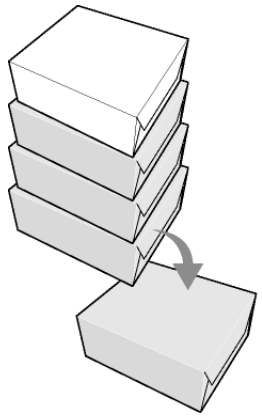
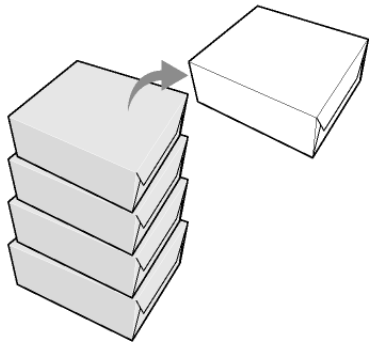
스택

목차

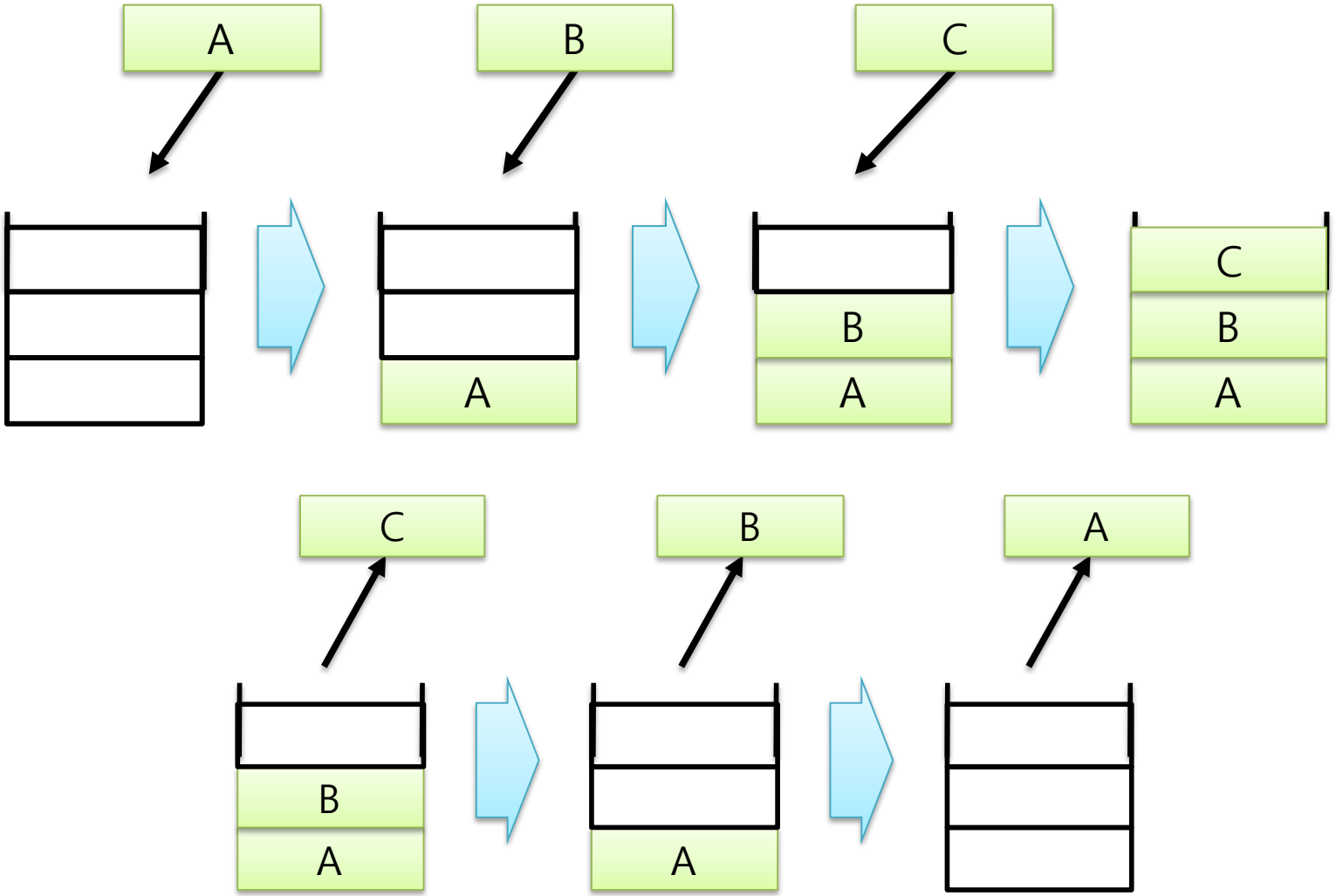
1. 스택의 개념 (1/3)

- 스택(Stack)
  - 일반적인 의미
    - 쌓다, 더미
  - 자료구조에서의 의미
    - 선형 자료구조 & LIFO
  - LIFO(Last-In-First-Out)
    - 후입선출(後入先出)
  - 사용 예
    - 택배에 사용되는 상자 더미(스택)





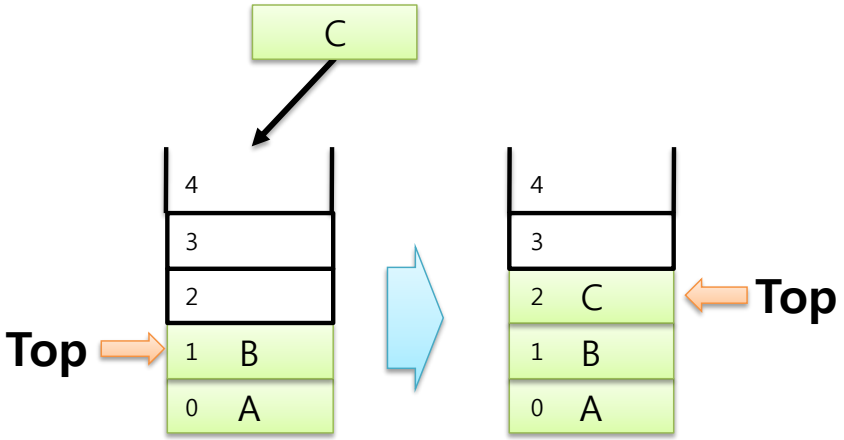
1. 스택의 개념 (2/3)



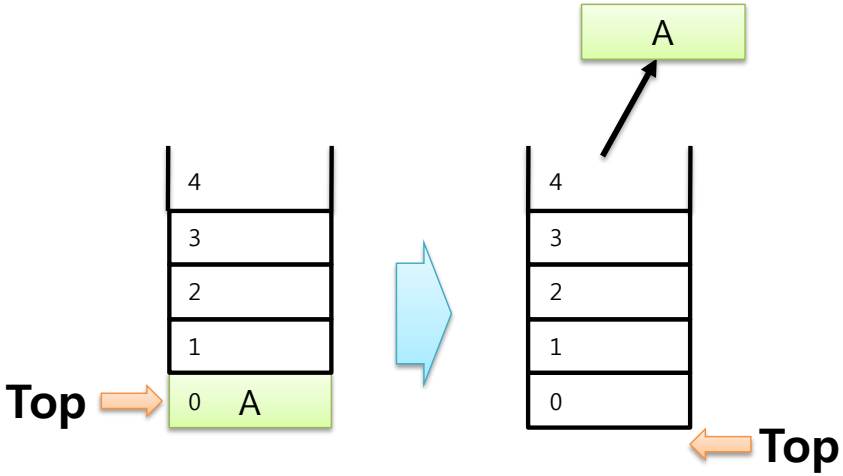
탑(Top) : **가장 최근에 스택에 삽입된 자료를 가리키는 것**

1. 스택의 개념 (3/3)

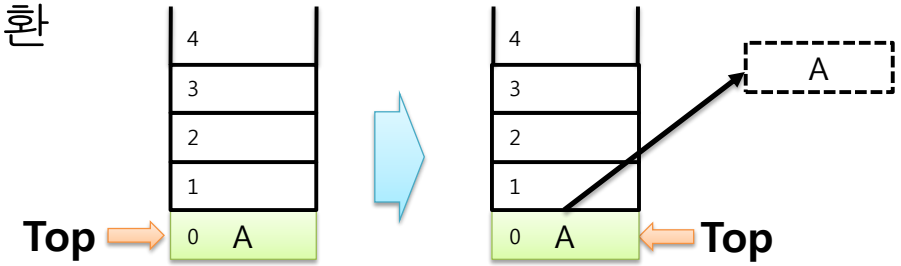
- 3가지 연산
  - 푸시(Push)
    - 넘침(Overflow) 현상



  - 팝(Pop)
    - 부족(Underflow) 현상



  - 피크(Peek)
    - 스택의 맨 위에 있는 원소를 반환 (스택에서 제거하지는 않음)



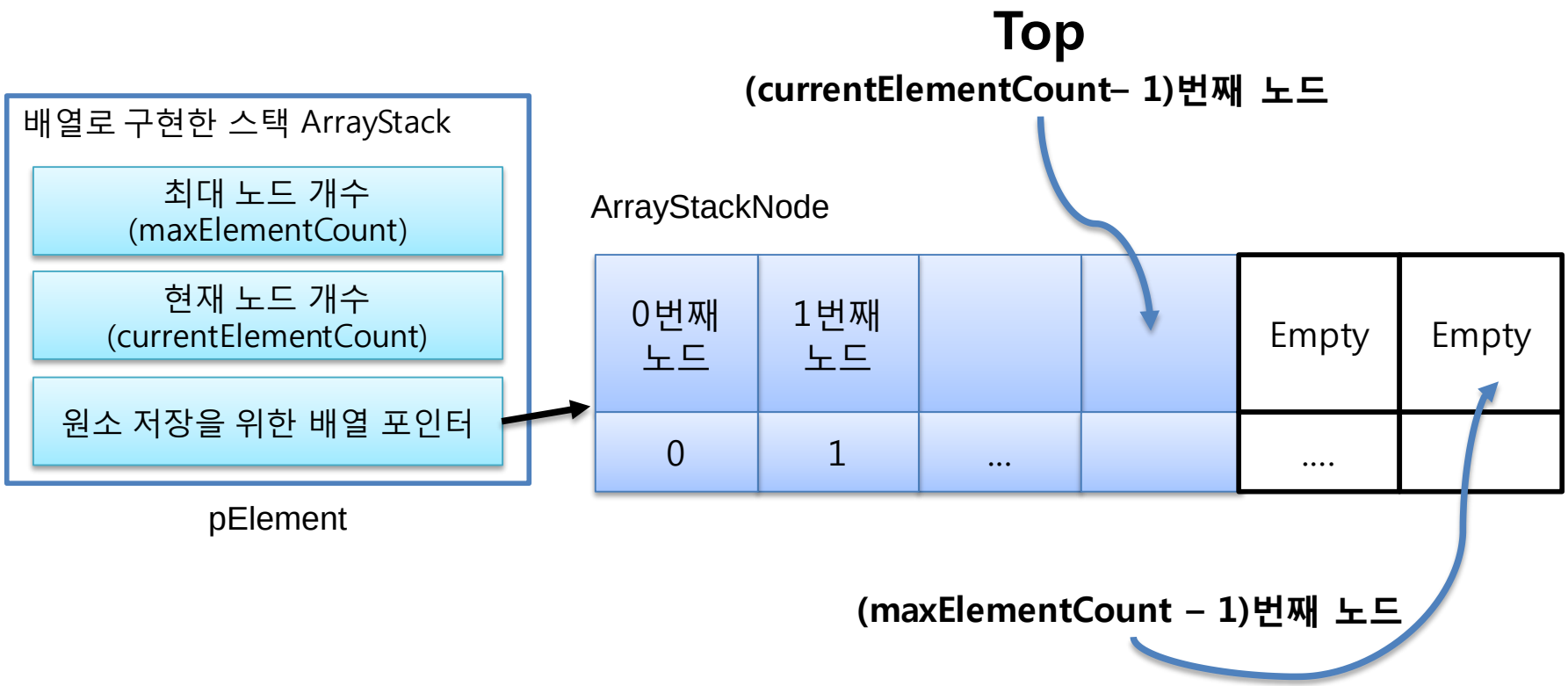
2. 스택 추상 자료형

- 스택 생성
  - 스택의 크기 n

- 푸시(Push)
  - 원소 추가 가능 여부 판단
- 팝(Pop)
  - 공백 스택인지 여부 판단
- 피크(Peek)

- 스택 삭제

3. 배열로 구현한 스택 (2/7)

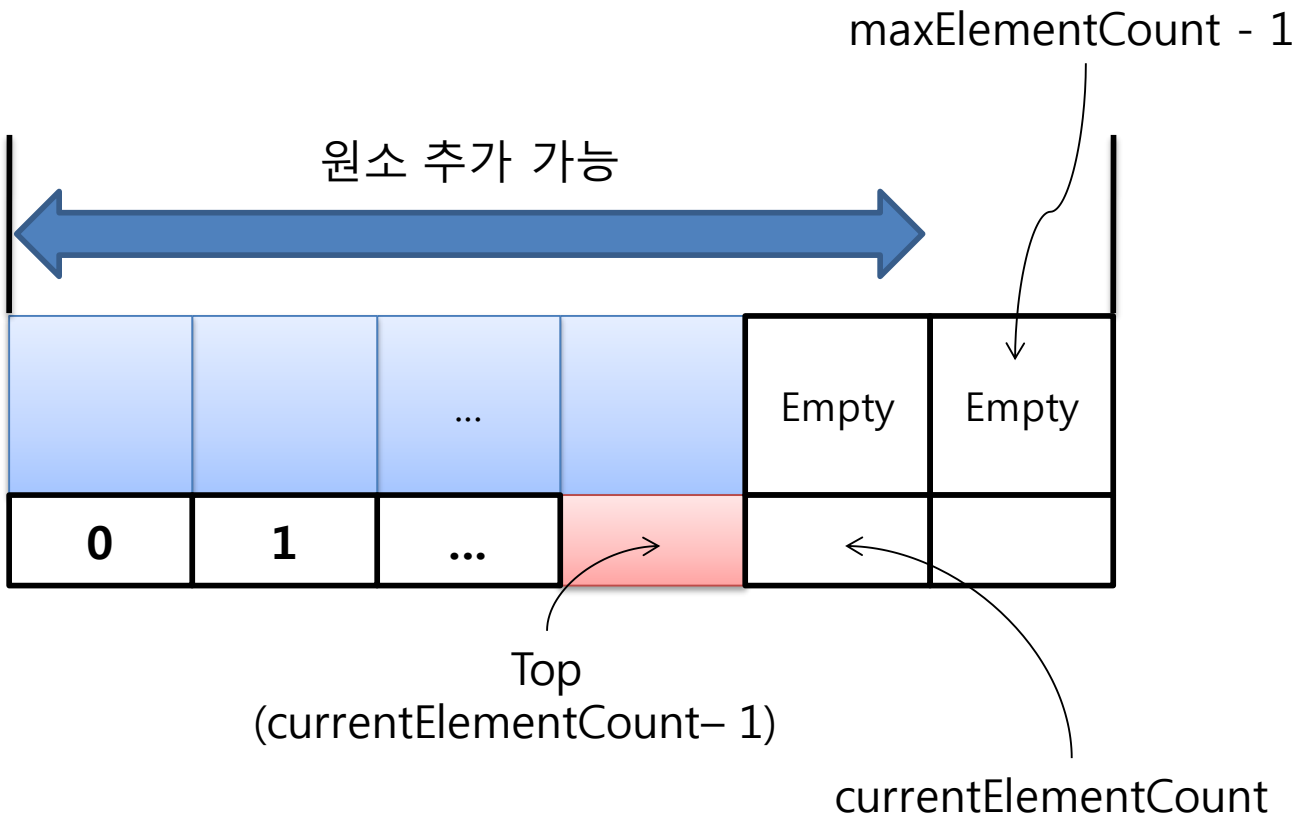


3. 배열로 구현한 스택 (3/7)

- 스택의 생성
- 푸시
- 팝과 피크
- 기타
- 예제 프로그램
  - example04_01.c

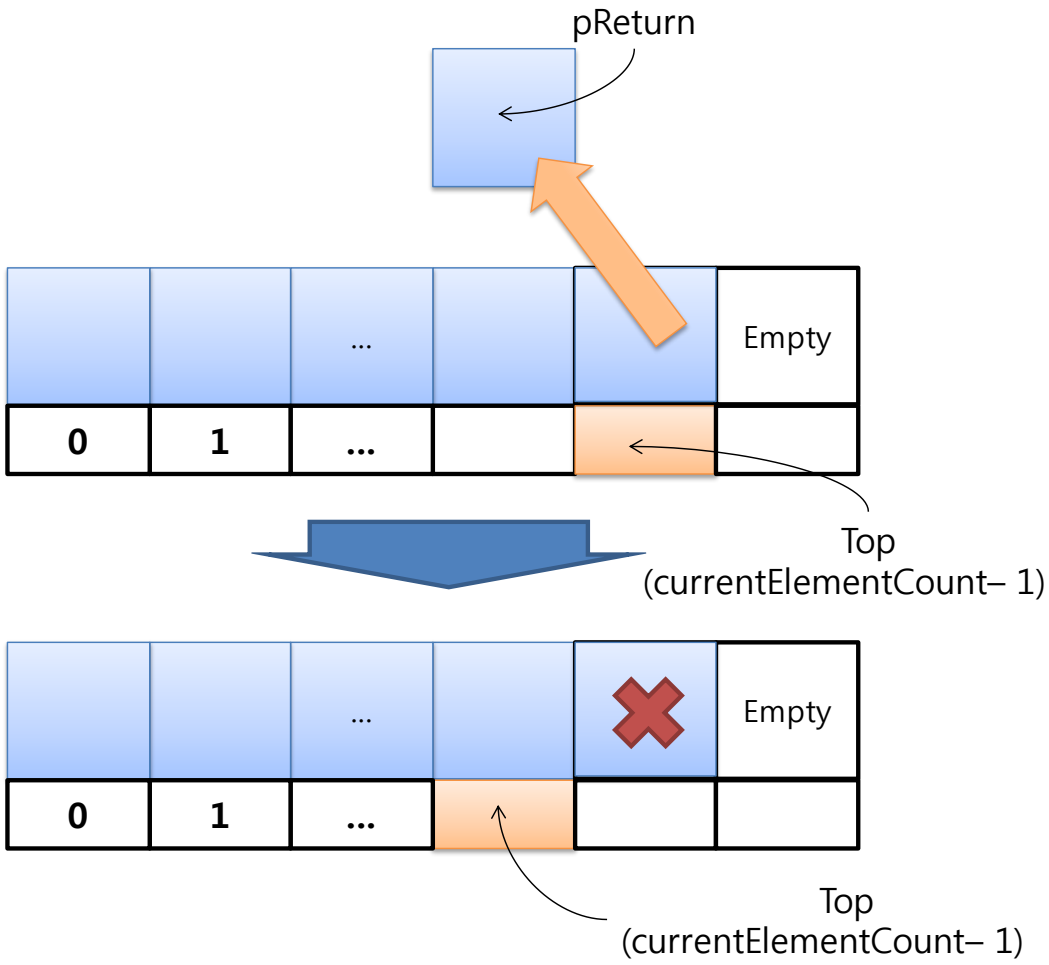
3. 배열로 구현한 스택 (4/7)

- 푸시



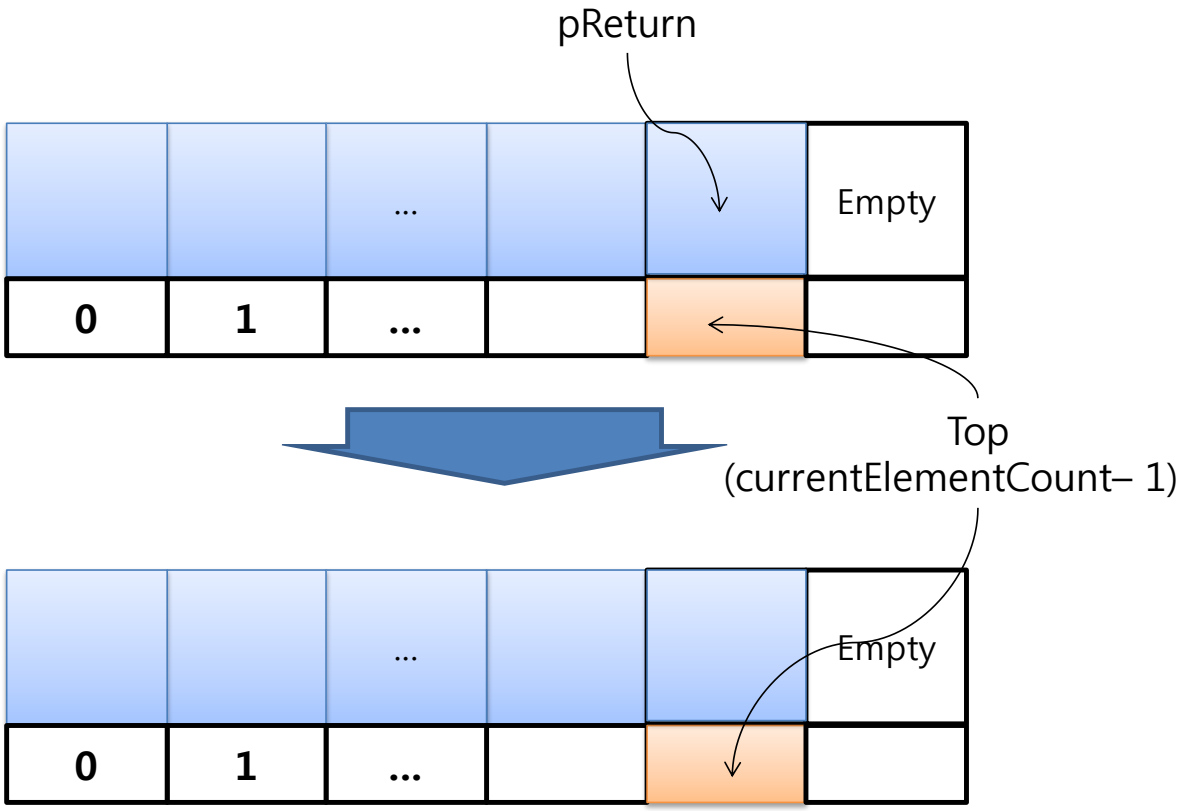
3. 배열로 구현한 스택 (5/7)

- 팝



3. 배열로 구현한 스택 (6/7)

- 피크

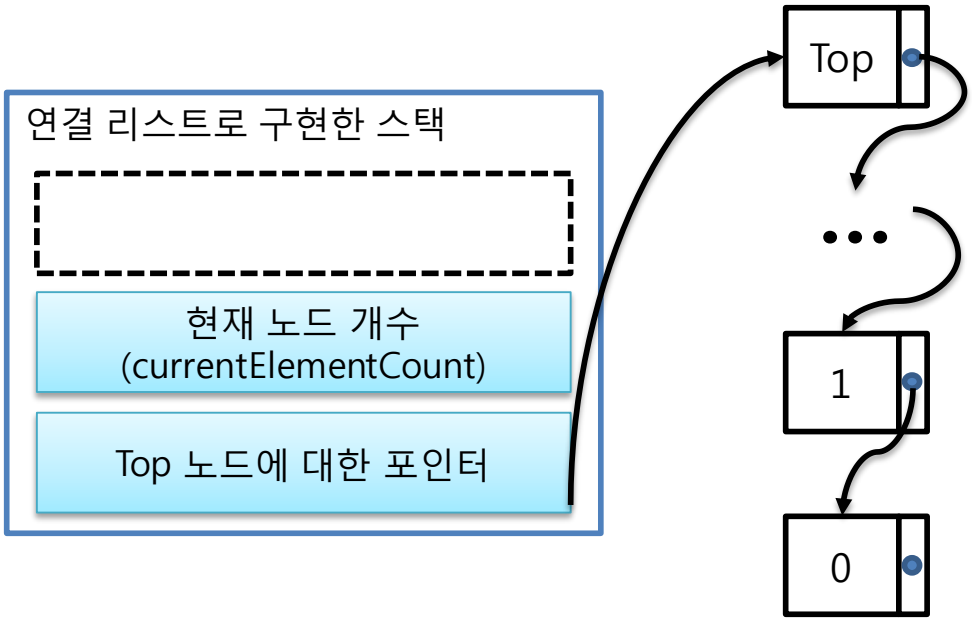


3. 배열로 구현한 스택 (7/7)

- 예제 프로그램
  - 시나리오
    - 푸시: A B C D
    - 팝 1회
    - 피크 1회

4. 연결 리스트로 구현한 스택 (1/8)

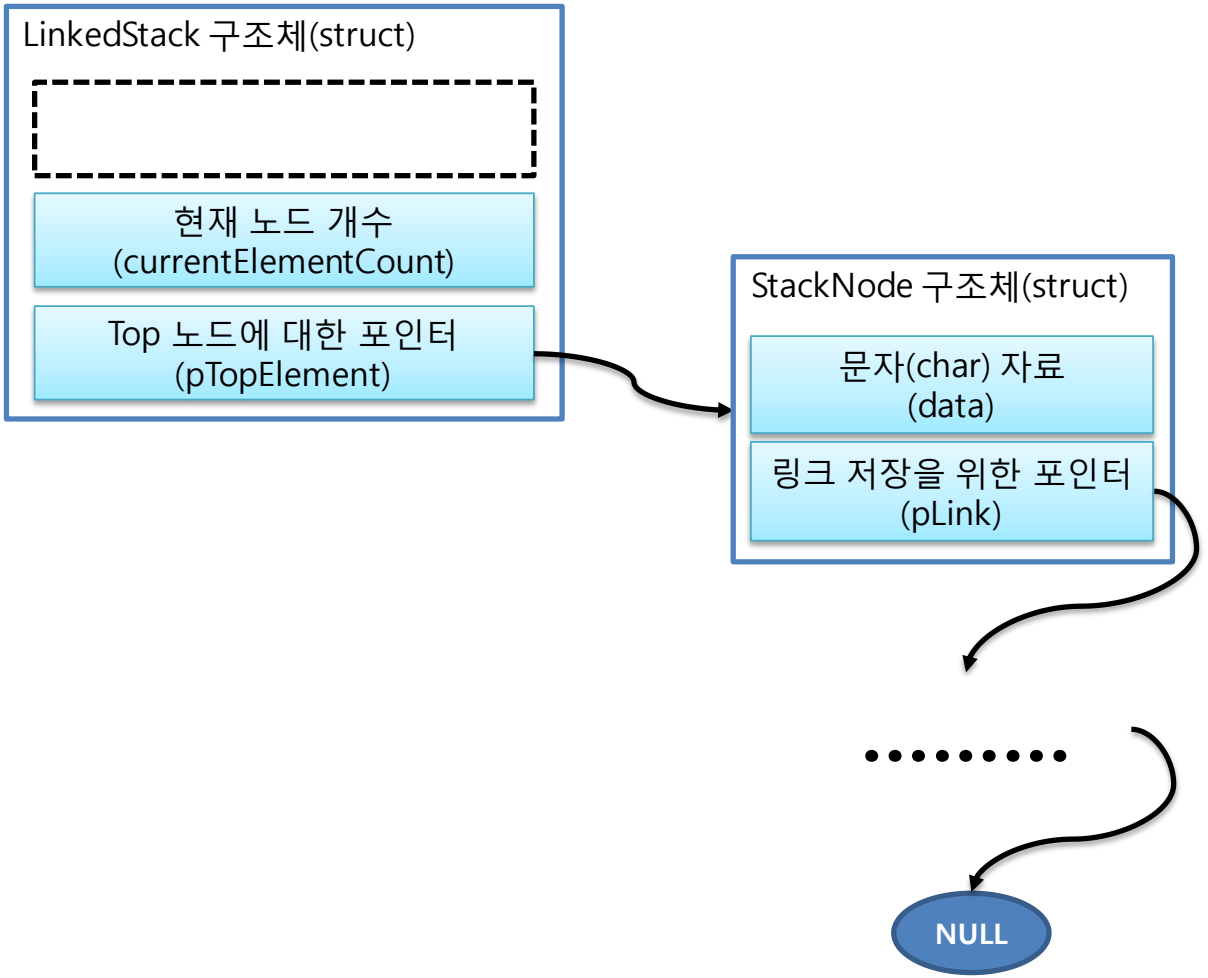
- 연결 리스트(Linked List)로 구현한 스택



- 차이점
  - 스택의 크기 지정

4. 연결 리스트로 구현한 스택 (3/8)

- 구조체

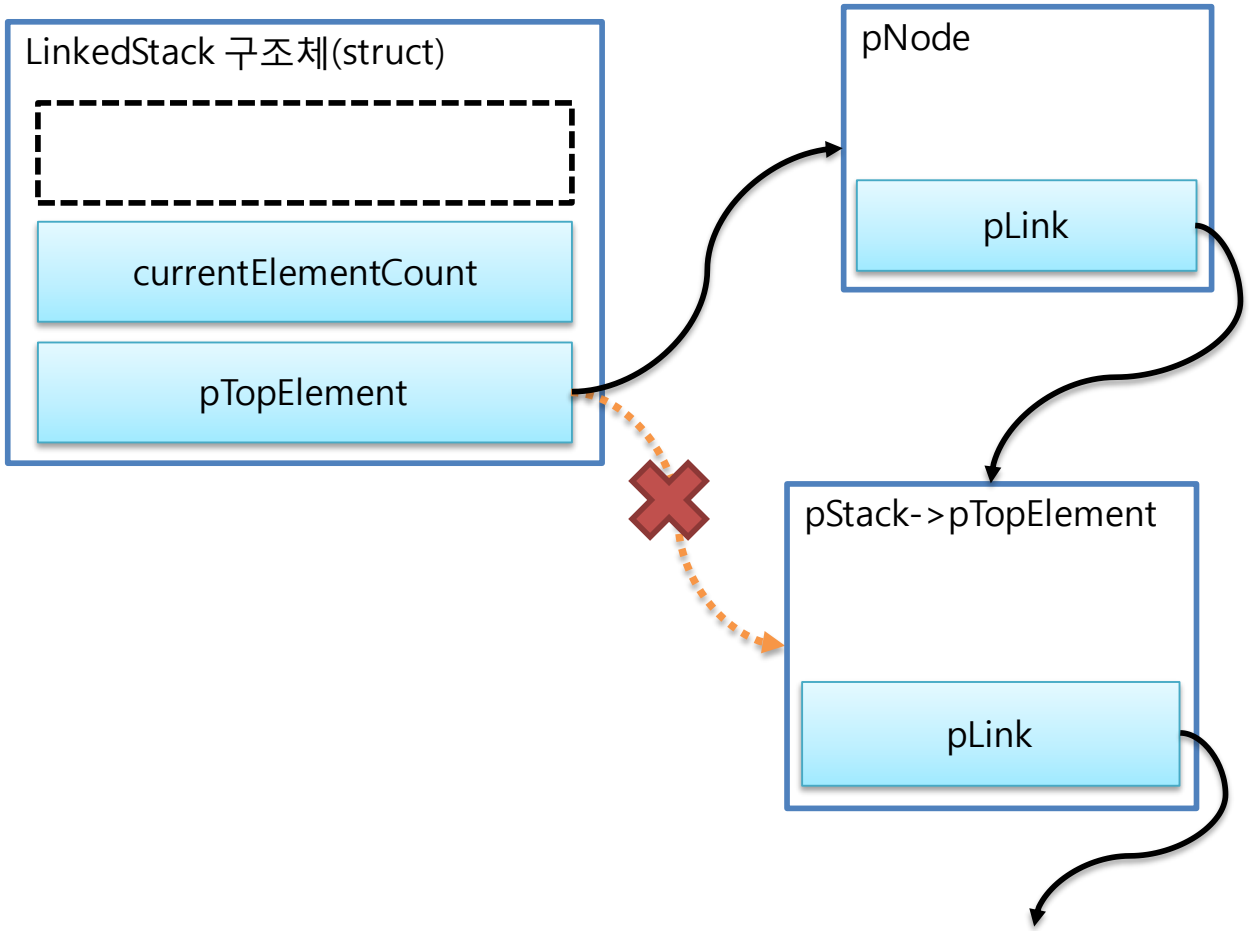


4. 연결 리스트로 구현한 스택 (4/8)

- 스택의 생성
- 푸시
- 팝과 피크
- 기타
- 예제 프로그램

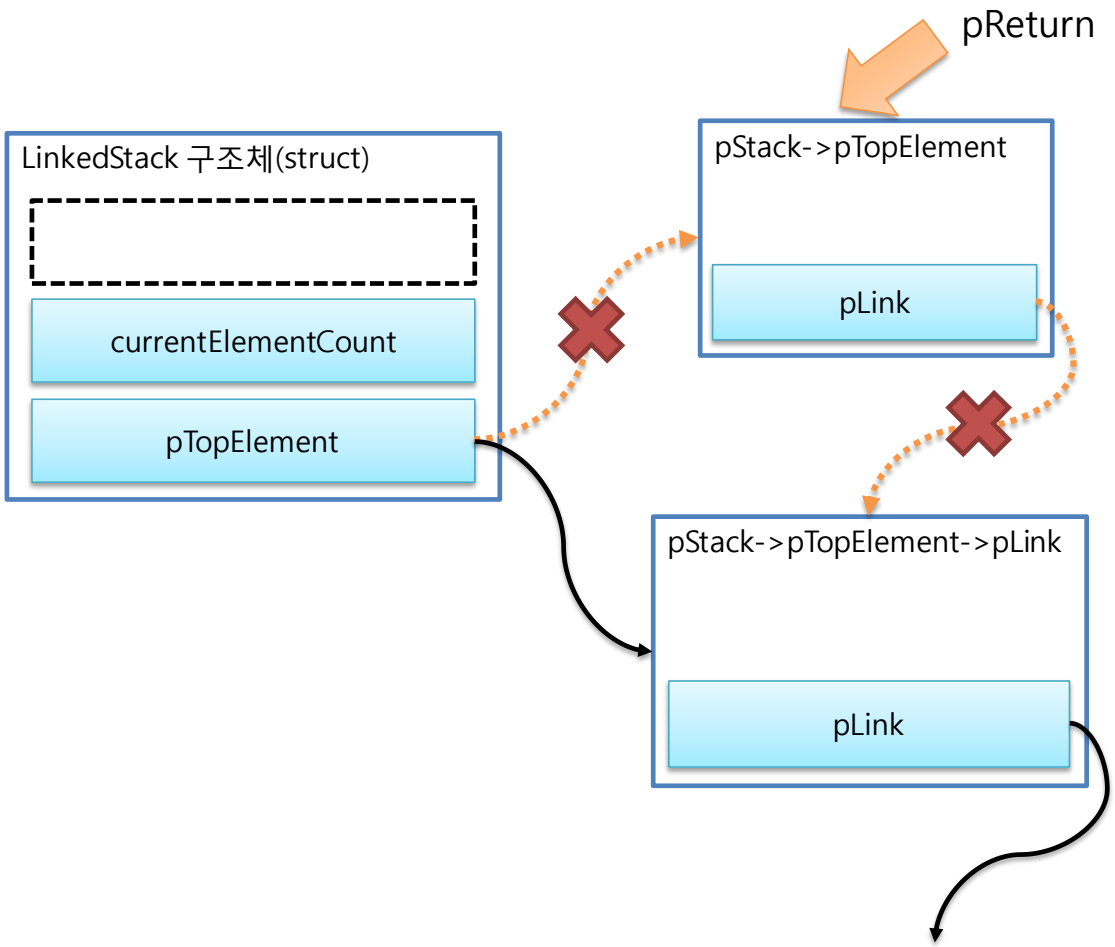
4. 연결 리스트로 구현한 스택 (5/8)

- 푸시



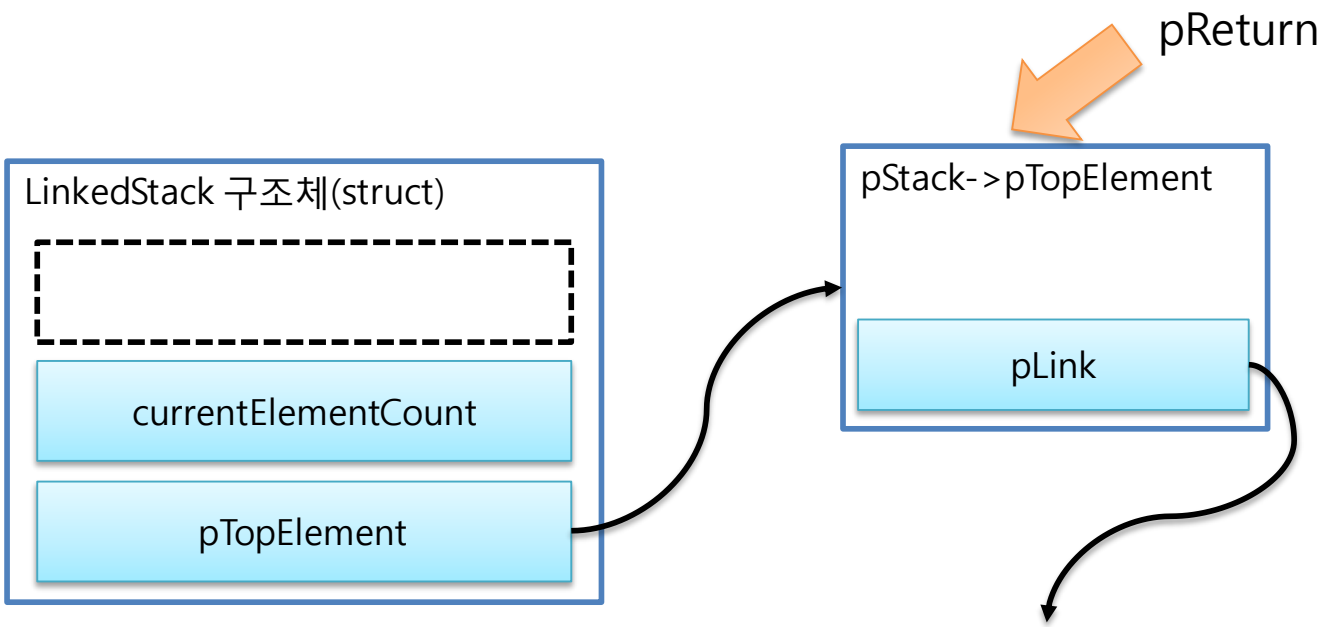
4. 연결 리스트로 구현한 스택 (6/8)

- 팝



4. 연결 리스트로 구현한 스택 (7/8)

- 피크



4. 연결 리스트로 구현한 스택 (8/8)

- 기타 연산들
  - 스택 제거
    - deleteLinkedList()

5.1 스택 응용 1: 역순 문자열 만들기 (1/4)

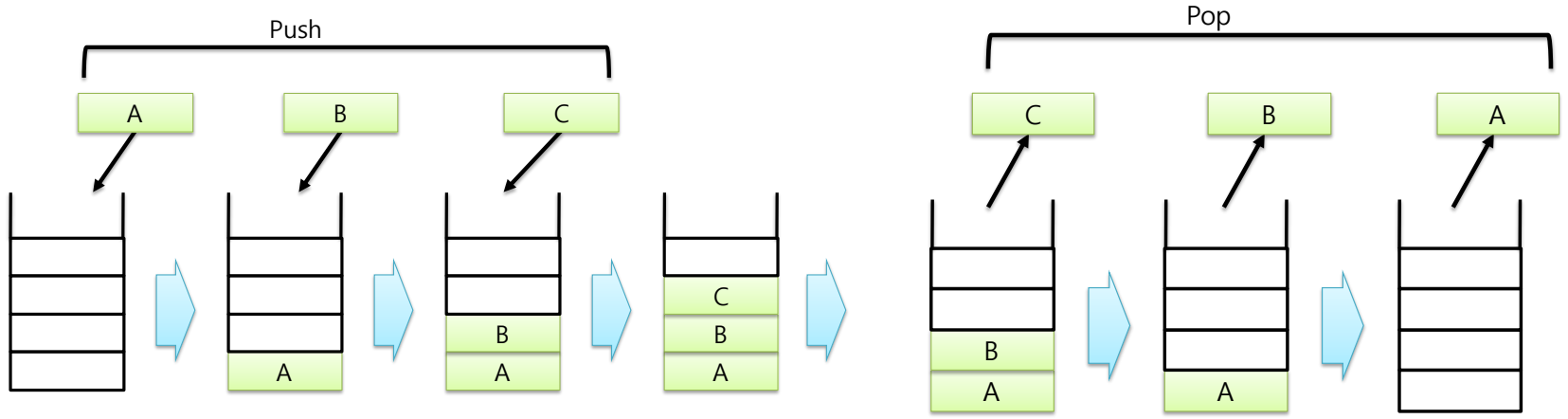
- 소스의 구성

파일 이름
04_02.vcproj
linkedstack.h
linkedstack.c
stackutil.h
stackutil.c
exampl04_03.c

5.1 스택 응용 1: 역순 문자열 만들기 (2/4)

- 역순 문자열 만들기

'ABC'의 문자열	A→B→C
'ABC'의 역순 문자열 'CBA'	C→B→A



5.1 스택 응용 1: 역순 문자열 만들기 (4/4)

```
030:  pStack = createLinkedStack();
031:  if (pStack != NULL) {
032:              StackNode node;
033:              for(i = 0; i < size; i++) {
034:                        node.data = pSource[i];
035:                        pushLS(pStack, node);
036:              }
037:
038:              for(i = 0; i < size; i++) {
039:                        pNode = popLS(pStack);
040:                        if (pNode != NULL) {
041:                                  pReturn[i] = pNode->data;
042:                                  free(pNode);
043:                        }
044:              }
045:              pReturn[i] = '₩0';
046:              deleteLinkedStack(pStack);
047:  }
```

5.2 스택 응용 1: 수식 괄호 검사 (1/4)

- 정상 수식

1)	(A + B) * C
2)	{ (A + B) * C } * D

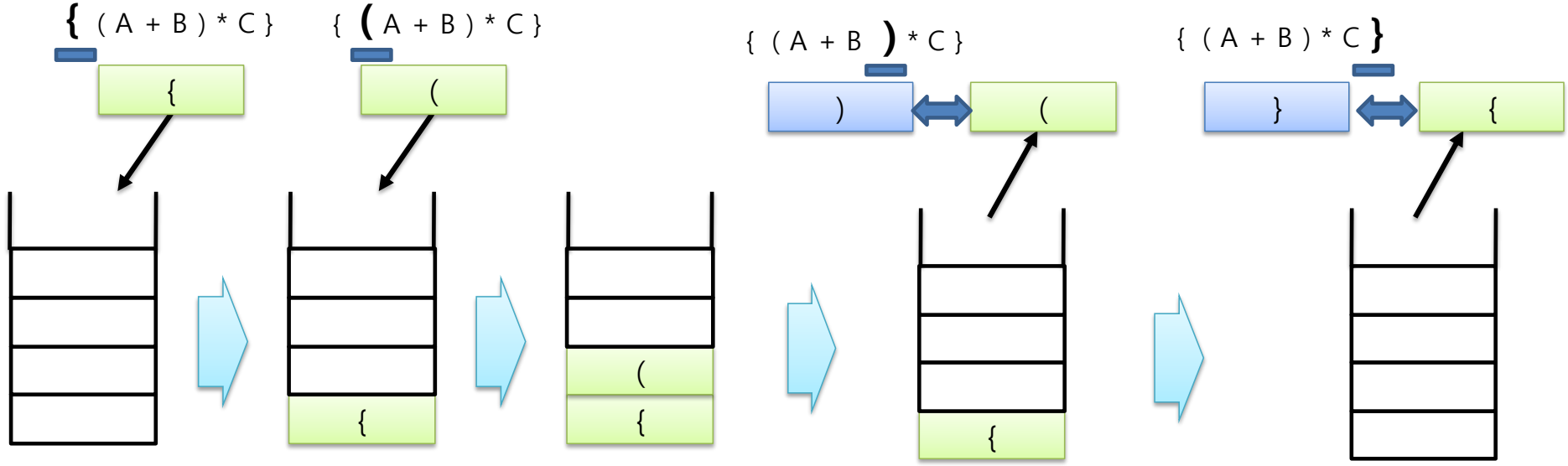
☞ **소괄호** ( )
중괄호 { }
대괄호 [ ]

- 비정상 수식

3)	(A + B) * C)
4)	((A + B) * C
5)	{ (A + B }) * C * D

5.2 스택 응용 1: 수식 괄호 검사 (2/4)

- 수식 괄호 검사의 예
  - { (A + B) * C }



5.2 스택 응용 1: 수식 괄호 검사-의사코드 (3/4)

```
checkBracketMatching( expression)  {
    result ← 성공
    while (expression의 끝이 아닌 경우 && result != 오류) {
        symbol ← expression의 다음 글자
        switch( symbol ) {
            case ‘(‘: case ‘[‘: case ‘{‘:
                symbol을 스택에 푸시
                break;
            case ‘)’: case ‘]’: case ‘}’:
                if (스택이 공백 상태) {
                    result ← 오류
                }
                else {
                    checkSymbol ← 스택에서 팝
                    if (symbol과 checkSymbol이 쌍이 맞지 않는 경우) {
                        result ← 오류
                    }
                }
                break;
        }
        if (스택이 비어 있지 않다면) {
            result ← 오류
        }
    return result
}
```

5.2 스택 응용 1: 수식 괄호 검사 (4/4)

- 예제 프로그램
  - example04_03.c
  - 시나리오

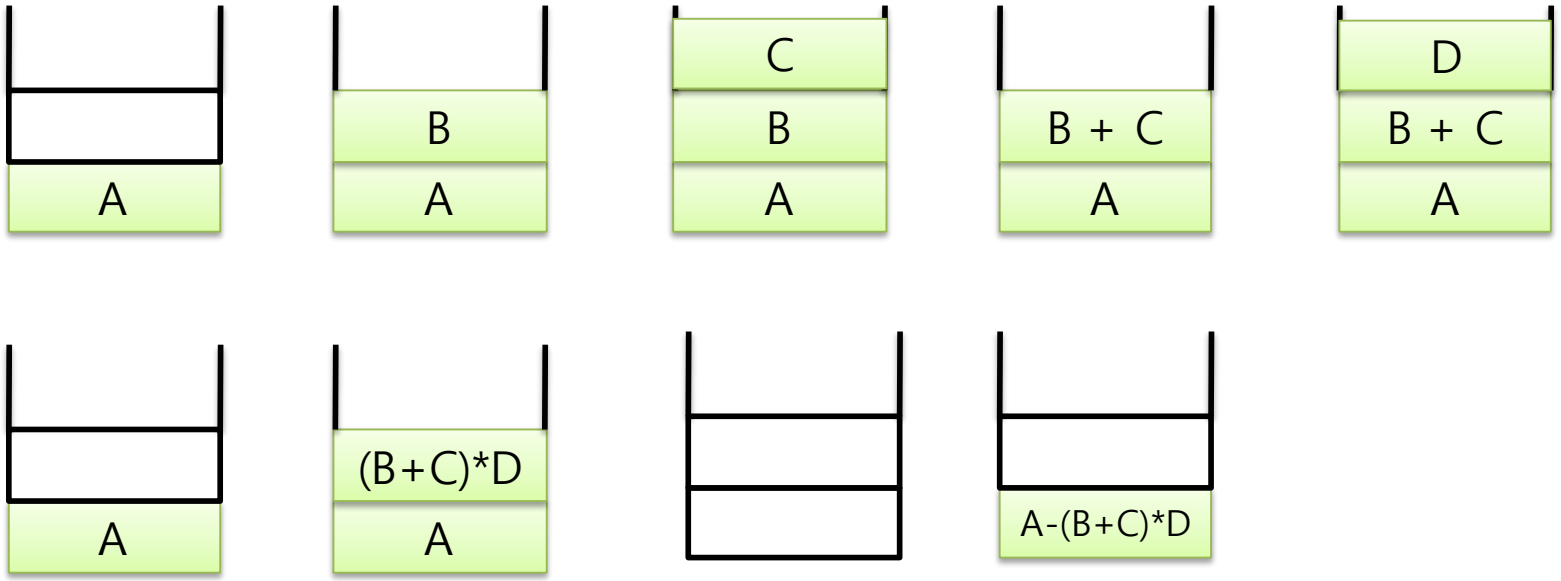
```
((A*B)/C)-{(D+E)&&(F-G)}
((A*B)/C-{(D+E)&&(F-G))}
```

6.1 스택 응용2: 수식계산 (1/8)

- 수식 계산 1
  - A * B
  - A B *
- 수식 계산 2
  - A + B * C
  - A B C * +
- 토큰
  - 피연산자 + 연산자

6.1 스택 응용2: 수식계산 (2/8)

- 수식 계산 3
  - A – (B + C) * D
  - A B C + D * –



6.1 스택 응용2: 수식계산 (3/8)

- 의사 코드

```
calcExpr ( expression ) {
    while ( 토큰 in expression ) {
        if ( 토큰 == 피연산자 ) {
                토큰을 스택에 푸시
        else if ( 토큰 == 연산자 ) {
            피연산자2 ← 스택에서 팝
            피연산자1 ← 스택에서 팝
            결과 ← 피연산자1 연산자 피연산자2  // 연산자는 + ,-,*,/ 중 하나
            결과를 스택에 푸시
        }
    }
    result ← 스택에서 팝
    return result
}
```

6.1 스택 응용2: 수식계산 (4/8)

- 소스의 구성

파일 이름
04_04.vcproj
exprdef.h
exprlinkedstack.h
exprlinkedstack.c
stackcalc.h
stackcalc.c
exampl04_04.c

4. **연결 리스트로 구현한 스택**

6.1 스택 응용2: 수식계산 (5/8)

- 연산자의 종류
  - C의 열거 타입(enumeration type)

```
04:     typedef enum PrecedenceType { lparen, rparen, times, divide, plus, minus,
05:                                   operand } Precedence;
```

열거 타입	설명	예
lparen	여는(left) 괄호	(
rparen	닫는(right) 괄호	)
times	곱하기	*
divide	나누기	/
plus	더하기	+
minus	빼기	-
operand	피연산자	0, 1, 2, ... 등

6.1 스택 응용2: 수식계산 (6/8)

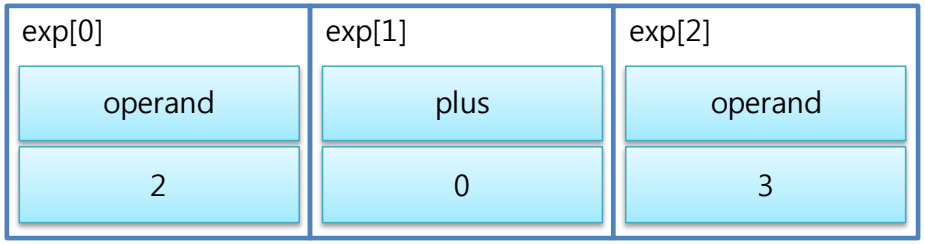
- 토큰

```
07:     typedef struct ExprTokenType {
08:             float value;
09:             Precedence type;
10:     } ExprToken;
```

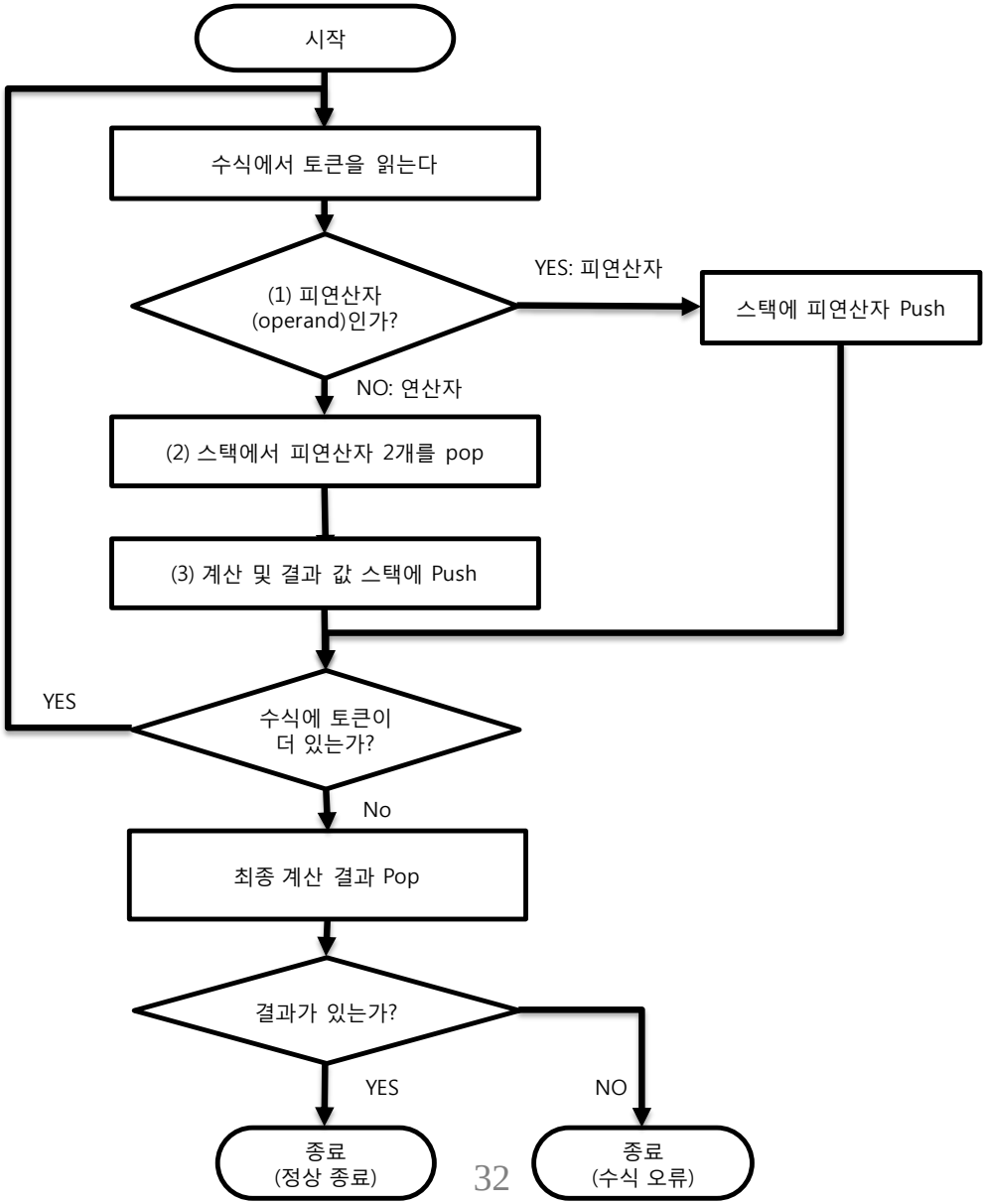
- 예
  - 2 + 3

```
ExprToken exp[3] = {0,};
exp[0].type = operand;
exp[0].value = 2;
exp[1].type = plus;
exp[1].value = 0;
exp[2].type = operand;
exp[2].value = 3;
```

2 + 3



6.1 스택 응용2: 수식계산 (8/8)



6.2 중위 표기에서 후위 표기로의 변환 (1/7)

- 4가지 예

	중위 표기법	후위 표기법
1)	A * B	A B *
2)	A * B + C	A B * C +
3)	A + B * C	A B C * +
4)	A * (B + C)	A B C + *

6.2 중위 표기에서 후위 표기로의 변환 (2/7)

- 1번째 예
  - A * B

1	A * B
2	A * B
3	A * B

6.2 중위 표기에서 후위 표기로의 변환 (3/7)

- 2번째 예
  - A * B + C

1	A * B + C
2	A * B + C
3	A * B + C
4	A * B + C
5	A * B + C

6.2 중위 표기에서 후위 표기로의 변환 (4/7)

- 3번째 예
  - A + B * C

1	A + B * C
2	A + B * C
3	A + B * C
4	A + B * C
5	A + B * C

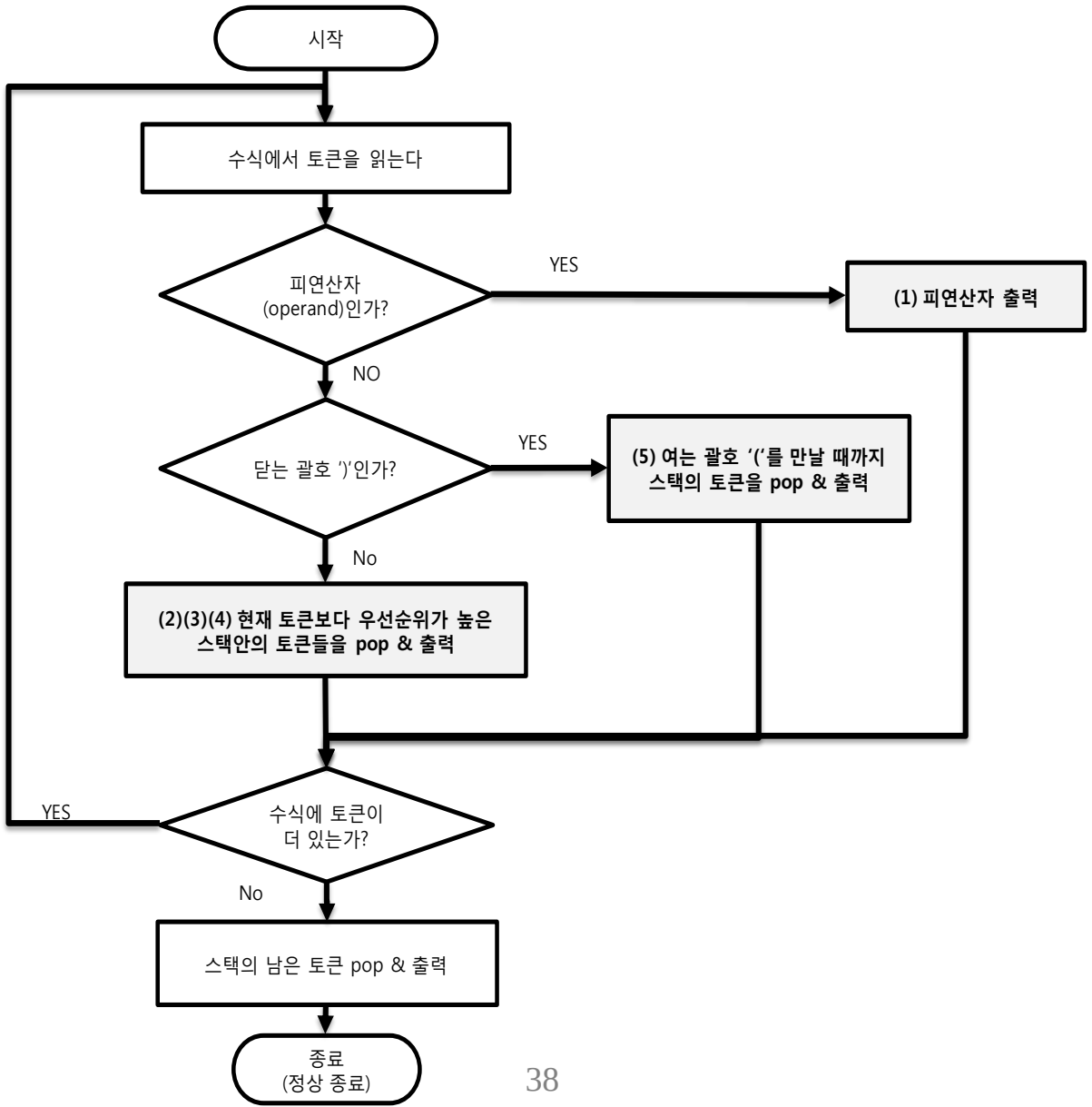
6.2 중위 표기에서 후위 표기로의 변환 (5/7)

- 4번째 예
  - A * (B + C)

1	A * (B + C)
2)	A * (B + C)
3	A * (B + C)
4	A * (B + C)
5	A * (B + C)
6	A * (B + C)
7	A * (B + C)

연산자	High ←	우선순위		→ Low
스택 내부	)	* /	+ –	(
스택 외부	( )	* /	+ –	

6.2 중위 표기에서 후위 표기로의 변환 (6/7)



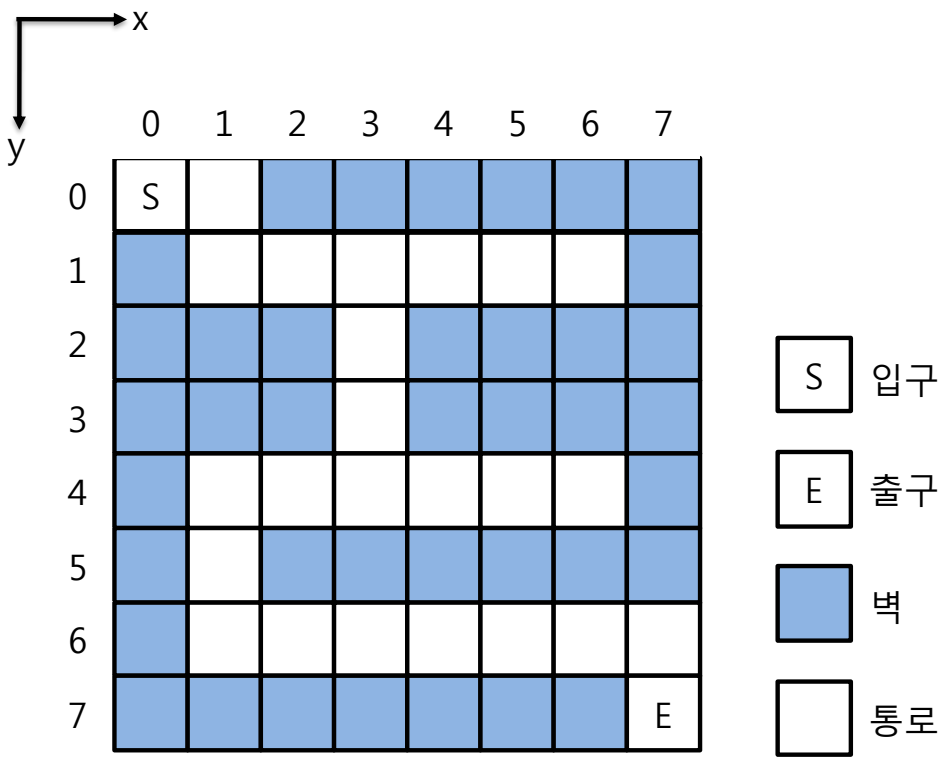
6.2 중위 표기에서 후위 표기로의 변환 (7/7)

- 예제 프로그램
  - example04_04.c
  - 시나리오

```
2 - ( 3 + 4 ) * 5
2 3 4 + 5 *
```

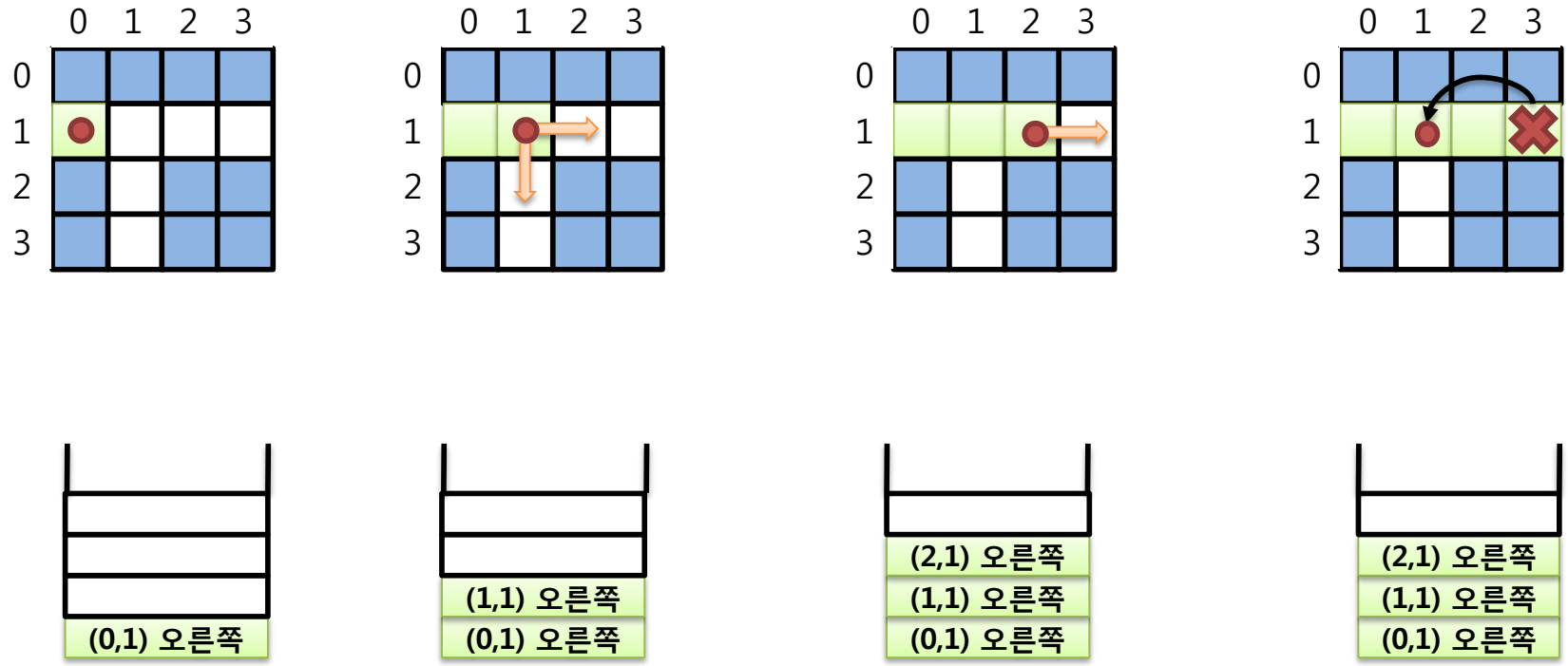
7. 스택 응용 3: 미로 찾기 (1/9)

- 미로 찾기 문제



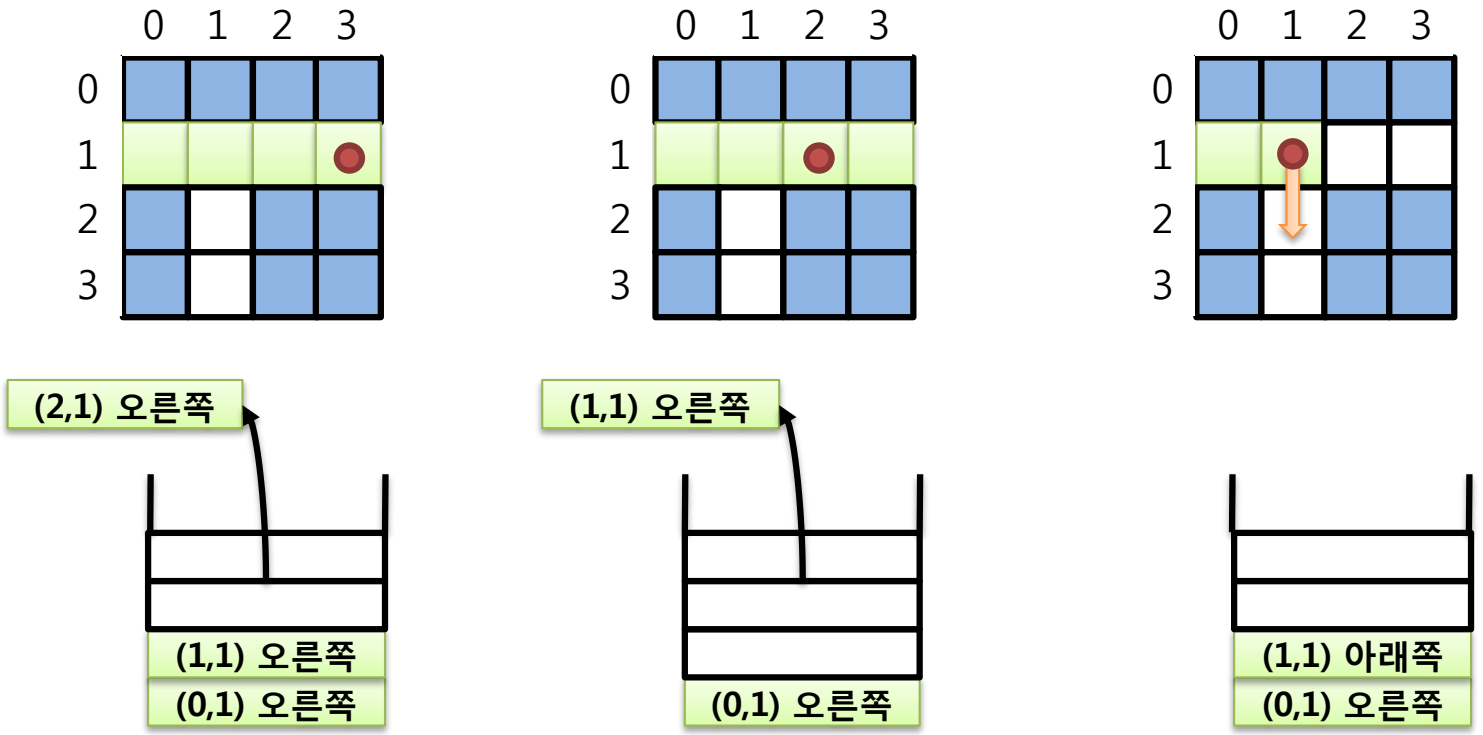
7. 스택 응용 3: 미로 찾기 (2/9)

- 미로 찾기 알고리즘 (1)



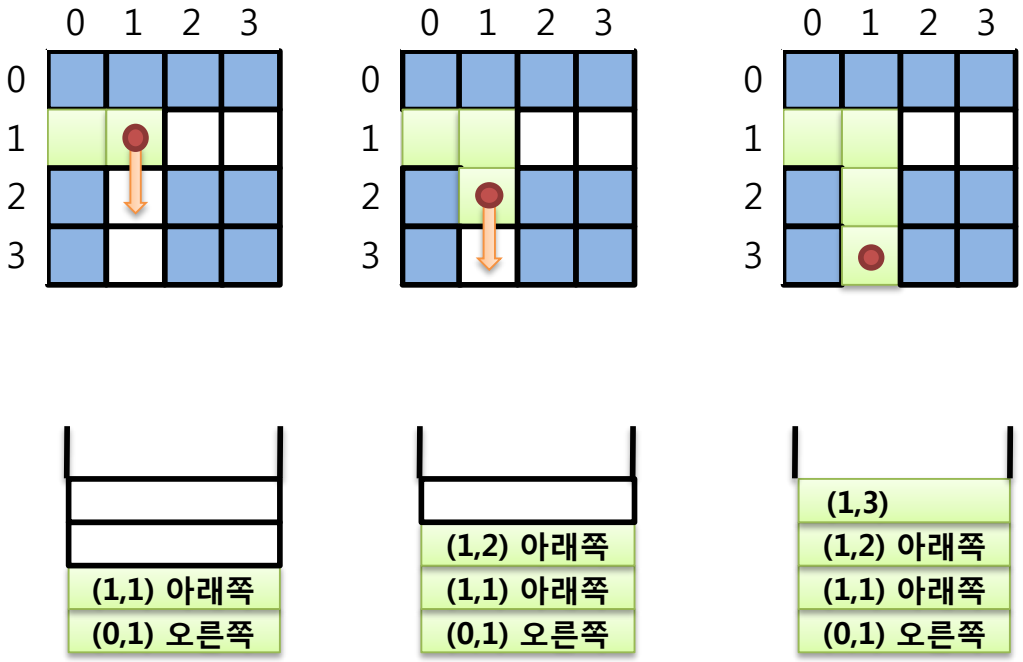
7. 스택 응용 3: 미로 찾기 (3/9)

- 미로 찾기 알고리즘 (2)

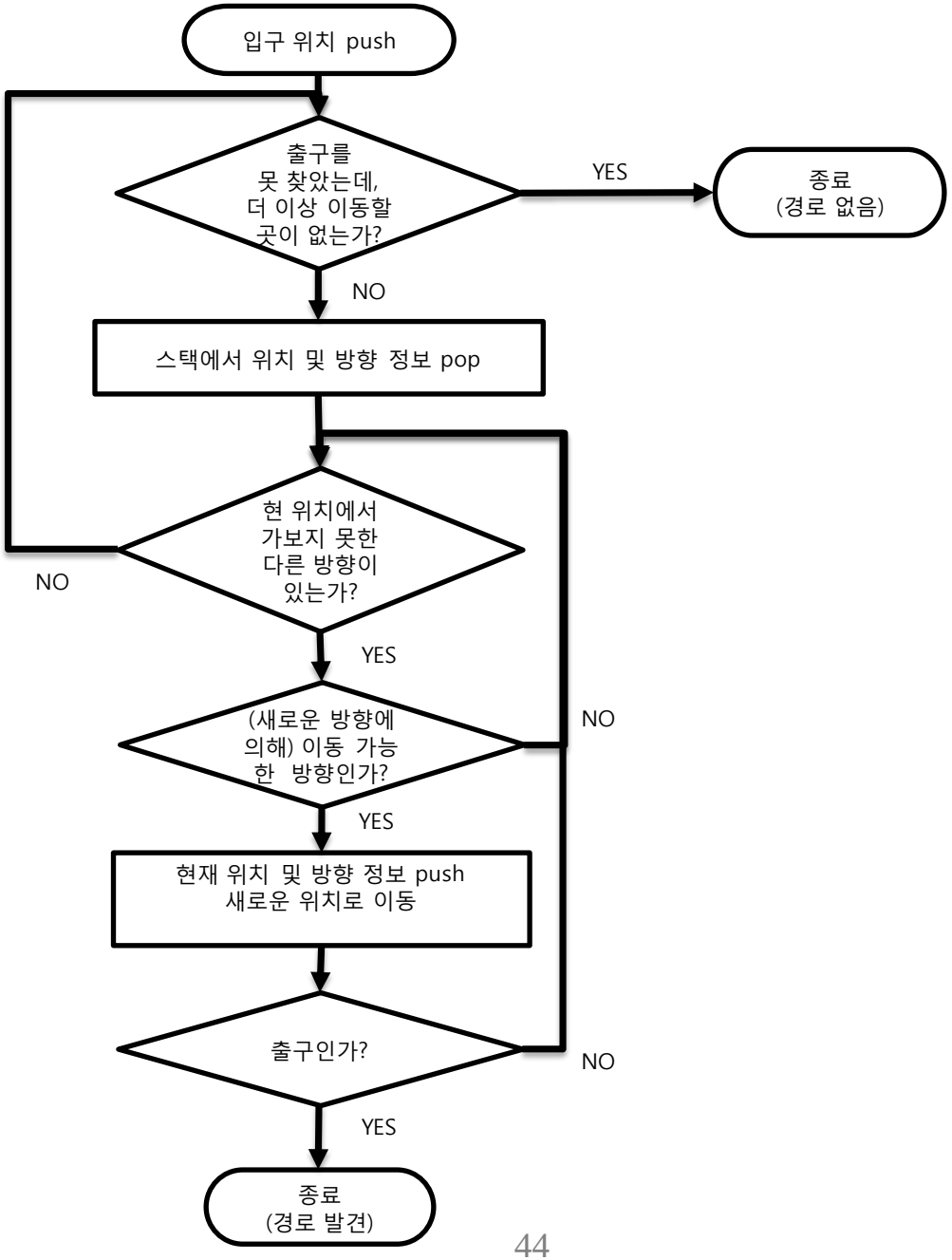


7. 스택 응용 3: 미로 찾기 (4/9)

- 미로 찾기 알고리즘 (3)

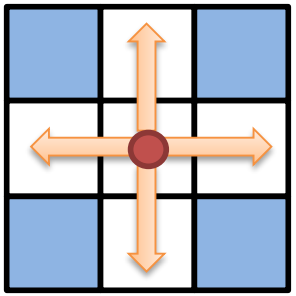


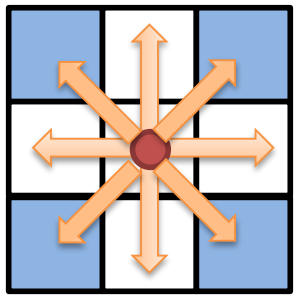
- 최종 경로 정보는?


입구 위치 push
출구를
못 찾았는데,
더 이상 이동할
곳이 없는가?
YES
종료
(경로 없음)
NO
스택에서 위치 및 방향 정보 pop
현 위치에서
가보지 못한
다른 방향이
있는가?
NO
YES
(새로운 방향에
의해) 이동 가능
한 방향인가?
NO
YES
현재 위치 및 방향 정보 push
새로운 위치로 이동
출구인가?
NO
YES
종료
(경로 발견)

7. 스택 응용 3: 미로 찾기 (7/9)

- 제약 조건과 문제 모델링





	0	1	2	3	4	5	6	7
0	2	2	1	1	1	1	1	1
1	1	2	0	0	0	0	0	1
2	1	1	1	0	1	1	1	1
3	1	1	1	0	1	1	1	1
4	1	0	0	0	0	0	0	1
5	1	0	1	1	1	1	1	1
6	1	0	0	0	0	0	0	0
7	1	1	1	1	1	1	1	0

7. 스택 응용 3: 미로 찾기 (9/9)

- 미로 찾기 알고리즘의 구현

```
04: void findPath(int mazeArray[HEIGHT][WIDTH],
05:                         MapPosition startPos,
06:                         MapPosition endPos,
07:                         LinkedStack *pStack);
08: int pushLSMapPosition(LinkedStack* pStack, MapPosition data);
09: void showPath(LinkedStack *pStack, int mazeArray[HEIGHT][WIDTH]);
10: void printMaze(int mazeArray[HEIGHT][WIDTH]);
```

- 예제 프로그램

이번 장에서는

- 스택의 개념
- 스택 추상 자료형
- 배열로 구현한 스택
- 연결 리스트로 구현한 스택
- 스택 응용 1: 역순 문자열 및 괄호 검사
- 스택 응용 2: 수식의 계산 및 표기법 변환
- 스택 응용 3: 미로 찾기