실습과제 5.1

[과제 설명]

스택(Stack)은 삽입과 삭제 연산이 한쪽 끝에서 이루어지는 자료구조인데, LIFO(Last-In-First-Out)이라 불리는 먼저 입력된 것이 나중에 출력되는 후입선출 구조이다.

배열을 이용해 구현할 때는 데이터가 어디까지 차있는지 위치를 나타내줄 인덱스 변수 top이 필요하며 이것을 이용해 자료의 삽입과 삭제가 이루어진다.
삽입할 때는 해당하는 칸에 데이터를 새로 삽입하는 형태로 이루어진다. 또, 배열 기반의 스택은 POP을 할 때 top만 옮겨주고 따로 그 칸을 다른 값으로 초기화할 필요는 없다. 동적 할당이 없는 정적 할당으로 만든 배열이기 때문에 크기가 정해져 있어, 배열이 다 차있는지 혹은 전부 비어있는지를 체크하는 함수도 필요하다.

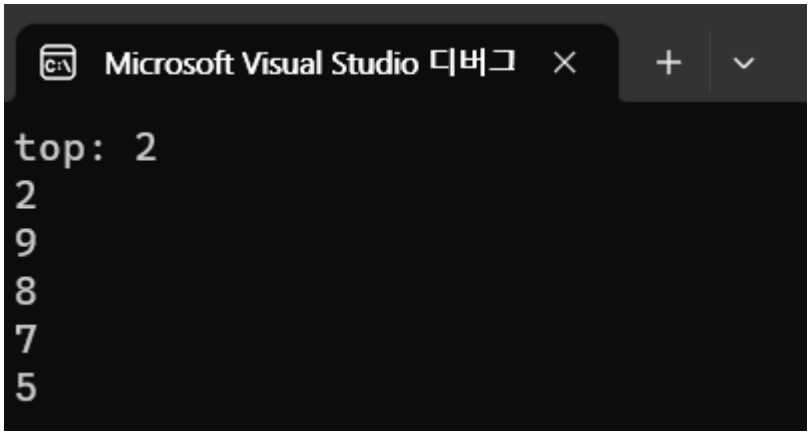
[코드]

메인 함수와 header 파일은 교안에서 미리 구현된 것을 사용했으므로 보고서에는 생략하고 구현된 함수 세부 내용만 첨부한다.

```
#include "MyArrayStack.h"

//배열 스택 초기화
void Stack_Init(ArrStack* pStack) {
  pStack->top = -1;
}

//데이터 삽입
void Stack_Push(ArrStack* pStack, Data item) {
  if (Stack_IsFull(pStack) == F) {
    pStack->top += 1;
    pStack->arr[pStack->top] = item;
  }
  else //스택이 가득 차 있는 경우 종료
    return;
}

//데이터 인출
Data Stack_Pop(ArrStack* pStack) {
```

```
  if (Stack_IsEmpty(pStack) == F) {
    pStack->top -= 1;
    return pStack->arr[pStack->top + 1];
  }

  //top의 위치를 하나 줄이고 그 전의 데이터를 반환한다.
}

//최상단 데이터 확인
Data Stack_Peek(ArrStack* pStack) {
  if (Stack_IsEmpty(pStack) == T)
    return; //스택이 비어있는 경우 종료
  else
    return pStack->arr[pStack->top];
  //비어있지 않다면 top 위치에 있는 데이터를 반환
}

//스택이 비어있는지 확인
Bool Stack_IsEmpty(ArrStack* pStack) {
  if (pStack->top == -1)
    return T;
  else
    return F;
}

//스택이 가득 차 있는지 확인
Bool Stack_IsFull(ArrStack* pStack) {
  if (pStack->top == STACK_LEN - 1)
    return T;
  else
    return F;
}
```

[실행 결과]



[과제에 대한 고찰]

배열 기반의 스택을 사용할 때의 단점을 보안하는 방법으로 연결 리스트를 이용한 스택을 구현해보는 걸 흔히 사용하지만, 실습을 하면서 정적 할당으로 한 배열이 아닌 (배열 자체를) 동적 할당으로 하는 배열을 하면 어떨까라는 생각이 들었다. 하지만, 이렇게 하면 일정 크기 이상을 넘어갈 때마다 새로 할당을 해줘야 해서 시간적 자원이 많이 소모 되는데다가 메모리에 선형으로 공간이 남아있어야 하기 때문에 실무, 실생활 등에서 대량으로 사용하는 경우에는 오히려 맞지 않을 거라는 결론이 나왔던 것 같다.

발상 자체는 생소해서 괜찮지 않을까 라는 생각이 들었지만, 상당히 오래 전부터 사용한 자료구조인만큼 서치를 해봐도 나오지 않는 방법은 역시 비효율적이라서 그런 것 같다는 생각이 굳어지게 만들었다.