Lecture 9 : 실전 : STL Stack 활용하기

Stack이 필요한 경우 우리가 할 수 있는 대응은 2가지로 구분된다. 가장 직접적인 방법은 우리가 직접 STACK을 만들어 사용하는 것이다. 교재 대부분에서는 linear array<type>를 이용하는 방법을 설명하고 있다. 이 경우 사용자는 해당 배열, top pointer, push( ), pop( ) function을 준비해야 하고 더 중요한 것은 각 push( ), pop( ) 함수의 exception handling까지 잘 준비해야 한다. 다른 방법은 STL에서 제공하는 stack을 활용하는 것이다. 당연히 이 두 번째 방법에 훨씬 간결하고 안정적인 대처이다.

여러분이 STL stack을 잘 활용하기 위해서는 자신이 정의한 추상 자료를 원소로 하는 stack을 만들어 있어야 하고, push( ) pop( ) 함수에서 발생하는 다양한 오류를 제대로 해석할 수 있어야 한다.

STL stack에 제공되는 member function은 다음과 같다.

function name	함수의 내용
(constructor)	Stack을 생성함
empty( )	Stack이 비어있는지를 검사함
size( )	원소의 개수
top( )	top 원소를 copy 함.
push( )	새로운 원소를 insert
emplace( )	Construct and insert element
pop( )	top 원소를 stack에서 삭제함.
swap( )	C++11에서만 가능

9.1 STL에서 제공하는 stack을 사용하기 위해서는 미리 **#include <stack>**을 선언해야 한다. 아니면 모든 built-in STL을 한꺼번에 불러오기 위하여 다음과 같이 선언을 할 수도 있다. **#include <bits/stdc++.h>.**

단 이 방법은 표준이 아닌 de facto standard[1]이다. 즉 GNU C++의 표준 라이브러리 헤더가 아니라 gcc 컴파일러에서 사용되는 헤더이므로, 따로 설정해야지만 사용할 수 있다. 선언과 사용은 편하지만 system load가 올라갈 수 있다. 따라서 이런 선언은 프로그래밍 대회나 jon interview coding에 사용하면 좋다. 그러나 속도가 중요한 상황에는 필요한 module만 올려서 사용하는 것을 권장한다. UNIX, LINUX 등에서 사용한다면 따로 각 OS에 맞는 절차를 통하여 설치해야 한다. 코딩 시합, testing이 아닌 경우라면 사용하지 않는 편이 좋다. 이것을 너무 즐겨 쓰면 약간 게으른 프로그래머라고 오해를 받을 수 있다.

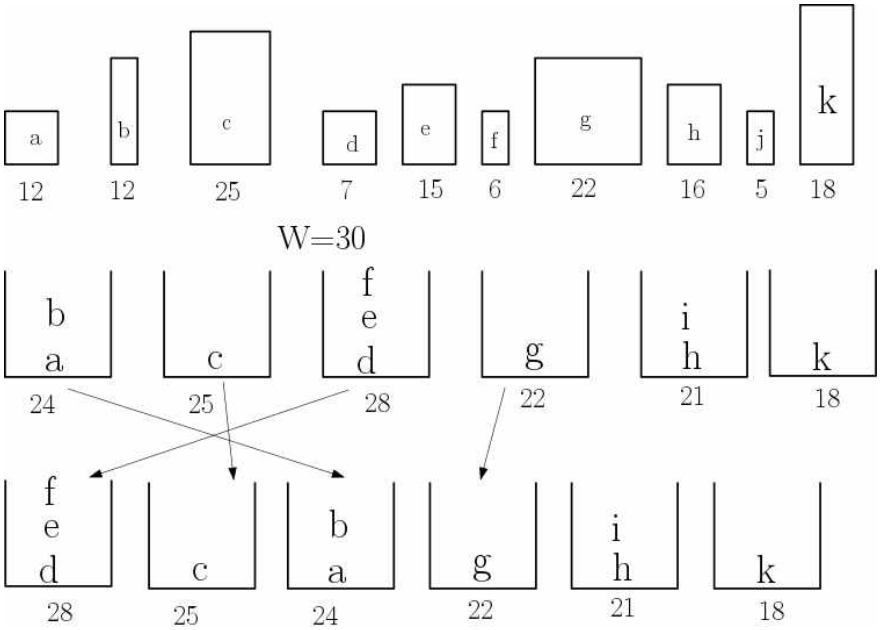
1) 표준은 아니지만, 현장에서 실질적인 표준으로 활용되는 상황. 예를 들어 문자 메시지를 “카톡”, 복사를 “제록스”, 굴착기를 “불도저”라고 부르는 것과 같다.

emplace( )와 push( )의 차이

```
#include <bits/stdc++.h>
using namespace std;

int main () {
  stack<string> mystack;
  string stemp = "토마토 주스" ;

  mystack.emplace ("데낄라");
  mystack.emplace ("레몬 즙");   // 데이터를 그 자체로 push( )
  mystack.emplace ("탄산수");
  mystack.emplace ("얼음");
  mystack.push( stemp ) ;     // 데이터를 그릇(변수) 담아서 push( )

  cout << "mystack contains:\n\n";
  int i = 0 ;
  while (!mystack.empty()) {
    cout << ++i << ".  " << mystack.top() << '\n';
    mystack.pop();
  }
  return 0;
}
```

9.2 (실전응용) 공항 수화물 문제. 맡긴 수화물은 작은 단위 stack으로 저장된 다음 이 작은 stack의 무게대로 정렬이 된 다음 단위 스택이 비행기 화물칸에 다시 stack으로 저장된 다음 도착하여 다시 Baggage Claim에서 나온다.

```
typedef stack<int>  intstack ;

int main() {
    intvec  myvec ;
    intstack mystack ;

    stack   <intstack> StackStack ;  // 원소가 Stack이 Stack
    vector  <intstack>  StackVec ;    // 원소가 Stack이 Vector
    vector  <intstack>::iterator iStackVec ;
```



이 경우 최종적으로 나오는 화물의 순서를 계산하는 프로그램을 생각해보자. 단 W은 한 단위 container에 담을 수 있는 최대 무게를 의미한다.

9.3 STL에서 제공해주는 함수에는 stack.pop( )과 stack.top( )이 분리되어 있다. 여러분은 안전한 stack pop 함수인 TYPE safepop(Stack)을 만들어 본다. 이것은 stack이 비어있는지 check을 해보고 아니라면 스택 top 원소, TYPE top을 return 시키고 하나의 원소를 삭제한다. 만일 스택이 비어있다면 오류를 발생시키고 control을 main driver routine을 넘긴다.

9.4 두 Stack의 "내용물"이 동일한 지의 여부를 == operator로 판단할 수 있다. 즉 == 연산자는 generic하게 활용된다.

9.5 Stack의 단위 원소를 내가 원하는 type으로도 만들 수 있다.

```
struct Mytype {
  string name ;
  int age ;
  int work[12] ;
  } t1, t2 ;

int main() {
     stack <Mytype> Mystack ;
     t1.name= "Good"  ;
     t1.age = 23 ;
     Mystack.push( t1 ) ;
     return 0;
}
```

9.6 여러분은 제시된 괄호 문자열이 제대로(valid) 짝이 지워진 것인지를 계산하여 RIGHT, WRONG을 출력하시오. (괄호 문자열의 끝은 '$' 문자로 표시되어 있다.)

- 준비된 괄호 기호가 열린 기호라면 →
- 준비된 괄호 기호가 닫힌 기호라면 →
- 준비된 기호가 끝을 나타내는 $ 기호라면 →

준비할 function으로는 어떤 것이 있을까?

```
a. match( this, that ) ;
b. final_symbol( ) ;
c. empty( ) ;
```

9.7 STL stack에 적용되는 모든 가능한 operation을 제시하시오. (어디서 찾지?)
Codeblock에서 auto completion을 사용해보기 / reset 하기

9.8 STL 스택을 동작시켜 어떤 오류가 발생하는지 모두 조사해보자. 그리고 그때 어떤 오류메시지를 만들어 주는지 그것을 모두 조사하여 기록하시오.

- Stack underflow
- 중앙값 access 하거나 update하기
- Stack copy

9.9 STL에서 제공해주는 stack에서 알려주는 다양한 오류에는 어떤 것이 있는지 확인해 보고 실제 그러한 오류를 생성시켜 보시오.

Q) 이런 오류생성 작업의 의미를 설명하시오. (멀쩡할 때 병원을 가야 하는 이유)

다음 문제를 STL stack을 이용해서 작성하시오. 단 코드는 짧으면 짧을수록 좋다.[2] 단 이 과정에서 여러분은 복수의 stack만을 사용할 수 있으며 다른 보조 배열(array)이나 vector는 사용할 수 없다. 사용할 수 있는 stack의 수는 최대 3개를 초과할 수 없다. 만일 이 제한으로 제시한 작업이 불가능하면 그 사실을 표현해야 한다.

9.10 [코딩 문제] Stack Middle Popping
stack에 N개의 원소가 저장되어 있다. 이 stack에서 int(N/2) 번째 원소를 제거한 stack을 만들어 보시오.

2) 코드의 "길이"는 statement의 개수로 계산한다.

[1, 2, 3, 4, 5, 6, 7, 8] top ==> [1, 2, 3, 5, 6, 7, 8]

9.11 [코딩 문제] Stack Reversing

N개의 원소가 저장된 stack의 원소를 역순으로 정렬하여 저장된 stack을 만들어 보시오.

[1, 2, 3, 4, 5, 6, 7, 8] top ==> [8, 7, 6, 5, 4, 3, 2, 1,]

9.12 [코딩 문제] Stack Rotate

크기 N인 stack의 rotate한다. 즉 top 원소가 바닥에 깔리도록 한다. 나머지는 순서를 그대로 유지해야 한다.

[1, 2, 3, 4, 5, 6, 7, 8] top ==> [8, 1, 2, 3, 4, 5, 6, 7]