1 스택의 개념과 연산

1 스택(Stack)

- 접시를 쌓듯이 자료를 차곡차곡 쌓아 올린 형태의 자료구조

[스택의 개념 예]

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

1 스택(Stack)

- 스택에 저장된 원소는 top으로 정한 곳에서만 접근 가능
  - top의 위치에서만 원소를 삽입하므로, 먼저 삽입한 원소는 밑에 쌓이고, 나중에 삽입한 원소는 위에 쌓이는 구조
  - 마지막에 삽입(Last-In)한 원소는 맨 위에 쌓여 있다가 가장 먼저 삭제(First-Out)됨
    ⇨ 후입선출 구조 (LIFO, Last-In-First-Out)

1 스택(Stack)

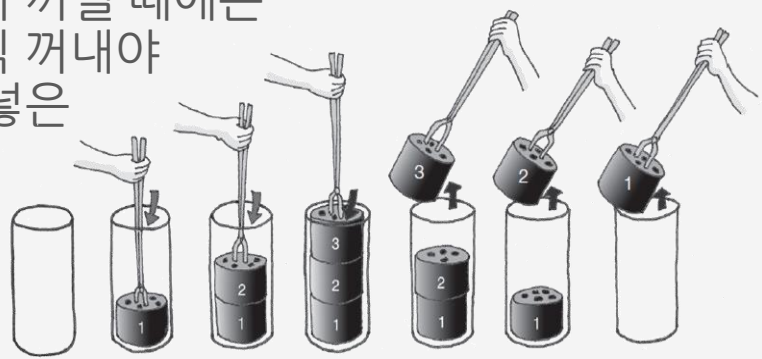
[스택의 구조]

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

2 후입선출 구조의 예

연탄 아궁이

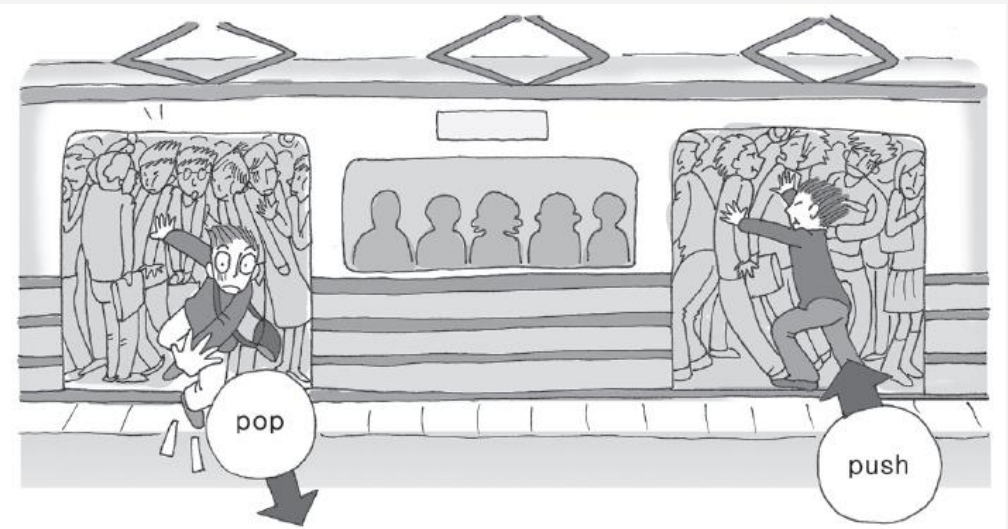
- 연탄을 하나씩 쌓으면서 아궁이에 넣으므로 마지막에 넣은 3번 연탄이 가장 위에 쌓여 있음
- 연탄을 아궁이에서 꺼낼 때에는 위에서부터 하나씩 꺼내야 하므로 마지막에 넣은 3번 연탄을 가장 먼저 꺼내게 됨



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

3 스택의 연산

- 스택에서의 삽입 연산 : **push**
- 스택에서의 삭제 연산 : **pop**

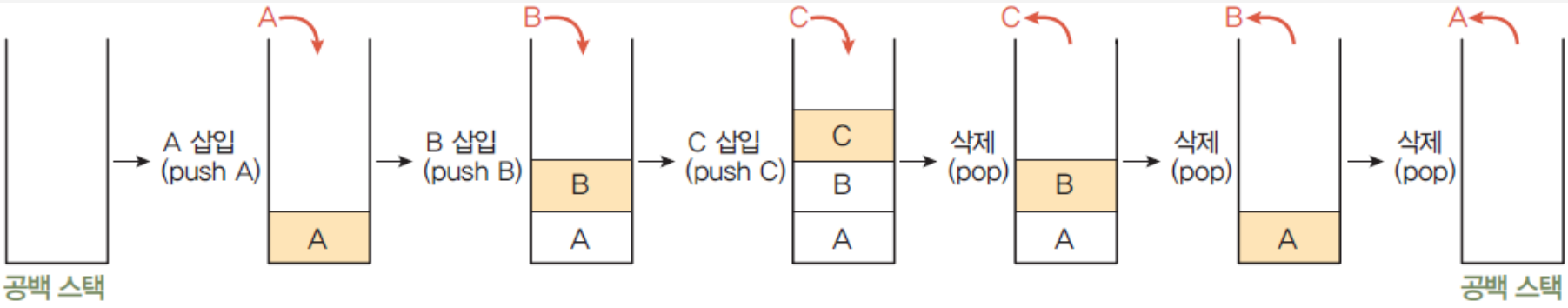


[만원 전철에서의 pop와 push]

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

4 스택에서의 원소 삽입/삭제 과정

- 공백 스택에 원소 A, B, C를 순서대로 삽입하고 한번 삭제하는 연산과정 동안의 스택 변화



[스택의 데이터 삽입(push)과 삭제(pop) 과정]

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

4 스택에서의 원소 삽입/삭제 과정

ADT 5-1 스택의 추상 자료형

```
ADT Stack
데이터 : 0개 이상의 원소를 가진 유한 순서 리스트
연산 :
    S ∈ Stack; item ∈ Element;

    // 공백 스택 S를 생성하는 연산
    createStack(S) ::= create an empty Stack S;

    // 스택 S가 공백인지 확인하는 연산
    isEmpty(S) ::= if (S is empty) then return true
                   else return false;
```

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

4 스택에서의 원소 삽입/삭제 과정

```
    // 스택 S의 top에 item(원소)을 삽입하는 연산
    push(S, item) ::= insert item onto the top of Stack S;

    // 스택 S의 top에 있는 item(원소)을 삭제하는 연산
    pop(S) ::= if (isEmpty(S)) then return error
               else delete and return the top item of Stack S;

    // 스택 S의 top에 있는 item(원소)을 반환하는 연산
    peek(S) ::= if (isEmpty(S)) then return error
                else return the top item of the Stack S;
End Stack
```

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

5 스택의 push() 알고리즘

❶ top ← top+1;

- 스택 S에서 top이 마지막 자료를 가리키고 있으므로 그 위에 자료를 삽입하려면 먼저 top의 위치를 하나 증가
- 만약 이때 top의 위치가 스택의 크기(Stack_SIZE)보다 크다면 오버플로우(Overflow)상태가 되므로 삽입 연산을 수행하지 못하고 연산 종료

5 스택의 push() 알고리즘

❷ S(top) ← x;

- 오버플로우 상태가 아니라면
  스택의 top이 가리키는 위치에 x 삽입

알고리즘 5-1 스택의 원소 삽입

```
push(S, x)
    ❶ top ← top + 1;
      if (top > stack_SIZE) then overflow;
      else
       ❷ S(top) ← x;
end push()
```

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

6 스택의 pop() 알고리즘

❶ return S(top);

- 공백 스택이 아니면 top에 있는 원소를 삭제하고 반환

❷ top ← top-1;

- 스택의 마지막 원소가 삭제되면 그 아래 원소, 즉 스택에 남아 있는 원소 중에서 가장 위에 있는 원소가 top이 되어야 하므로 top 위치를 하나 감소

6 스택의 pop() 알고리즘

알고리즘 5-2 스택의 원소 삭제

```
pop(S)
    if (top = 0) then underflow;
    else {
        ❶ return S(top);
        ❷ top ← top - 1;
    }
end pop()
```

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

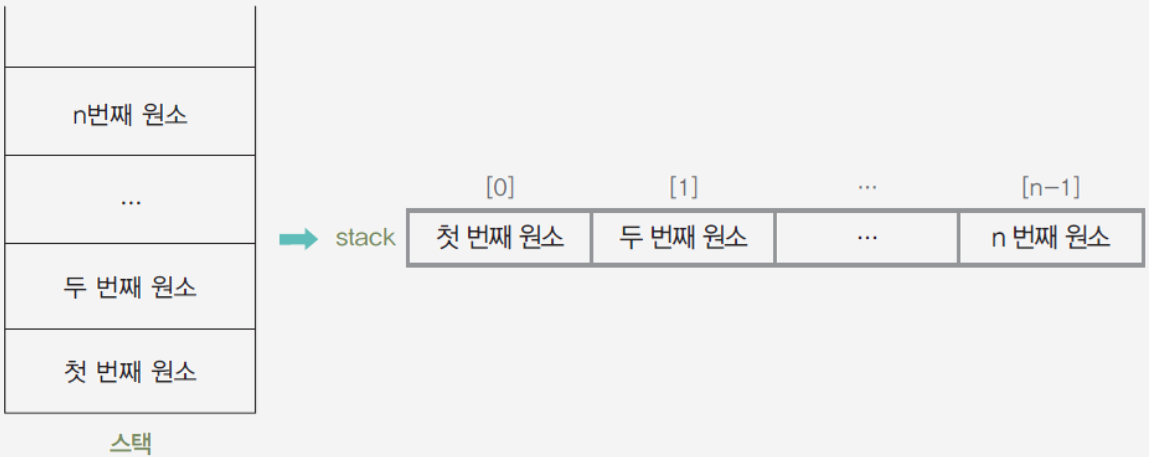
2 순차 자료구조를 이용한 스택의 구현

1 순차 자료구조를 이용한 스택의 구현

- 순차 자료구조인 1차원 배열을 이용하여 구현
  - 스택의 크기 : 배열의 크기
  - 스택에 저장된 원소의 순서 : 배열 원소의 인덱스
    - 인덱스 0번 : 스택의 첫번째 원소
    - 인덱스 n-1번 : 스택의 n번째 원소
  - 변수 top : 스택에 저장된 마지막 원소에 대한 인덱스 저장
    - 공백 상태 : top = -1 (초기값)
    - 포화 상태 : top = n-1

1 순차 자료구조를 이용한 스택의 구현

- 순차 자료구조인 1차원 배열을 이용하여 구현

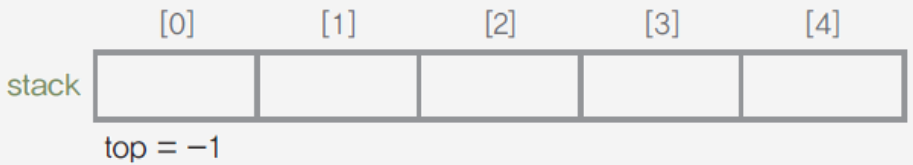


[1차원 배열을 이용한 순차 스택]

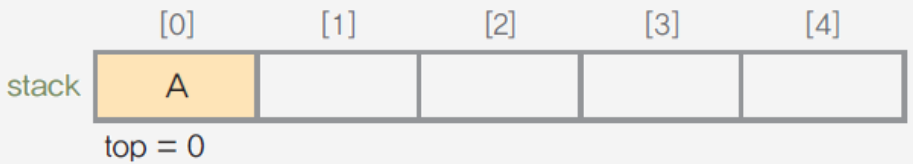
※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

2 크기가 5인 스택을 생성하여 원소 A, B, C를 순서대로 삽입한 후에 원소 하나를 삭제하는 과정

1 공백 스택 생성 : createStack(S, 5);



2 원소 A 삽입 : push(S, A);

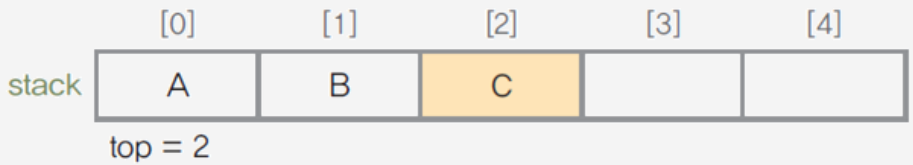


※출처: 이지영(2016). IT CookBook, C로 배우는 쉬운 자료구조(개정3판). 한빛미디어

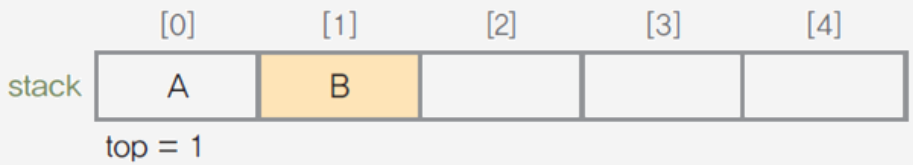
2 크기가 5인 스택을 생성하여 원소 A, B, C를 순서대로 삽입한 후에 원소 하나를 삭제하는 과정

3 원소 B 삽입 : push(S, B);

4 원소 C 삽입 : push(S, C);



5 원소 삭제 : pop(S);



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

3 순차 자료구조를 구현한 스택의 장점과 단점

장점	단점
• 순차 자료구조인 1차원 배열을 사용하여 쉽게 구현	• 물리적으로 크기가 고정된 배열을 사용하므로 스택의 크기 변경 어려움 • 순차 자료구조의 단점을 가짐

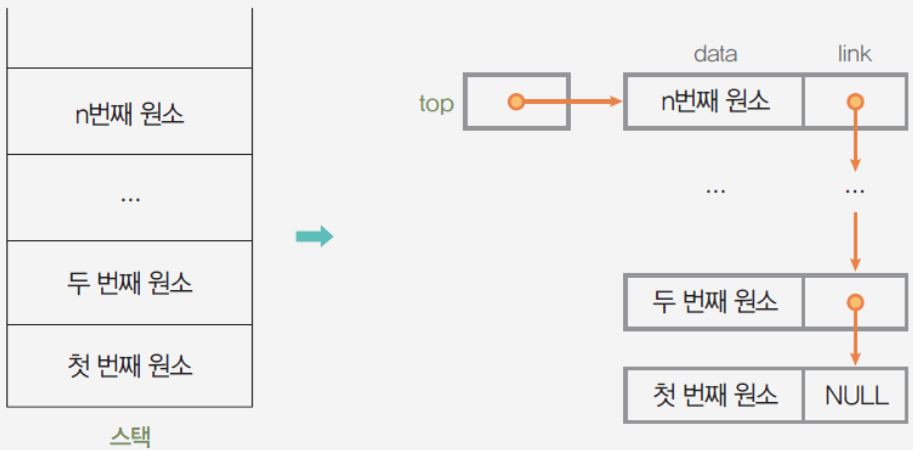
3 연결 자료구조를 이용한 스택의 구현

1 연결 자료구조를 이용한 스택의 구현

- 단순 연결 리스트를 이용하여 구현
  - 스택의 원소 : 단순 연결 리스트의 노드
    - 스택 원소의 순서 : 노드의 링크 포인터로 연결
    - push : 리스트의 마지막에 노드 삽입
    - pop : 리스트의 마지막 노드 삭제
  - 변수 top : 단순 연결 리스트의 마지막 노드를 가리키는 포인터 변수
    - 초기 상태 : top = null

1 연결 자료구조를 이용한 스택의 구현

단순 연결 리스트를 이용하여 구현



[단순 연결 리스트를 이용한 연결 스택]

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

1 연결 자료구조를 이용한 스택의 구현

- 스택 원소(element)의 자료형을 int로 정의

```
typedef int element;
```

- 스택의 노드를 구조체로 정의

```
typedef struct  stackNode {
                element data;
                struct stackNode *link;
} stackNode;
```

1 연결 자료구조를 이용한 스택의 구현

- 스택의 top 노드를 지정하기 위해 포인터 top 선언

```
stackNode* top;
```

- 스택이 공백 상태인지 확인하는 연산

```
int isEmpty() {
        if (top == NULL) return 1;
        else return 0;
}
```

1 연결 자료구조를 이용한 스택의 구현

- 스택의 top에 원소를 삽입하는 연산

```
void push(element item) {
    stackNode* temp = (stackNode *)malloc(sizeof(stackNode)); //공백 스택 생성
    temp->data = item;
    temp->link = top;     // 삽입 노드를 top의 위에 연결
    top = temp;           // top 위치를 삽입 노드로 이동
}
```

1 연결 자료구조를 이용한 스택의 구현

- 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
    }
}
```

2 원소 A, B, C를 순서대로 삽입하면서 스택을 생성한 후에 원소 하나를 삭제하는 과정

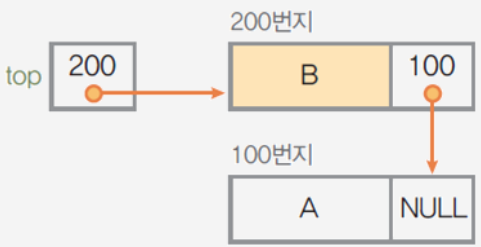
1 공백 스택 생성 : createStack(S);

2 원소 A 삽입 : push(S, A);

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

2 원소 A, B, C를 순서대로 삽입하면서 스택을 생성한 후에 원소 하나를 삭제하는 과정

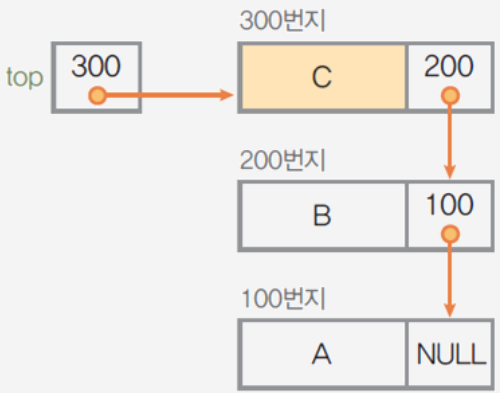
3 원소 B 삽입 : push(S, B);



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

2 원소 A, B, C를 순서대로 삽입하면서 스택을 생성한 후에 원소 하나를 삭제하는 과정

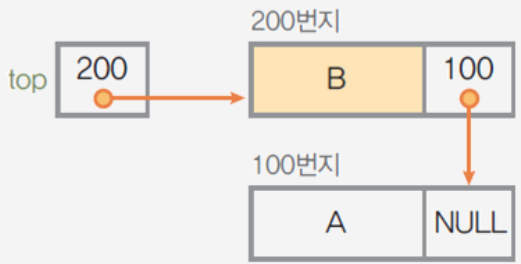
4 원소 C 삽입 : push(S, C);



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

2 원소 A, B, C를 순서대로 삽입하면서 스택을 생성한 후에 원소 하나를 삭제하는 과정

5 원소 삭제 : pop(S);



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

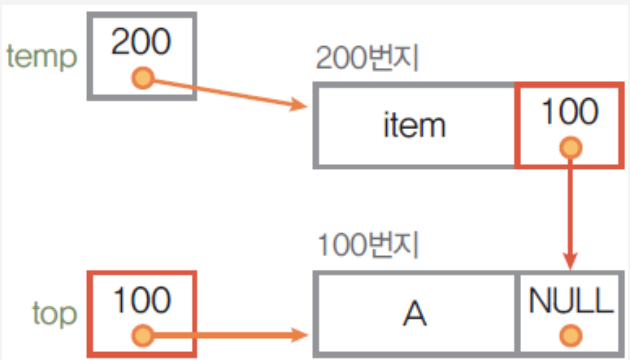
3 연결 스택에서 삽입 연산을 수행하는 과정

- 원소 item을 저장할 노드에 대한 메모리 할당, 포인터 temp 설정
  - stackNode* temp
    = (stackNode *)malloc(sizeof(stackNode));
- 삽입할 노드의 데이터 필드에 원소 item을 저장
  - temp->data = item;

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

3 연결 스택에서 삽입 연산을 수행하는 과정

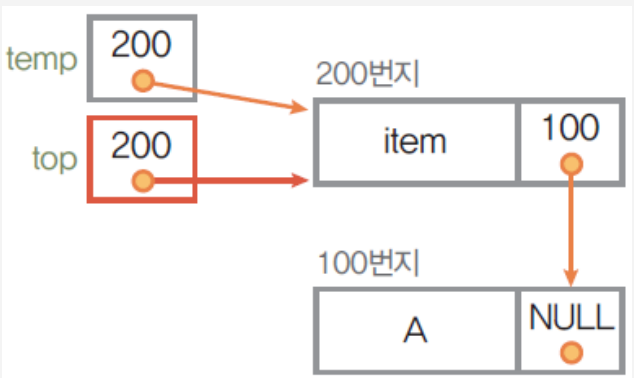
- 삽입할 노드의 링크 필드에 포인터 top의 값 저장하면, 새로 삽입한 노드가 현재 스택의 마지막 노드(현재 top이 가리키는 노드)로 연결
  - temp->link = top;



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

3 연결 스택에서 삽입 연산을 수행하는 과정

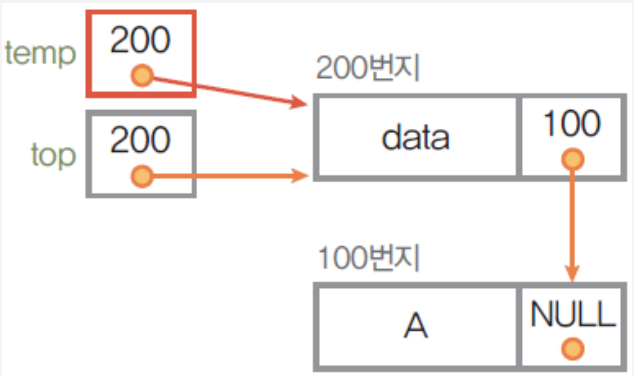
- 포인터 temp의 값(삽입 노드의 주소)을 포인터 top에 설정, 새로 삽입한 노드가 스택의 top 노드가 되도록 조정
  - top = temp;



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

4 연결 스택에서 삭제 연산을 수행하는 과정

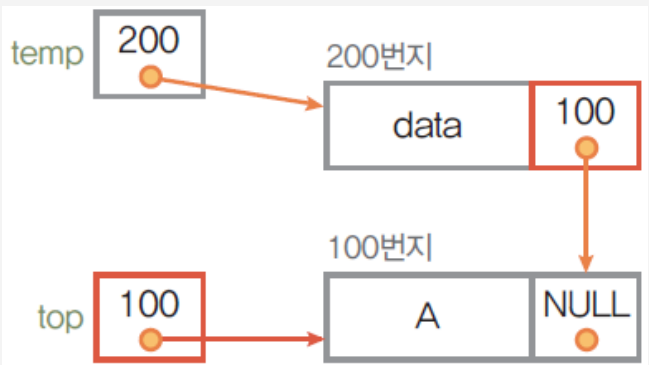
- 포인터 temp를 top 노드에 설정하여 삭제할 노드를 가리킴
  - stackNode* temp = top;



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

4 연결 스택에서 삭제 연산을 수행하는 과정

- 스택의 마지막 노드의 데이터 필드값을 변수 item에 저장
  - element item = temp->data;
- 포인터 top의 위치를 현재 마지막 노드의 아래 노드로 이동
  - top = temp->link;



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

4 연결 스택에서 삭제 연산을 수행하는 과정

- 포인터 temp가 가리키는 노드를 메모리 해제
  - free(temp);
- 변수 item의 값,
  즉 스택의 top이었던 노드의 데이터를 반환
  - return item;

5 연결 스택에서 스택의 원소를 top에서 bottom 순서로 출력하는 연산

▶ 출력하는 연산 프로그램

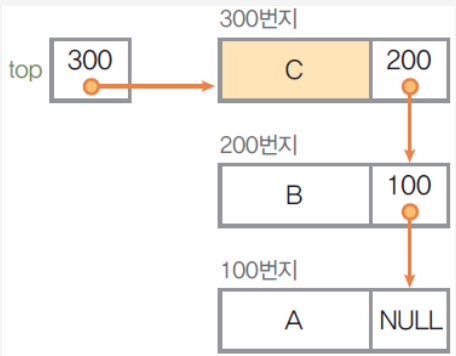
```
void printStack() {
    stackNode* p = top;
     printf("₩n STACK [ ");
     while (p) {  // 연결 스택에 노드가 있는 동안
                printf("%d ", p->data); //스택의 원소 출력
                p = p->link; //링크 필드를 따라  아래 노드로 이동
    }
    printf("] ");
}
```

5 연결 스택에서 스택의 원소를 top에서 bottom 순서로 출력하는 연산

- 연결 스택에 노드가 있는 동안, top 노드부터 링크 필드를 따라 아래 노드로 이동하면서 데이터 필드 값을 출력
- top 노드부터 출력, 스택의 top 원소부터 차례로 왼쪽에서 오른쪽으로 출력됨
  - 예) push(A) → push(B) → push(C) 과정 후 데이터 출력

5 연결 스택에서 스택의 원소를 top에서 bottom 순서로 출력하는 연산

- top 노드부터 출력, 스택의 top 원소부터 차례로 왼쪽에서 오른쪽으로 출력됨
  - 예) push(A) → push(B) → push(C) 과정 후 데이터 출력
    - 결과 : STACK[C B A]
    - 논리적인 형태 ⟶



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