

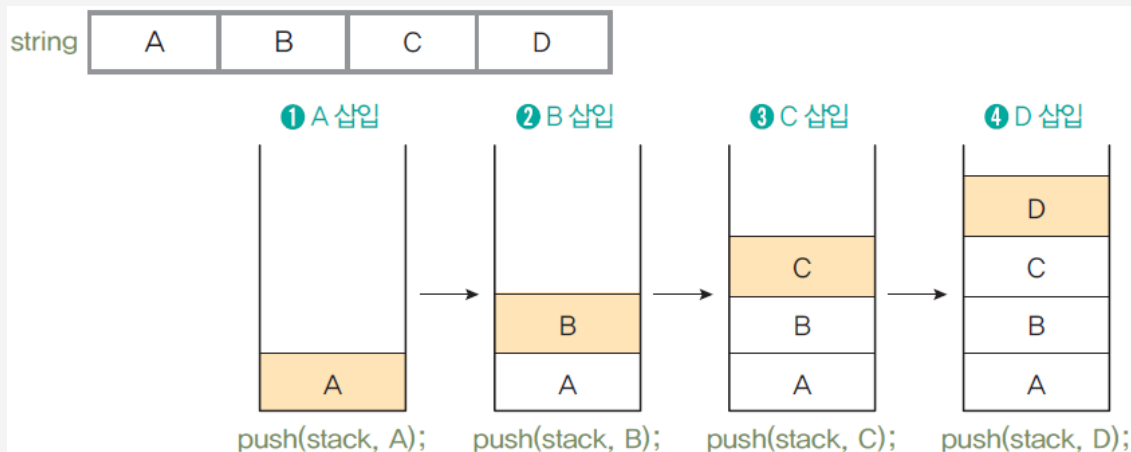
1 | 역순 문자열 만들기

1 | 역순 문자열 만들기

1 역순 문자열 만들기

▶ 스택의 후입선출(LIFO) 성질을 이용

① 문자열을 순서대로 스택에 삽입



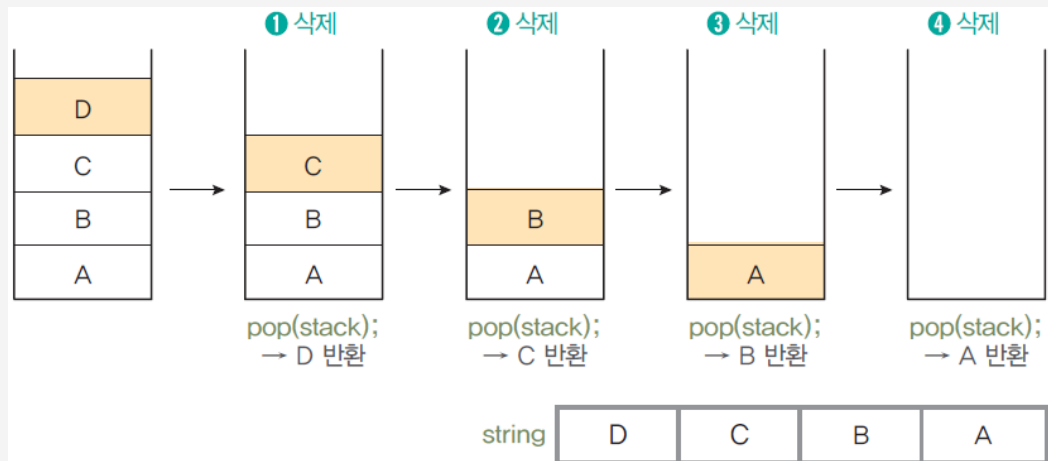
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

1 | 역순 문자열 만들기

1 역순 문자열 만들기

▶ 스택의 후입선출(LIFO) 성질을 이용

② 스택에서 삭제하여 문자열을 만들기



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

2 | 프로그램 호출과 복귀에 따른 수행 순서

2 | 프로그램 호출과 복귀에 따른 수행 순서

1 시스템 스택

- ▶ 프로그램에서의 호출과 복귀에 따른 수행 순서를 관리
 - 가장 마지막에 호출된 함수가 가장 먼저 실행을 완료하고 복귀하는 후입선출 구조이므로, 후입선출 구조의 스택을 이용하여 수행순서 관리
 - 함수 호출이 발생하면 호출한 함수 수행에 필요한 지역변수, 매개변수 및 수행 후 복귀할 주소 등의 정보를 스택 프레임(Stack Frame)에 저장하여 시스템 스택에 삽입

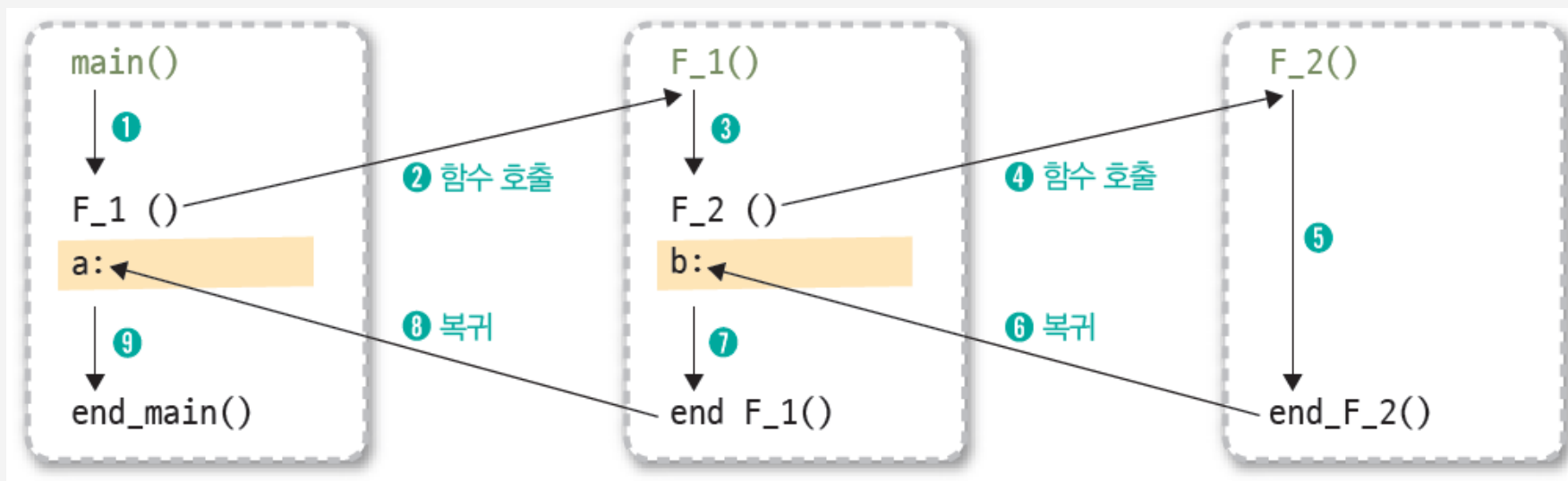
2 | 프로그램 호출과 복귀에 따른 수행 순서

1 시스템 스택

- ▶ 프로그램에서의 호출과 복귀에 따른 수행 순서를 관리
 - 함수의 실행이 끝나면 시스템 스택의 top 원소 (스택 프레임)를 삭제(pop)하면서 프레임에 저장되어있던 복귀주소를 확인하고 복귀
 - 함수 호출과 복귀에 따라 이 과정을 반복하여 전체 프로그램 수행이 종료되면 시스템 스택은 공백스택이 됨

2 | 프로그램 호출과 복귀에 따른 수행 순서

2 함수 호출과 복귀에 따른 전체 프로그램의 수행 순서



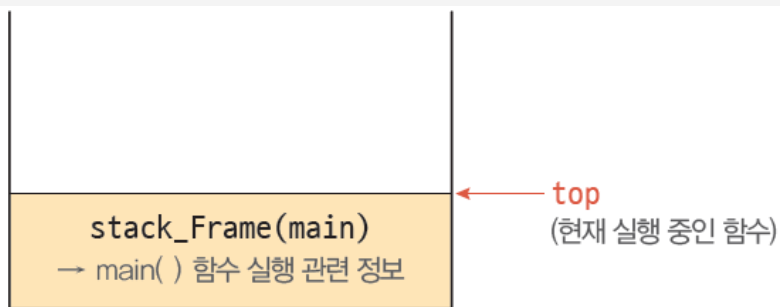
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

2 | 프로그램 호출과 복귀에 따른 수행 순서

2 함수 호출과 복귀에 따른 전체 프로그램의 수행 순서

- ① `main()` 함수 실행 시작 : 시작하면 `main()` 함수가 호출되어 실행,
`main()` 함수 시작과 관련된 정보를 스택 프레임에 저장, 시스템 스택에 삽입

```
push(System_stack, stack_Frame(main));
```



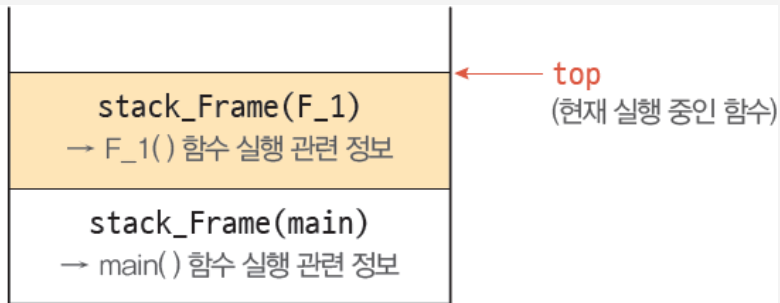
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

2 | 프로그램 호출과 복귀에 따른 수행 순서

2 함수 호출과 복귀에 따른 전체 프로그램의 수행 순서

- ② F_1() 함수 호출 : main() 함수 실행 중 F_1() 함수 호출을 만나면 함수 호출과 복귀에 필요한 정보를 스택 프레임에 저장, 시스템 스택에 삽입, 호출된 함수인 F_1() 함수로 이동. 이때 스택 프레임에는 호출된 함수의 수행이 끝나고 main() 함수로 복귀할 주소 a를 저장

```
push(System_stack, stack_Frame(F_1));
```



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

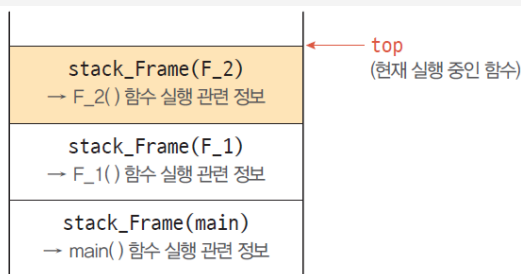
2 | 프로그램 호출과 복귀에 따른 수행 순서

2 함수 호출과 복귀에 따른 전체 프로그램의 수행 순서

③ 호출된 함수 F_1() 함수 실행

④ F_2() 함수 호출 : F_1() 함수 실행 중에 F_2() 함수 호출을 만나면 다시 함수 호출과 복귀에 필요한 정보를 스택 프레임에 저장하여 시스템 스택에 삽입하고, 호출된 함수인 F_2() 함수를 실행. 스택 프레임에는 F_1() 함수로 복귀할 주소 b를 저장

```
push(System_stack, stack_Frame(F_2));
```



※출처: 이지영(2016). IT CookBook,
C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

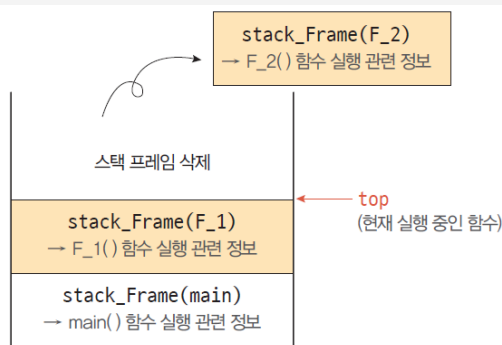
⑤ 호출된 함수 F_2() 함수 실행

2 | 프로그램 호출과 복귀에 따른 수행 순서

2 함수 호출과 복귀에 따른 전체 프로그램의 수행 순서

- ⑥ F₂() 함수 실행 종료, F₁() 함수로 복귀 : F₂() 함수 실행이 끝나면 F₂() 함수를 호출했던 이전 위치로 복귀하여 이전 함수 F₁()의 작업을 계속해야 하며 시스템 스택의 top에 있는 스택 프레임을 pop하여 정보를 확인하고 복귀 및 작업 전환 실행함

pop(System_stack);



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

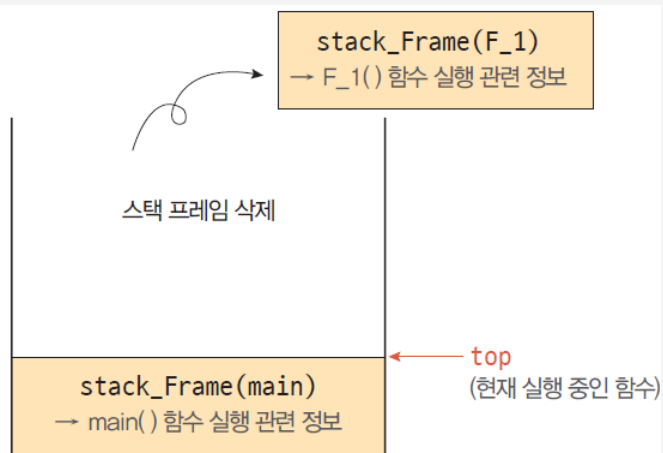
- ⑦ F₁()함수로 복귀하여 F₁()함수의 나머지 부분 실행

2 | 프로그램 호출과 복귀에 따른 수행 순서

2 함수 호출과 복귀에 따른 전체 프로그램의 수행 순서

- ⑧ F_1() 함수 실행 종료, main() 함수로 복귀 : 스택의 top에 있는 스택 프레임을 pop하여 정보를 확인하고 복귀 및 작업 전환을 실행

`pop(System_stack);`



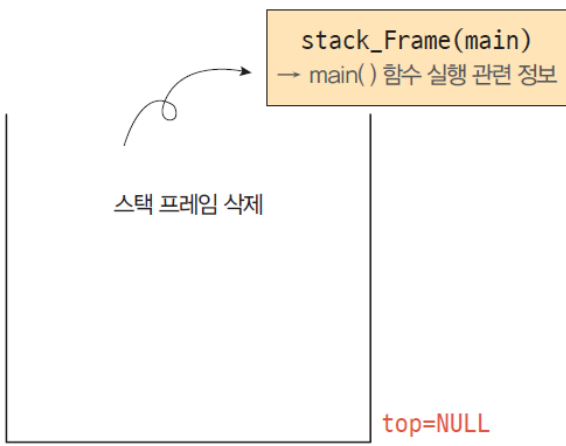
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

2 | 프로그램 호출과 복귀에 따른 수행 순서

2 함수 호출과 복귀에 따른 전체 프로그램의 수행 순서

- ⑨ `main()` 함수 실행 완료(전체 프로그램 실행 완료) :
정상적인 함수 호출과 복귀가 모두 완료되었으므로 시스템 스택은 공백이 됨

`pop(System_stack);`



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

3 | 수식의 괄호 검사와 수식의 후위 표기

1 수식의 괄호 검사

- ▶ 수식에 포함되어있는 괄호는 가장 마지막에 열린 괄호를 가장 먼저 닫아 주어야 하는 후입선출 구조로 구성되어있으므로, 후입선출 구조의 스택을 이용하여 괄호를 검사
- ▶ 수식을 왼쪽에서 오른쪽으로 하나씩 읽으면서 괄호 검사
 - ① 왼쪽 괄호를 만나면 스택에 **push**
 - ② 오른쪽 괄호를 만나면 스택을 **pop**하여 마지막에 저장한 괄호와 같은 종류인지를 확인
 - 같은 종류의 괄호가 아닌 경우 괄호의 짝이 잘못 사용된 수식임

3 | 수식의 괄호 검사와 수식의 후위 표기

1 수식의 괄호 검사

- ▶ 수식에 대한 검사가 모두 끝났을 때 스택은 공백 스택이 됨
 - 수식이 끝났어도 스택이 공백이 되지 않으면 괄호의 개수가 틀린 수식임

3 | 수식의 괄호 검사와 수식의 후위 표기

1 수식의 괄호 검사

알고리즘 5-3 수식의 괄호 쌍 검사

```
testPair()
  exp ← Expression;
  Stack ← null;
  while (true) do {
    symbol ← getSymbol(exp);
    case {
      symbol = "(" or "[" or "{" :
        push(Stack, symbol);
      symbol = ")" :
        open_pair ← pop(Stack);
        if (open_pair ≠ "(") then return false;
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

1 수식의 괄호 검사

```
symbol = "]" :  
    open_pair ← pop(Stack);  
    if (open_pair ≠ "[") then return false;  
symbol = "]" :  
    open_pair ← pop(Stack);  
    if (open_pair ≠ "{") then return false;  
symbol = null :  
    if (isEmpty(Stack)) then return true;  
    else return false;  
else :  
}  
}  
end testPair()
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

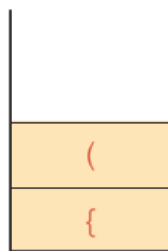
2 예문 : 수식 처리에 따른 스택 사용 과정

$$\{(A+B)-3\}*5+[\{\cos(x+y)+7\}-1]*4$$

{ (A + B) - 3 } * 5 + [{ cos (x + y) + 7 } - 1] * 4

└─ ② push(stack, ()

└─ ① push(stack, {)

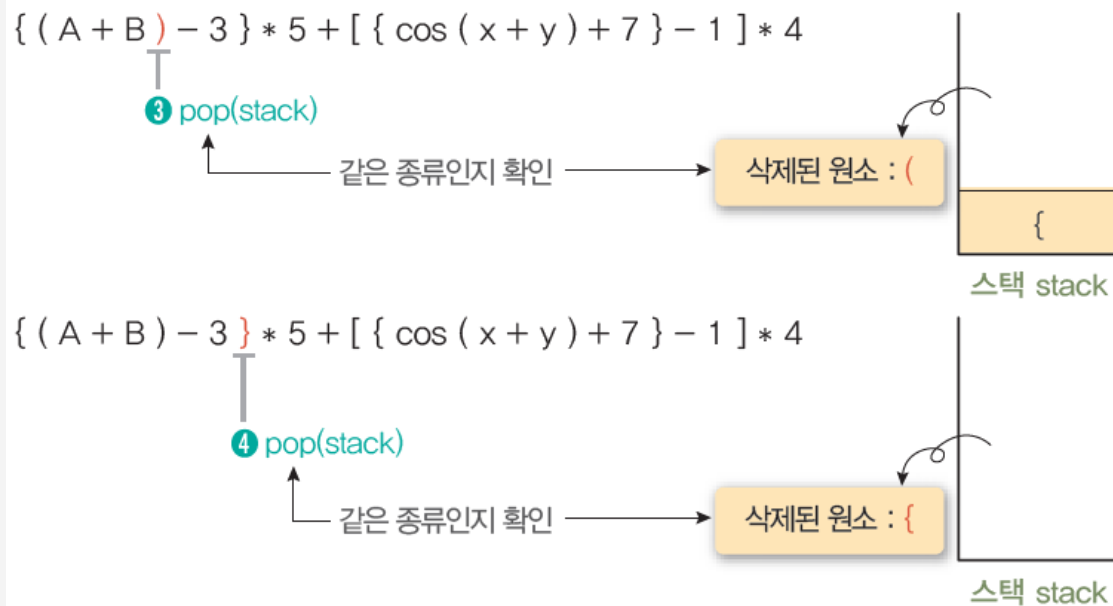


스택 stack

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

2 예문 : 수식 처리에 따른 스택 사용 과정

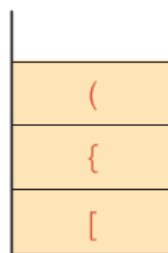
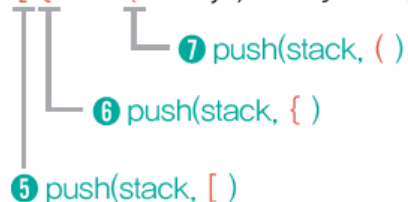


※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

2 예문 : 수식 처리에 따른 스택 사용 과정

$\{ (A + B) - 3 \} * 5 + [\{ \cos (x + y) + 7 \} - 1] * 4$



스택 stack

$\{ (A + B) - 3 \} * 5 + [\{ \cos (x + y) + 7 \} - 1] * 4$

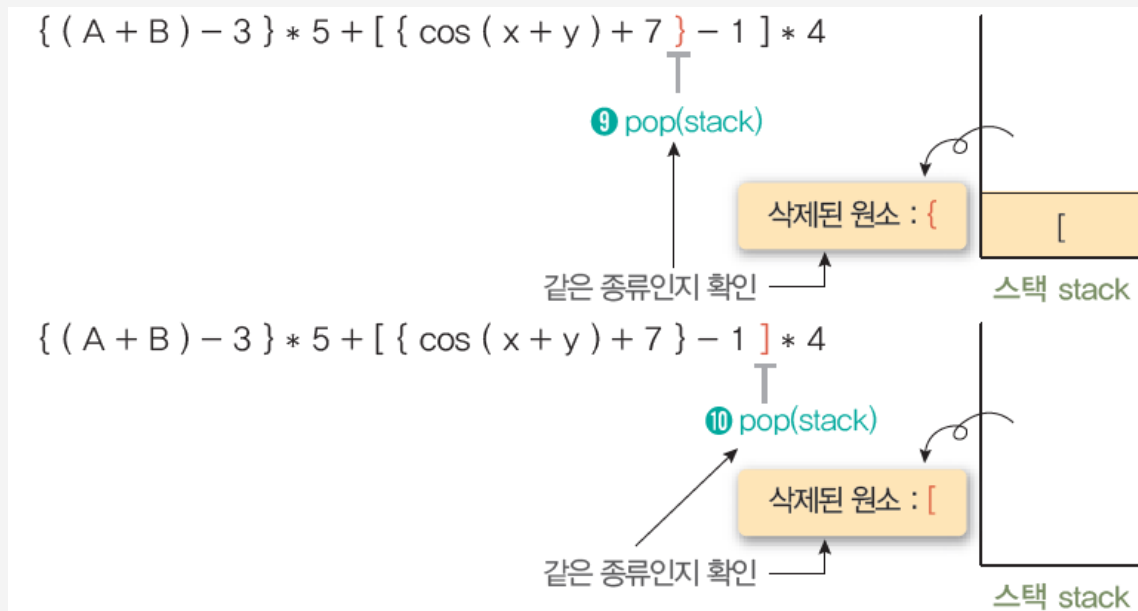


스택 stack

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

2 예문 : 수식 처리에 따른 스택 사용 과정



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

3 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) $AB+$

3 | 수식의 괄호 검사와 수식의 후위 표기

4 중위표기식의 전위표기식 변환 방법

- ▶ 수식의 각 연산자에 대해서 우선순위에 따라 괄호를 사용하여 다시 표현
 - $((A * B) - (C / D))$
- ▶ 각 연산자를 그에 대응하는 왼쪽 괄호의 앞으로 이동시킴
 - $-(*(A B) /(C D))$
- ▶ 괄호를 제거
 - $- * A B / C D$

3 | 수식의 괄호 검사와 수식의 후위 표기

5 중위표기식의 후위표기식 변환 방법

- ▶ 수식의 각 연산자에 대해서 우선순위에 따라 괄호를 사용하여 다시 표현
 - $((A * B) - (C / D))$
- ▶ 각 연산자를 그에 대응하는 오른쪽 괄호의 앞으로 이동시킴
 - $((A B) * (C D) /) -$
- ▶ 괄호를 제거
 - $A B * C D / -$

3 | 수식의 괄호 검사와 수식의 후위 표기

5 중위표기식의 후위표기식 변환 방법

▶ 컴퓨터 내부에서 스택을 사용해 중위 표기법을 후위 표기법으로 바꾸는 방법

- ❶ 왼쪽 괄호를 만나면 무시하고 다음 문자를 읽는다.
- ❷ 피연산자를 만나면 출력한다.
- ❸ 연산자를 만나면 스택에 삽입한다.
- ❹ 오른쪽 괄호를 만나면 스택을 pop하여 출력한다.
- ❺ 수식이 끝나면 스택이 공백이 될 때까지 pop하여 출력한다.

[컴퓨터 내부에서 중위 표기법을 후위 표기법으로 바꾸는 방법]

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

5 중위표기식의 후위표기식 변환 방법

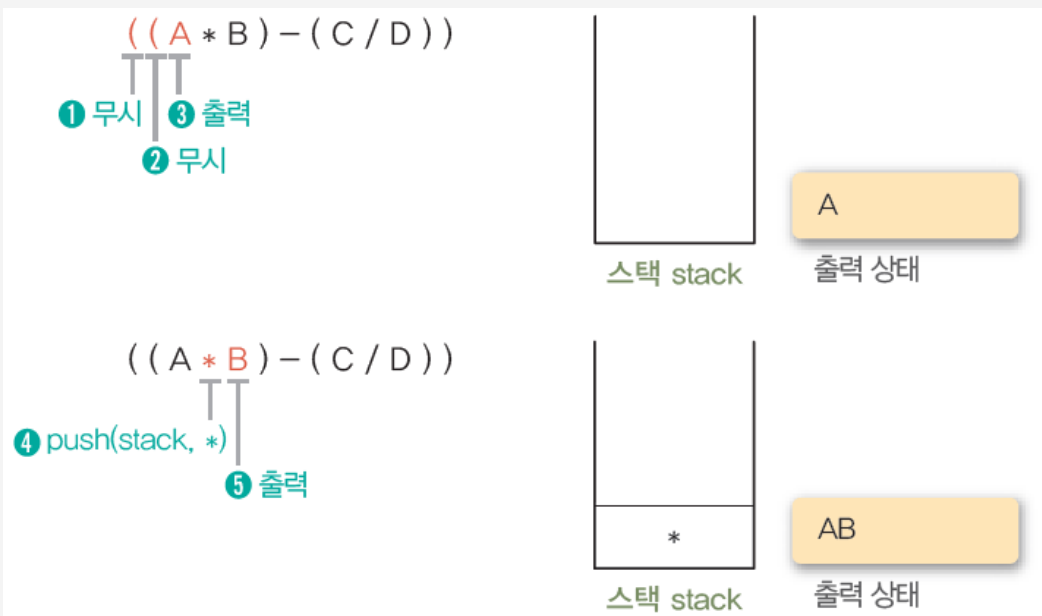
알고리즘 5-4 후위 표기법으로 변환

```
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));
    }
  }
end infix_to_postfix()
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

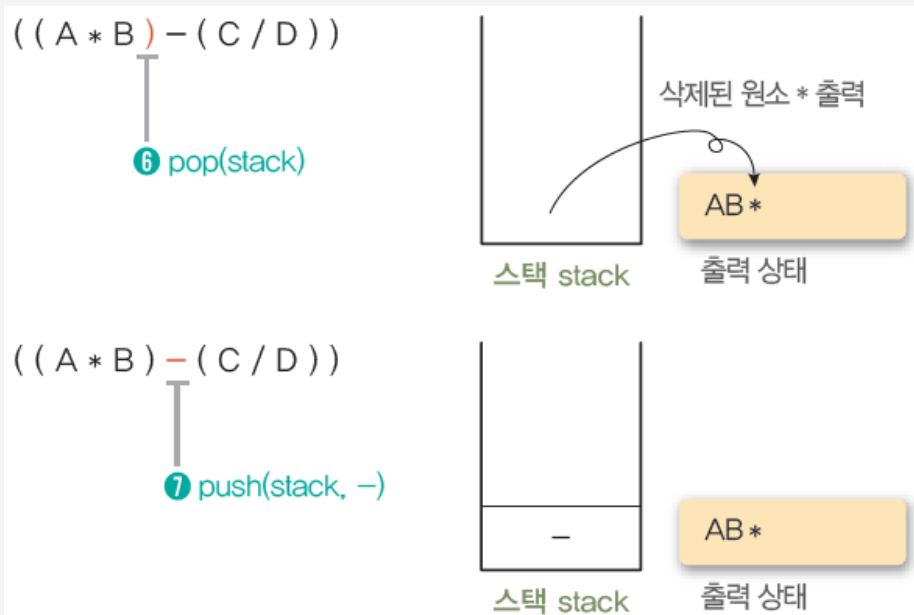
6 스택을 이용하여 수식 $A*B-C/D$ 를 후위 표기법으로 변환



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

6 스택을 이용하여 수식 $A*B-C/D$ 를 후위 표기법으로 변환



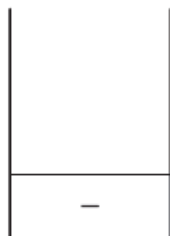
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

6 스택을 이용하여 수식 $A*B-C/D$ 를 후위 표기법으로 변환

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

⑧ 무시
⑨ 출력



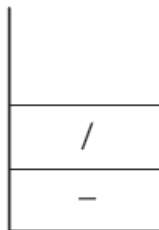
스택 stack

AB * C

출력 상태

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

⑩ push(stack, /)
⑪ 출력



스택 stack

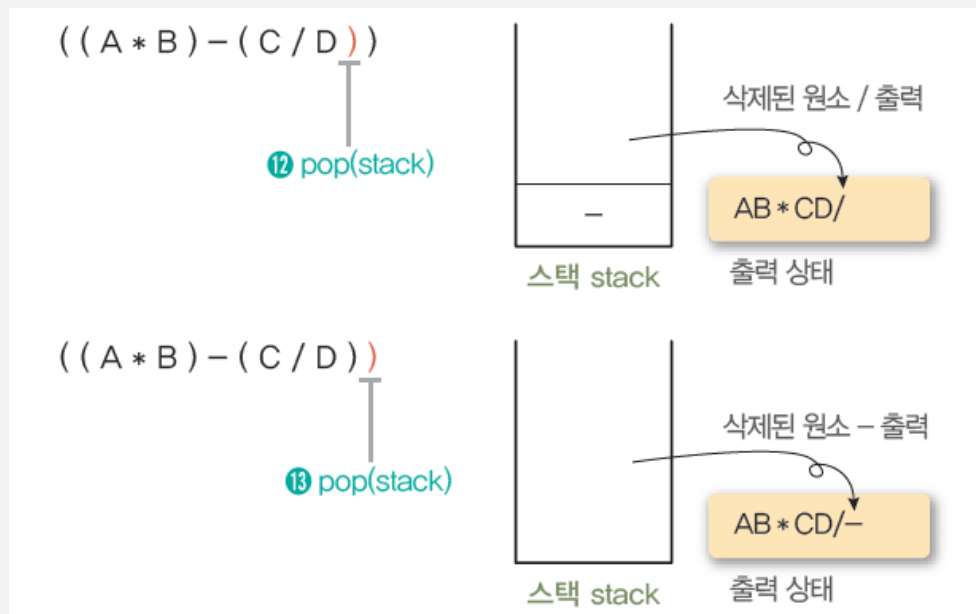
AB * CD

출력 상태

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

6 스택을 이용하여 수식 $A*B-C/D$ 를 후위 표기법으로 변환



※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

7 스택을 이용한 후위 표기법 수식의 연산

▶ 스택을 사용해 후위 표기법 수식을 계산하는 방법

- 1 피연산자를 만나면 스택에 push한다.
- 2 연산자를 만나면 필요한 만큼의 피연산자를 스택에서 pop하여 연산하고, 연산 결과를 다시 스택에 push한다.
- 3 수식이 끝나면 마지막으로 스택을 pop하여 출력한다.

[스택을 사용해 후위 표기법 수식을 계산하는 방법]

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

7 스택을 이용한 후위 표기법 수식의 연산

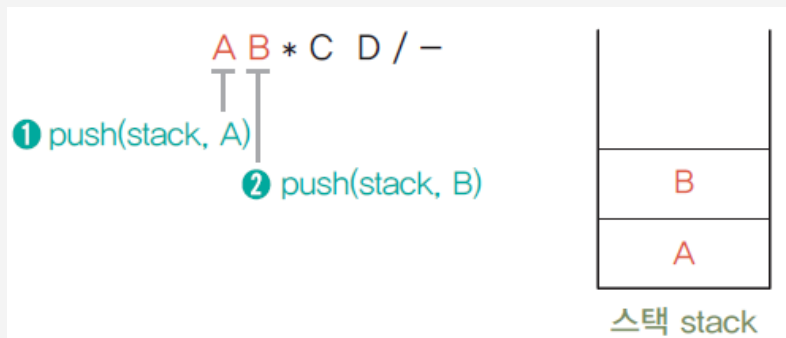
알고리즘 5-5 후위 표기법으로 수식 연산

```
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));
    }
}
```

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

8 위의 알고리즘으로 수식 $AB*CD/-$ 를 연산

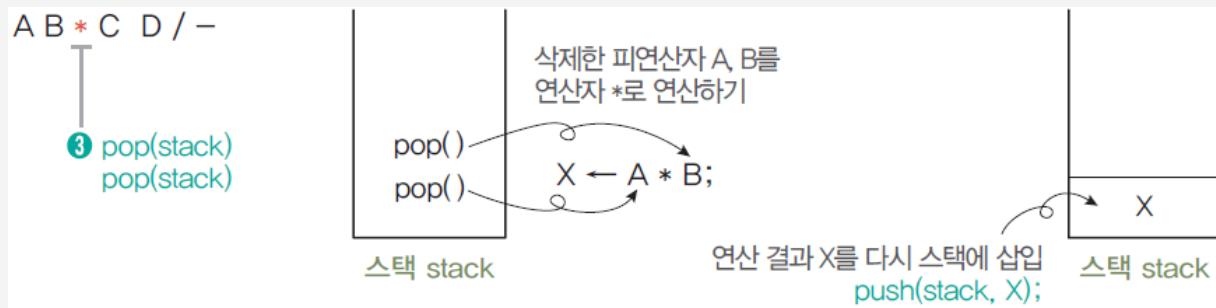


[스택을 사용한 후위 표기법 수식의 연산과정]

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

8 위의 알고리즘으로 수식 $AB*CD/-$ 를 연산

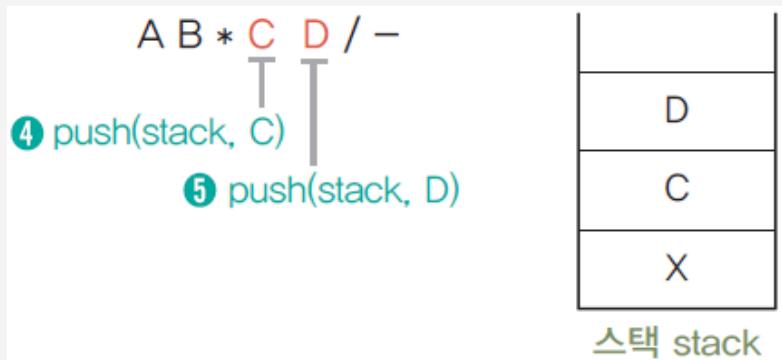


[스택을 사용한 후위 표기법 수식의 연산과정]

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

8 위의 알고리즘으로 수식 $AB*CD/-$ 를 연산

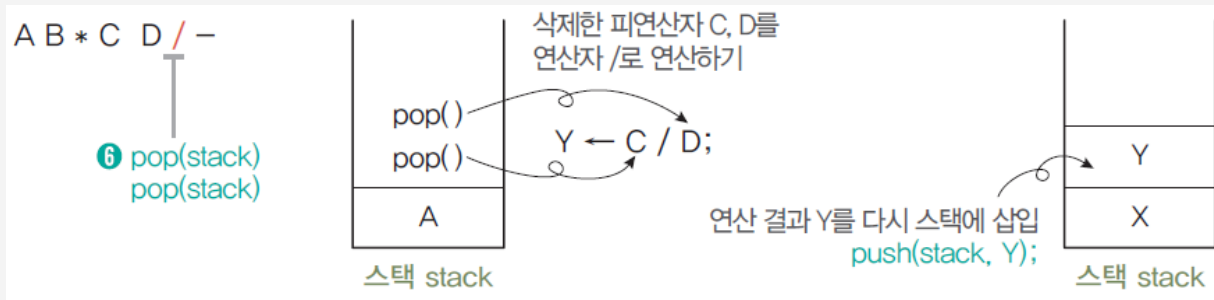


[스택을 사용한 후위 표기법 수식의 연산과정]

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

3 | 수식의 괄호 검사와 수식의 후위 표기

8 위의 알고리즘으로 수식 $AB*CD/-$ 를 연산

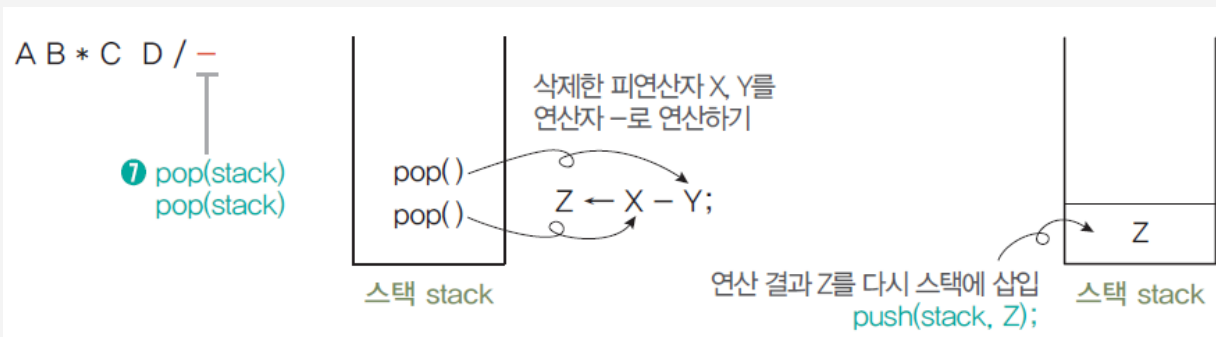


[스택을 사용한 후위 표기법 수식의 연산과정]

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어



8 위의 알고리즘으로 수식 $AB*CD/-$ 를 연산



[스택을 사용한 후위 표기법 수식의 연산과정]

※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어