REPORT #4



|  |
| --- |
| **제 출 일** : 2019년 4월 5일 |
| **과 목 명** : 자료구조및실습 |
| **담당교수 : 김희철 교수님** |
| **학 과** : 컴퓨터전자시스템공학부 |
| **학 번** : 201803818 |
| **성 명 : 한아영** |

**HANKUK UNIVERSITY OF FOREIGN STUDIES**



**Problem Definition**

Postfix 계산기

* 피연산자는 실수형 상수이고 연산자는 + - \* /
* 수식을 입력하면 해당하는 답을 구하는 계산기
* 연산자에 대한 피연산자, 피연산자에 대한 연산자가 없는 경우 error 출력

**Solution**

Stack 클래스는 과제 3과 같다.

main클래스에서는

피연산자를 저장하기 위해 Stack s를 선언하였고

사용자의 입력을 저장할 string 형 변수 token을 선언하였습니다.

첫번째 피연산자, 두번째 피연산자를 각각 ItemType형 first, second라고 ,

연산 결과를 저장할 함수를 ItemType형 result라고 선언하였습니다.

while문에서 token을 계속 사용자로부터 입력 받으며 token에 대한 case를 if-else구문으로 분리하였습니다.

token이

1. ; 라면 while문을 탈출하게 됩니다.
2. + , \* , - , /라면

* 연산자가 나왔을 때에는 스택에 쌓은 피연산자 두개를 pop해 연산을 실행해야 하므로 먼저 스택이 비어 있는지 검사합니다. 비어 있다면 error1을 출력한다. 첫번째 피연산자, 두번째 피연산자를 pop해야하므로 검사도 총 두 번 하게 됩니다.

1. 음수일 경우

* token이 음수일 경우를 판별하기 위해서 먼저 token의 0번째가 – 인지와 동시에 1번째가 정수인지 검사합니다. 두 조건을 모두 만족한다면 token의 – 문자는 연산자가 아닌 부호입니다.
* 음수가 맞다고 판별되면 push합니다. 이때 실습과 달리 ItemType은 float이므로 atoi함수가 아닌 atof함수를 사용합니다.

1. 정수일 경우

* 양수 피연산자의 경우이므로 atof함수를 이용하여 push합니다.

사용자의 입력이 끝난 경우에는 마지막 스택에 저장되어 있는 data를 pop해 result에 저장하고 스택이 비어 있는지 검사합니다. 비어 있다면 피연산자에 대한 연산자가 부족한 상태이므로 error 2를 출력합니다. 비어 있지 않다면 result를 출력해 계산 결과를 알려 줍니다.

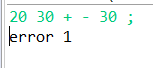
**Result**

<결과 1>



<결과 2>

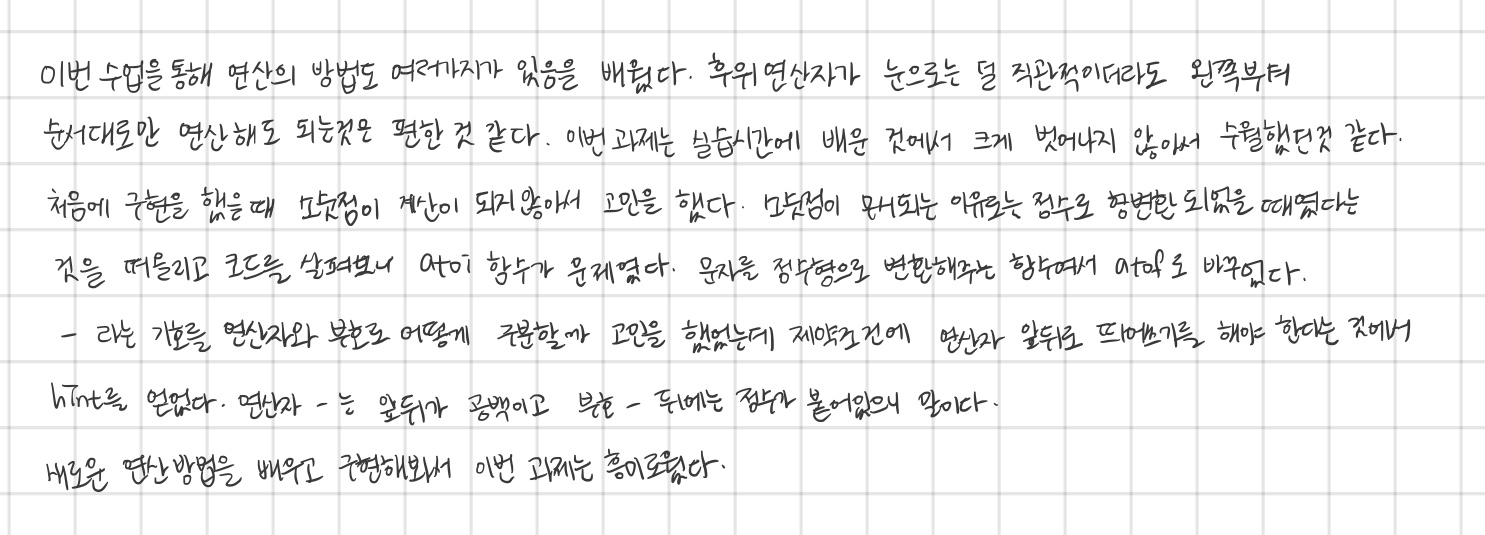
<결과 3>



<결과 4>



**Conclusion**



**Source Code**

**#include**<iostream>

**#include**<cstdlib>

**#include**<iomanip>

**using** **namespace** std;

**const** **int** MAXS = 100;

**typedef** **float** ItemType;

**class** Stack{

**int** top; //구현할 스택에서 가장 마지막에 저장 된 원소의 index

ItemType c\_data[MAXS]; //데이터를 저장할 배열

**public**:

**Stack**();

**~Stack**();

**bool** **is\_empty**(); //스택이 비어있는지 검사하는 함수

**bool** **is\_full**(); //스택이 포화상태인지 검사하는 함수

**void** **push**(ItemType item); //스택에 새로운 data를 저장하는 함수

ItemType **pop**(); //스택의 데이터를 반환하고 지우는 함수

};

**int** **main**(){

Stack s; //데이터를 저장할 스택

string token; //사용자로부터 입력받는 string을 저장할 변수

**while**(cin>>token){

**if**(token==";"){ // ;이면 while문 탈출

**break**;

}

**if**(**isdigit**(token[0])){ //token 앞자리가 정수라면 push

s.push(**atof**(token.c\_str()));

**continue**;

}

**else** **if**((**isdigit**(token[1]))&&(token[0]=='-')){ //앞자리가 -이고 다음 자리가 정수라면 이때는 -가 연산자가 아닌 부호이므로 이 때 피연산자는 음수이다.

s.push(**atof**(token.c\_str()));

**continue**;

}

/\*

\* 연산자일 경우

\* ㄱ) 스택이 비어있는지 검사하고 첫번째 피연산자를 pop (비어있다면 이는 연산자에대한 피연산자가 없는 경우이다 - error 1)

\* ㄴ) 스택이 비어있는지 검사하고 두번째 피연산자를 pop

\* ㄷ) 해당 연산 실행 후 결과값을 push

\*/

**else** **if**(token == "+"){

**if**(s.is\_empty()){

cout<<"error 1"<<**endl**;

**return** 0;

}

ItemType second = s.pop();

**if**(s.is\_empty()){

cout<<"error 1"<<**endl**;

**return** 0;

}

ItemType first = s.pop();

s.push(first+second);

**continue**;

}

**else** **if** (token=="-"){

**if**(s.is\_empty()){

cout<<"error 1"<<**endl**;

**return** 0;

}

ItemType second = s.pop();

**if**(s.is\_empty()){

cout<<"error 1"<<**endl**;

**return** 0;

}

ItemType first = s.pop();

s.push(first-second);

**continue**;

}

**else** **if**(token=="\*"){

**if**(s.is\_empty()){

cout<<"error 1"<<**endl**;

**return** 0;

}

ItemType second = s.pop();

**if**(s.is\_empty()){

cout<<"error 1"<<**endl**;

**return** 0;

}

ItemType first = s.pop();

s.push(first\*second);

**continue**;

}

**else** **if** (token=="/"){

**if**(s.is\_empty()){

cout<<"error 1"<<**endl**;

**return** 0;

}

ItemType second = s.pop();

**if**(s.is\_empty()){

cout<<"error 1"<<**endl**;

**return** 0;

}

ItemType first = s.pop();

s.push(first/second);

**continue**;

}

}

ItemType result = s.pop(); //result에 마지막 결과값을 저장

**if**(!s.is\_empty()){ //스택이 비어있다면 피연산자에 대한 연산자가 없는 경우이다. -error 2

cout<<"error 2" <<**endl**;

}

**else**{

cout << fixed << setprecision(2) << result << **endl**; //소수 둘째자리까지 출력

}

**return** 0;

}

**Stack::Stack**(){

top = -1;

}

**Stack::~Stack**(){}

**bool** **Stack::is\_empty**(){

**return** top==-1; //top 이 -1 일때 true 반환

}

**bool** **Stack::is\_full**(){

**return** top==MAXS -1;//top이 MAXS-1일때 true 반환

}

**void** **Stack::push**(ItemType item){

**if**(is\_full()){

cout<<"스택 포화 오류"<<**endl**;

**exit**(1);

}

top++;

c\_data[top]=item;

}

ItemType **Stack::pop**(){

**if**(is\_empty()){

cout<<"스택 비어있음 오류" <<**endl**;

**exit**(1);

}

ItemType item;

item = c\_data[top];

top--;

**return** item;

}