22. 수식 계산

수식은 연산자 위치에 따라 다음과 같이 구분한다.

전위표기식(prefix)	+ A B
중위표기식(infix)	A + B
후위표기식(postfix)	A B +

[예] 중위표기식 (A + B) * C를 전위 및 후위표기식으로 바꾸어 보자.

① 전위표기식	② 후위표기식
(A + B) * C	(A + B) * C
↓주어진 식을 연산순서로 ( )로 묶음	↓주어진 식을 연산순서로 ( )로 묶음
((A + B) * C)	((A + B) * C)
↓연산자를 해당 괄호 앞으로 이동	↓연산자를 해당 괄호 뒤로 이동
* (+ (A B) C)	((A B) + C) *
↓ ( )를 제거한다.	↓ ( )를 제거한다.
* + A B C	A B + C *

- 주어진 수식에 괄호가 있으면 원래의 식에 있는 괄호는 그대로 두고
- 연산우선순위에 맞추어 식을 괄호로 묶은 후 다른 표기식으로 바꾼다.

◈ 수식의 중위 / 전위 / 후위표기식

중위표기식	전위표기식	후위표기식
3 + 4 * 5	+ 3 * 4 5	3 4 5 * +
3 * 4 + 5	+ * 3 4 5	3 4 * 5 +
(3 + 4) * 5	* + 3 4 5	3 4 + 5 *
a * (b - c) + d / (e + f)	+ * a - b c / d + e f	a b c - * d e f + / +

- 피연산자들의 순서는 3가지 표현 모두가 같다. 수식에서 연산자 순서만 서로 다르다.

◉ 후위표기식 특징

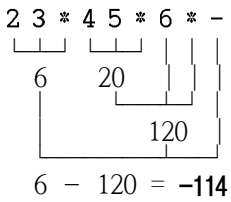
- 괄호가 필요 없다.
- 연산자의 우선순위를 고려할 필요가 없다.
- 후위표기식의 연산 원리는 수식을 앞에서 뒤로 읽어 가면서 연산자를 만나면 연산한다.
- 후위표기식의 연산이 중위표기식을 이용하는 연산보다 연산처리가 단순하다.

1. 후위표기식(postfix)을 이용한 연산 → 컴파일러에서 많이 이용

〈중위표기식을 후위표기식으로 변경〉

2 * 3 − 4 * 5 * 6	← **중위표기식**
↓	주어진 식을 연산우선순위에 따라 ( )로 묶는다.
((2 * 3) − ((4 * 5) * 6))	
↓	각 연산자를 해당 괄호 **뒤로** 옮긴다.
((2 3) * ((4 5) * 6) *) −	
↓	마지막으로, 괄호 ( )를 제거한다.
2 3 * 4 5 * 6 * −	← **후위표기식**

〈후위표기식을 이용한 연산〉

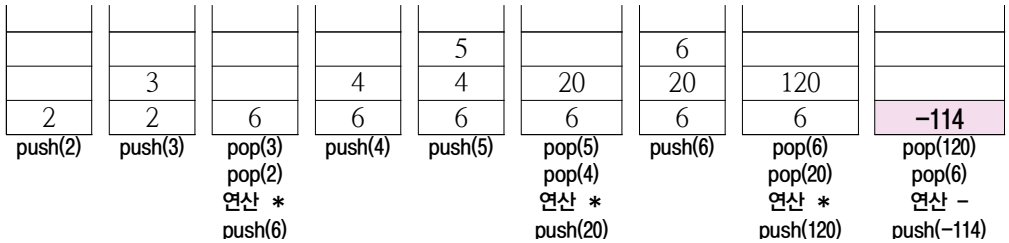


~ **후위표기식**

→ 후위표기식의 수식을 연산하는 과정을 나타낸다.

- 후위표기식의 연산은 수식을 **좌에서 우**로 읽어가면서 **연산자**를 만났을 때 연산한다.
- 연산 대상은 연산자를 만나기 **직전**에 읽은 **피연산자**들이다.

// 후위표기식의 연산 과정을 스택(stack)으로 나타내면 다음과 같다.



◈ **후위표기식의 연산 과정**

① 후위표기식의 수식을 **왼쪽에서 오른쪽**으로 차례로 읽으면서 하나씩 취한다.

- 읽은 자료가 **피연산자**이면 그냥 **스택**에 넣는다.(push)
- 읽은 자료가 **연산자**이면 스택에서 필요한 만큼의 **피연산자를 꺼내**(pop) **연산**한다.
- 그리고, 연산된 **결과만**을 다시 스택에 넣는다.(연산은 나중에 출력된 피연산자가 앞에 온다)

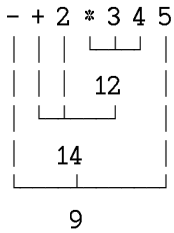
② 위의 연산 과정 ①이 완료된 후, 스택에 저장된 **최종 값**이 수식의 연산 **결과 값**이다.

2. 전위표기식(prefix)을 이용한 연산

〈중위표기식을 전위표기식으로 변경〉

```
2 + 3 * 4 - 5          ← 중위표기식
      ↓                주어진 식을 연산우선순위에 따라 ( )로 묶는다.
( (2 + (3 * 4) ) - 5)
      ↓                각 연산자를 해당 괄호 앞으로 옮긴다.
- ( + (2 * (3 4) ) 5)
      ↓                마지막으로, 괄호 ( )를 제거한다.
- + 2 * 3 4 5          ← 전위표기식
```

〈전위표기식을 이용한 연산〉

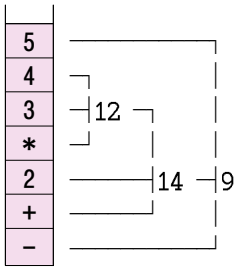


~ 전위표기식

→ 전위표기식의 수식을 연산하는 과정을 나타낸다.

- 전위표기식의 연산은 수식을 **우에서 좌**로 읽어가면서 **연산자**를 만났을 때 연산한다.
- 연산 대상은 연산자를 만나기 **직전**에 읽은 **피연산자**들이다.
- 전위표기식의 연산은 후위표기식의 연산 원리와 반대로 수식을 읽는다.

// 전위표기식의 연산 과정을 스택(stack)으로 나타내면 다음과 같다.



[연산 순서]
① 3 * 4 = 12
② 2 + 12 = 14
③ 14 - 5 = 9

- 수식이 이항 연산자로 구성된 경우,
- 스택을 운행하면서 '**피연산자, 피연산자, 연산자**' 순이면 **연산**하고,
- 연산 결과만을 다시 스택에 넣는다.
- 전위표기식의 연산 원리는 후위표기식의 연산 원리에 비해 시간이 많이 소요된다.

기출문제 분석

1. 다음 중위표기식을 후위표기식으로 바꾼 것으로 옳은 것은? [2022년 국회 9급]

$$a+(b-c)\times d\div(e-f)$$

① $ab+c-def-\div\times$
② $abc-d\times ef-\div+$
③ $abc-d\times ef\div-+$
④ $a+b-c\times d\div e-f$
⑤ $+a\div\times-bcd-ef$

♤ **중위표기식을 후위표기식으로**

a + (b − c) × d ÷ (e − f)	← **중위표기식**
↓	주어진 식을 연산우선순위에 따라 ( )로 묶는다.
(a + (((b − c) × d) ÷ (e − f)))	
↓	연산자 **+**를 해당 괄호 **뒤로** 옮긴다.
(a (((b − c) × d) ÷ (e − f))) +	
↓	연산자 **−**를 해당 괄호 **뒤로** 옮긴다.
(a (((b c) − × d) ÷ (e − f))) +	
↓	연산자 **×**를 해당 괄호 **뒤로** 옮긴다.
(a (((b c) − d) × ÷ (e − f))) +	
↓	연산자 **÷**를 해당 괄호 **뒤로** 옮긴다.
(a (((b c) − d) × (e − f))÷) +	
↓	연산자 **−**를 해당 괄호 **뒤로** 옮긴다.
(a (((b c) − d) × (e f) −)÷) +	
↓	마지막으로, 괄호 ( )를 제거한다.
a b c − d × e f − ÷ +	← **후위표기식**

// 중위표기식을 후위표기식으로 바꾸는 방법(다양)

방법 1	• 먼저, 수식을 연산 순서로 괄호로 묶는다. • 수식을 **왼쪽에서 오른쪽**으로 읽어가면서 연산자를 만났을 때, **연산자를 뒤로 이동**한다. • **연산자를 뒤로 이동**하는 방법은 연산 순서에 해당하는 **괄호 뒤로 이동**한다. • 수식이 긴 경우는 이 방법이 실수할 가능성이 적다. 위에서 풀이한 방법이다.
방법 2	• 수식의 연산 순서로 **연산자**를 **괄호 뒤로 이동**한다.

정답 : ②

2. 다음 전위(prefix) 표기식의 계산 결과는? [2019년 국가 9급]

```
+ - 5 4 × 4 7
```

① -19 ② 7 ③ 28 ④ 29

♤ 전위표기식

• + - 5 4 × 4 7
  1 28 → 전위표기식의 수식을 연산하는 과정을 나타낸다.
  29

• 전위표기식의 연산은 수식을 뒤에서 앞으로 읽어가면서 연산자를 만났을 때 연산한다.
• 연산 대상은 연산자를 만나기 직전에 읽은 피연산자들이다.

정답 : ④

3. 다음 후위표기식을 전위표기식으로 변환하였을 때 옳은 것은? [2022년 국가 9급]

```
3 1 4 1 - * +
```

① 3 + 1 * 4 - 1 ② 4 - 1 * 1 + 3
③ + 3 * 1 - 4 1 ④ + 3 - 4 1 * 1

♤ 후위표기식을 전위표기식으로 변환

• 후위표기식을 **왼쪽에서 오른쪽**으로 읽어가면서 **연산자**를 만났을 때 처리한다.
• 연산자를 만났을 때, 해당 연산자를 **연산 단위 앞으로 이동**한다.

3 1 4 1 − * +	← **후위표기식**
↓	연산자 −를 앞으로 이동
3 1 (− 4 1) * +	
↓	연산자 *를 앞으로 이동
3 (* 1 (− 4 1)) +	
↓	연산자 +를 앞으로 이동
(+ 3 (* 1 (− 4 1)))	
↓	마지막으로, 괄호 ( )를 제거한다.
+ 3 * 1 − 4 1	← **전위표기식**

정답 : ③

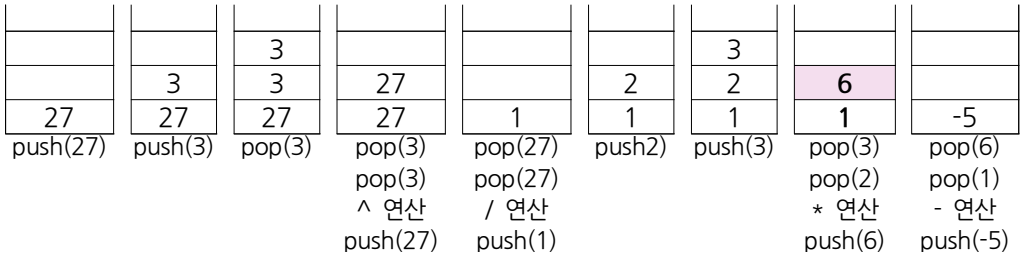
4. 다음은 postfix 수식이다. 이 postfix 수식은 스택을 이용하여 연산을 수행한다. 그리고 ^는 지수함수 연산자이다. 처음 *(곱하기 연산) 계산이 되고 난 후 스택의 top과 top-1에 있는 두 원소는 무엇인가? (단, 보기의 (top)은 스택의 top 위치를 나타낸다) [2017년 서울 9급]

```
-------------------------------
27  3  3  ^  /  2  3  *  -
-------------------------------
```

① (top) 6, 1 ② (top) 9, 1 ③ (top) 2, 3 ④ (top) 2, 7

♤ **후위식 연산**

- 후위식의 연산은 수식을 **왼쪽에서 오른쪽**으로 읽어가면서 연산자를 만났을 때 연산한다.
- 연산 대상은 연산자를 만나기 직전에 읽은 피연산자들이다.
- 후위식의 연산 과정을 Stack으로 나타내면 다음과 같다.



*(곱하기 연산) 계산이 되고 난 후 스택의 top과 top-1 : 6, 1

정답 : ①