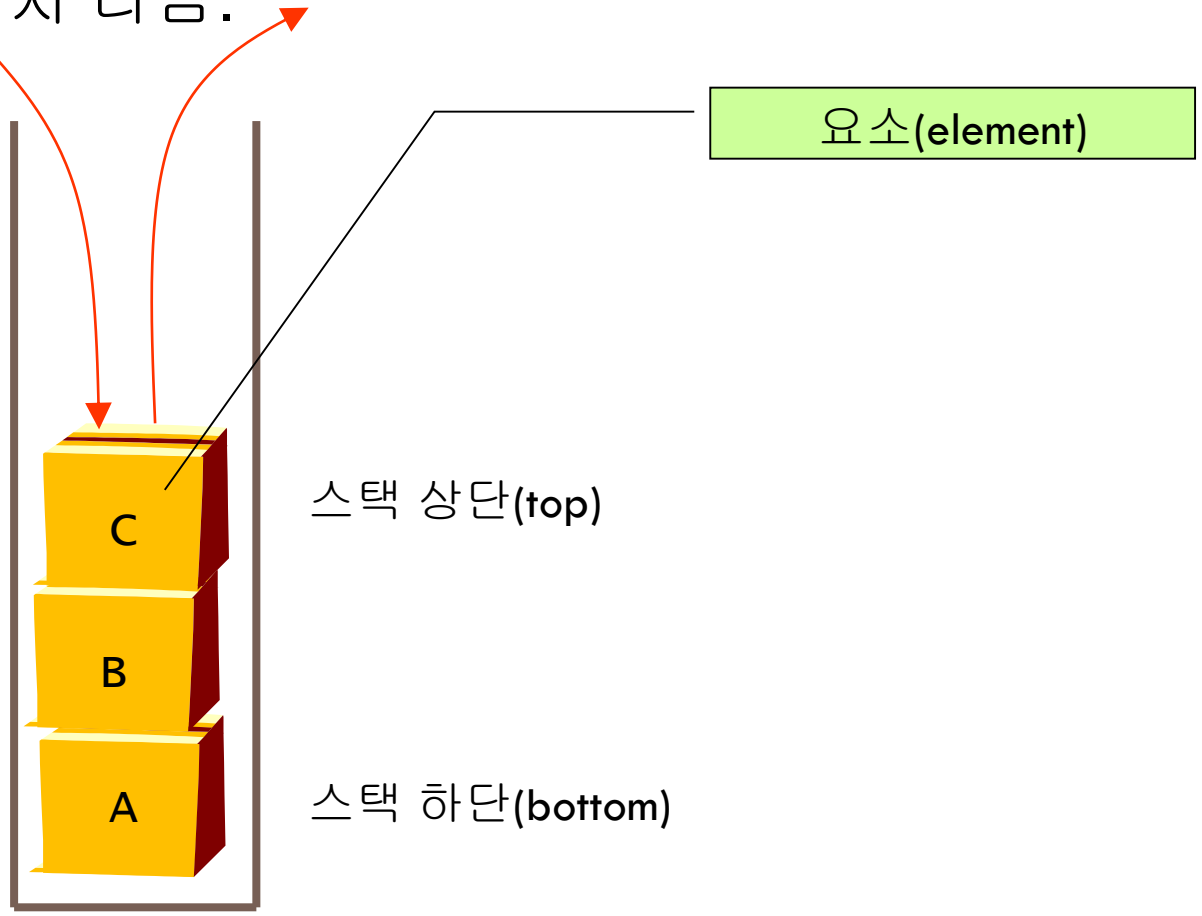
4장 스택

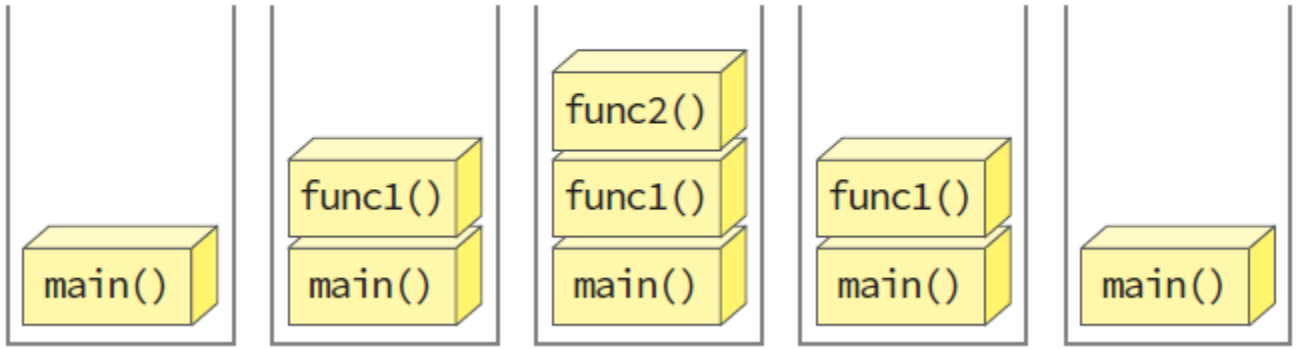
스택의 특징 및 구조

- **후입선출(LIFO:Last-In First-Out)**: 가장 최근에 들어온 데이터가 가장 먼저 나감.



예제: 시스템 스택을 이용한 함수 호출

```
void func2(){
    return;
}

void func1(){
    func2();
}

int main(void){
    func1();
    return 0;
}
```



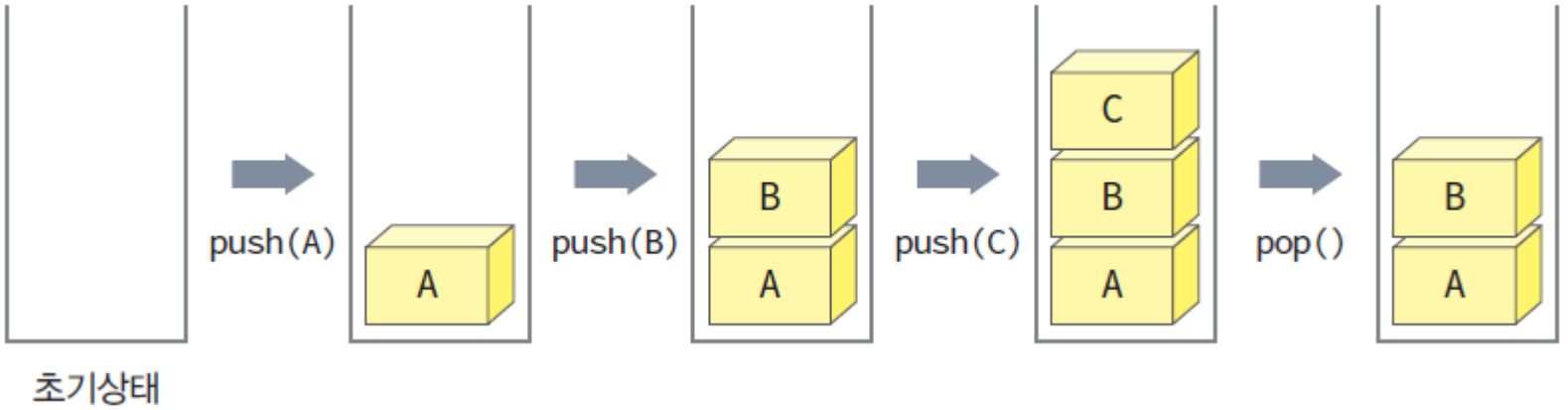
C로 쉽게 풀어쓴 자료구조

스택 추상데이터타입(ADT)

·객체: 0개 이상의 원소를 가지는 유한 선형 리스트
·연산:
- create(size) ::= 최대 크기가 size인 공백 스택을 생성한다.
- is_full(s) ::=
  if(스택의 원소수 == size) return TRUE;
  else return FALSE;
- is_empty(s) ::=
  if(스택의 원소수 == 0) return TRUE;
  else return FALSE;
- push(s, item) ::=
  if(is_full(s)) return ERROR_STACKFULL;
  else 스택의 맨 위에 item을 추가한다.
- pop(s) ::=
  if(is_empty(s)) return ERROR_STACKEMPTY;
  else 스택의 맨 위의 원소를 제거해서 반환한다.
- peek(s) ::=
  if(is_empty(s)) return ERROR_STACKEMPTY;
  else 스택의 맨 위의 원소를 제거하지 않고 반환한다.

스택의 연산

- push(): 스택에 데이터를 추가
- pop(): 스택에서 데이터를 삭제

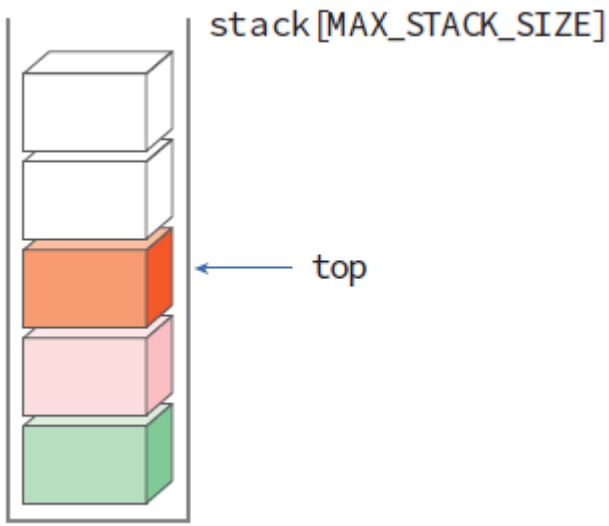


스택의 연산

- is_empty(s): 스택이 공백상태인지 검사
- is_full(s): 스택이 포화상태인지 검사
- create(): 스택을 생성
- peek(s): 요소를 스택에서 삭제하지 않고 보기만 하는 연산
- pop(): 연산은 요소를 스택에서 완전히 삭제하면서 가져온다.

배열을 이용한 스택의 구현

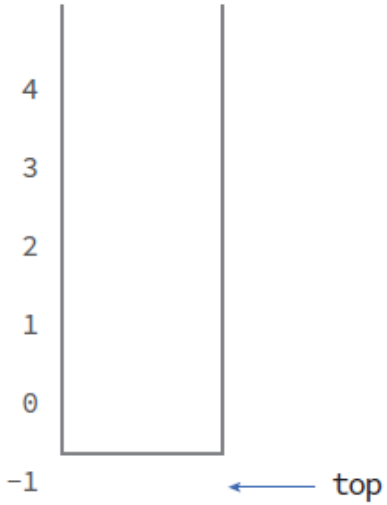
- 1차원 배열 stack[ ]
- 스택에서 가장 최근에 입력되었던 자료를 가리키는 top 변수
- 가장 먼저 들어온 요소는 stack[0]에, 가장 최근에 들어온 요소는 stack[top]에 저장
- 스택이 공백상태이면 top은 -1



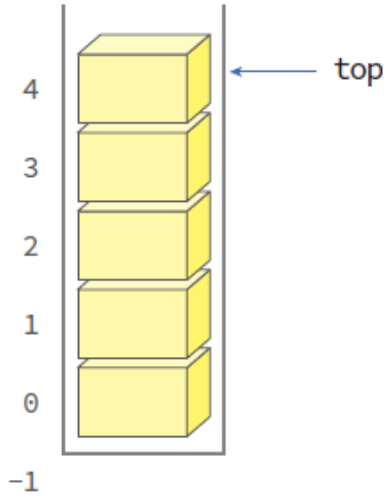
is_empty, is_full 연산의 구현

```
is_empty(S):

if top == -1
     then return TRUE
     else return FALSE
```

```
is_full(S):

if top == (MAX_STACK_SIZE-1)
     then return TRUE
     else return FALSE
```



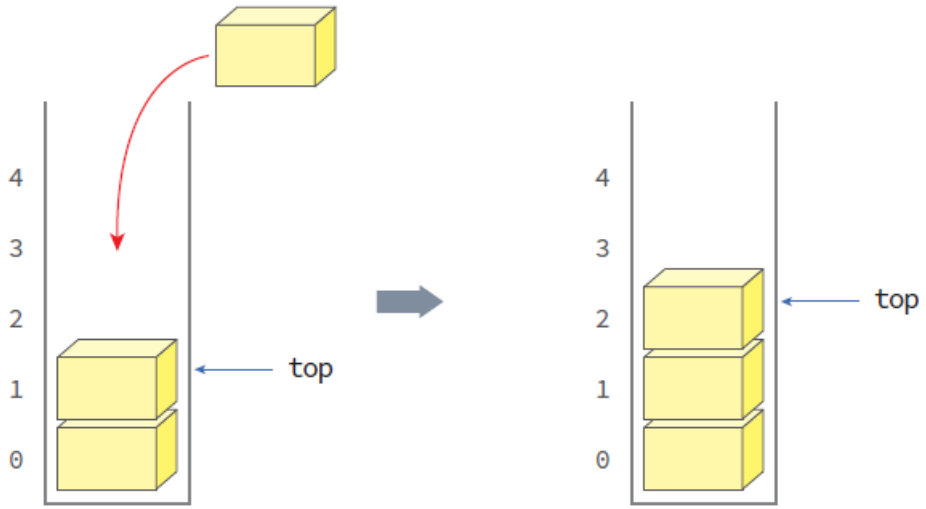
(a) 공백상태



(b) 포화상태

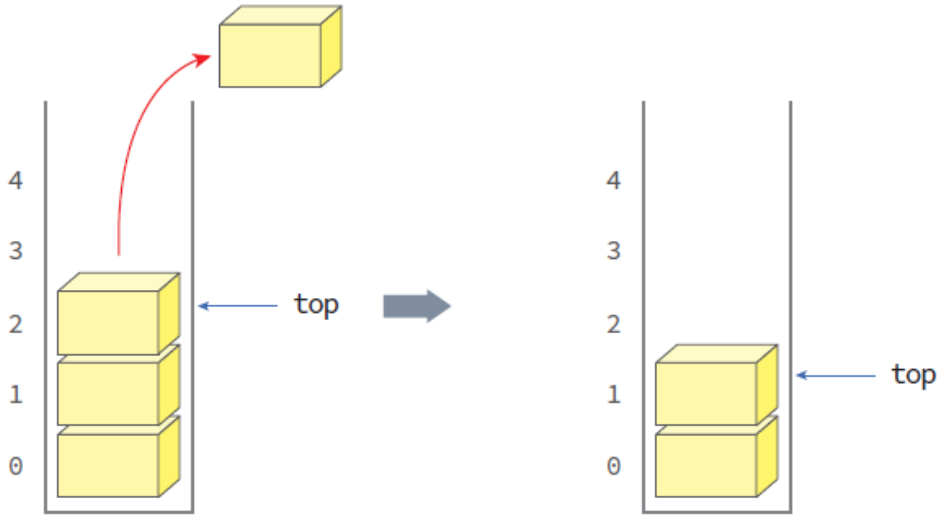
push 연산

```
push(S, x):

if is_full(S)
     then error "overflow"
     else top←top+1
          stack[top]←x
```



pop 연산

```
pop(S, x):

if is_empty(S)
     then error "underflow"
     else e←stack[top]
          top←top-1
          return e
```



peek 연산

```
peek():

if is_empty(S)
     then error "underflow"
     else e←stack[top]
          return e
```

스택의 top번째 값만 가져오고 스택은 변함이 없다.

```c
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<stdlib.h>
#define SIZE 5

typedef int element;
int top = -1;
element stack[SIZE];

int is_empty()
{
    return (top == -1);
}

int is_full()
{
    return (top == (SIZE - 1));
}

```

```
19
20  void push(element item)
21  {
22      if (is_full())
23      {
24          fprintf(stderr, "스택 포화 에러 ₩n");
25          return;
26      }
27      else {
28          top++;
29          stack[top] = item;
30          printf("스택에 push성공 ₩n");
31      }
32  }
33
```

```
34  element pop()
35  {
36      if (is_empty()) {
37          fprintf(stderr, "스택이 비어있습니다₩n");
38          exit(1);
39      }
40      else {
41          element tmp = stack[top];
42          top--;
43          return tmp;
44      }
45  }
46
47  element peek()
48  {
49      if (is_empty())
50      {
51          fprintf(stderr, "스택이 비어있습니다₩n");
52          exit(1);
53      }
54      else return stack[top];
55  }
```

```c
int main(void)
{
    int menu, value;
    while (1) {
        printf("스택에 push = 1, pop = 2, peek = 3, 종료 = 4 를 입력하시오>>");
        scanf("%d", &menu);
        if (menu == 1) {
            printf("push할 값 입력하시오.");
            scanf("%d", &value);
            push(value);
        }
        else if (menu == 2) {
            printf("pop한 값은 : %d 입니다₩n", pop());
        }
        else if (menu == 3) {
            printf("peek한 값은 : %d 입니다₩n", peek());
        }
        else if (menu == 4) break;
        else printf("1~4 중 하나를 선택하세요");

        printf("전체스택의 요소의 %d개 입니다. ₩n", (top+1));
    }
}
```

C로 쉽게 풀어쓴 자료구조