자료구조 (특별)

Chap 04. 스택 2 (수식 연산)

2019년 하계 특별 학기

컴퓨터과학과
민경하

Contents

4. 스택 2 (수식 연산)

4.1 수식

4.2 수식 연산

4.3 수식의 종류

4.4 연산 전략

4.5 후위 표기 변환

4.6 후위 표기 연산

4.1 수식

- 수식 (Expression)
  - 연산자 (operator)와 피연산자(operand)의 규칙에 따른 조합

```
3+4*6-7/2
```

```
x = a/b –c + d/g – a*c;
```

```
((rear+1 == front) || ((rear == MAX_SIZE) && !front))
```

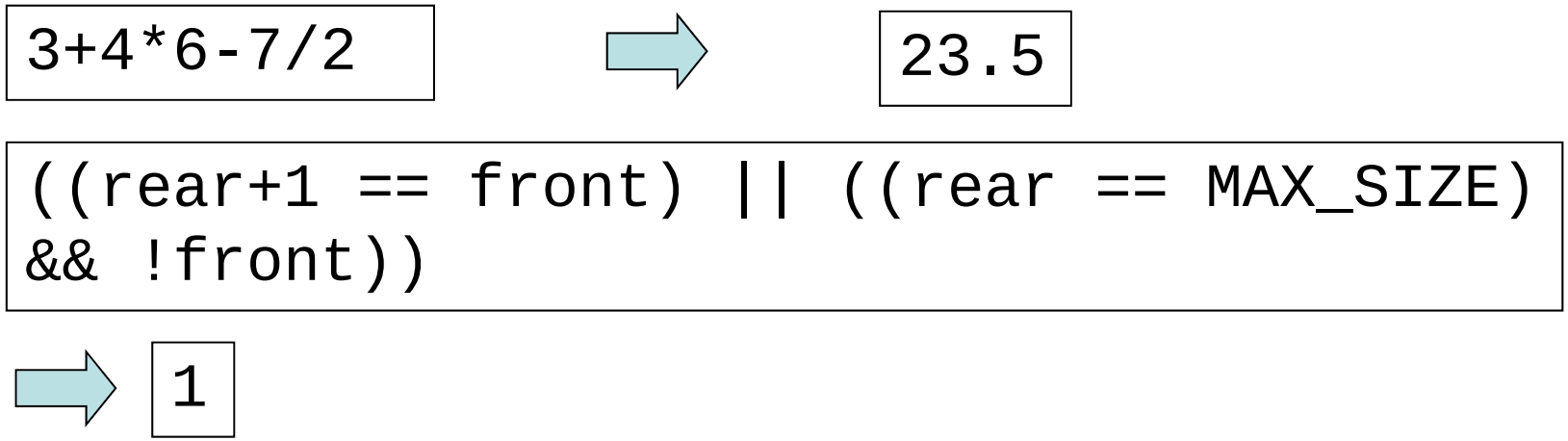
  - 피연산자: 3, 4, x, a, b, front, MAX_SIZE, …
  - 연산자: +, –, *, /, (,), ==, &&, !, …

4.1 수식

- 연산자의 종류
  - **1**항 연산자 **(Unary** operator)
    - 하나의 피연산자를 갖는 연산자
    - OPERATOR OPERAND or OPERAND OPERATOR
    - Example: **–**4, !front, x++
    - **OPERATOR (OPERAND) → OPERAND**
  - **2**항 연산자 **(Binary** operator)
    - 2 개의 피연산자를 갖는 연산자
    - OPERAND1 OPERATOR OPERAND2
    - Example: 4 – 5, index != 3, a && b
    - **OPERATOR (OPERAND1, OPERAND2) → OPERAND**

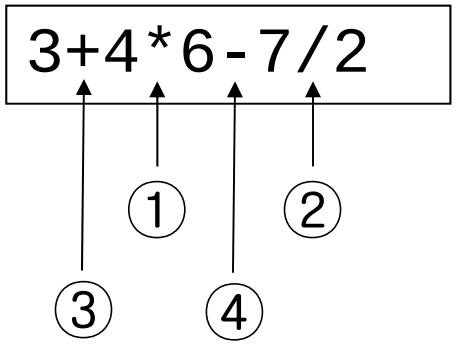
4.2 수식의 연산

- 수식의 연산 (evaluation of expression)
  - 적절한 순서로 연산을 반복해서 주어진 수식과 동등한 하나의 값을 찾는 과정



4.2 수식의 연산

- 연산의 문제점 (1)
  - 연산의 순위를 결정해야 함 → **precedence**



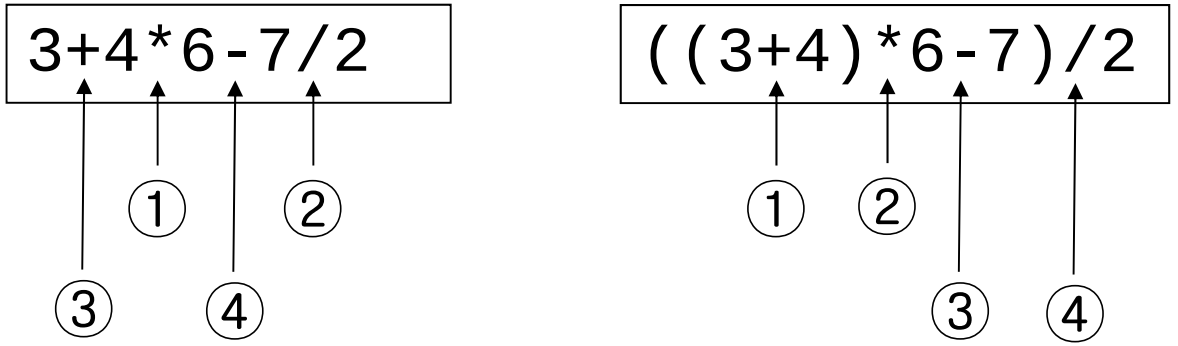
4.2 수식의 연산

- 연산의 문제점(1)
  - 우선 순위 표 (Precedence table)

우선순위	연산자 (연산기호)	연산자 명
1	(), [], ->, .	괄호, 포인터, 도트
2	!, ~, ++, --, *, &, sizeof(), (자료형)	단항 연산자
3	*, /, %	산술 연산자
4	+, -	
5	<<, >>	비트이동 연산자
6	<, >, <=, >=	관계 연산자
7	==, !=	
8	&	비트논리 연산자
9	^	
10	|	
11	&&	논리 연산자
12	||	
13	?:	조건연산자
14	=, +=, *=, ... , <<=	할당 및 복합 할당연산자
15	,	콤마 연산자

4.2 수식의 연산

- 연산의 문제점 (2)
  - 우선 순위의 무력화 (Overriding the precedence)
    - 괄호 (**parenthesis**)의 사용
    - 가장 안쪽의 괄호 (**innermost parenthesis**)가 가장 먼저 연산됨



4.2 수식의 연산

- 연산의 문제점 (3)
  - 동일한 우선 순위를 갖는 연산자들의 연산 순서 → **associativity**
    - Left-to-right
    - Right-to-left

4.3 수식의 종류

- 수식 표기의 종류
  - 중위 표기 (Infix notation)
    - OPERAND1 **OPERATOR** OPERAND2

      3 + 4
  - 전위 표기 (Prefix notation)
    - **OPERATOR** OPERAND1 OPERAND2

      + 3 4
  - 후위 표기 (Postfix notation)
    - OPERAND1 OPERAND2 **OPERATOR**

      3 4 +

4.3 수식의 종류

- 괄호의 사용 여부
  - 중위 표기 (필요함)

```
3 + 4 * 5
```

```
(3 + 4) * 5
```

  - 전위 표기 (필요 없음)

```
+ 3 * 4 5
```

```
* + 3 4 5
```

  - 후위 표기 (필요 없음)

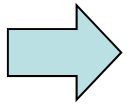
```
3 4 5 * +
```

```
3 4 + 5 *
```

4.3 수식의 종류

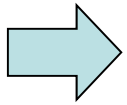
- 표기법 변환의 예

3 + 4 * 8 – 7 / 2



3 4 8 * + 7 2 / -

((3 + 4) * 8 – 7) / 2



3 4 + 8 * 7 – 2 /

4.4 연산 전략

- 토큰 (Token) 단위 연산
  - 토큰
    - 서로 구별되는 단위
    - 피연산자

```
(3 + 4) * 5
```

```
24.6 + a_3
```

    - 연산자

```
(3 + 4) * 5
```

```
24.6 + a_3
```

```
x++
```

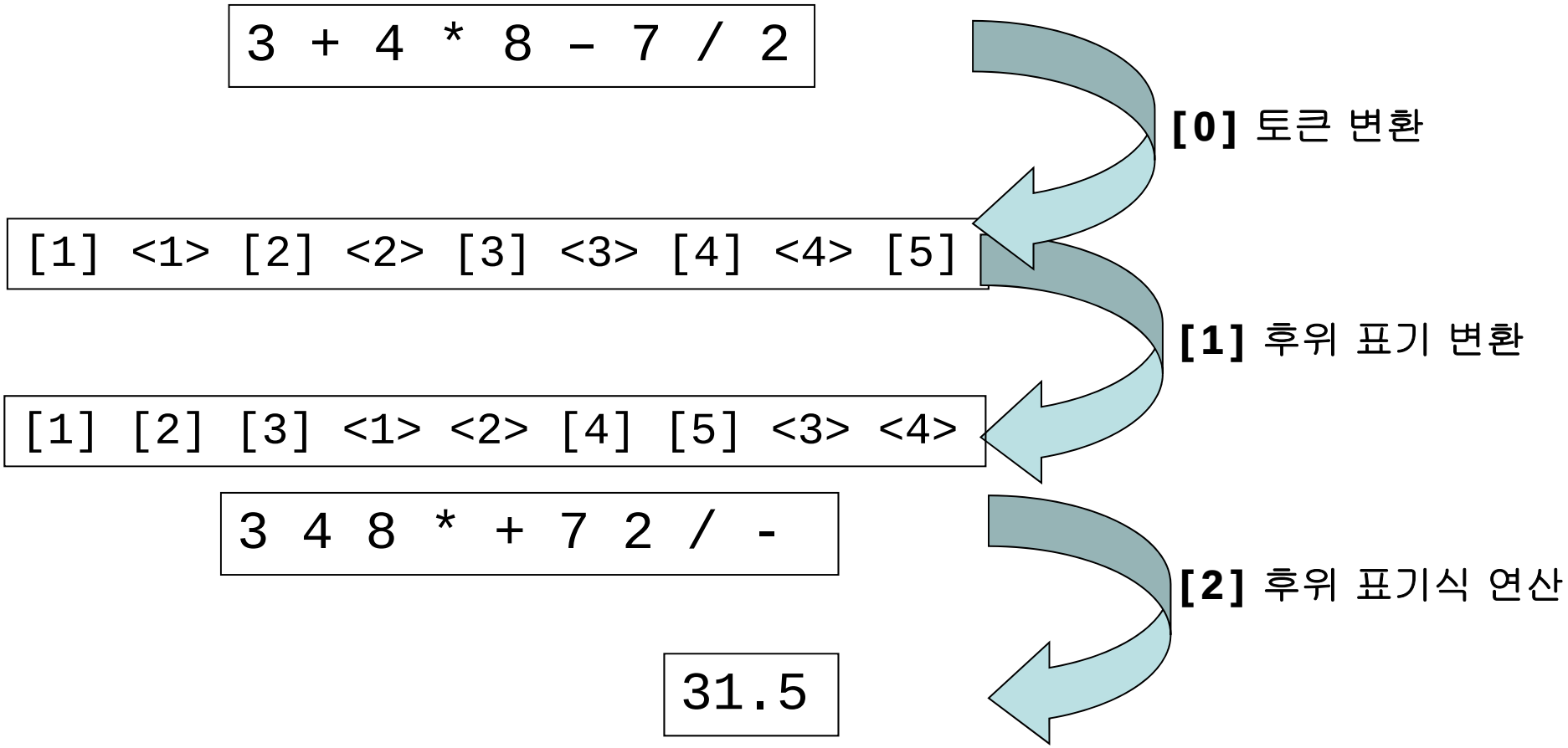
  - 수식

```
3 + 4 * 8 - 7 / 2
```

```
opd1 opr1 opd2 opr2 opd3 opr3 opd4 opr4 opd5
```

4.4 연산 전략

- 연산 과정


3 + 4 * 8 – 7 / 2
[0] 토큰 변환
[1] <1> [2] <2> [3] <3> [4] <4> [5]
[1] 후위 표기 변환
[1] [2] [3] <1> <2> [4] [5] <3> <4>
3 4 8 * + 7 2 / -
[2] 후위 표기식 연산
31.5

4.5 후위 표기 변환

- 중위 표기 연산을 후위 표기 연산으로 변환?

  ➔ **stack**을 사용

  - 단순한 경우
    - 괄호가 없는 연산
  - 복잡한 경우
    - 괄호가 있는 연산

4.5 후위 표기 변환

- 알고리즘 (복잡한 경우)

```
for all tokens in an expression
    if (current token is an operand)
        print (current token);
    else
        if ( current token == ‘(‘ )
            stack.push (current token);
        if ( current token == ‘)’)
            while ( stack.TOP() != ‘(‘)
                print(stack.pop());
            stack.pop();
        if ( stack.TOP() <= current token )
            print(stack.pop());
        stack.push(current token);

while (!stack.is_empty())
    print (stack.pop());
```

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(							
(							
3							
+							
4							
)							
*							
8							
–							
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(							
3							
+							
4							
)							
*							
8							
–							
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(	(	(				1	
3							
+							
4							
)							
*							
8							
–							
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(	(	(				1	
3	(	(				1	3
+							
4							
)							
*							
8							
–							
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(	(	(				1	
3	(	(				1	3
+	(	(	+			2	3
4							
)							
*							
8							
–							
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(	(	(				1	
3	(	(				1	3
+	(	(	+			2	3
4	(	(	+			2	3 4
)							
*							
8							
–							
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(	(	(				1	
3	(	(				1	3
+	(	(	+			2	3
4	(	(	+			2	3 4
)	(					0	3 4 +
*							
8							
–							
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(	(	(				1	
3	(	(				1	3
+	(	(	+			2	3
4	(	(	+			2	3 4
)	(					0	3 4 +
*	(	*				1	3 4 +
8							
–							
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(	(	(				1	
3	(	(				1	3
+	(	(	+			2	3
4	(	(	+			2	3 4
)	(					0	3 4 +
*	(	*				1	3 4 +
8	(	*				1	3 4 + 8
–							
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(	(	(				1	
3	(	(				1	3
+	(	(	+			2	3
4	(	(	+			2	3 4
)	(					0	3 4 +
*	(	*				1	3 4 +
8	(	*				1	3 4 + 8
–	(	–				1	3 4 + 8 *
7							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
(	(					0	
(	(	(				1	
3	(	(				1	3
+	(	(	+			2	3
4	(	(	+			2	3 4
)	(					0	3 4 +
*	(	*				1	3 4 +
8	(	*				1	3 4 + 8
–	(	–				1	3 4 + 8 *
7	(	–				1	3 4 + 8 * 7

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
)						-1	3 4 + 8 * 7 –
/							
2							
EOS							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
)						-1	3 4 + 8 * 7 –
/	/					0	3 4 + 8 * 7 –
2							
EOS							

4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
)						-1	3 4 + 8 * 7 –
/	/					0	3 4 + 8 * 7 –
2	/					0	3 4 + 8 * 7 – 2
EOS							

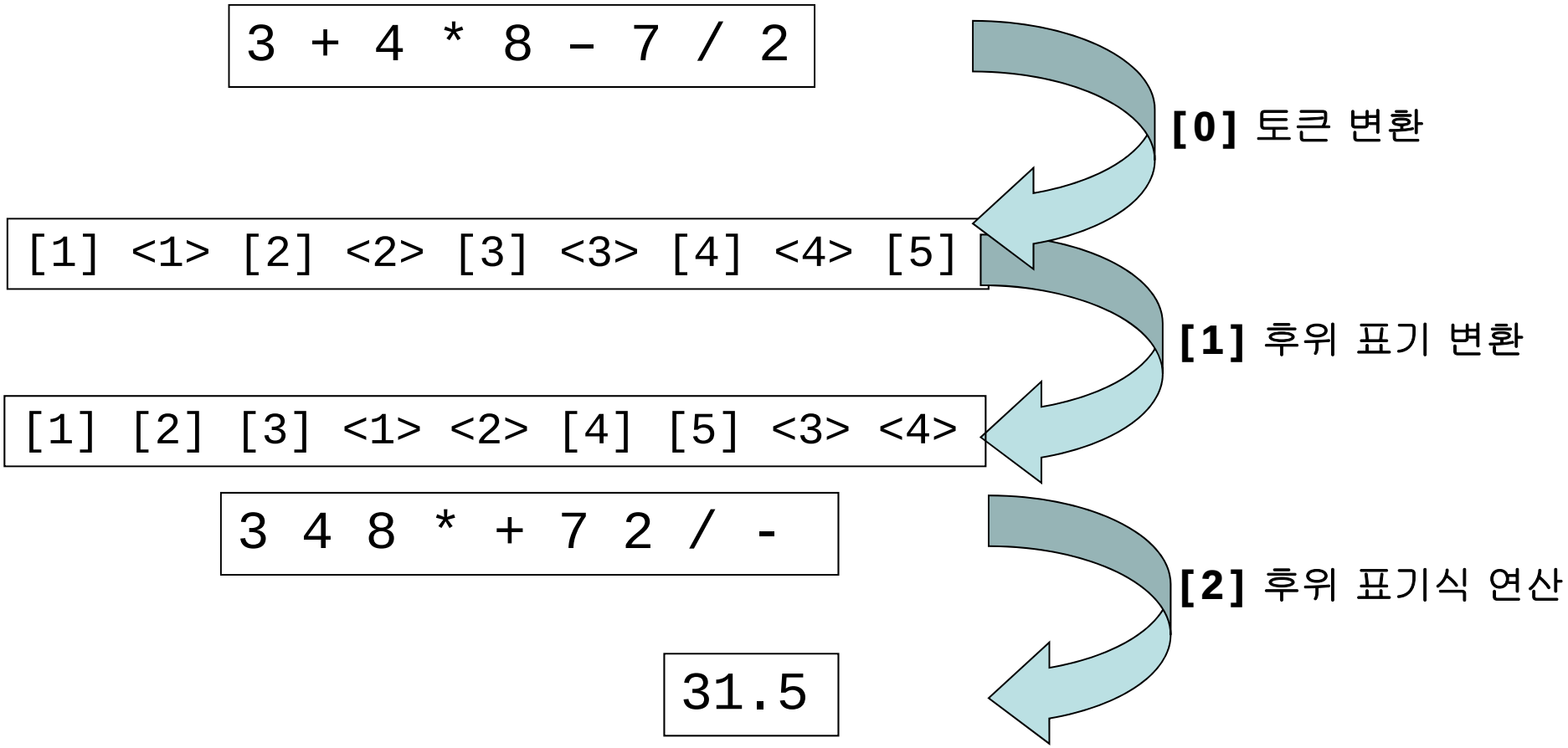
4.5 후위 표기 변환

- Example: `((3 + 4) * 8 – 7) / 2`

Token	Stack					TOP	Output
	[0]	[1]	[2]	[3]	[4]		
)						-1	3 4 + 8 * 7 –
/	/					0	3 4 + 8 * 7 –
2	/					0	3 4 + 8 * 7 – 2
EOS						-1	3 4 + 8 * 7 – 2 /

4.6 후위 표기 연산

- 연산 과정



4.6 후위 표기 연산

- 후위 표기 수식의 연산 ➔**Use stack!!**

```
for all tokens in a postfix expression
    if (current token is an operand)
        stack.push ( current token );
    else
        od1 = stack.pop ();
        od2 = stack.pop ();
        switch ( current token )
            case ‘+’:
                stack.push ( od2 + od1 );
            ……

print(stack.pop());
```

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3							
4							
+							
8							
*							
7							
-							
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4							
+							
8							
*							
7							
-							
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+							
8							
*							
7							
-							
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+				0	3	4	7
8							
*							
7							
-							
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8							
*							
7							
-							
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8	7	8		1			
*							
7							
-							
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8	7	8		1			
*				0	7	8	56
7							
-							
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8	7	8		1			
*	56			0	7	8	56
7							
-							
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8	7	8		1			
*	56			0	7	8	56
7	56	7		1			
-							
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8	7	8		1			
*	56			0	7	8	56
7	56	7		1			
-				0	56	7	49
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8	7	8		1			
*	56			0	7	8	56
7	56	7		1			
-	49			0	56	7	49
2							
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8	7	8		1			
*	56			0	7	8	56
7	56	7		1			
-	49			0	56	7	49
2	49	2		1			
/							

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8	7	8		1			
*	56			0	7	8	56
7	56	7		1			
-	49			0	56	7	49
2	49	2		1			
/				0	49	2	24.5

4.6 후위 표기 연산

- Example: `3 4 + 8 * 7 - 2 /`

Token	Stack			Top	Op1	Op2	Result
	[0]	[1]	[2]				
3	3			0			
4	3	4		1			
+	7			0	3	4	7
8	7	8		1			
*	56			0	7	8	56
7	56	7		1			
-	49			0	56	7	49
2	49	2		1			
/	24.5			0	49	2	24.5

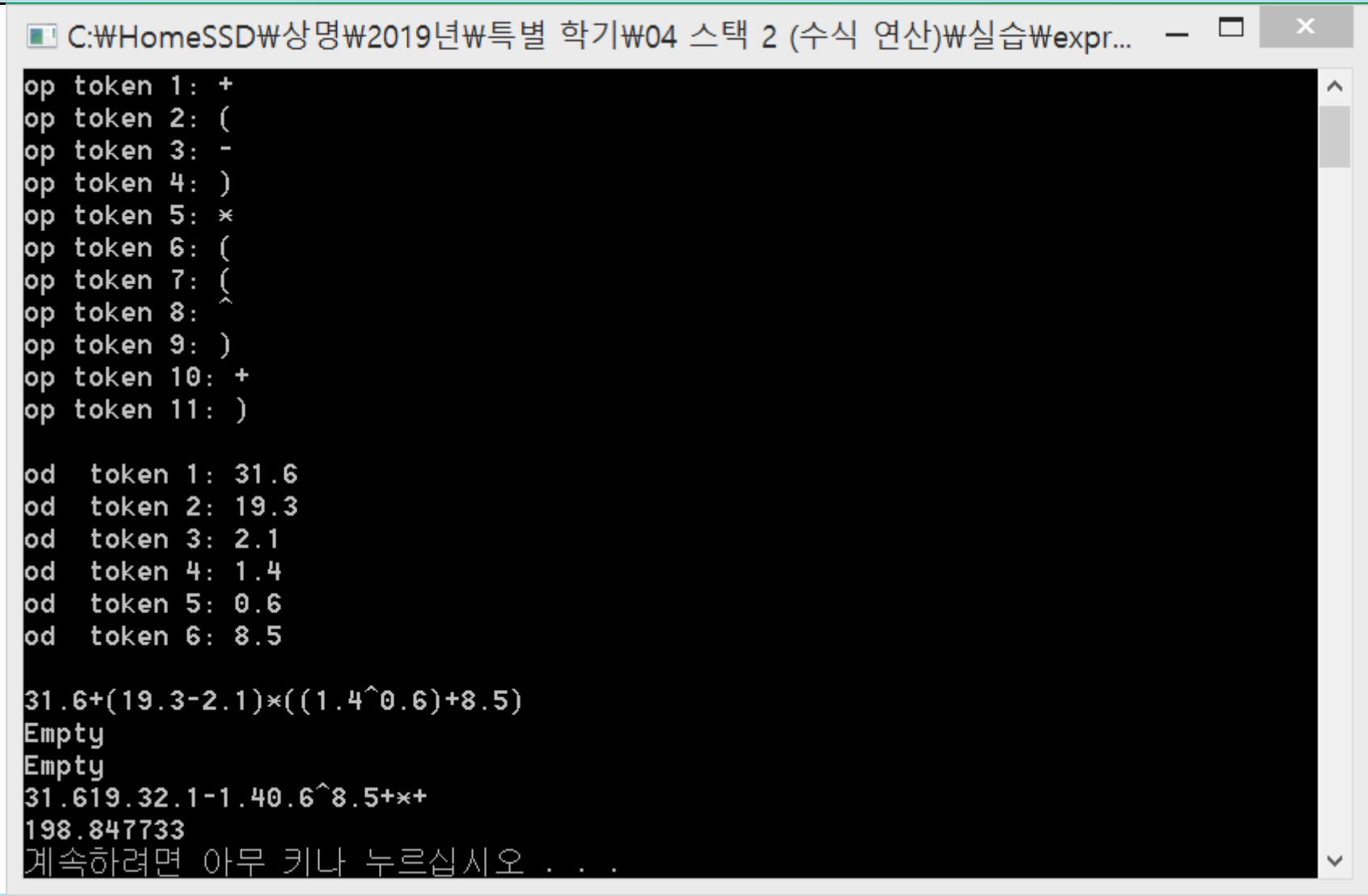
프로그램 구조

- stack1.h
  - 후위 표기식 생성을 위한 stack
- Stack1.cpp
- stack2.h
  - 후위 표기식 연산을 위한 stack
- stack2.cpp
- emain.cpp

main 함수

```
int main()
{
    char expr[256] = "31.6 + (19.3 -  2.1)*((1.4^0.6) + 8.5)";
    char expr2[256];

    remove_delimiters(expr2, expr);
    token_parser(expr2);
    print_tok(tcnt, tok);

    postpix( );
    print_tok(ptcnt, ptok);

    evaluate();

    system("pause");
    return 0;
}
```

출력 예


C:₩HomeSSD₩상명₩2019년₩특별 학기₩04 스택 2 (수식 연산)₩실습₩expr...
op token 1: +
op token 2: (
op token 3: -
op token 4:)
op token 5: *
op token 6: (
op token 7: (
op token 8: ^
op token 9:)
op token 10: +
op token 11:)
od token 1: 31.6
od token 2: 19.3
od token 3: 2.1
od token 4: 1.4
od token 5: 0.6
od token 6: 8.5
31.6+(19.3-2.1)*((1.4^0.6)+8.5)
Empty
Empty
31.619.32.1-1.40.6^8.5+*+
198.847733
계속하려면 아무 키나 누르십시오 . . .