Infix To Postfix Convertion (중위 → 후위)

Input : infix expression (예 : A+B*C-D) exp

Output : postfix expression (예: ABC*+D-) res

* key point → operator를 stack에 넣는다.

입력 수식을 첫번째항 (index 1) ~> 마지막 항까지 (index (length-1)) Scan을 하면서

① empty stack 생성

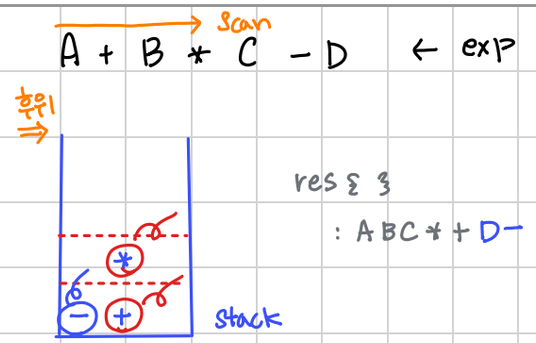
② 만약 operand 이면, 후위 수식으로 옮김

③ 그렇지 않고 만약 operator이면 stack에 push.

* 단, stack에 넣을 때 stack이 비어있지 않고, top 에 있는 연산자의 우선순위가 push 하려는 원소의 우선순위보다 높거나 같으면 pop()으로 꺼내서 후위 수식으로 옮김

④ ②의 과정을 수식 끝까지 반복

⑤ stack 에 남아있는 원소들을 pop()으로 꺼내서 후위 수식으로 옮김.




res { }

: ABC*+D-

evaluate → 평가 방법!

① 왼쪽 → 오른쪽 Scan

② operand 이면 stack에 넣음

③ operator, → stack에서 두개를 꺼내서 계산결과 다시 stack에 넣음.

④ ② ~ ③ 과정을 반복.

ALGORITHM Infix To Postfix Convert

```
// Input : exp (중위표기수식)
// Output : res (후위 표기 수식)
1. Creat an empty stack S
2. res ← { }  // empty list
3. For i ← 0 To i ← (length(exp) - 1)  // scan
4. IF exp(x) is operand  // 피연산자
5.     res ← res + exp(i)  // res에 append
6. ELSE IF exp(i) is operator  // 연산자
7. WHILE (!isempty(S) and
          Precedence(Top(S)) ≥ Precedence(exp(i)))
8.     res ← res + pop(S)
9.     PUSH(S, exp(i))
10. WHILE (! isempty(S))  // stack에 남아 있는 연산자를
                          꺼내서 후위 표기식으로 옮김.
11.     res ← res + pop(S)
12. RETURN res
```