

HW7: Infix → Postfix

제출 방법

클래스룸 과제 제출

제출 기한

실습일로부터 2주 뒤 00시까지.(지각제출: 조교 이메일, 1주 추가 기한, 50% 감점)

예) 금요일에 실습했다면 2주 뒤 목요일 23:59:59까지

제출파일: 필수파일 + 알파

- 소스코드
 - StackInterface.h
 - ArrayStack.h
 - ArrayStack.cpp
 - post.h
 - post.cpp
 - hw7.cpp
 - *.tex
 - *.pdf
- 이미지파일
 - post.in, post2.in, post3.in, post4.in 각 입력 파일에 대한 출력 결과 사진
- 그 외 기타 필요한 사진/파일들

실행결과에 본인 학번이 같이 출력될 수 있도록 코드를 임의로 수정

latex 내용:

1. stack에 관한 설명
2. 무슨 작업을 한 건지 알아볼 수 있게 latex에 작성부분 코드와 설명.
3. 최대한 자세하게 작성.
4. 어려운 점이 있었다면 어려웠던 점.

실습 시작 전 만들어야 하는 파일들:

(a) infix notation으로 된 표현식들로 이루어진 각 파일별로 4가지 테스트 케이스들을 만든다

post.in

```
A * B * C
-A + B - C + D
A * -B + C
(A + B) * D + E / (F + A * D) + C
```

post2.in

```
A && B || C || !(E>F)
!(A&&!((B<C)|| (C>D))) || (C<E)
```

post3.in

```
34*56+11/2
1+2*3+-4*2
```

post4.in

```
33+55*2
an77=2+7*5
b=2
an77+b*2
a+5
```

2. 다음 같이 동작하는 프로그램을 작성하려 한다.

```
>> ./ hw5 < post.in

A B * C *
A -u B + C - D +
A B -u * C +
A B + D * E F A D * + / + C +
```

(a) main 프로그램(hw.cpp)은 다음과 같다. 아래와 같이 작성 후 빈칸을 채우시오.

※ main 프로그램 안의 구성 변경 가능

※ 주어진 함수는 적절하게 변형 가능

```
#include <iostream>
#include "post.h"

using namespace std;

void Postfix(Expression);

int main() {
    char line[MAXLEN];
    while (/* 어떤 함수를 써야할지 고민 */(line, MAXLEN)) {
        Expression e(line); // line 버퍼를 이용하여 Expression 을 읽음
        try {
            Postfix(e);
        }
        catch (char const *msg) {
            cerr << "Exception: " << msg << endl;
        }
    }
}
```

(b) 다음 프로그램을 post.h로 저장.

```
#ifndef __POSTFIX_H__
#define __POSTFIX_H__
#include <iostream>
// token types for non one-char tokens
#define ID 257
#define NUM 258
#define EQ 259
#define NE 260
#define GE 261
#define LE 262
#define AND 263
#define OR 264
#define UMINUS 265
#define MAXLEN 80

struct Expression {
    Expression(char* s) : str(s), pos(0) {
        for (len = 0; str[len] != '\0'; len++)
            ;
    }
    char* str;
    int pos;
    int len;
};

struct Token {
    bool operator==(char);
    bool operator!=(char);
    Token();
    Token(char); // 1-char token: type equals the token itself
    Token(char, char, int); // 2-char token(e.g. <=) & its type(e.g.LE)
    Token(char*, int, int); // operand with its length & type(defaulted to ID)
    bool IsOperand(); // true if the token type is ID or NUM
    int type; // ascii code for 1-char op; predefined for other tokens
    char* str; // token value
    int len; // length of str
    int ival; // used to store an integer for type NUM; init to 0 for ID
};

using namespace std;
ostream& operator<<(ostream&, Token t);
Token NextToken(Expression&, bool); // 2nd arg = true for infix expression
#endif
```

(c) 다음 post.cpp을 완성하라.

```
#include "post.h"

#include <iostream>

#include "ArrayStack.h"

using namespace std;

bool Token::operator==(char b) { return len == 1 && str[0] == b; }

bool Token::operator!=(char b) { return len != 1 || str[0] != b; }

Token::Token() {}

Token::Token(char c) : len(1), type(c) {
    str = new char[1];
    str[0] = c;
} // default type = c itself

Token::Token(char c, char c2, int ty) : len(2), type(ty) {
    str = new char[2];
    str[0] = c;
    str[1] = c2;
}

Token::Token(char* arr, int l, int ty = ID) : len(l), type(ty) {
    str = new char[len + 1];
    for (int i = 0; i < len; i++) str[i] = arr[i];
    str[len] = '\0';
    if (type == NUM) {
        ival = arr[0] - '0';
        for (int i = 1; i < len; i++) ival = ival * 10 + arr[i] - '0';
    } else if (type == ID)
        ival = 0;
    else
        throw "must be ID or NUM";
}

bool Token::IsOperand() { return type == ID || type == NUM; }

ostream& operator<<(ostream& os, Token t) {
    if (t.type == UMINUS)
        os << "-u";
    else if (t.type == NUM)
        os << t.ival;
    else
        for (int i = 0; i < t.len; i++) os << t.str[i];
    os << " ";
    return os;
}

bool GetID(Expression& e, Token& tok) {
    char arr[MAXLEN];
```

```

int idlen = 0;
char c = e.str[e.pos];
if (!(c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z')) return false;
arr[idlen++] = c;
e.pos++;
while ((c = e.str[e.pos]) >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z' ||
        c >= '0' && c <= '9') {
    arr[idlen++] = c;
    e.pos++;
}
arr[idlen] = '\0';
tok = Token(arr, idlen, ID); // return an ID
return true;
}

bool GetInt(Expression& e, Token& tok) {
    // 이 부분을 작성하세요
}

void SkipBlanks(Expression& e) {
    char c;
    while (e.pos < e.len && ((c = e.str[e.pos]) == ' ' || c == '\t')) {
        e.pos++;
    }
}

bool TwoCharOp(Expression& e, Token& tok) {
    // 7 가지 두글자 토큰들 <= >= == != && || -u 을 처리
    char c = e.str[e.pos];
    char c2 = e.str[e.pos + 1];
    int op; // LE GE EQ NE AND OR UMINUS
    if (c == '<' && c2 == '=')
        op = LE;
    else if // 이 부분 작성: 각 두 글자 토큰에 대해 알맞은 type 값 op 에 저장
           // 코드 작성
        else {
            return false;
        } // 맞는 두 글자 토큰이 아니면 false 를 return
    tok = Token(c, c2, op);
    e.pos += 2;
    return true;
}

bool OneCharOp(Expression& e, Token& tok) {
    char c = e.str[e.pos];
    if (c == '-' || c == '!' || c == '*' || c == '/' || c == '%' || c == '+' ||
        c == '<' || c == '>' || c == '(' || c == ')' || c == '=') {
        tok = Token(c);
        e.pos++;
        return true;
    }
    return false;
}

Token NextToken(Expression& e, bool INFIX = true) {
    static bool oprrrFound = true; // 종전에 연산자 발견되었다고 가정.
    Token tok;

```

```

SkipBlanks(e);          // skip blanks if any
if (e.pos == e.len) {   // No more token left in this expression
    if (INFIX) oprrFound = true;
    return Token('#');
}
if (GetID(e, tok) || GetInt(e, tok)) {
    if (INFIX) oprrFound = false;
    return tok;
}
if (TwoCharOp(e, tok) || OneCharOp(e, tok)) {
    if (tok.type == '-' && INFIX && oprrFound) // operator 후 - 발견
        tok = Token('-', 'u', UMINUS); // unary minus(-u)로 바꾸시오
    if (INFIX) oprrFound = true;
    return tok;
}
throw "Illegal Character Found";
}

int icp(Token& t) { // in-coming priority
    int ty = t.type;
    // 이 부분 작성
    // ty 가 '('면 0,
    // UMINUS 나 '!'면 1,
    // '*' 나 '/' 나 '%'면 2,
    // '+' 나 '-'면 3,
    // '<' 나 '>' 나 LE 나 GE면 4,
    // EQ 나 NE면 5,
    // AND면 6,
    // OR 이면 7,
    // '=' 이면 8,
    // '#'면 9 를 return 한다.
}

int isp(Token& t) // in-stack priority
{
    int ty = t.type; // stack 에서의 우선순위 결정
    // 이 부분 작성
}

void Postfix(Expression e) {
    // infix expression e 를 postfix form 으로 바꾸어 출력
    // e 에 토큰이 없으면 NextToken 은 '#' 토큰을 반환한다.
    // 스택의 밑에도 '#'를 넣고 시작한다.

    // HINT:
    // expression e 에서...
    // token 이 피연산자인 경우
    // token 이 ')' 인 경우
    // 그 외의 경우

    // expression e 는 다 끝났고 stack 이 비어있지 않은 경우
}

```

Stack 관련 파일들(교재 참고해서 구현)

StackInterface.h

```
#ifndef __STACK_INTERFACE__
#define __STACK_INTERFACE__

#include <stdexcept>
#include <string>

class PrecondViolatedExcep : public std::logic_error {
public:
    PrecondViolatedExcep(const std::string& message = "")
        : std::logic_error("Precondition Violated Exception: " + message) {}
};

template <typename T>
class StackInterface {
public:
    /**
     * Stack 이 비었는지 여부를 확인합니다.
     * @return 비었으면 true, 아니면 false
     */
    virtual bool is_empty() const = 0;

    /**
     * Stack 의 최상단에 새로운 아이템을 추가합니다
     * @post item 이 Stack 의 최상단에 추가됩니다.
     * @param item 추가할 아이템
     * 교재와 차이점: 더 추가할 수 없을 경우 용량을 늘리게끔 설정
     */
    virtual void push(const T& item) = 0;

    /**
     * Stack 의 최상단 아이템을 제거합니다.
     * @post 성공적으로 수행된다면, Stack 의 최상단 아이템이 제거됩니다.
     * @return 성공적으로 제거되었다면 true, 아니면 false
     */
    virtual bool pop() = 0;

    /**
     * Stack 의 최상단 아이템의 사본을 반환합니다.
     * @pre Stack 이 비어있지 않아야 합니다.
     * @post Stack 의 최상단 항목의 사본이 반환되고, stack 의 상태는 변하지
     * 않습니다.
     * @return Stack 의 최상단 아이템의 사본
     */
    virtual T peek() const = 0;

    /**
     * stack 의 모든 아이템을 제거합니다.
     * @post Stack 이 비어있게 됩니다.
     */
    virtual void clear() = 0;
};
```

```

/**
 * 할당된 메모리를 해제하고, stack 을 소멸시킵니다.
 */
virtual ~StackInterface() {}
};

#endif // __STACK_INTERFACE__

```

ArrayStack.h

```

#ifndef __ARRAY_STACK__
#define __ARRAY_STACK__

#include "StackInterface.h"

template <typename T>
class ArrayStack : public StackInterface<T> {
private:
    static const int DEFAULT_CAPACITY = 10; // 기본 용량
    int capacity = DEFAULT_CAPACITY;
    T* items; // 스택 아이템을 저장할 배열
    int top; // 스택의 최상단 인덱스
public:
    ArrayStack();
    bool is_empty() const noexcept;
    void push(const T& item);
    bool pop();
    T peek() const;
    void clear() { top = -1; }
    ~ArrayStack();
};

#include "ArrayStack.cpp"

#endif // __ARRAY_STACK__

```

ArrayStack.cpp

```

#include "ArrayStack.h"

#include <cassert>

template <typename T>
ArrayStack<T>::ArrayStack()

template <typename T>
ArrayStack<T>::~~ArrayStack() {
}

template <typename T>
bool ArrayStack<T>::is_empty() const noexcept {
}

```

```
template <typename T>
void ArrayStack<T>::push(const T& item) {
}

template <typename T>
bool ArrayStack<T>::pop() {
}

template <typename T>
T ArrayStack<T>::peek() const {
}
```

Makefile

```
CC = clang++

# C++ 컴파일러 옵션
CXXFLAGS = -Wall -O2 -g -std=c++11

# 링커 옵션
LDFLAGS =

# 소스 파일 디렉토리
SRC_DIR = ./src

# 오브젝트 파일 디렉토리
OBJ_DIR = ./obj

# 생성하고자 하는 실행 파일 이름
TARGET = hw7

SRCS = hw7.cpp post.cpp

OBJS = $(SRCS:.cpp=.o)

OBJECTS = $(patsubst %.o,$(OBJ_DIR)/%.o,$(OBJS))
DEPS = $(OBJECTS:.o=.d)

all: $(TARGET)

$(OBJ_DIR)/%.o : $(SRC_DIR)/%.cpp
    $(CC) $(CXXFLAGS) -c $< -o $@ -MD $(LDFLAGS)

$(TARGET) : $(OBJECTS)
    $(CC) $(CXXFLAGS) $(OBJECTS) -o $(TARGET) $(LDFLAGS)

.PHONY: clean all
clean:
    rm -f $(OBJECTS) $(DEPS) $(TARGET)

-include $(DEPS)
```

tasks.json

```
{
  "tasks": [
    {
      "label": "Build with makefile",
      "type": "shell",
      "command": "make",    // window->"mingw32-make"
      "group": {
        "kind": "build",
        "isDefault": true
      },
      "problemMatcher": ["$gcc"]
    }
  ],
  "version": "2.0.0"
}
```

launch.json

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "lldb",
      "request": "launch",
      "name": "Debug with Makefile",
      "program": "${workspaceFolder}/hw7",    // window: -> "hw7.exe"
      "args": [],
      "cwd": "${workspaceFolder}",
      "preLaunchTask": "Build with makefile",
      "stopOnEntry": false,
      "stdio": ["${workspaceFolder}/post.in", null, null]
    }
  ]
}
```

올바른 설정

```
.
├── .vscode
│   ├── launch.json
│   ├── settings.json
│   └── tasks.json
├── makefile
├── obj          <- Caution! directory
├── post.in
├── post2.in
├── post3.in
├── post4.in
└── src
    ├── ArrayStack.cpp
    ├── ArrayStack.h
    ├── hw7.cpp
    ├── post.cpp
    ├── post.h
    └── StackInterface.h
```