

Data Structures

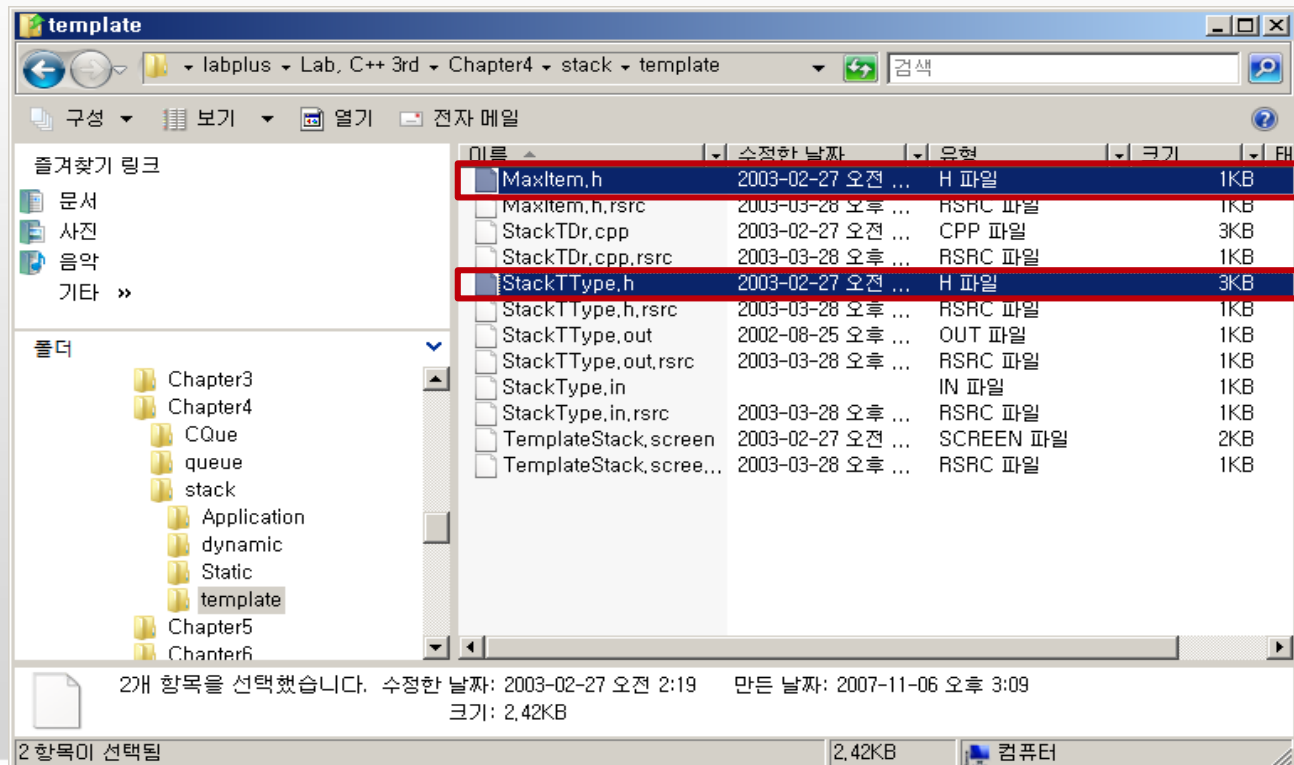
Lab # 04



1. Exercise

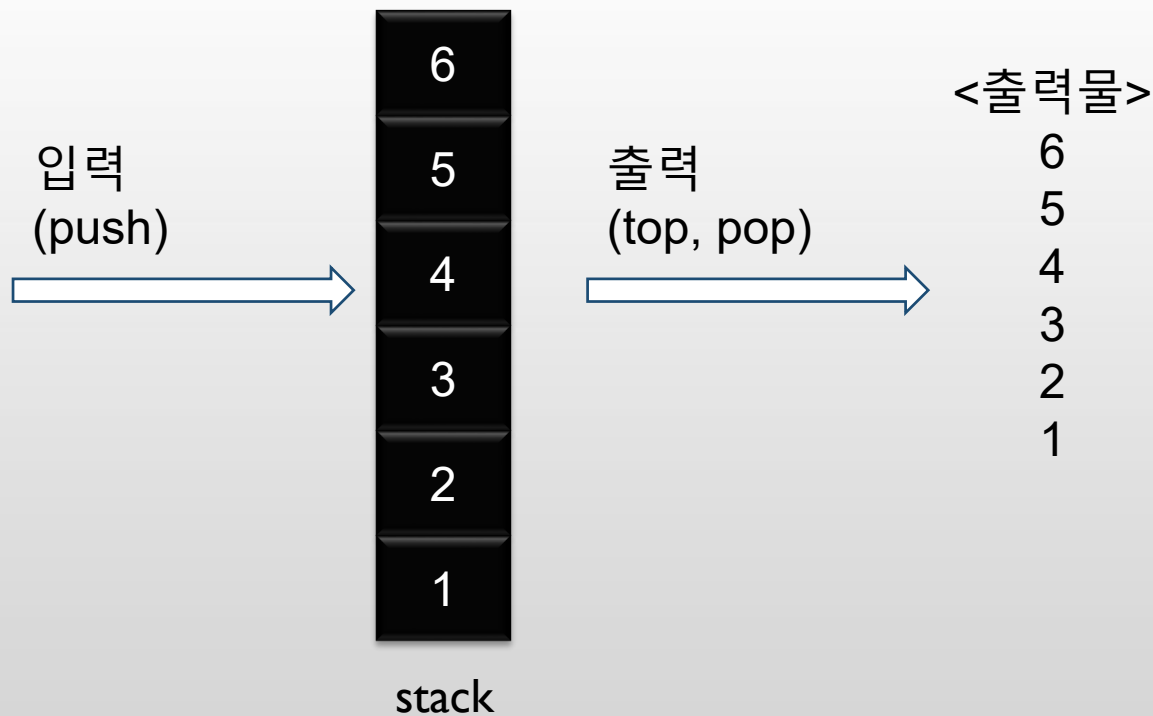
■ 문제

- ❖ 스택의 동작방법을 이해하기 위하여 "StackTType.h"에 정의된 StackType클래스를 분석
 - StackType내의 멤버함수인 Push, Pop, Top 함수에 대해서 분석
- ❖ 템플릿으로 정의된 스택을 정수형 타입으로 선언하고 스택에 1,2,3,4,5,6을 순서대로 삽입하고 하나씩 꺼내서 출력하는 프로그램을 작성함



1-help slides (1/2)

- Push 함수를 이용하여 스택에 1,2,3,4,5,6을 입력한다.
- Top 함수를 이용하여 가장 최근에 넣은 아이템을 가져와서 출력
Pop 함수를 이용하여 가장 최근에 넣은 아이템을 제거
 - ❖ 이 과정을 반복하여 스택의 모든 아이템을 출력 및 제거

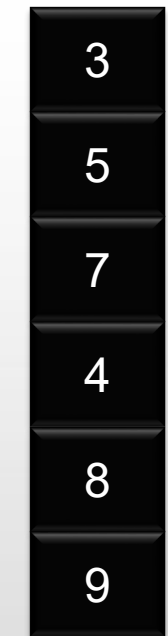


■ 예제코드

```
int main()
{
    StackType<int> stack;
    stack.Push(1); // 1을 스택에 넣는다.
    ... // 나머지를 스택에 전부 넣는다.
    while(!stack_isEmpty()) // 스택에 원소가 없을때 까지 반복한다.
    {
        int last = .... ; // 가장 최근에 넣은 아이템 값을 가져온다.
        .... // 가장 최근에 넣은 아이템을 제거한다.
        cout << last << endl;
    }
}
```

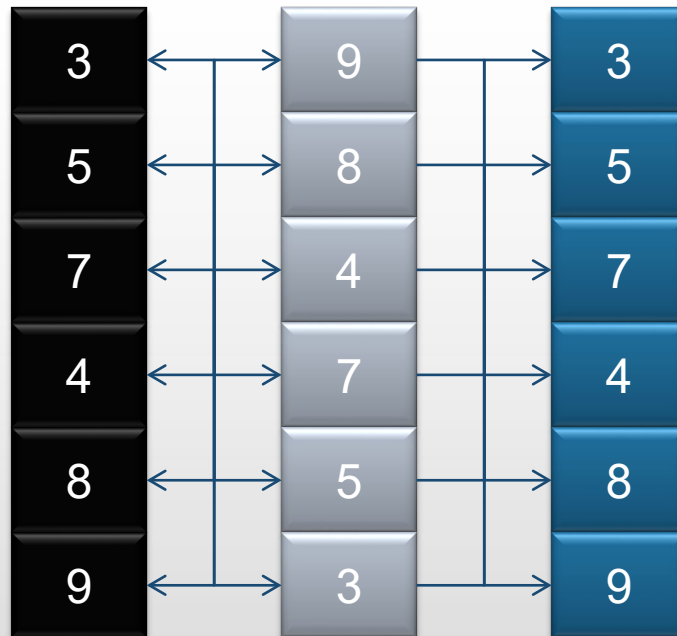
2. Exercise

- 스택의 데이터를 변경하지 않고, 기존의 스택과 동일한 값을 가지는 스택을 만드는 클라이언트 함수를 작성



stack A

[입력값]

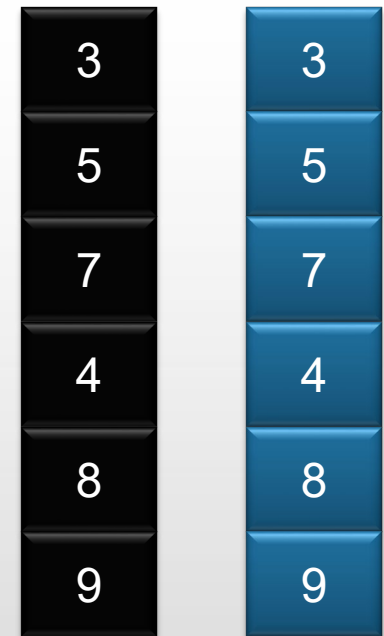


stack A

stack C

stack B

Hint: 임시 저장을 위한 stack 필요



stack A

stack B

[출력값]

이번 문제는 **클라이언트 함수**로 작성해야 하며, 주어진 stack 구현 코드는 변경하면 안됩니다. Stack의 push, top, pop등을 사용하여 주어진 문제 a를 구현하세요.

3. Exercise

- 하나의 배열을 이용하여, 두 개의 스택을 구현하는 Double stack 클래스를 작성하세요.
 - ❖ 첫 번째 스택은 1000 이하의 수를 저장합니다.
 - ❖ 두 번째 스택은 1000을 초과하는 수를 저장합니다.
 - ❖ Double stack의 최대 아이템 저장 갯수는 200입니다.
 - ❖ 각 스택의 개수는 정해지지 않았습니다.
 - 1000이하의 수로 200개를 저장할 수 있고, 1000이하의 수가 하나도 없을 수도 있습니다.

■ A. 하나의 배열에 stack 2개를 어떻게 구현할 것인가?

array[200]



1. Push 할 때 값을 비교하여 0번부터 채워 넣거나, 199번부터 채워 넣습니다.
2. Pop, Top 구현은 생략
3. 2개의 flag를(가장 최근에 넣은 수를 기록하는 변수) 사용해서 stack을 관리합니다.
4. IsFull은 2개의 flag의 주소가 연속이면 full입니다.

- B. A에서 생각한 Double stack을 클래스로 정의해 보세요.
 - ❖ Double stack 클래스의 멤버 함수 중 Push 연산 부분을 구현해 보세요.
- C. 채점을 위해 저장된 아이템들을 확인 하는 Print()함수를 작성하세요.
 - ❖ Stack Pop 순서로 출력을 해야하며, 1000이하 스택을 출력 후에 1000초과 스택을 출력하세요.

```
const int MAX_ITEMS = 200;

class doublestack
{
private:
    int top_small; //1000보다 작거나 같은 스택의 top
    int top_big; // 1000보다 큰 스택의 top
    int items[MAX_ITEMS];

public:
    void Push(int item); // B에서 구현할 push 연산
    void Print(); // stack 의 상황을 보여줄 수 있는 함수(채점시)
    ... // (필요하다고 생각되는 멤버 함수 ...)
}
```


4. Exercise

■ 각 스택의 복사본을 다른 것으로 치환하는 함수를 작성하세요.

❖ 다음과 같은 사양을 사용

Replaceltem

동작 : 모든 oldItem을 newItem으로 바꾼다.

조건 : 스택은 초기화되어 있다.

결과 : 스택에 있는 각각의 oldItem은 newItem으로 바뀌진다.

■ A. Replaceltem 함수를 클라이언트 함수로 작성한다.

■ B. Replaceltem 함수를 StackType의 멤버 함수로 작성한다.

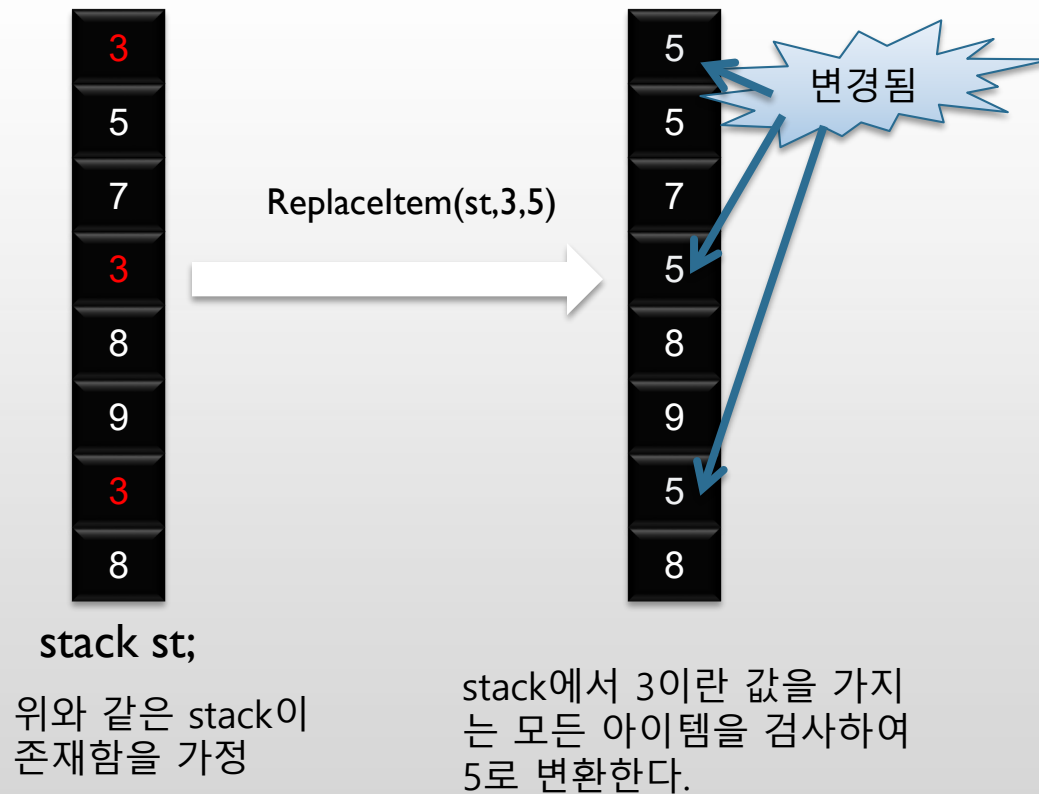
❖ *Template를 사용하지 않는 StackType 클래스를 사용한다.

(경로 : \\lappus\\Lab,C++3rd\\Chapter4\\stack\\Static)

■ 문제 추가 설명

❖ 예제

- stack의 3이란 값을 가지는 item을 5로 모두 변경하고자 할때



- A. 클라이언트로 함수로 작성하여 스택의 구조는 변경하지 않고 위의 동작을 하는 함수를 만들어라.

❖ Client function prototype :

- void ReplaceItem(StackType &st, int oldItem, int newItem);

Hint: 임시 저장을 위한 stack 필요

```
void ReplaceItem(StackType &st, int oldItem, int
newItem);

int main()
{
    StackType stack;
    ... // stack에 Push연산을 통해 값을 입력
    ReplaceItem(...); // replace 수행
    while(!stack.IsEmpty())          //스택 내용 출력
    {
        item = stack.Top();
        stack.Pop();
        cout <<"Item : "<<item << endl;
    }
    return 0;
}
```

- B. 스택의 구조를 변경하여 Replaceltem이라는 함수를 클래스 내부에 선언하고 구현하라. (a와 달리 oldItem, newItem 2개의 파라미터를 갖는다.)
 - ❖ 클라이언트 함수와 달리 멤버함수는 클래스의 멤버변수에 직접적으로 접근할 수 있다.

```
class StackType
{
private:
    ... //기본 선언
public:
    ... //기본 구현
    void Replaceltem(int, int);
}
```

```
void StackType::Replaceltem(int oldItem, int newItem)
{
    for(item 개수만큼)
    {
        if(items[i]가 oldItem과 같은가?)
            items[i]는 newItem;
    }
}
```