HW2

1. 소스코드

```
#include<iostream>
#include<stack>
#include<vector>
#include<string>
#include<sstream>

usingnamespacestd;

structCharIntConstruct{
    inttype; // 0 == 숫자, 1 == 연산자
    intnumber;
    charsymbol;
};

// 우선순위 정하는 함수
intprecedence(charitem) {
    if(item =='+'||item =='-') return1;
    if(item =='*'||item =='/') return2;
    if(item =='^') return3;
    return0;
}

// 후위 표기법 변환 함수
vector<CharIntConstruct> make_postfix(istream&ins) {
    vector<CharIntConstruct>postfix;// 후위 표기법을 저장할 벡터
    stack<char>item;// 연산자를 저장할 스택

    charch;
    boolreadingNumber =false; // 숫자를 읽고 있는 중인지 확인
    intnumber =0;          // 읽은 숫자를 저장할 변수

    while(ins >>ch) {
        // 공백을 건너뛰기
        if(isspace(ch)) continue;
```

```
        // 숫자를 처리
        if(ch >='0'&&ch <='9') { // 숫자인지 확인
            number =number *10+(ch -'0'); // 각 자리 수 계산
            readingNumber =true;          // 숫자를 읽고 있음을 표시
        }
        // 연산자나 괄호가 나오면
        else{
            // 숫자를 읽고 있으면, 숫자를 종료하고 벡터에 추가
            if(readingNumber) {
                postfix.push_back({0, number, ' '});
                number =0; // 숫자 초기화
                readingNumber =false;
            }

            // 여는 괄호 처리
            if(ch =='(') {
                item.push('(');
            }
            // 닫는 괄호 처리
            elseif(ch ==')') {
                while(!item.empty() &&item.top() !='(') {
                    postfix.push_back({1, 0, item.top()});// 스택에서 연산자 꺼내기
                    item.pop();
                }
                if(item.empty()) return{};// 괄호 불일치 오류
                item.pop();// 여는 괄호 제거
            }
            // 연산자 처리
            elseif(ch =='+'||ch =='-'||ch =='*'||ch =='/'||ch =='^') {
                while(!item.empty() &&precedence(item.top()) >=precedence(ch)) {
                    postfix.push_back({1, 0, item.top()});// 스택에서 연산자 꺼내기
                    item.pop();
                }
                item.push(ch);// 연산자 스택에 추가
            }
```

```
        // 잘못된 입력 처리
        else{
            return{};// 잘못된 입력
        }
    }
}

// 마지막으로 읽은 숫자가 있으면 추가
if(readingNumber) {
    postfix.push_back({0, number, ' '});
}

// 남아 있는 연산자를 모두 처리
while(!item.empty()) {
    if(item.top() =='(') return{};// 괄호 불일치 오류
    postfix.push_back({1, 0, item.top()});// 스택에서 연산자 꺼내기
    item.pop();
}

returnpostfix;
}

boolevaluate_stack(stack<int>&numbers, constcharsymbol) {
// 스택에 숫자 2개 이상이라고 가정
if(numbers.size() <2) returnfalse;

intvalue2 =numbers.top();
numbers.pop();
intvalue1 =numbers.top();
numbers.pop();

intresult;// 결과 저장할 변수
switch(symbol) {
    case'+': result =value1 +value2; break;
    case'-': result =value1 -value2; break;
    case'*': result =value1 *value2; break;
    case'/':
```

```
            if(value2 ==0) returnfalse;// 0으로 나누기 방지
            result =value1 /value2;
            break;
        case'^':
            result =1;// 제곱을 위한 초기값
            for(inti =0; i <value2; ++i) result *=value1;
            break;
        default: returnfalse;
    }

    numbers.push(result);
    returntrue;
}

intevaluate_postfix(constvector<CharIntConstruct>&postfix) {
    if(postfix.empty()) return-1;

    stack<int>numbers;
    for(constauto&token : postfix) {
        if(token.type==0) {
            numbers.push(token.number);
        } else{
            if(!evaluate_stack(numbers, token.symbol)) {
                return-1;// 계산 오류
            }
        }
    }

    if(numbers.size() !=1) {
        return-1;// 오류
    }

    returnnumbers.top();
}

intmain() {
    cout <<"수식을 입력하세요."<<endl;
```

```
    cout <<"'EOI'를 입력하면 결과를 출력합니다."<<endl;

    string input;
    vector<string>expressions; // 입력된 수식을 저장할 벡터

    while(true) {
        cout <<"> ";
        getline(cin, input);

        // EOI가 입력 되었다면 계산 시작
        if(input =="EOI") {
            break; // 루프를 종료하고 계산으로 넘어감
        } else{
            expressions.push_back(input); // 입력된 수식을 저장
        }
    }

    // 모든 수식을 처리
    for(conststring&expr : expressions) {// expressions 벡터 요소만큼 반복
        istringstream expr_stream(expr);
         vector<CharIntConstruct>postfix =make_postfix(expr_stream);// 후위 표기법으로 변환
        intresult =evaluate_postfix(postfix);// 수식 계산

        cout <<"결과: ";
        if(result ==-1) {// 계산 중 오류 출력
            cout <<"오류"<<endl;
        } else{
            cout <<result <<endl;
        }
    }

    return0;
}
```

2. 코드를 작성할 때 가지고 한 생각

코드 예시에 있는 것들은 다 이유가 있을 것이라고 생각하여 예시 코드를 먼저 분석하고 필요한 헤더 파일을 찾아보았다. 찾은 헤더 파일에는 어떤 명령어가 있는지 충분한 검색과 공부를 하는 것이 먼저라고 생각하여 공부를 한 후에 코드를 작성하였다.

3. 고찰

유용하게 사용 가능한 다양한 헤더 파일이 존재하는 것 같다. 하지만 헤더 파일을 많이 찾지 못하여 아쉬웠다.