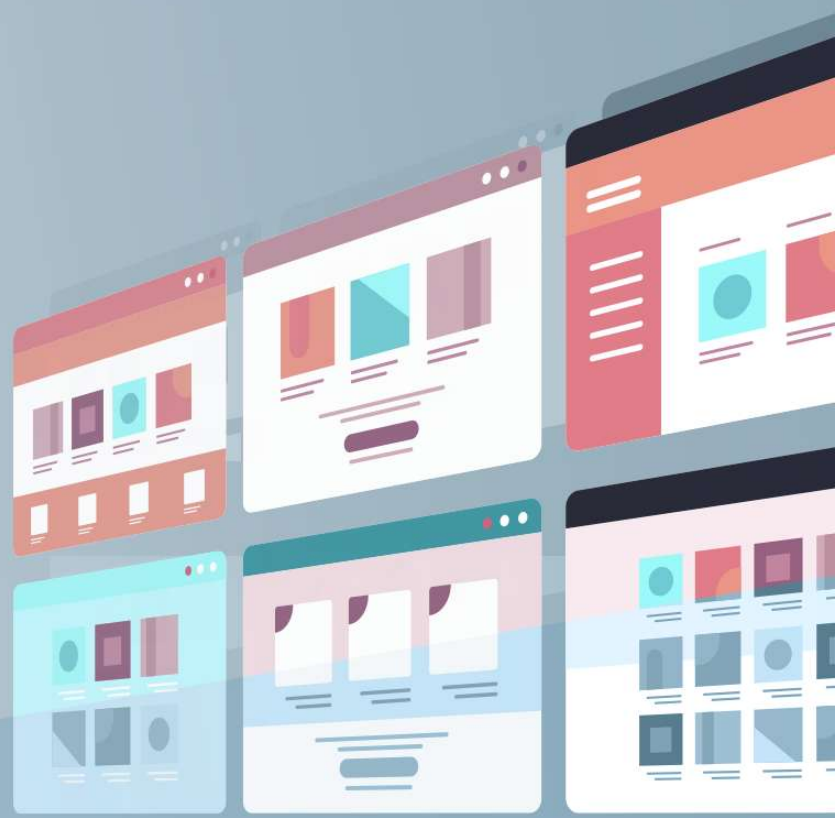
3. 스택의 응용

스택의 다양한 응용

- 변수에 대한 메모리의 할당과 수집을 위한 시스템 스택
- 서브루틴 호출 관리를 위한 스택
- 연산자들 간의 우선순위에 의해 계산 순서가 결정되는 **수식 계산**
- 인터럽트의 처리와 되돌아갈 명령 수행 지점을 저장하기 위한 스택
- 컴파일러, 순환 호출 관리

04

스택의 연산



4. 스택의 연산

스택의 삭제 연산

- 'top--'에서 사용된 '--' 연산자의 위치에 따라 연산의 적용순서가 달라질 수 있음

```
a-- ==  a - 1
--a ==  a - 1
```

```
int a, b;
a = 5;
b = a--;
```

```
b = 5
a = 4
```

```
int a, b;
a = 5;
b = --a;
```

```
b = 4
a = 4
```

4. 스택의 연산

스택의 생성

```
#define STACK_SIZE 10
typedef int element;
element stack[STACK_SIZE];
int top = -1;
```

4. 스택의 연산

스택의 삭제 연산

```
① int pop( ) {
②       if (top == -1) {
③           return StackIsEmpty( );
④         else return stack[top--]; }
```

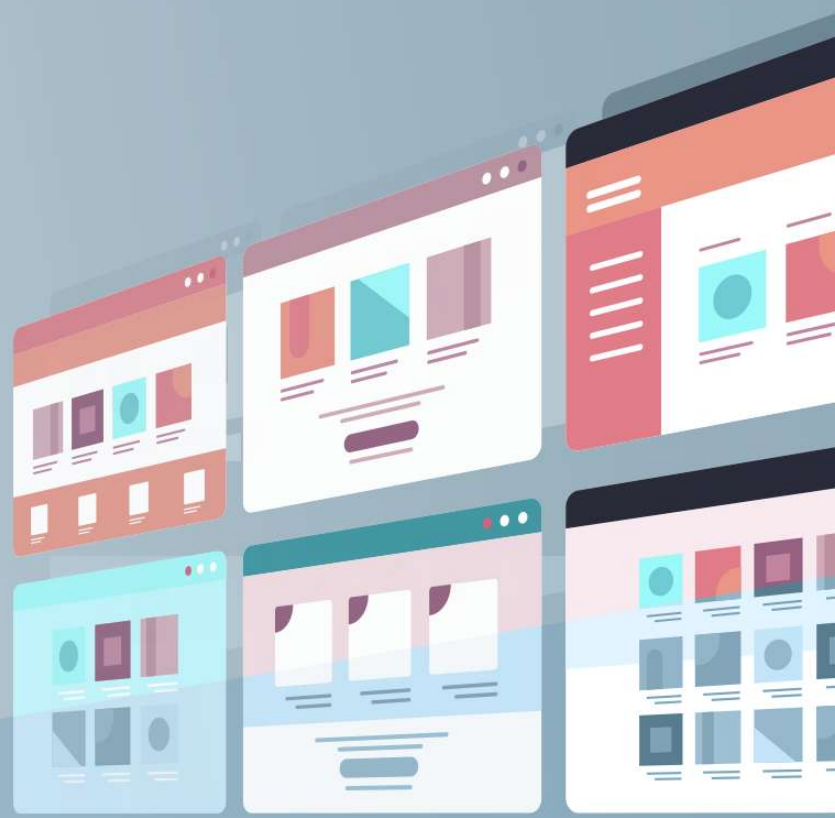
4. 스택의 연산

스택의 삽입 연산

```
① void push(int item) {
②       if (top >= STACK_SIZE -1)
③             return StackIsFull( );
④         else stack[++top] = item; }
```

05

사칙연산의 전위•후위•중위 표현



5. 사칙연산의 전위 • 후위 • 중위 표현

수식의 계산

- 연산자의 계산 우선순위를 생각해야 함

- A + B * C + D
  - ✓ ((A + (B * C)) + D)

5. 사칙연산의 전위 • 후위 • 중위 표현

수식의 표기 방법

- 중위 표기법(infix notation)
  - ✓ 연산자를 피연산자 사이에 표기하는 방법
  - ✓ A + B

- 전위 표기법(prefix notation)
  - ✓ 연산자를 피연산자 앞에 표기하는 방법
  - ✓ + A B

- 후위 표기법(postfix notation)
  - ✓ 연산자를 피연산자의 뒤에 표기하는 방법
  - ✓ A B +

5. 사칙연산의 전위·후위·중위 표현

후위 표기법

A + B => AB +

- (A -((B+K)/D))
  - ⇨ (A -((BK+) / D))
    - ⇨ (A- ((BK+)D/))
      - ⇨ A ((BK+)D/) -

5. 사칙연산의 전위 • 후위 • 중위 표현

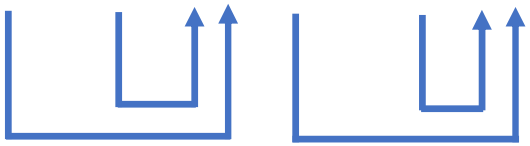
중위 표기식의 후위 표기식 변환 방법

- 먼저 중위 표기식을 연산자의 우선순위를 고려하여 (피연산자, 연산자, 피연산자)의 형태로 괄호로 묶어줌
- 각 계산뭉치를 묶고 있는 괄호 안에서 연산자를 계산뭉치의 가장 오른쪽으로 이동시킴
- 각 계산뭉치를 하나의 피연산자로 고려하여 위를 반복함
- 괄호를 모두 제거함

5. 사칙연산의 전위 • 후위 • 중위 표현

중위 표기식의 후위 표기식 변환 방법

- ((A - (B / C)) - (D * E))



- ((A (B C /) -) (D E *) -)

5. 사칙연산의 전위·후위·중위 표현

후위 표기식의 계산 알고리즘

```
// 후위 표기식(369*+)을 계산하는 연산
element evalPostfix(char *exp) {
    int oper1, oper2, value, i=0;
    int length = strlen(exp);
    char symbol;
    top = -1;
```

5. 사칙연산의 전위·후위·중위 표현

후위 표기식의 계산 알고리즘

```
for(i=0; i<length; i++) {
  symbol = exp[i];
  if(symbol != '+' && symbol != '-'
        && symbol != '*' && symbol != '/') {
            value = symbol - '0';
            push(value);   }
```

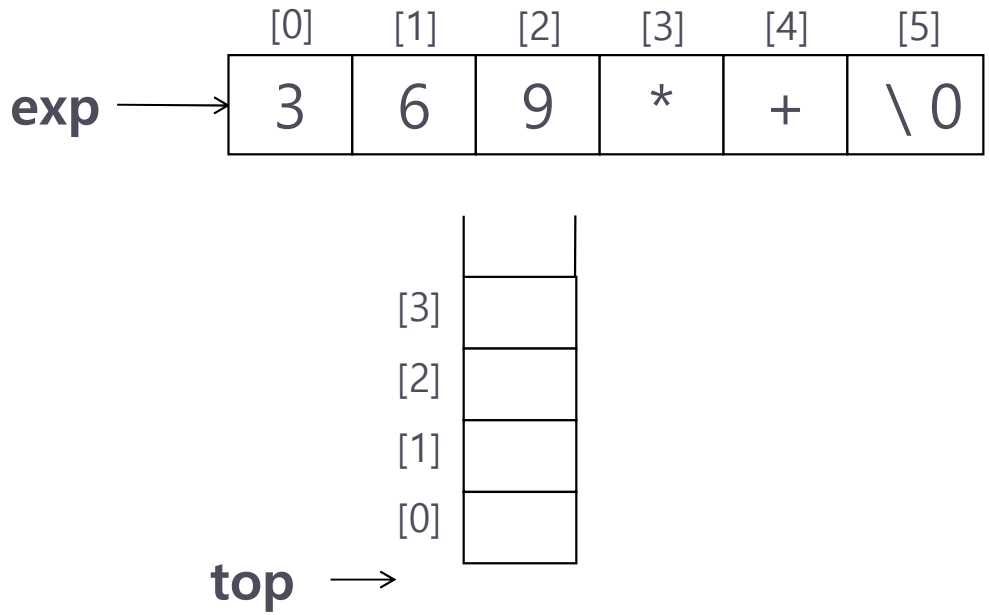
5. 사칙연산의 전위·후위·중위 표현

후위 표기식의 계산 알고리즘

```
else { oper2 = pop( );  oper1 = pop( );
  switch(symbol) {
    case '+' : push(oper1 + oper2); break;
    case '-' : push(oper1 - oper2); break;
    case '*' : push(oper1 * oper2); break;
    case '/' : push(oper1 / oper2); break;
  }   } }
return pop( );                }
```

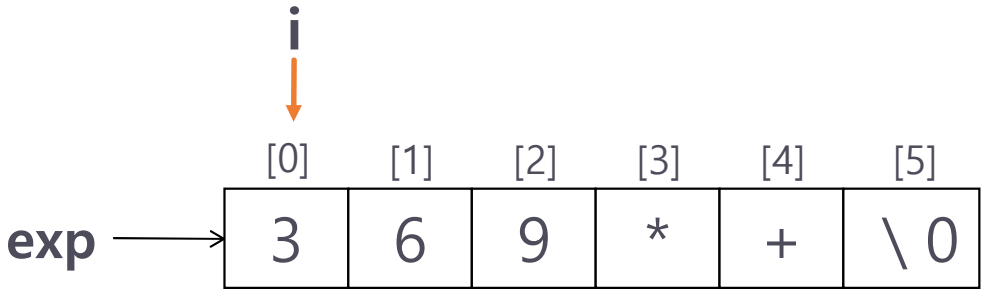
5. 사칙연산의 전위 • 후위 • 중위 표현

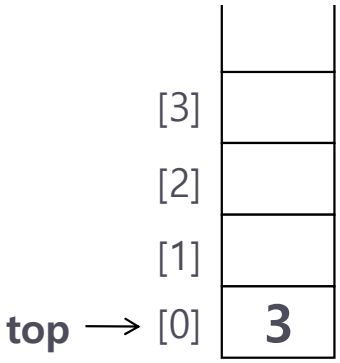
- 수식이 저장된 배열과 스택의 초기 모습



5. 사칙연산의 전위 • 후위 • 중위 표현

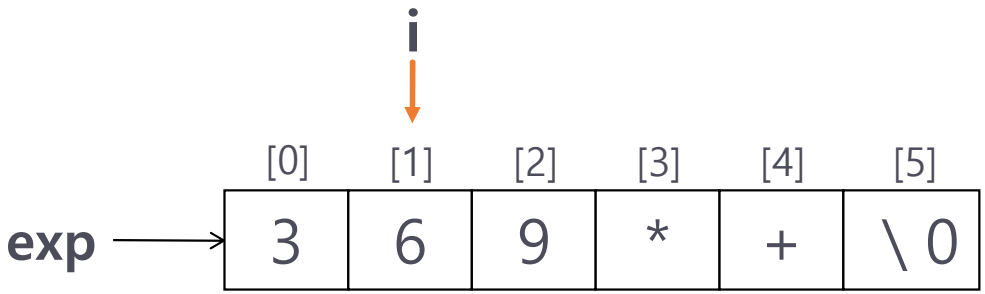
첫번째 피연산자 계산

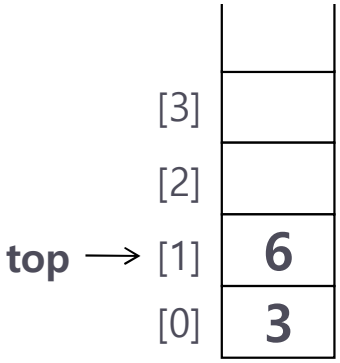




5. 사칙연산의 전위 • 후위 • 중위 표현

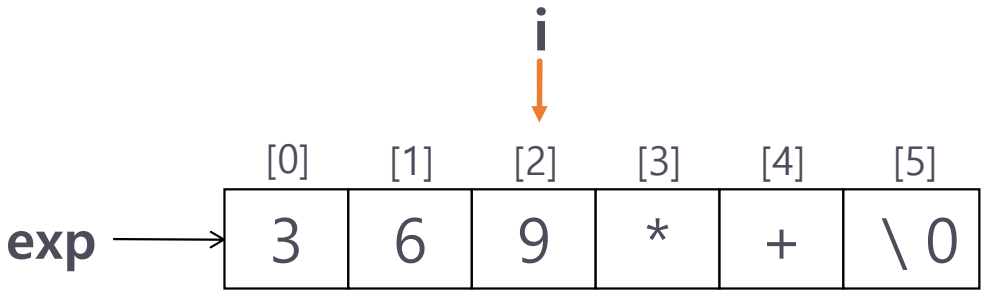
두번째 피연산자 계산

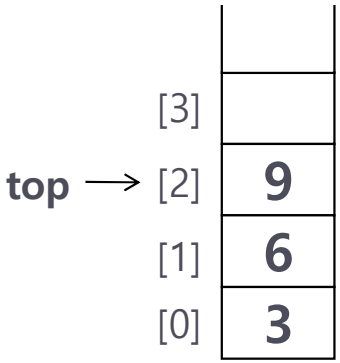




5. 사칙연산의 전위 • 후위 • 중위 표현

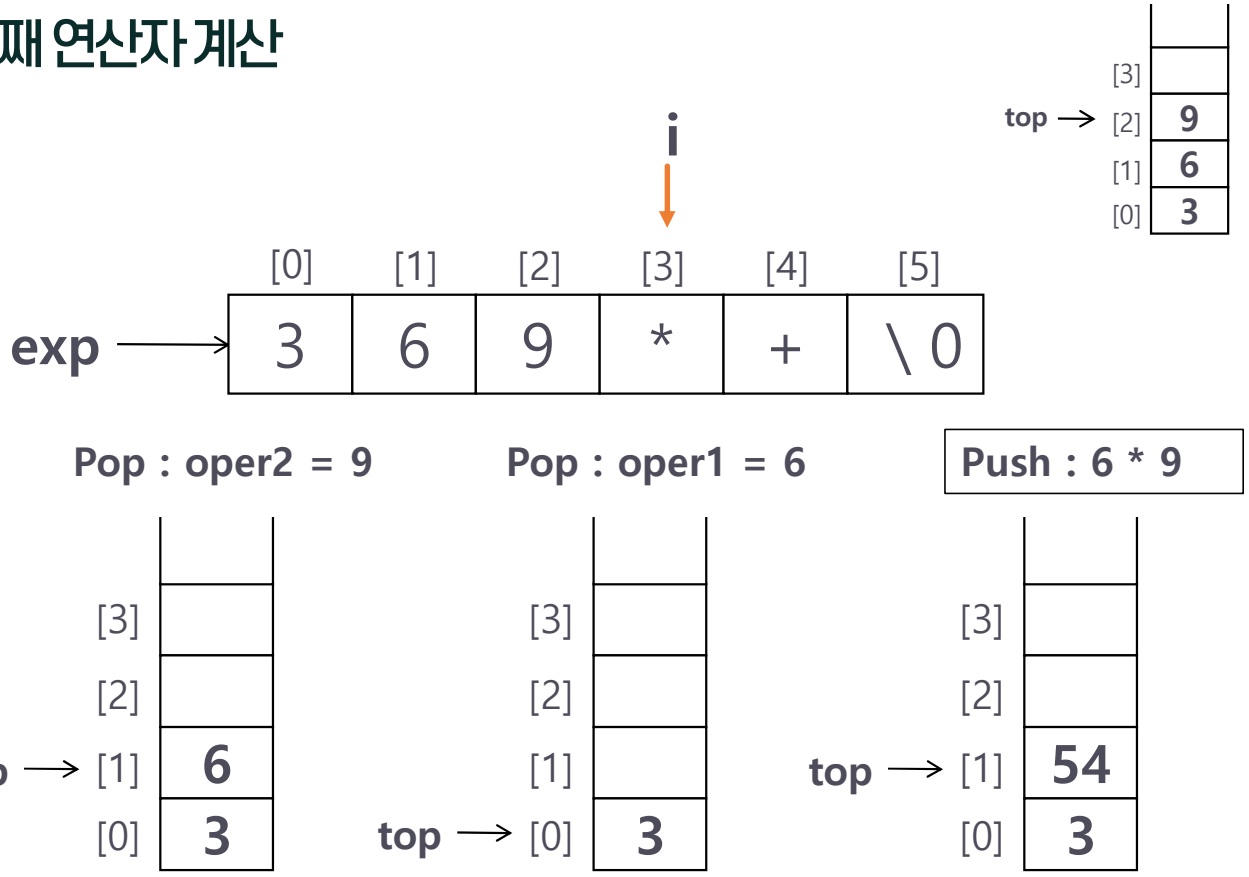
세번째 피연산자 계산





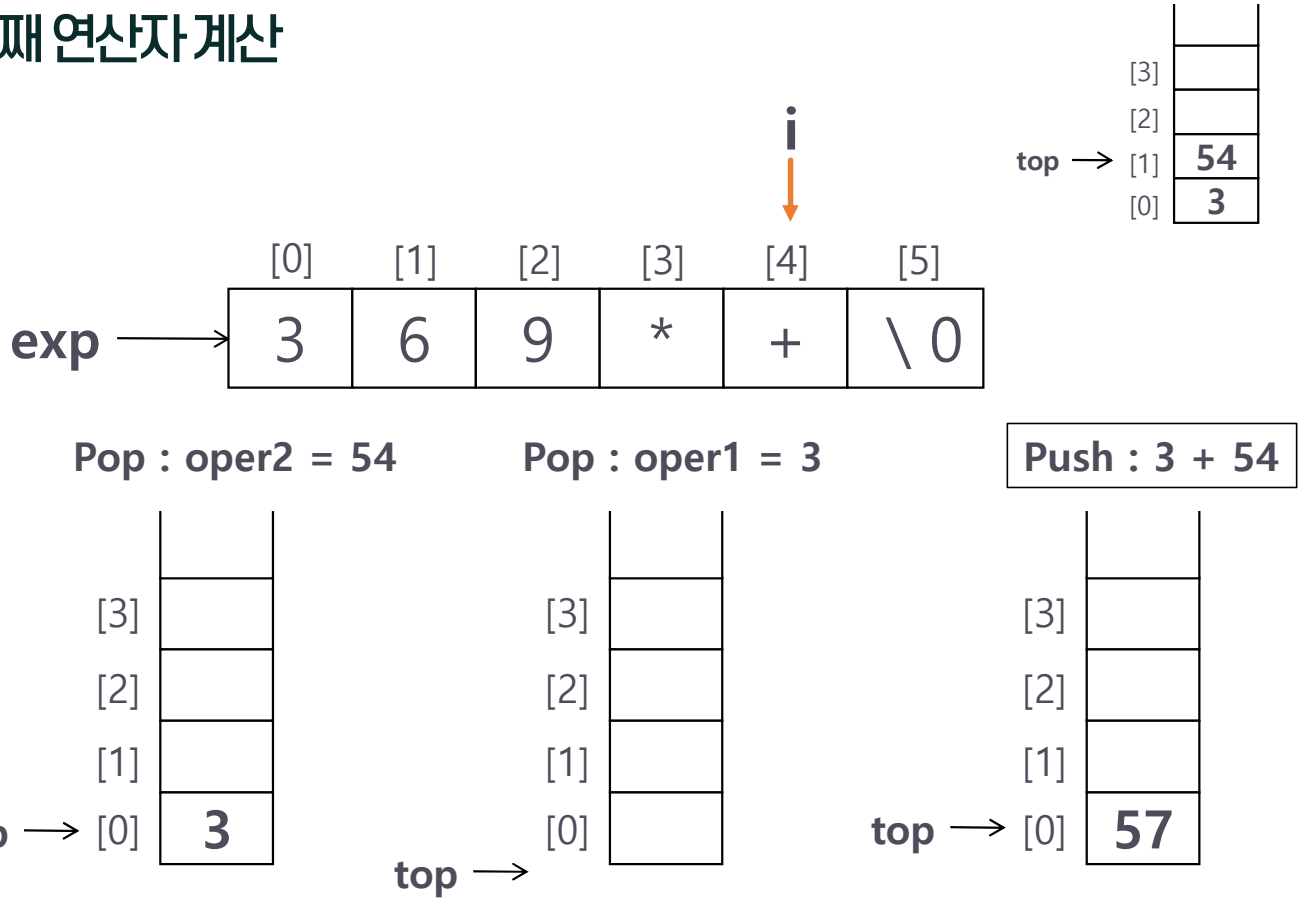
5. 사칙연산의 전위 • 후위 • 중위 표현

첫번째 연산자 계산



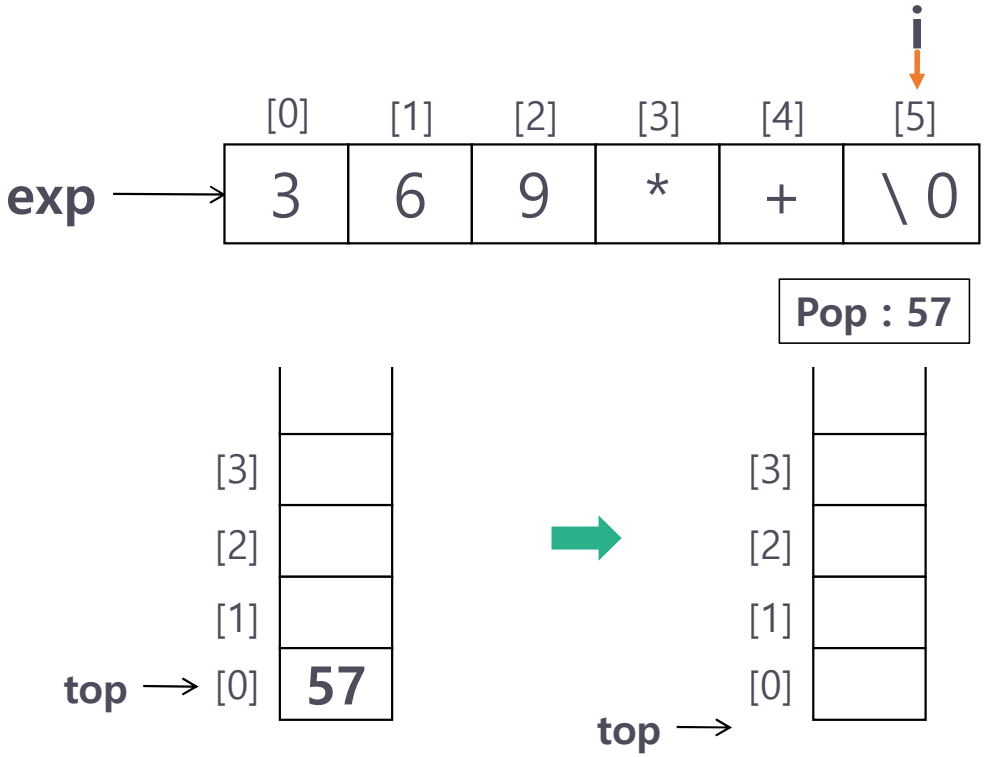
5. 사칙연산의 전위 • 후위 • 중위 표현

두번째 연산자 계산



5. 사칙연산의 전위·후위·중위 표현

- pop(57) 결과



정리하기

- 스택(stack) : 객체와 그 객체가 저장되는 순서를 기억하는 방법에 관한 추상 자료형
- 시스템 스택(system stack) : 변수에 대한 메모리의 할당과 수집을 위해 운영체제가 관리하는 스택
- 스택의 삭제 연산 : 스택의 가장 위에 있는 원소를 삭제하는 연산
- 스택의 삽입 연산 : 스택의 가장 위에 있는 원소 위에 원소 하나를 추가하는 연산

정리하기

- 중위 표기법(infix notation) : 연산자를 피연산자의 사이에 표기하는 방법이며 일반적으로 가장 많이 사용되는 표기 방법(A+B)
- 전위 표기법(prefix notation) : 연산자를 피연산자의 앞에 표기하는 방법(+AB)
- 후위 표기법(postfix notation) : 연산자를 피연산자의 뒤에 표기하는 방법(AB+)

04

다음시간 안내

큐