자료구조
Programming Report #1

제출일자 : 2014년 11월 5일

담당 교수: 이상호

학과: 컴퓨터공학과

학번: 1315071

이름: 한혜정

1. 문제 기술

다음의 조건 하에서 중위 표기법(infix notation)으로 된 여러 개의 수식들을 파일로 부터 입력 받아서 각각의 수식을 후위 표기법(postfix notation)으로 변환한 후, 수식의 값을 계산하는 프로그램을 작성하라.

• 조건

(1) 중위 표기법 수식에는 괄호가 들어갈 수 있다.
(2) 중위 표기법 및 후위 표기법 수식은 세미콜론(;)으로 끝난다.
(3) 피연산자(operand)는 하나의 숫자 문자('0', '1', ... , '9')로 된 상수이다.
(4) 연산자(operator)는 +, -, *, /, %, ^ 로만 구성된다. (단, /는 정수형 나눗셈, ^는 지수 연산을 의미한다)
(5) 연산자 스택은 반드시 배열을 사용하여 구현하고, 피연산자 스택은 단순 연결 리스트를 사용하여 구현한다.
(6) 모든 수식은 문법적으로 맞는다고 가정한다.

2. 입출력의 예

• 입력 파일(infix.txt)의 예 :

```
3                       /* 입력으로 들어올 수식의 개수
2+4*2-1;                /* 첫 번째 수식
9+3^2^(3-1)*2;          /* 두 번째 수식
2*((7-2)/3+4)^2%3;      /* 세 번째 수식
```

• 출력의 예 : 각 수식에 대해서 중위 표기법 수식(입력), 이를 변환한 후위 표기법 수식, 후위 표기법을 이용한 계산 값을 차례로 화면에 출력한다. (단, 출력시 각 문자 사이에는 빈칸 하나씩을 둠. 세 번째 수식의 예)

infix notation = 2 * ((7 – 2) / 3 + 4) ^ 2 % 3 ;

postfix notation = 2 7 2 – 3 / 4 + 2 ^ * 3 % ;

value = 2

3. 문제풀이 방법(알고리즘)

1. 피연산자를 만나면 바로 출력한다.
2. 연산자를 만나면 해당 연산자보다 우선순위가 높거나 같은 것들을 output으로 출력하고 스택에 담는다.
3. 여는 괄호 '('는 무조건 스택에 담는다.
4. 닫는 괄호 ')'을 만나면 여는 괄호 '('을 만날 때까지 스택에서 출력한다.

4. 소스 프로그램

[소스]

```
//
//  InfixToPostfixConvertAndCalculate.cpp
//  Data Structure Programming Report#1
//  Convert infix expression to Postfix expression and calculate the result
//
//  Created by 한혜정 on 06/01/2019.
//  Copyright © 2019 한혜정. All rights reserved.
//

#include <iostream>
#include <fstream>
#include <stack>
#include <string>
#include <cmath>

using namespace std;

//check if the character is operator
bool isOperator(char c);

//put weight to operator
int GetOperatorWeight(char op);

//compare weight between two operators
bool hasHigherPrecedence(char op1, char op2);

//convert infix expression to postfix expression
string infixPostfixTranslation(string exp);
```

```
//print results
void printResult(string exp, int num);

//insert spaces to expression
string insertSpace(string exp);

//calculate the expression
double calculatePostfixExp(string exp);

int main(int argc, const char * argv[]){
    int count = 1;
    string line;
    ifstream fin;
    fin.open("infix.txt", ios::in);
    if(!fin) cout << "File Open Error" << endl;

    getline(fin, line);
    while (getline(fin, line)){
        printResult(line, count);
        count++;
    }

    fin.close();

    return 0;
}

void printResult(string exp, int num){
    exp = exp.substr(0,exp.length()-1);
    string postfixExp = infixPostfixTranslation(exp);
    cout << "#" << num << endl;
    cout << "infix notation = " << insertSpace(exp) << endl;
    cout << "postfix notation = " << insertSpace(postfixExp) << endl;
    cout << "value = " << calculatePostfixExp(postfixExp) << endl;
    cout << endl;
}

string infixPostfixTranslation(string exp){
    stack<char> Stack;
    string postfix = "";
    for(int i=0;i<exp.length();i++){
        char token = exp[i];
```

```
        if(token==';') break;
        else if (token>='0'&&token<='9') postfix+=token;
        else if (token=='(') Stack.push(token);
        else if (token==')'){
            while (!Stack.empty()&&Stack.top()!='(') {
                postfix += Stack.top();
                Stack.pop();
            }
            Stack.pop();
        }else if (isOperator(token)){
            while (!Stack.empty() && Stack.top()!='(' &&
hasHigherPrecedence(Stack.top(), token)) {
                postfix += Stack.top();
                Stack.pop();
            }
            Stack.push(token);
        }
    }
    while (!Stack.empty()) {
        postfix += Stack.top();
        Stack.pop();
    }
    postfix += ';';

    return postfix;
}

bool isOperator(char c){
    if (c == '+'||c == '-'||c == '*'||c == '/'||c == '%'||c == '^')
        return true;
    else
        return false;
}

bool hasHigherPrecedence(char op1, char op2){
    int op1Weight = GetOperatorWeight(op1);
    int op2Weight = GetOperatorWeight(op2);

    return op1Weight>=op2Weight?true:false;
}

int GetOperatorWeight(char op)
{
```

```
    int weight = -1;
    switch(op)
    {
        case '+':
        case '-':
            weight = 1;
            break;
        case '*':
        case '/':
        case '%':
            weight = 2;
            break;
        case '^':
            weight = 3;
            break;
    }
    return weight;
}

string insertSpace(string exp){
    string output = "";
    for (int i=0; i<exp.length(); i++) {
        output += exp.substr(i,1) + " ";
    }
    return output;
}

double calculatePostfixExp(string exp){
    stack<double> Stack;
    double num1, num2;
    double num3 = 0.0;
    double result;
    for(int i=0;i<exp.length();i++) {
        char token = exp[i];
        if (token>='0'&&token<='9') Stack.push((int)token - 48);
        else if (isOperator(token)){
            num2 = Stack.top();
            Stack.pop();
            num1 = Stack.top();
            Stack.pop();

            switch (token) {
                case '+':
```

```
                num3 = num1+num2;
                break;

            case '-':
                num3 = num1-num2;
                break;

            case '*':
                num3 = num1*num2;
                break;

            case '/':
                num3 = num1/num2;
                break;

            case '%':
                num3 = (int)num1%(int)num2;
                break;

            case '^':
                num3 = pow(num1, num2);
                break;

            default:
                break;
        }
        Stack.push(num3);
    }
  }
  result = Stack.top();
  Stack.pop();

  return result;
}
```

[결과]

```
#1
infix notation = 2 * 3 / 3 + 4 - 5 ;
postfix notation = 2 3 * 3 / 4 + 5 - ;
value = 1
```

#2
infix notation = 1 + 2 * 3 - 4 / 2 + 5 ;
postfix notation = 1 2 3 * + 4 2 / - 5 + ;
value = 10

#3
infix notation = (2 + 4) * (2 - 4) ;
postfix notation = 2 4 + 2 4 - * ;
value = -12

#4
infix notation = 8 + (3 ^ 2) ^ 2 ^ (3 - 1) * 4 ;
postfix notation = 8 3 2 ^ 2 ^ 3 1 - ^ 4 * + ;
value = 26252

#5
infix notation = (2 * ((7 - 2) / 3 + 4)) ^ (8 % 3) ;
postfix notation = 2 7 2 - 3 / 4 + * 8 3 % ^ ;
value = 128.444

#6
infix notation = (((1 + 2) * 3) - 2) % 5 ;
postfix notation = 1 2 + 3 * 2 - 5 % ;
value = 2

#7
infix notation = 9 + 3 ^ 2 ^ (3 - 1) * 2 ;
postfix notation = 9 3 2 ^ 3 1 - ^ 2 * + ;
value = 171

#8
infix notation = 2 * ((7 - 2) / 3 + 4) ^ 2 % 3 ;
postfix notation = 2 7 2 - 3 / 4 + 2 ^ * 3 % ;
value = 1

#9
infix notation = ((1 - 2)) ^ 9 ;
postfix notation = 1 2 - 9 ^ ;
value = -1

#10
infix notation = 3 ;

postfix notation = 3 ;
value = 3

Program ended with exit code: 0

5. 결과 분석 및 토의

5번 값이 1.2222가 나와야 하는데 1이 나와서 아쉬웠다.
처음에 C로 프로그래밍하였으나 포인터가 헷갈려서 C++로 다시 작성하였다. 포인터를 조금 더 공부해야겠다.
Xcode를 사용하였는데 파일 입출력을 할 때 Product-Scheme-Edit Scheme-Run-Option에서 Use Custom Working Directory 체크박스를 활성화시켜야 한다는 것을 알았다.
후위표기법이 잘못 나와서 한참 헤맸는데 operator에 weight를 줄때 switch에서 break를 빠트려서 그랬다는 것을 깨달았다. 이런 간단한 부분에서 실수하지 않도록 노력해야겠다고 생각했다.
참고자료

https://www.mathblog.dk/tools/infix-postfix-converter/
https://gist.github.com/mycodeschool/7867739