Chapter 06.
스택

9주차

계산기 프로그램
(계산할 식의 형태)

다음과 같은 문장의 수식을 계산할 수 있어야 한다.

(3 + 4) * (5 / 2) + (7 + (9 – 5))

그러기 위해서는 아래 두 가지를 고려해야 한다.

소괄호 파악 -> 그 부분을 먼저 연산.

연산자의 우선순위 반영.

스택을 이용하여 구현!

계산기 프로그램
(세 가지 수식의 표기법: 전위, 중위, 후위)

중위 표기법(infix notation) **예) 5 + 2 / 7**

수식 내에 연산 순서에 대한 정보가 없다.

-> 소괄호와 연산자의 우선순위를 고려해 줘야 함.

전위 표기법(prefix notation) **예) + 5 / 2 7**

수식 내에 연산의 순서에 대한 정보가 있다.

-> 소괄호와 연산의 우선순위를 고려할 필요 없다.

후위 표기법(postfix notation) **예) 5 2 7 / +**

수식 내에 연산의 순서에 대한 정보가 있다.

-> 소괄호와 연산의 우선순위를 고려할 필요 없다.

계산기 프로그램
(중위표기식 -> 후위표기식으로 전환)

(1) 피 연산자는 그대로 결과로 출력.

(2) 연산자는 스택에 push

- (스택 top 연산자의 우선순위) < (새로 push할 연산자의 우선순위)

  될 때까지 pop하여 결과로 출력.

(3) 식을 모두 처리했으면 스택이 빌 때까지 popup하여 출력.

계산기 프로그램
(중위표기식 -> 후위표기식으로 전환)

▶ [그림 06-3: 수식 변환의 과정 1/7]

▶ [그림 06-5: 수식 변환의 과정 3/7]

▶ [그림 06-7: 수식 변환의 과정 5/7]

▶ [그림 06-9: 수식 변환의 과정 7/7]

계산기 프로그램
(중위표기식 -> 후위표기식으로 전환 : 소괄호 고려)

여는 소괄호:

무조건 push.

스택 안에서는 우선순위가 가장 낮아져서 모든 연산자를 받아들임.

닫는 소괄호:

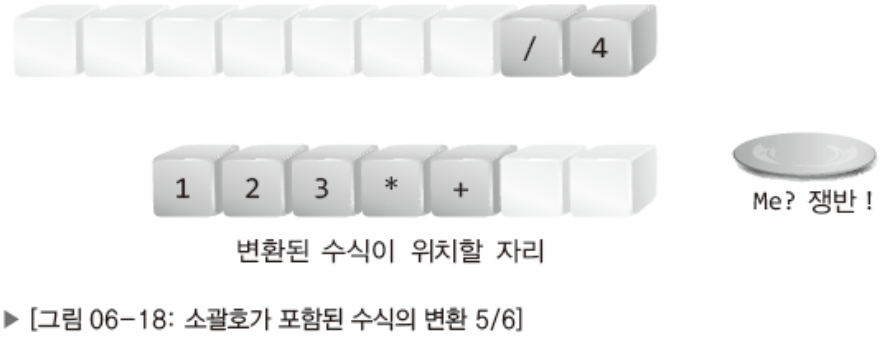
여는 소괄호가 나올 때까지 pop하여 결과로 출력.

여는 소괄호는 pop하여 버림

계산기 프로그램
(중위표기식 -> 후위표기식으로 전환 : 소괄호 고려)

▶ [그림 06-14: 소괄호가 포함된 수식의 변환 1/6]

▶ [그림 06-17: 소괄호가 포함된 수식의 변환 4/6]



▶ [그림 06-18: 소괄호가 포함된 수식의 변환 5/6]

▶ [그림 06-19: 소괄호가 포함된 수식의 변환 6/6]

계산기 프로그램
(후위표기식 계산)

(1) 피 연산자 : 스택에 push.

(2) 연산자 : 스택에서 피연산자를 pop하여 연산 결과를 push.

- op1 op op2

- 먼저 pop된 피연산자가 op2, 나중 pop된 피연산자가 op1

(3) 식을 모두 처리했을 때 스택에 남는 수가 결과.

실습 소스 파일

- **ListBaseStack.c**
  - 스택 구현 소스 파일
  - 지난 주 작업한 “연결리스트 기반 스택” 구현에서 main 함수만 제외
- **ListBaseStack.h**
  - 지난 주 작업한 소스에 대한 헤더 파일
- **Canculator.c**
  - 계산기 소스

[실행 결과]

1+2*3 = 7
(1+2)*3 = 9
((1-2)+3)*(5-2) = 6

실습
(ListBaseStack.h)

지난 주 작업한 소스에 대한 헤더 파일

```
#ifndef _LISTBASESTACK_H_
#define _LISTBASESTACK_H_

#define TRUE      1
#define FALSE     0

typedef int Data;
typedef struct _node {
          Data data;
          struct _node * next;
} Node;
typedef struct _listStack {
          Node * head;
} ListStack;
typedef ListStack Stack;

void StackInit(Stack * pstack);
int SIsEmpty(Stack * pstack);
void SPush(Stack * pstack, Data data);
Data SPop(Stack * pstack);
Data SPeek(Stack * pstack);

#endif // _LISTBASESTACK_H_
```