세종사이버대학교

자료구조와알고리즘

9주차 -2

스택의 응용 2

학습내용

4. 스택을 이용한 수식의 후위 표기법 변환

- 4.1 수식의 표기법
- 4.2 중위 표기법을 후위 표기법으로 변환 방법
- 4.3 중위 표기법을 후위 표기법으로 변환 알고리즘
- 4.4 중위 표기법을 후위 표기법으로 변환 과정

5. 스택을 이용한 후위 표기법 수식의 연산

- 5.1 후위 표기법 수식의 연산 방법
- 5.2 후위 표기법 수식의 연산 알고리즘
- 5.3 후위 표기법 수식의 연산 과정
- 5.4 후위 표기법 수식 연산의 구현

학습목표

- 스택을 이용한 수식의 후위 표기법 변환을 설명할 수 있다.
- 변환된 후위 표기법 수식의 연산을 구현할 수 있다.

4 스택을 이용한 수식의 후위 표기법 변환

4.1 수식의 표기법

4.2 중위 표기법을 후위 표기법으로 변환 방법

4.3 중위 표기법을 후위 표기법으로 변환 알고리즘

4.4 중위 표기법을 후위 표기법으로 변환 과정

4.1 수식의 표기법

- 전위표기법 (prefix notation)
  - 연산자를 피연산자를 앞에 표기하는 방법

    예 +AB

- 중위표기법 (infix notation)
  - 연산자를 피연산자의 가운데 표기하는 방법

    예 A+B

- 후위표기법 (postfix notation)
  - 연산자를 피연산자 뒤에 표기하는 방법

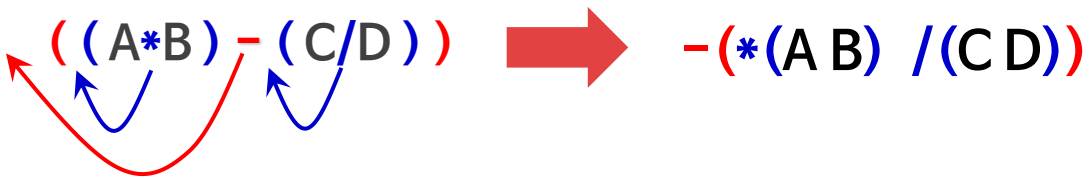
    예 AB+

1 A*B-C/D인 중위표기식을 전위표기식으로 변환 방법

❶ 수식의 각 연산자에 대해서 우선순위에 따라 괄호를 사용하여 다시 표현한다.

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

❷ 각 연산자를 그에 대응하는 왼쪽 괄호의 앞으로 이동시킨다.



❸ 괄호를 제거한다.

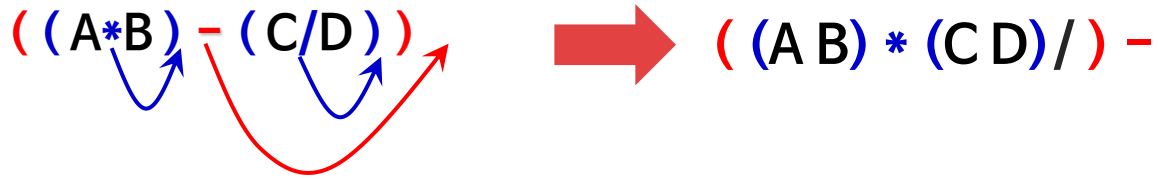
- * A B / C D

2 A*B-C/D인 중위표기식을 후위표기식으로 변환 방법

❶ 수식의 각 연산자에 대해서 우선순위에 따라 괄호를 사용하여 다시 표현한다.

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

❷ 각 연산자를 그에 대응하는 왼쪽 괄호의 뒤로 이동시킨다.



❸ 괄호를 제거한다.

A B * C D / -

4.1 수식의 표기법

- 일상에서 사용하는 표기법은 중위 표기법이지만
- 컴퓨터 내부에서 수식을 처리하는 데 가장 효율적인 방법은 후위 표기법임
- 후위 표기법을 사용하면 괄호나 연산자 우선순위를 따로 처리하지 않고, 왼쪽에서 오른쪽으로 표기된 순서대로 처리할 수 있음
- 사용자가 컴퓨터에 중위 표기법 형태의 수식을 입력하면 컴퓨터 내부에서는 효율적인 처리를 위해 스택을 사용하여 입력된 수식을 후위 표기법으로 변환함

4.2 중위 표기법을 후위 표기법으로 변환 방법

❖ 컴퓨터 내부에서 중위 표기법을 후위 표기법으로 바꾸는 방법

❶ 왼쪽 괄호를 만나면 무시하고 다음 문자를 읽는다.

{, (

❷ 피연산자를 만나면 출력한다.

A, B

❸ 연산자를 만나면 스택에 삽입한다.

+, /

❹ 오른쪽 괄호를 만나면 스택을 pop하여 출력한다.

), }

연산자, +, /

❺ 수식이 끝나면 스택이 공백이 될 때까지 pop하여 출력한다.

연산자, +, /

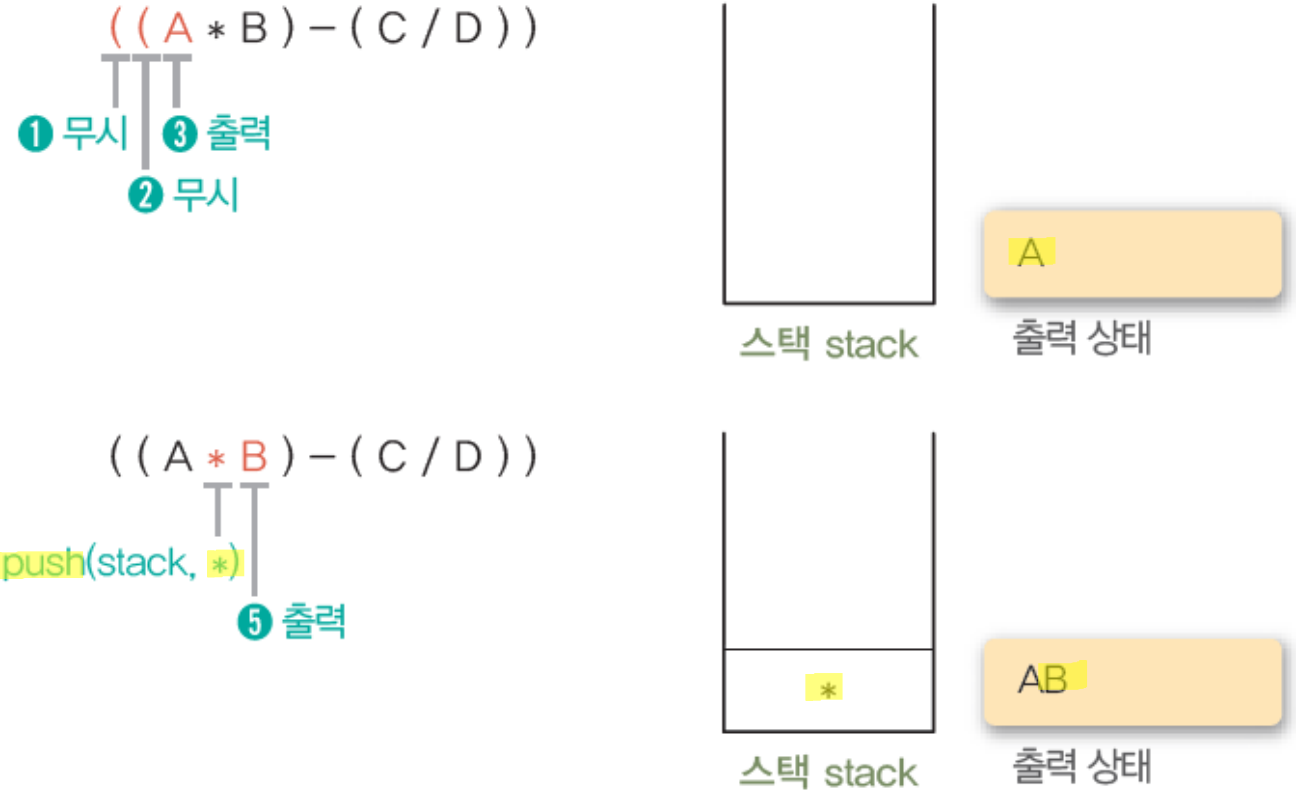
4.3 중위 표기법을 후위 표기법으로 변환 알고리즘

알고리즘 5-4 후위 표기법으로 변환

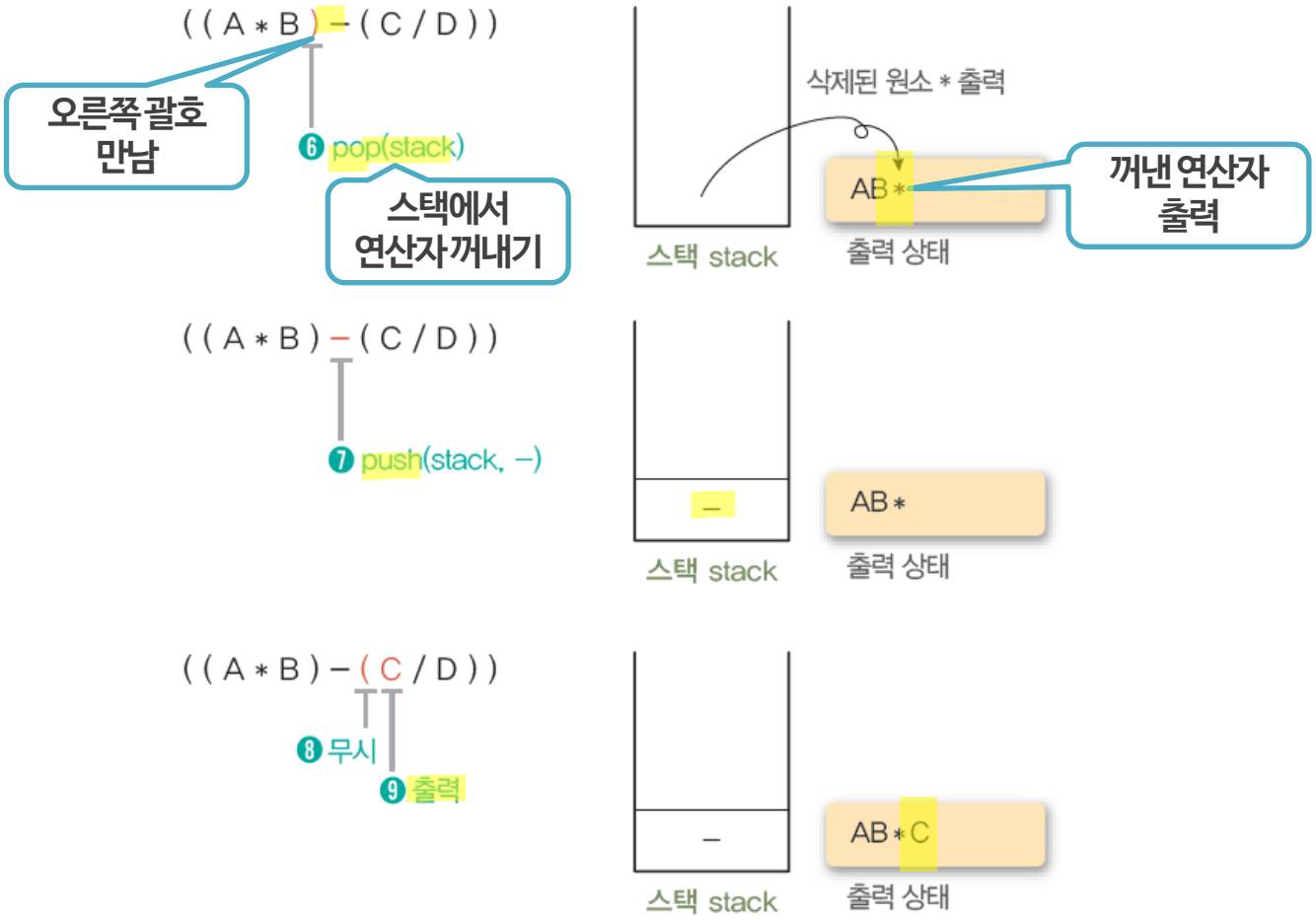
```
infix_to_postfix(exp)
    while (true) do {
        symbol ← getSymbol(exp);
        case {
            symbol = operand :  // 피연산자 처리: 출력
                      print(symbol);
            symbol = operator : // 연산자 처리: 스택에 push
                      push(stack, symbol);
            symbol = ")" :      // 오른쪽 괄호 처리: 스택을 pop하여 출력
                      print(pop(stack));
            symbol = null :     // 수식의 끝 처리:
                      while (top > -1) do // 스택이 공백이 될 때까지 pop하여 출력
                          print(pop(stack));
            else :
        }
    }
end infix_to_postfix()
```

4.4 중위 표기법을 후위 표기법으로 변환 과정

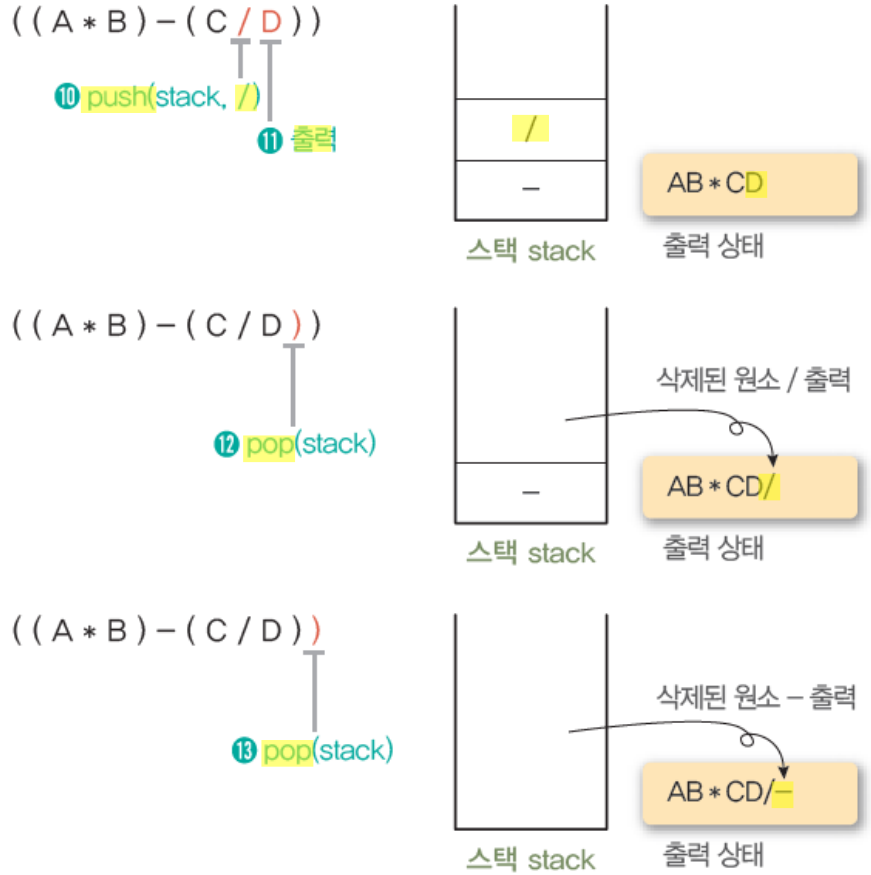
❖ 스택을 이용하여 수식 A*B-C/D를 후위 표기법으로 변환하는 예)



4.4 중위 표기법을 후위 표기법으로 변환 과정



4.4 중위 표기법을 후위 표기법으로 변환 과정



[스택을 사용해 수식 A*B-C/D를 후위 표기법으로 바꾸는 과정 예]

5 스택을 이용한 후위 표기법 수식의 연산

5.1 후위 표기법 수식의 **연산 방법**
5.2 후위 표기법 수식의 **연산 알고리즘**
5.3 후위 표기법 수식의 **연산 과정**
5.4 후위 표기법 수식 **연산의 구현**

5.1 후위 표기법 수식의 연산 방법

❖ 스택을 사용해 후위 표기법 수식을 연산하는 방법

❶ 피연산자를 만나면 스택에 push한다.

A, B

❷ 연산자를 만나면 필요한 만큼의 피연산자를 스택에서 pop하여 연산하고,

+, /

A, B

연산 결과를 다시 스택에 push한다.

❸ 수식이 끝나면 마지막으로 스택을 pop하여 출력한다.

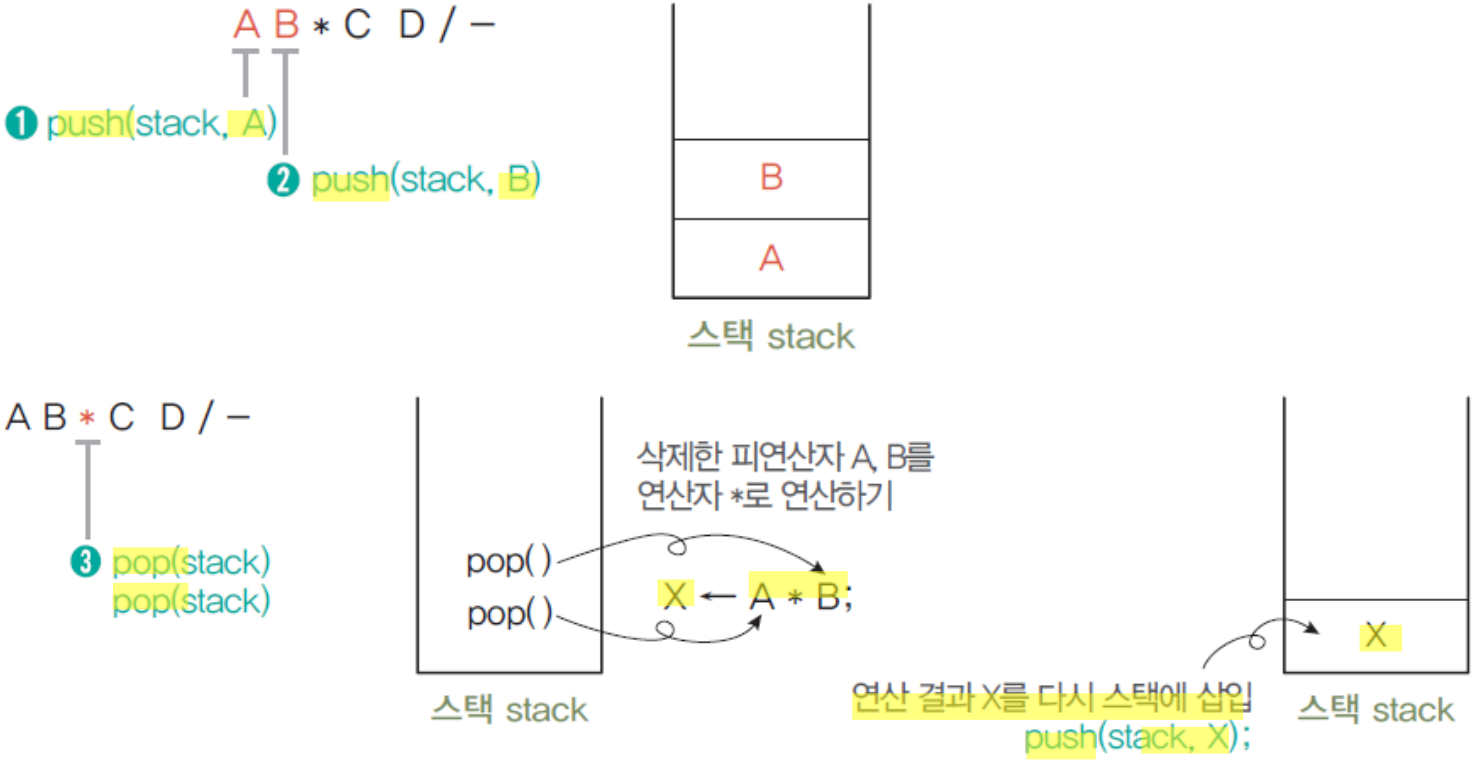
5.2 후위 표기법 수식의 연산 알고리즘

알고리즘 5-5 후위 표기법으로 수식 연산

```
evalPostfix(exp)
    while (true) do {
        symbol ← getSymbol(exp);
        case {
            symbol = operand :  // 피연산자 처리
                        push(Stack, symbol);
            symbol = operator : // 연산자 처리
                        opr2 ← pop(Stack);
                        opr1 ← pop(Stack);
                        // 스택에서 꺼낸 피연산자들을 연산자로 연산
                        result ← opr1 op(symbol) opr2;
                        push(Stack, result);
            symbol = null :     // 후위 수식의 끝
                        print(pop(Stack));
        }
    }
end evalPostfix()
```

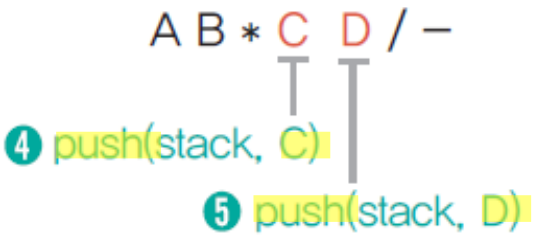
5.3 후위 표기법 수식의 연산 과정

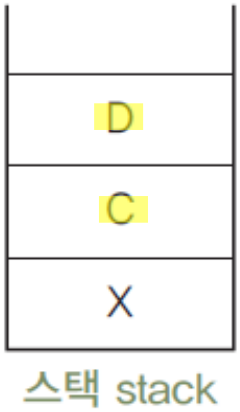
❖ 후위표기법 수식의 연산 알고리즘으로 수식 AB*CD/-를 계산하는 예)



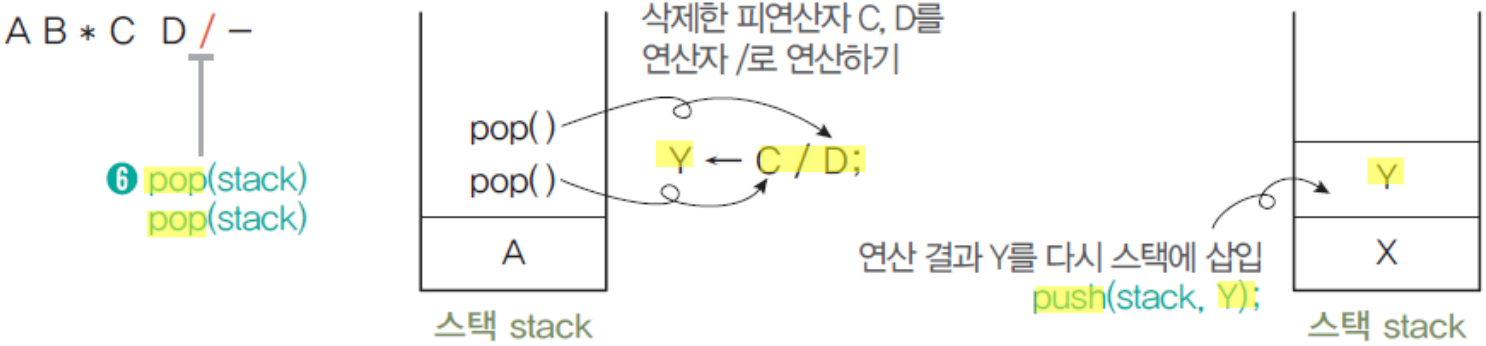
5.3 후위 표기법 수식의 연산 과정

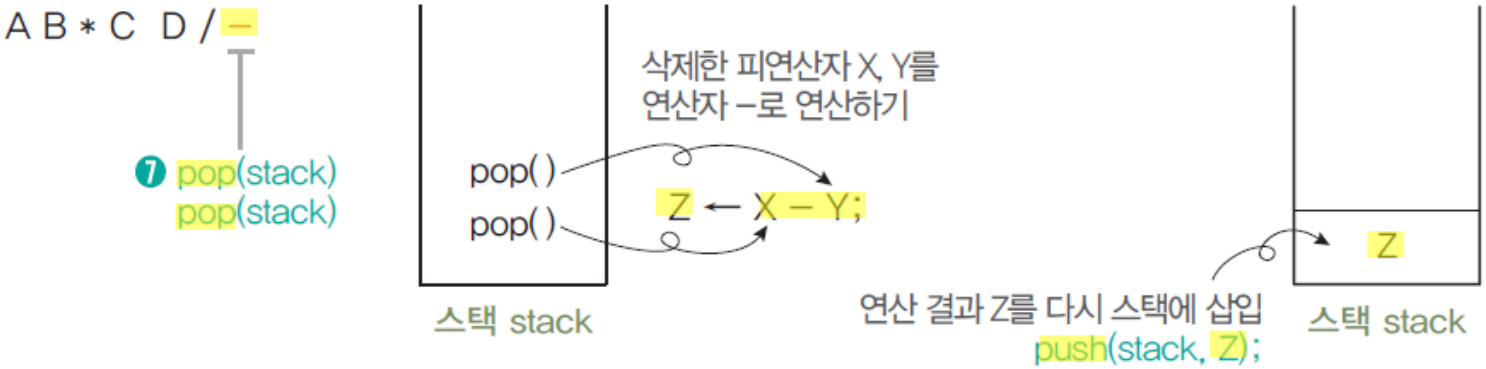
❖ 후위표기법 수식의 연산 알고리즘으로 수식 AB*CD/-를 계산하는 예)





5.3 후위 표기법 수식의 연산 과정





5.4 후위 표기법 수식 연산의 구현

[예제5-4] 수식을 후위 표기법으로 연산하기 프로그램

연결자료구조 스택 코딩 [예제5-2]와 같음

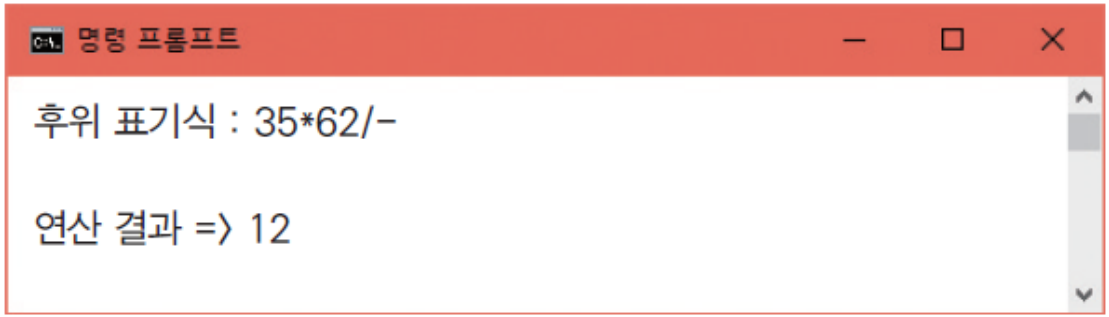
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef int element;// 스택 원소(element)의 자료형을 int로 정의

typedef struct  stackNode {// 스택의 노드를 구조체로 정의
        element data;
        struct stackNode *link;
} stackNode;

stackNode* top;// 스택의 top 노드를 지정하기 위해 포인터 top 선언

// 스택이 공백 상태인지 확인하는 연산
int isEmpty() {
        if (top == NULL) return 1;
        else return 0;
}
```

```
// 스택의 top에 원소를 삽입하는 연산
void push(element item) {
    stackNode* temp = (stackNode *)malloc(sizeof(stackNode));
    temp->data = item;
    temp->link = top;// 삽입 노드를 top의 위에 연결
    top = temp;// top 위치를 삽입 노드로 이동
}
// 스택의 top에서 원소를 삭제하는 연산
element pop() {
    element item;
    stackNode* temp = top;

    if (top == NULL) {// 스택이 공백 리스트인 경우
        printf("₩n₩n Stack is empty !₩n");
        return 0;
    }
    else {// 스택이 공백 리스트가 아닌 경우
        item = temp->data;
        top = temp->link;// top 위치를 삭제 노드 아래로 이동
        free(temp);// 삭제된 노드의 메모리 반환
        return item;// 삭제된 원소 반환
    }
}
```

연결자료구조 스택 코딩 [예제5-2]와 같음

```
// 후위 표기법 수식을 연산(계산)하는 코딩
element evalPostfix(char *exp) {
    int opr1, opr2, value, i = 0;
    int length = strlen(exp); // exp의 길이(수식의 길이)를 계산하여 length 변수에 저장
    char symbol;
    top = NULL;

    for (i = 0; i<length; i++) {    //수식의 길이만큼 반복
        symbol = exp[i];    //자동으로 포인터 배열 exp[i]이 됨. symbol은 문자 1개씩 받음
        if (symbol != '+' && symbol != '-' && symbol != '*' && symbol != '/') { // symbol이 연산자가 아니라면(숫자라면)
            value = symbol - '0';    //char형을 int형으로 바꾸기 위해, 0을 빼서 숫자로 바꿔 value 변수에 저장
            push(value);            //value에 저장된 숫자를 스택에 push한다.
        }
        else {    //연산자라면
            opr2 = pop();
            opr1 = pop();
            switch (symbol) {    // 변수 opr1과 opr2에 대해 symbol에 저장된 연산자를 연산
                case '+': push(opr1 + opr2); break;    //계산 결과를 push
                case '-': push(opr1 - opr2); break;
                case '*': push(opr1 * opr2); break;
                case '/': push(opr1 / opr2); break;
            }
        }

    return pop();  // 수식 exp에 대한 처리를 마친 후 스택에 남아 있는 최종 결과값을 pop하여 반환
}
```

```
void main(void) {
int result;
char* express = "35*62/-";   //수식의 주소를 express에게 줌
printf("후위 표기식 : %s", express);   //포인터 express가 가리키는 곳의 문자열 찍기

result = evalPostfix(express);   //후위표기식 연산 함수 호출
printf("\n\n연산 결과 => %d", result);

getchar();
}
```



학습정리

1 수식의 표기법

- 전위표기법 (prefix notation)
  - 예 +AB
- 중위표기법 (infix notation)
  - 예 A+B
- 후위표기법 (postfix notation)
  - 예 AB+

학습정리

2 후위표기법

- 컴퓨터 내부에서 수식을 처리하는 데 가장 효율적인 방법은 후위 표기법임
- 후위 표기법을 사용하면 괄호나 연산자 우선순위를 따로 처리하지 않고
- 왼쪽에서 오른쪽으로 표기된 순서대로 처리할 수 있음
- 사용자가 컴퓨터에 중위 표기법 형태의 수식을 입력하면
- 컴퓨터 내부에서는 효율적인 처리를 위해 스택을 사용하여 입력된 수식을 후위 표기법으로 변환함