Data Structures

Lab # 04

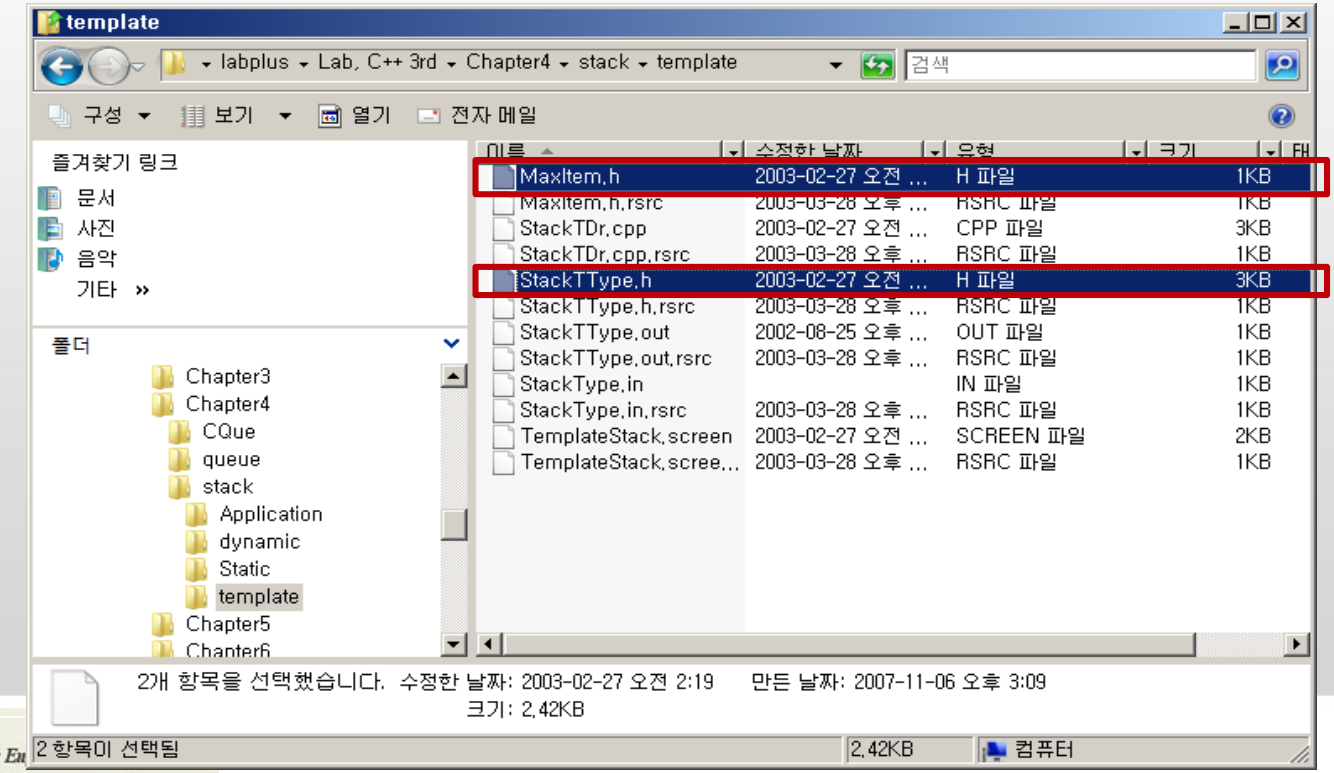
Data & Knowledge Engineering Laboratory
Kyung Hee University, Korea

Chapter 4 Programming Exercise

1. Exercise

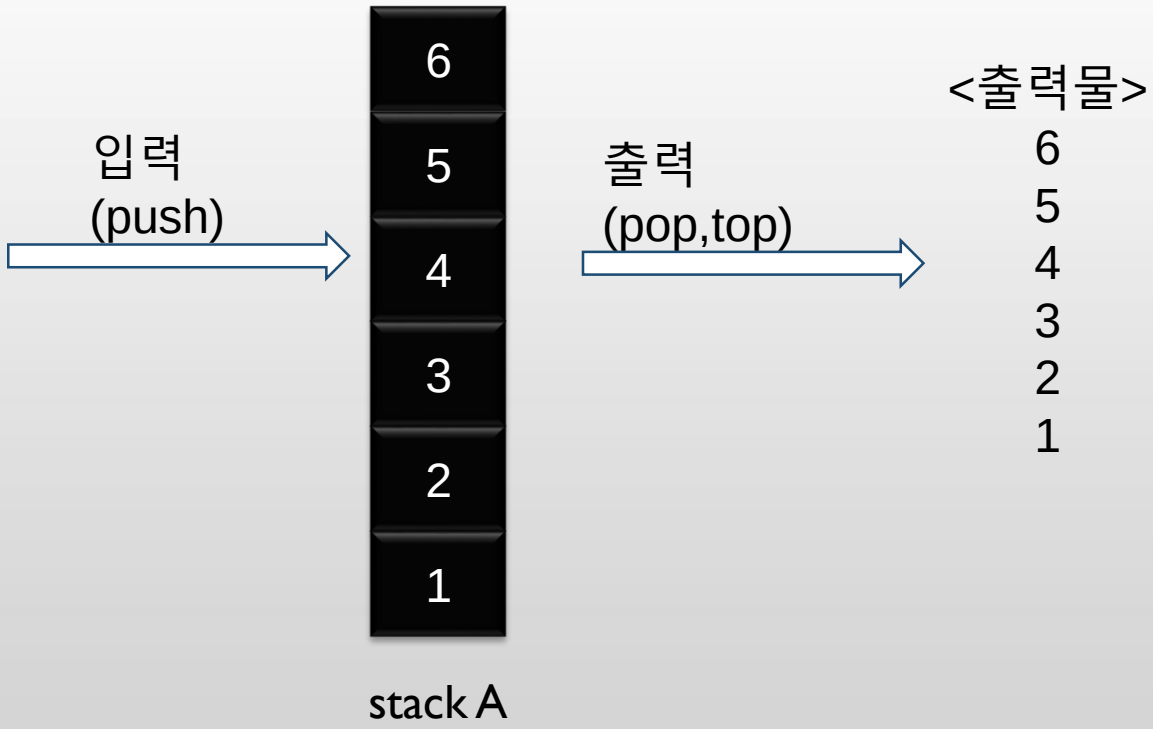
■ 문제

- ❖ 스택의 동작방법을 이해하기 위하여 "StackTType.h"에 정의된 StackType클래스를 분석
  - StackType내의 멤버함수인 Push, Pop, Top 함수에 대해서 분석
- ❖ 탬플릿으로 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 출력을하고 pop을 이용하여 가장 최근에 넣은 아이템을 제거한다. 이 과정을 반복하여 모든 아이템을 출력한다.



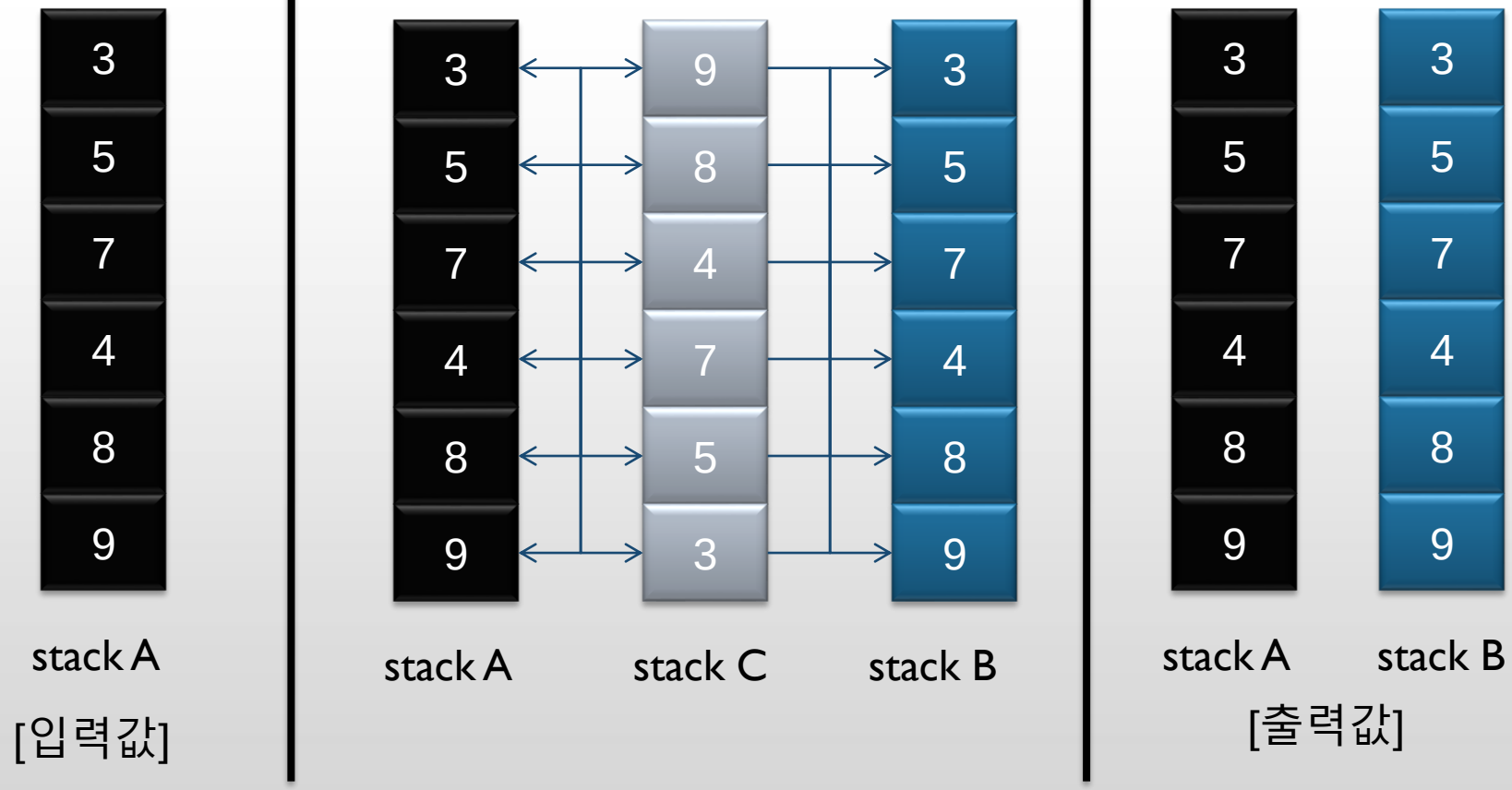
1-help slides(2/2)

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 result =  stack.top(); //   가장 최근에 넣은 아이템 값을 가져온다.
       stack.pop(); // 가장 최근에 넣은 아이템을 제거한다.
       cout << result << endl;
     }
}
```

2. Exercise

- 스택의 데이터를 변경하지 않고, 기존의 스택과 동일한 값을 가지는 스택을 만드세요.



A와 같은 stack인 B를 만들기 위해 임시로 저장할 C가 필요합니다.

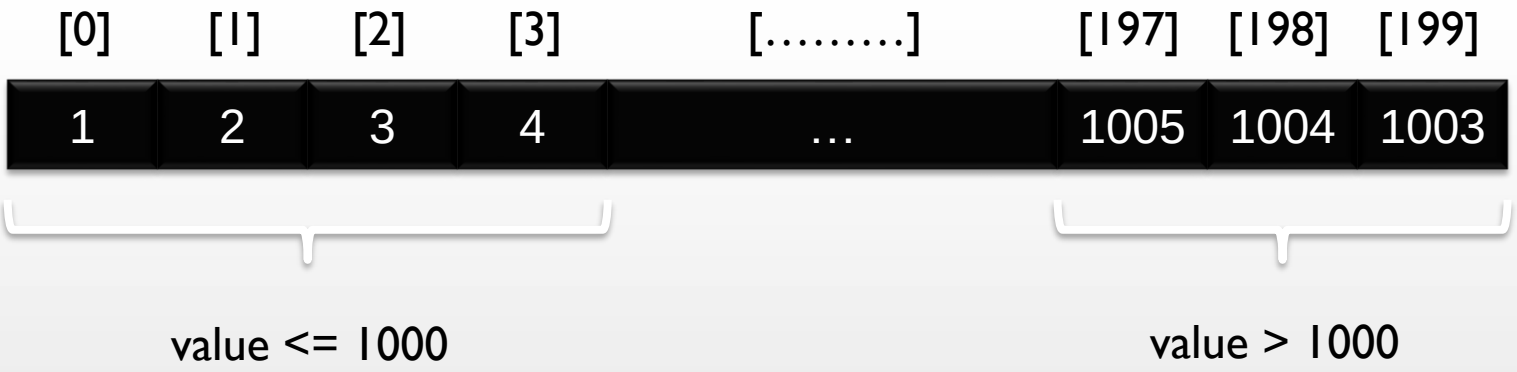
이번 문제는 클라이언트 함수로 작성해야 하며, 주어진 stack 구현 코드는 변경하면 안됩니다. Stack의 push, top, pop등을 사용하여 주어진 문제 a를 구현하세요.

3. Exercise

- **하나의 배열을 이용하여, 두 개의 스택을 구현하는 double stack클래스를 작성하세요.**
  - 첫 번째 스택은 1000이하의 수를 저장합니다.
  - 두 번째 스택은 1000을 넘는 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]



Push 할 때 값을 비교하여 0번부터 채워 넣거나, 199번부터 채워 넣습니다.
Pop, Top 연산은 생략하세요.
2개의 flag를(top을 기록하는 변수) 사용해서 stack을 관리합니다.
IsFull은 2개의 flag의 주소가 연속이면 full입니다.

3-help slides

- **B. A에서 생각한 double stack을 클래스로 정의해 보세요.**
- **C. double stack 클래스의 멤버 함수 중 Push 연산 부분을 구현해 보세요.**
- **D. 채점을 위해 저장된 아이템들을 확인 하는 Print()함수를 작성하세요.**
  - ❖ 스택pop 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); //C에서 구현할 push 연산
        void Print(); //stack 의 상황을 보여줄 수 있는 함수(채점시)
        // … (필요하다 생각되는…)
}
```

4. Exercise

■ 문제

❖ ◈각 스택의 복사본을 다른 것으로 치환하는 함수를 작성하여라. 다음과 같은 사양을 사용하여라(이 함수는 호출 프로그램이다.)

Replace Item
함수 : 모든 oldItem을 newItem으로 바꾼다.
조건 : 스택은 초기화되어 있다.
결과 : 스택에 있는 각각의 oldItem은 newItem으로 바뀌진다.

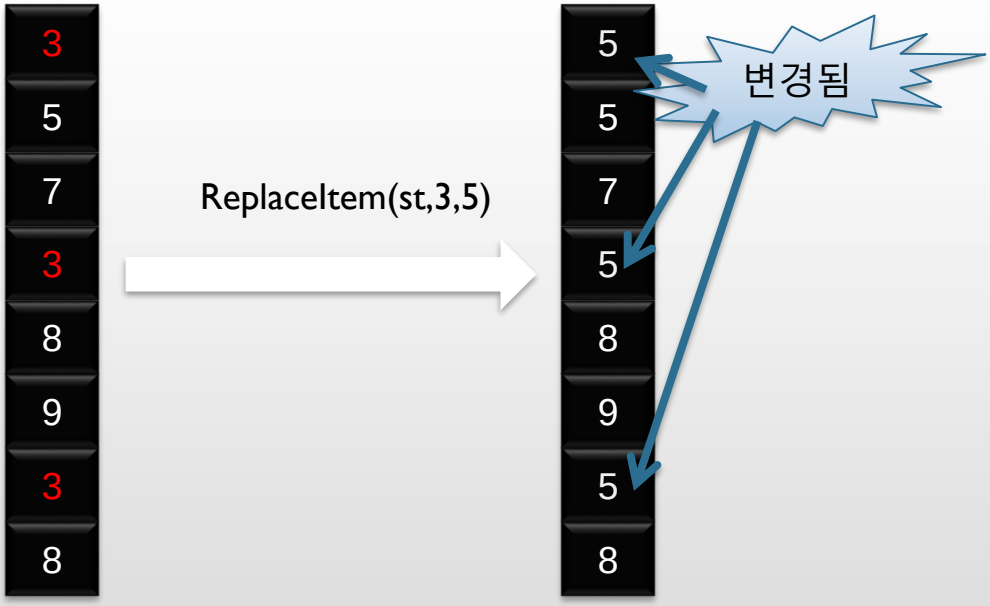
template로 작성한 stackType이 아닌 int 타입을 사용하는 StackType 클래스를 사용한다. (*경로 : ₩₩Iapplus₩₩Lab,C++3rd₩₩Chapter4₩₩stack₩₩Static)

A. ReplaceItem 함수를 StackType 클래스의 클라이언트로 작성한다.
B. ReplaceItem 함수를 StackType의 멤버 함수가 되도록 StackType을 수정한다.
*Template를 사용하지 않는 StackType 클래스를 사용한다.

문제 추가 설명

예제

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



stack st;

위와 같은 stack이 존재함을 가정

stack에서 3이란 값을 가지는 모든 아이템을 검사하여 5로 변환한다.

4-help slides (2/3)

- **a. 클라이언트로 작성은 스택의 구조는 변경하지 않고 외부에서 추가 함수를 작성하여 위의 동작을 하는 함수를 만들어라.**
  (함수는 스택, oldItem, newItem 3개의 파라메터를 갖는다.)

  ❖ Client function prototype :

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

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

int main()
{
   StackType stack;
   // stack에 Push연산을 통해 값을 입력함.
   Replace (stack, 바꾸고자 하는 값, 바뀌게 될 값);
   while(!stack.IsEmpty())            //스택 내용 출력
   {
      item = stack.Top();
      stack.Pop();
      cout <<"Item : "<<item << endl;
   }
   return 0;
}
```

```
void ReplaceItem(StackType &st, int oldItem,
int newItem)
{
//Exercise에서 행했던 것처럼, 또 다른 stack
을 정의하여 st의 item을 top, pop을 통해 꺼
내면서, 값을 비교. oldItem과 일치하는 값을
newItem으로 다른 stack에 저장합니다.
다음 임시로 생성했던 stack에서 아이템을
top, pop을 통해 꺼내면서 st에 다시 집어 넣
습니다.
}
```

4-help slides (3/3)

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

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

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