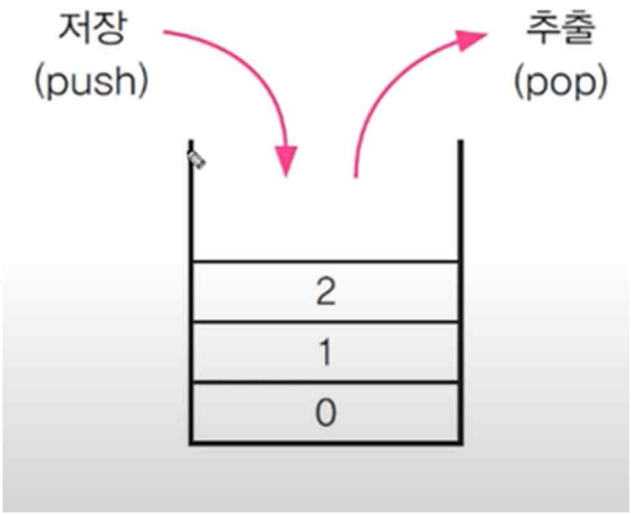
스택



후입선출(LIFO: Last In First Out)방식의 입출력 형태를 가지는 자료구조로, 가장 먼저 입력된 데이터가 맨 아래에 쌓이고 가장 최근에 입력된 데이터가 가장 위에 쌓이는 구조이다. 데이터의 입출력은 맨위에서만 일어나고 나머지 부분에서는 삭제하거나 추가할 수 없다.

대표적으로 편집 프로그램에서 Ctrl + z 키를 눌러 직전 작업을 순차적으로 취소하는 경우가 스택을 사용한 예라고 볼 수 있고, 그 외에는 함수에서 또다른 함수를 호출할 때 본래의 함수 주소를 저장하기 위해 스택의 형식을 사용한다.

스택에 새로 추가되는 데이터가 저장되는 위치를 top pointer이고, 데이터를 내보낼 때도 top pointer의 데이터가 출력된다. 이때 스택에 데이터를 저장하는 연산을 푸시(push), 추출하는 연산을 팝(pop)라고 부르며 가장 마지막 데이터를 조회하는 연산을 피크(peek)라고 한다. 피크 연산은 데이터의 변화없이 조회하는 기능만을 갖고 있다.

C에서 스택은 두가지 방법으로 표현할 수 있다.

- **배열**
- **연결리스트**

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

#define MAX_STACK_SIZE 100
typedef int element;
element stack[MAX_STACK_SIZE];
int top = -1;
//공백 상태 검출 함수
int is_empty()
{
    return (top == -1);
}
//포화 상태 검출 함수
int is_full()
{
    return (top == (MAX_STACK_SIZE-1));
}
//삽입 함수
void push(element item)
{
    if( is_full() ) {
        fprintf(stderr, "스택 포화 에러\n");
        return;
    }
    else stack[++top] = item;
}
// 삭제함수
element pop()
{
    if( is_empty() ) {
        fprintf(stderr, "스택 공백 에러\n");
        exit(1);
    }
    else return stack[top--];
}
//피크 함수
element peek()
{
    if( is_empty() ) {
        fprintf(stderr, "스택 공백에러 \n");
        exit(1);
    }
    else return stack[top];
}
// 주 함수
void main()
{
    push(1);
    push(2);
    push(3);
    printf("%d\n", pop());
    printf("%d\n", pop());
    printf("%d\n", pop());
    printf("%d\n", is_empty());
}
```