HW2

1. 소스코드

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

usingnamespacestd;

structCharIntConstruct{
  ininttype;// 0 == 숫자, 1 == 연산자
  inintnumber;
  chcharsymbol;
};

// 우선순위 정하는 함수
intprecedence(charitem) {
  ifif(item =='+'||item =='-') return1;
  ifif(item =='*'||item =='/') return2;
  ifif(item =='^') return3;
  rereturn0;
}

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

  sisize_ti =0;// 문자열 인덱스 초기화

  whwhile(i <input.size()) {
    // 공백을 건너뛰기
    whilwhile(i <input.size() &&input[i] ==' ') {
      i++; ضی++;
```

```
    }

    if(iif(i >=input.size()) break;

    // 숫자를 처리
    if(iif(isdigit(input[i])) {
      intnumintnumber =0;
      // 숫자를 계속 읽음 녑p蟳
      while(while(i <input.size() &&isdigit(input[i])) {
        number =number =number *10+(input[i] -'0');// 각 자릿수 계산
        i++; 이楠老++;
      }
      postfipostfix.push_back({0, number, ' '});// 숫자 추가
    }
    // 여는 괄호 처리
    elseelseif(input[i] =='(') {
      item.pitem.push('(');
      i++; 韝++;
    }
    // 닫는 괄호 처리
    elseelseif(input[i] ==')') {
      while(while(!item.empty() &&item.top() !='(') {
        postfix.postfix.push_back({1, 0, item.top()});// 스택에서 연산자 꺼내기
        item.popitem.pop();
      }
      if(iteif(item.empty()) return{};// 괄호 불일치 오류
      item.pitem.pop();// 여는 괄호 제거
      i++;  ++;
    }
    // 연산자 처리
    elseelseif(input[i] =='+'||input[i] =='-'||
        input[i] input[i] =='*'||input[i] =='/'||
        input[i] input[i] =='^') {
      charopcharop =input[i];
```

```
      while(while(!item.empty() &&precedence(item.top()) >=precedence(op)) {
        postfix.postfix.push_back({1, 0, item.top()});// 스택에서 연산자 꺼내기
        item.popitem.pop();
      } 瑙
      item.pitem.push(op);// 연산자 스택에 추가
      i++; ¸++;
    }
    // 잘못된 입력 처리 걃 ᄀ
    elseelse{
      returnreturn{};// 잘못된 입력
    }
  }

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

  rereturnpostfix;
}

boolevaluate_stack(stack<int>&numbers, constcharsymbol) {
  // 스택에 숫자 2개 이상이라고 가정
  ifif(numbers.size() <2) return0;

  inintvalue2 =numbers.top();
  nunumbers.pop();
  inintvalue1 =numbers.top();
  nunumbers.pop();

  inintresult;// 결과 저장할 변수
  swswitch(symbol) {
    casecase'+': result =value1 +value2; break;
```

```
    casecase'-': result =value1 -value2; break;
    casecase'*': result =value1 *value2; break;
    casecase'/':
      if(valif(value2 ==0) return0;// 0으로 나누기 방지
      result =value=value1 /value2;
      break;break;
    casecase'^':
      result =1;// =1;// 제곱을 위한 초기값
      for(infor(inti =0; i <value2; ++i) result *=value1;
      break;break;
    defadefault: return0;
  }

  nunumbers.push(result);
  rereturn1;
}

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

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

  ifif(numbers.size() !=1) {
    retureturn-1;// 오류: 결과가 하나가 아님
  }
```

```
  rereturnnumbers.top();
}

intmain() {
  cout <<<<"수식을 입력하세요."<<endl;
  cout <<<<"'EOI'를 입력하면 결과를 출력합니다."<<endl;

  vector<s<string>expressions;// 수식 저장할 벡터
  string input;//// 입력을 저장할 변수

  whwhile(true) {
    cout <<"><<"> ";
    getlgetline(cin, input);

    // EOI가 입력 되었다면 계산 시작
    if(iif(input =="EOI") {
      for(cofor(conststring&expr : expressions) {
        vector<CharInt<CharIntConstruct>postfix =make_postfix(expr);// 후위 표기법으로 변환
        intresulintresult =evaluate_postfix(postfix);// 수식 계

        cout <<"결과: "<<"결과: ";
        if(resulif(result ==-1) {// 계산 중 오류 출력
          cout <<"오류"<<en<<"오류"<<endl;
        } else{ Ä❥else{
          cout <<result <<<result <<endl;
        }
      }
      break;break;
    } 恰耀京

    exprexpressions.push_back(input);
  }

  rereturn0;
}
```

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.