자료구조론
Homework # 2

이름 : 윤성호

학번 : 12161756

분반 : 004

제출일 : 2019/04/12

- 개요 -

(1) ArrayStack

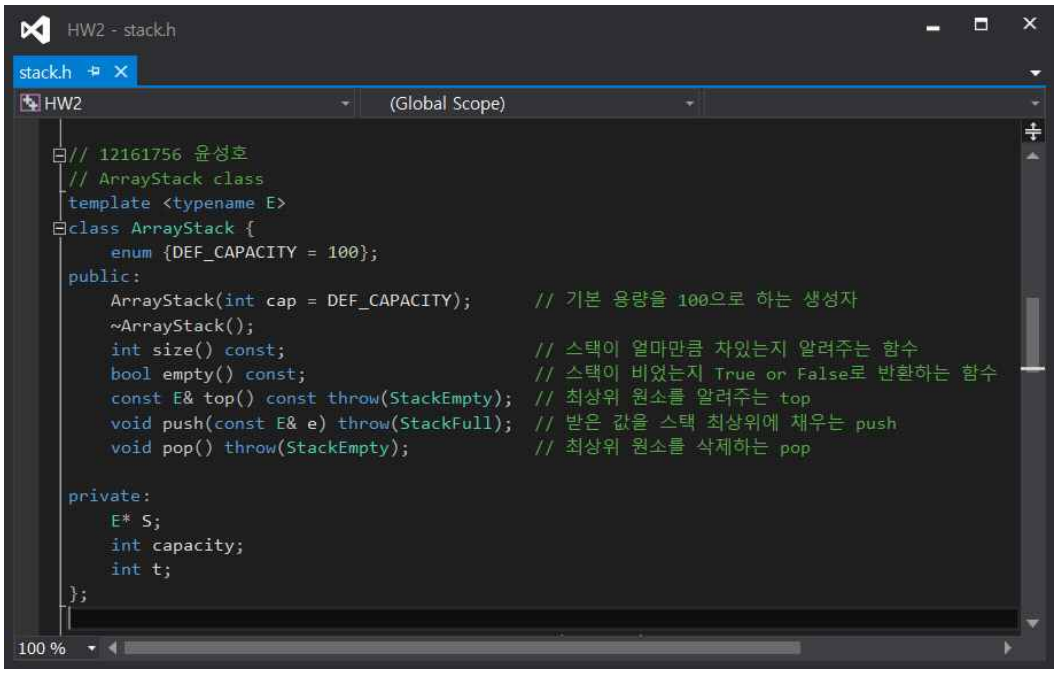
(2) 중위->후위표기식 변환 프로그램

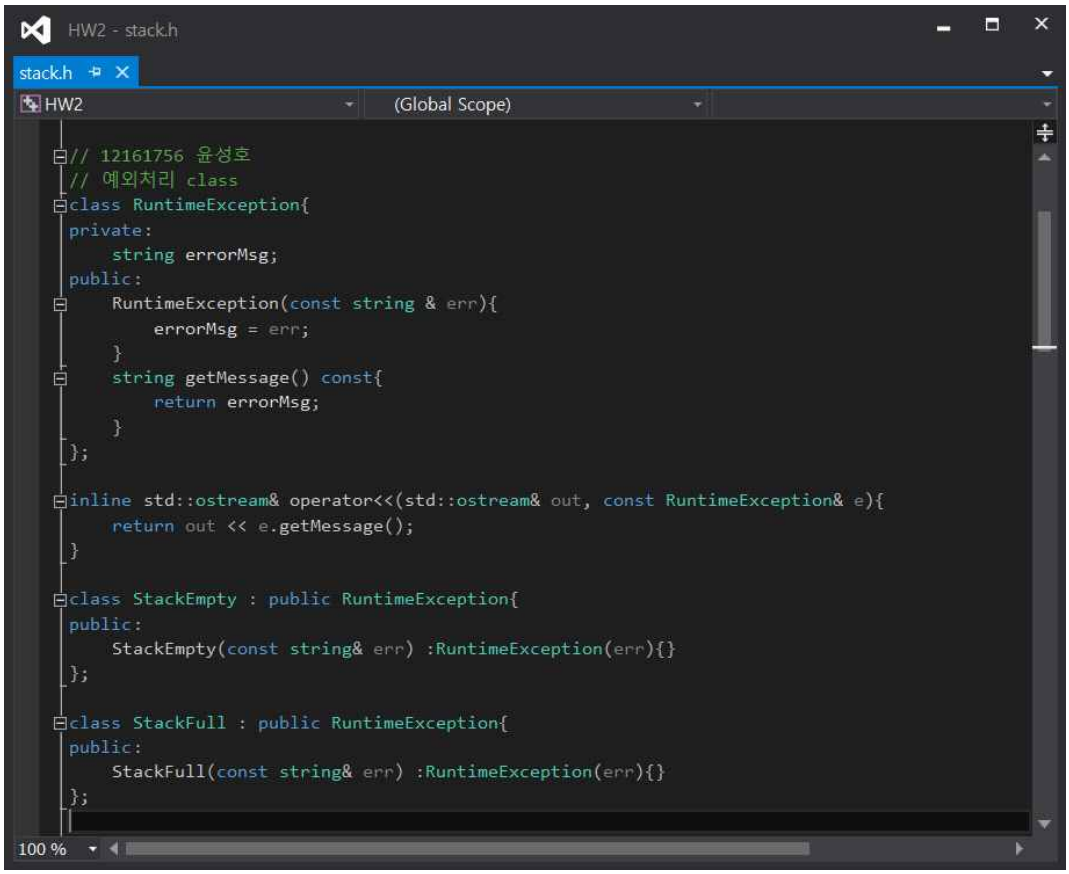
(1) ArrayStack

1. 구현상 특징

template 사용으로 인해 stack.h 와 main.cpp, 총 2 개의 파일로 구현하였습니다.

ArrayStack 과 예외처리 Class 의 대략적인 개요는 다음과 같습니다.





2. 실행 화면

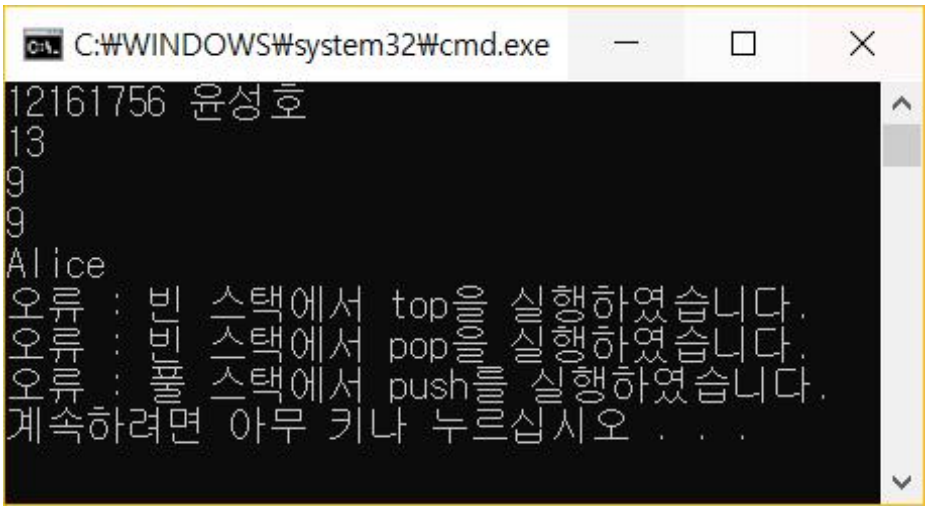
main.cpp 구현은 아래와 같으며, 3 가지의 예외 처리도 잘 이루어지고 있는 모습입니다.

Solution1

main.cpp

```cpp
// 12161756 윤성호
#include "stack.h"

int main(){
    cout << "12161756 윤성호" << endl;
    ArrayStack<int> A;                  // '*' : 스택 최상위 원소
    A.push(7);                          // A =[7*]
    A.push(13);                         // A =[7, 13*]
    cout << A.top() << endl; A.pop();   // A =[7*], output : 13
    A.push(9);                          // A =[7, 9*]
    cout << A.top() << endl;            // A =[7, 9*], output : 9
    cout << A.top() << endl; A.pop();   // A =[7*], output : 9

    ArrayStack<string> B(10);
    B.push("Bob");                      // B =[Bob*]
    B.push("Alice");                    // B =[Bob, Alice*]
    cout << B.top() << endl; B.pop();   // B =[Bob*], output : Alice
    B.push("Eve");                      // B =[Bob, Eve*]

    // 여기까지 현재 상태 : A =[7*], B =[Bob, Eve*], B의 capacity : 10

    A.pop();                            // A =[]
    A.top();                            // top 예외 발생 (empty 상태에서 top)
    A.pop();                            // pop 예외 발생 (empty 상태에서 pop)

    B.push("C"); B.push("D"); B.push("E"); B.push("F");  // B =[Bob, Eve, C, D, E, F*]
    B.push("G"); B.push("H"); B.push("I"); B.push("J");  // B =[Bob, Eve, C, D, ... , I, J*], 10번째 원소 : J
    B.push("K");                                        // push 예외 발생 (full인 상태에서 push)
}
```

100 %



(2) 중위->후위표기식 변환 프로그램

1. 구현상 특징

Solution1

stack2.h

```
#ifndef STACK2_H
#define STACK2_H

// 12161756 윤성호
// 기존 stack.h에서 추가된 부분(아래 3줄)
int getOpNum(char);    // token을 받아 그에 맞는 숫자를 반환하는 함수
bool confirm(char*);   // 중위표기식 배열을 받아 피연산자수 > 연산자수일때 true를 리턴하는 함수
void convert(char*);   // 중위표기식 배열을 받아 후위표기식으로 출력하는 함수
```

100 %

우선 과제요구사항에 맞게 (1)에서 작성한 코드를 재사용하였고 변환 프로그램 구현에 필요한 3 가지 함수만 기존 stack.h 파일 상단에 추가했습니다.

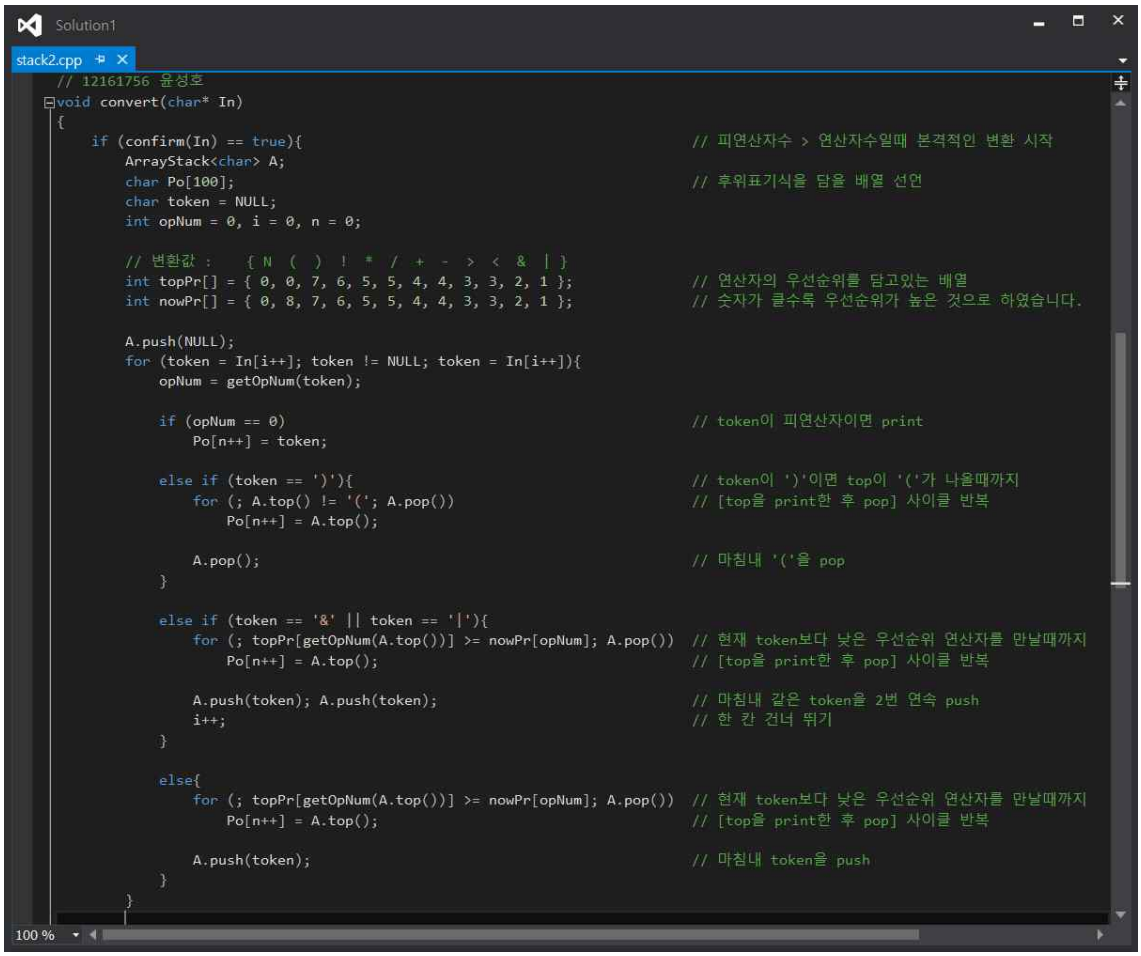
Solution1

stack2.cpp

```
// 12161756 윤성호
// 변환 함수 구현부
#include "stack2.h"

int getOpNum(char A)
{
    // 변환값 :          1   2   3   4   5   6   7   8   9   10  11
    char op[] = { NULL, '(', ')', '!', '*', '/', '+', '-', '>', '<', '&', '|' };
    int i;

    for (i = 1; i < 12; i++)
        if (op[i] == A)
            return i;                                   // A가 연산자이면 그에 따른 숫자(1~11)를 리턴
    return 0;                                           // A가 피연산자이면 0을 리턴
}

bool confirm(char* In){
    int i = 0, opNum = 0, countAnd = 0, countAtor = 0;
    char token = NULL;

    for (token = In[i++]; token != NULL; token = In[i++]){
        opNum = getOpNum(token);
        if (opNum == 0)
            countAnd++;
        else if (opNum >= 4 && opNum <= 9)              // 연산자수 계산에서 (, ), !는 제외하였습니다.
            countAtor++;
        else if (opNum >= 10){
            countAtor++;
            i++;
        }
        else
            continue;
    }

    return (countAnd > countAtor);                      // 피연산자수 > 연산자수일때 true 리턴
}
```

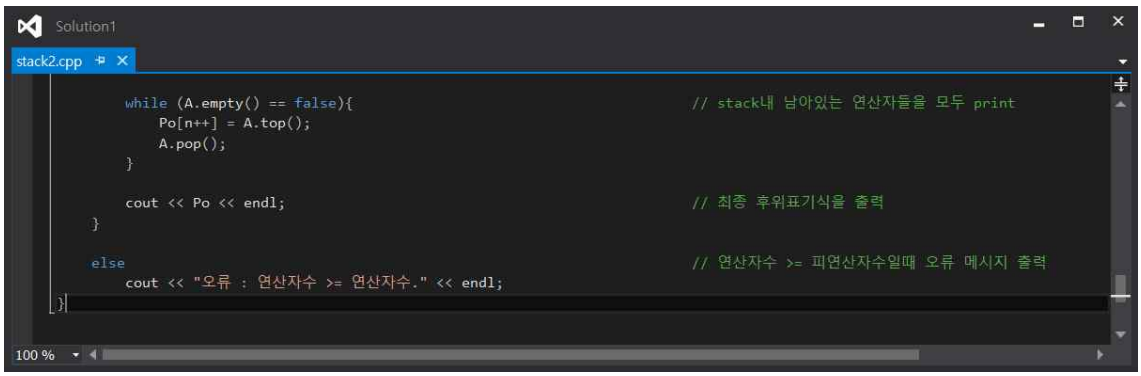
100 %

getOpNum 과 confirm 함수 구현부입니다. 과제에서 3)과 4)는 오류메시지 출력을 요구하고 있어 저는 이것을 연산자수가 피연산자수 이상일 때, 오류가 발생하는 것으로 가정했습니다. 6), 7)에서는 아무 언급이 없어서 3), 4)와 차이를 살펴보니 !를 연산자수에 포함하지 않으면 가정이 맞아떨어져 괄호와 함께 !도 연산자수 계산에서 제외하였습니다.



```
// 12161756 윤성호
void convert(char* In)
{
    if (confirm(In) == true){                                   // 피연산자수 > 연산자수일때 본격적인 변환 시작
        ArrayStack<char> A;
        char Po[100];                                           // 후위표기식을 담을 배열 선언
        char token = NULL;
        int opNum = 0, i = 0, n = 0;

        // 변환값 :    { N  (  )  !  *  /  +  -  >  <  &  | }
        int topPr[] = { 0, 0, 7, 6, 5, 5, 4, 4, 3, 3, 2, 1 };   // 연산자의 우선순위를 담고있는 배열
        int nowPr[] = { 0, 8, 7, 6, 5, 5, 4, 4, 3, 3, 2, 1 };   // 숫자가 클수록 우선순위가 높은 것으로 하였습니다.

        A.push(NULL);
        for (token = In[i++]; token != NULL; token = In[i++]){
            opNum = getOpNum(token);

            if (opNum == 0)                                     // token이 피연산자이면 print
                Po[n++] = token;

            else if (token == ')'){                             // token이 ')'이면 top이 '('가 나올때까지
                for (; A.top() != '('; A.pop())                 // [top을 print한 후 pop] 사이클 반복
                    Po[n++] = A.top();

                A.pop();                                        // 마침내 '('을 pop
            }

            else if (token == '&' || token == '|'){
                for (; topPr[getOpNum(A.top())] >= nowPr[opNum]; A.pop())  // 현재 token보다 낮은 우선순위 연산자를 만날때까지
                    Po[n++] = A.top();                          // [top을 print한 후 pop] 사이클 반복

                A.push(token); A.push(token);                   // 마침내 같은 token을 2번 연속 push
                i++;                                            // 한 칸 건너 뛰기
            }

            else{
                for (; topPr[getOpNum(A.top())] >= nowPr[opNum]; A.pop())  // 현재 token보다 낮은 우선순위 연산자를 만날때까지
                    Po[n++] = A.top();                          // [top을 print한 후 pop] 사이클 반복

                A.push(token);                                  // 마침내 token을 push
            }
        }
```

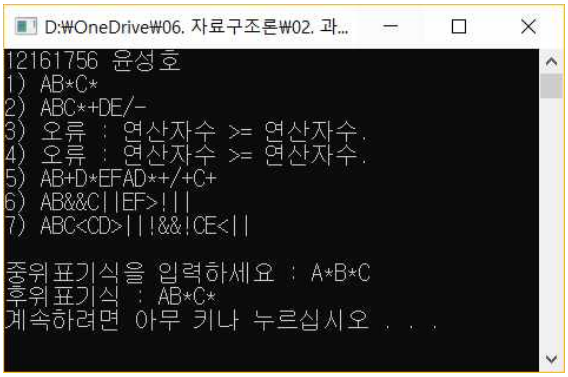
convert 함수 구현부(1/2)입니다. 후위표기식을 담을 배열의 크기는 임의로 100 을 주었습니다. 코드 중간에서 조금 밑에서 &와 | 토큰일 때, for 문을 돌고 빠져나와서 두 번 연속으로 push 한 후, 한 칸 건너뛰는 이유는 배열 한 칸에 하나의 연산자가 들어감으로써 발생한 일입니다. 다시 말하면, &&는 원래 하나의 연산자이지만, 배열에서는 &이 각각 두 개로 분리되어 들어가기 때문에, 저렇게 구현하지 따로 구현하지 않고, else 로 처리했다면 첫 &는 잘 push 되지만, 두 번째 &가 else 구문을 돌 때, 첫 &와 두 번째 &는 당연히 우선순위가 같으므로 첫 &는 pop 해버리기 때문에 이후 연산에 심각한 문제가 생깁니다.



```
        while (A.empty() == false){                             // stack내 남아있는 연산자들을 모두 print
            Po[n++] = A.top();
            A.pop();
        }

        cout << Po << endl;                                     // 최종 후위표기식을 출력
    }

    else                                                        // 연산자수 >= 피연산자수일때 오류 메시지 출력
        cout << "오류 : 연산자수 >= 연산자수." << endl;
}
```

convert 함수 구현부(2/2)입니다.

2. 실행 화면



문제없이 실행이 잘 이루어지고 있는 모습입니다.