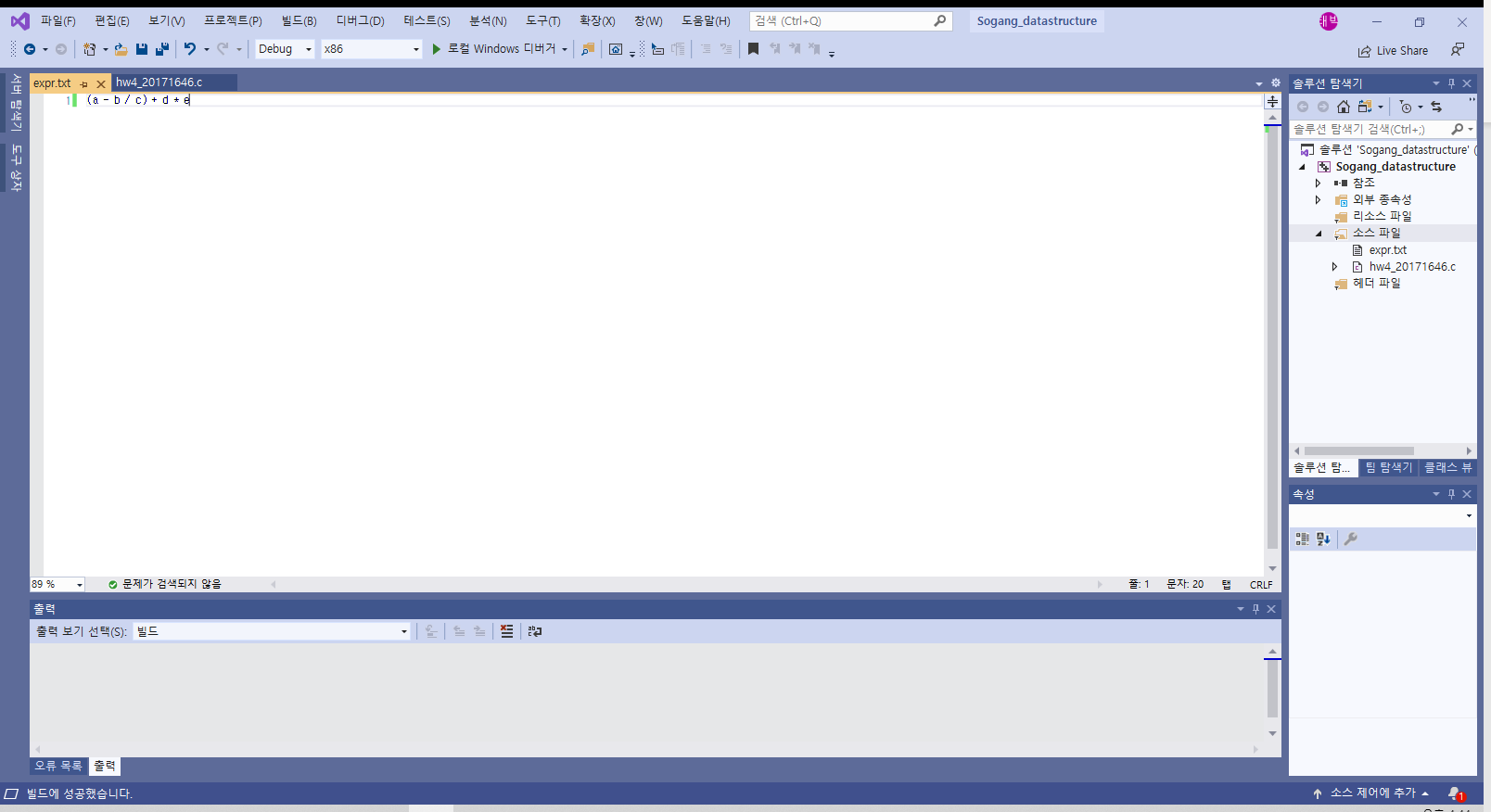
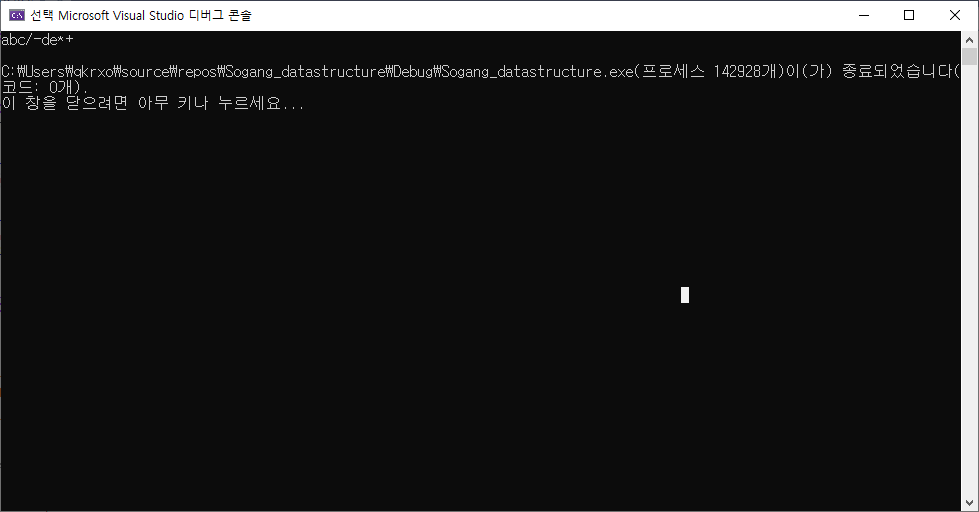
자료구조 오픈랩4

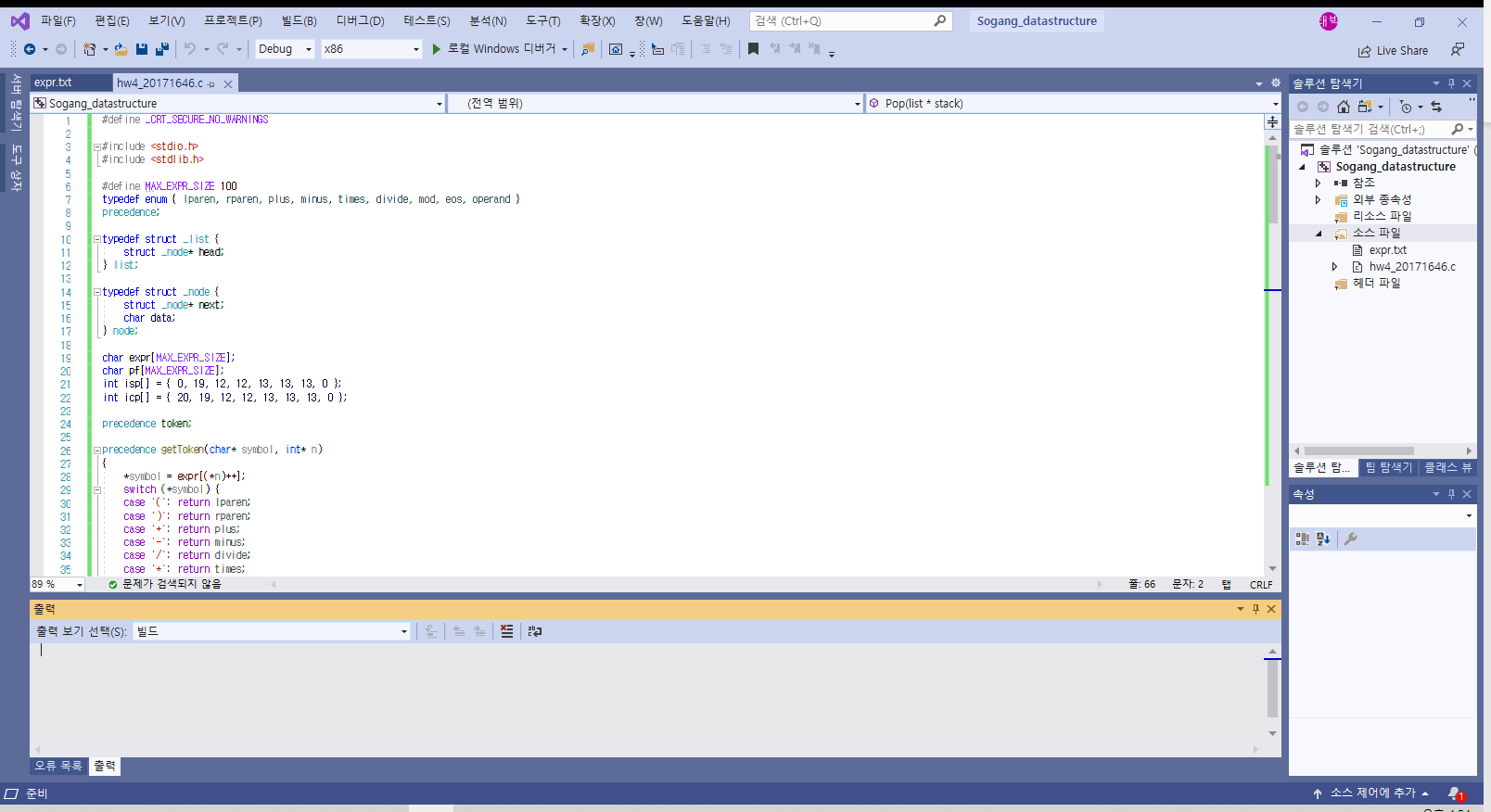
컴퓨터공학과 20171646 박태윤





Infix형태로 주어진 식이 있는 “expr.txt”와 이를 Postfix형태로 바꾸어 출력하는 코드를 실행시킨 결과이다.

(코드 / 알고리즘 설명)



소괄호를 포함한 연산자를 나타내기 위해 precedence라는 열거형을 선언하였으며 stack을 연결리스트로 구현하기 위해 다음과 같이 구조체 node와 list를 선언하였다. char형 배열 expr은 텍스트파일에 쓰여진 Infix형태의 식을 받기 위해, pf배열은 postfix형태로 변환된 식을 저장하기 위해 선언하였으며 int형 배열 isp와 icp는 연산자의 우선순위를 나타내기 위해 선언하였다. 마지막으로 precedence형 변수 token을 선언하였다.

**- main**

스크린샷, 노트북, 컴퓨터, 모니터이(가) 표시된 사진

자동 생성된 설명

메인 함수의 코드는 위의 그림과 같다. char형 변수 c를 선언하여 expr배열에 Infix형태의 식을 넣을 때 띄어쓰기를 제외하고 값을 넣었다. 이후 postfix함수를 호출하였다.

**- function of stack**

|  |  |
| --- | --- |
| 스크린샷, 노트북, 모니터, 컴퓨터이(가) 표시된 사진  자동 생성된 설명 | 스크린샷, 모니터, 컴퓨터, 노트북이(가) 표시된 사진  자동 생성된 설명 |

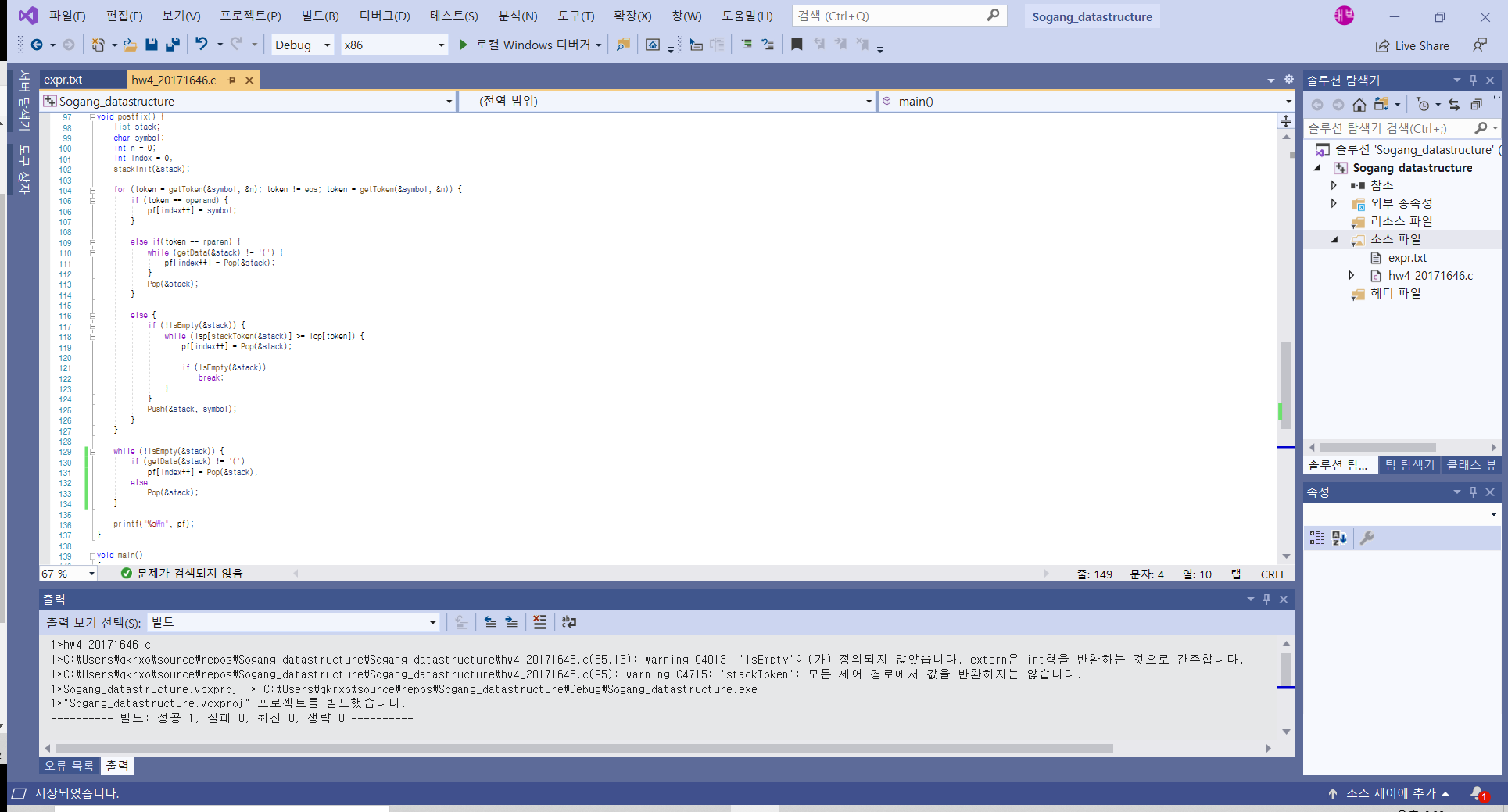
연결리스트로 구현한 stack에 관련한 함수들이다. Push함수는 새로운 노드를 만들어 그 노드에 데이터 값을 넣은 뒤 스택의 head에 추가해주는 함수이다. Pop함수는 스택에서 head가 가리키는 노드를 소멸시킨 뒤 소멸한 노드에 있던 데이터 값을 반환하는 함수이다. IsEmpty함수는 스택이 비어있는지를 확인시켜주는 함수이며 stackInit함수는 스택을 초기화시켜주는 함수이다. getData함수는 스택의 head가 가리키는 노드의 데이터 값을 반환하는 함수이며 마지막으로 stackToken함수는 스택의 head가 가리키는 데이터 값에 따른 precedence열거형의 값을 반환하는 함수이다.

**- postfix & getToken**

스크린샷, 노트북, 컴퓨터, 모니터이(가) 표시된 사진

자동 생성된 설명

getToken함수는 Infix형태의 식이 저장된 expr배열에서 문자 하나를 읽은 뒤 그 문자에 대응하는 precedence열거형 값을 반환하는 함수이다. 읽은 문자가 문장의 끝을 표현하는 ‘\0’ or 소괄호 or 연산자가 아니라면 이를 피연산자로 보고 operand를 반환한다.



postfix함수에서 stack을 만든다. expr배열의 값을 받아주는 symbol변수와 expr배열의 인덱스를 표현하는 n변수, pf배열의 인덱스를 표현하는 index변수를 선언하였다. 이후 for문을 선언하였는데, getToken함수에서 eos를 반환하기 전까지 token변수에 그 값을 저장하면서 반복문을 진행한다.

반복문 안에는 if문이 선언되어 있는데 symbol 값에 대응하는 열거형 값을 나타내는 token이 나타내는 값이 operand, 즉 피연산자라면 pf배열에 그대로 symbol 값을 넣어준 뒤 index를 하나 증가시킨다. token이 피연산자가 아니라 rparen( 오른쪽 괄호 ‘)’ )을 나타낸다면 스택에 쌓여있는 값이 왼쪽 괄호( ‘(‘ )를 나타낼 때 까지 Pop을 시키면서 pf배열에 그 반환 값을 저장한다. 이후 스택에 남아있는 왼쪽 괄호를 빼주기 위해 Pop을 한번 더 진행한다. token이 피연산자도 아니고 오른쪽 괄호도 아닌 왼쪽 괄호를 포함한 연산자들을 나타낸다면 else문을 진행한다. 스택이 비어있지 않다면 while문을 진행하는데 이 반복문은 스택에 쌓여있는 연산자들과 token이 나타내는 연산자의 우선순위를 비교하여 pf배열에 값을 넣어주는 역할을 한다. 반복문이 끝나면 최종적으로 symbol값을 스택에 Push해준다. stack에 쌓여있는 연산자가 token이 나타내는 연산자보다 우선순위가 높거나 같다면 pf배열에 Pop을 통해 값을 넣어주고 Pop을 한 뒤 스택이 비어있는 상태가 된다면 반복문을 끝낸다. 예를 들어, 현재 스택의 헤드가 가리키는 노드에 ‘\*’이 저장되어있고 expr[n] = ‘+’라 token에 열거형 값 중 하나인 plus가 저장되어 있다면 isp[stackToken(&stack)] = 13이고 icp[token] = 12이기 때문에 pf배열에 값을 집어넣는다.

isp와 icp로 우선순위를 나타내는 배열이 구분되어있는 이유는 왼쪽 괄호 때문인데, Infix식에서 괄호 안에 있는 식들은 연산에서 높은 우선순위를 가져가기 때문에 icp에서 왼쪽 괄호가 나타내는 값은 20으로 가장 높은 값을 가지며 최우선적으로 stack에 push해준다. 오른쪽 괄호가 나타나기 전까지 스택에 쌓여있는 왼쪽 괄호는 Pop되면 안되기때문에 isp에서는 가장 낮은 값인 0을 가진다.

이러한 과정을 진행하다가 expr에 저장된 값을 모두 읽어 for문이 끝난다면 스택에 쌓여있는 왼쪽 괄호인 ‘(‘을 제외한 연산자들을 Pop을 통해 pf배열에 저장하고 pf배열을 출력한다.