Name : 하준혁
Student ID : 20181705
Program ID : infix-postfix.cpp
Description: Infix를 Postfix 식으로 변환하고 이후 Postfix 식을 계산하기.
Algorithm: Stack

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>

using namespace std;

// Stack 관련 기본 변수 및 함수
const int stackSize = 10;
int stack[stackSize];
int top;
void create_stack();
void push(int);
int pop();
int isFull();
int isEmpty();
void displayStack();

int lastElement();
 -> Postfix 로 변환하기위해 Operator의 우선순위를 정하려면 스택 가장 위의 operator와 우선순위를 확인하기 위해 스택의 마지막 요소를 확인할 수 있는 함수

string infix_to_postfix(string);
-> infix 식을 postfix 식으로 변형시키는 함수

int postfix_evaluation(string);
-> postfix 식을 계산하는 함수

int exp_operator(int);
-> operator 의 우선순위를 알려주는 함수

int main()
{
	int num;
	char input[10];

	string filePath = "infix-postfix.txt";
	ifstream openFile(filePath.data());
	if( openFile.is_open() ){
		string line;
		while(getline(openFile, line)){
			string postfix;
			cout << "Echo Data (infix form) : "<< line << endl;
			-> 맨 윗줄은 기본 식 출력
			postfix = infix_to_postfix(line);
			cout << "Conversion (postfix form):" << postfix << endl;
			-> 두번째줄은 postfix로 변환한 식 출력
			cout << "Result : " << postfix_evaluation(postfix) << endl;
			-> 마지막줄은 postfix를 계산한 결과를 출력
		}
		openFile.close();
	}
```

```
}

// 스택 생성 함수
void create_stack() { top = -1; }
// 스택이 가득 찼는지 확인하는 함수
int isFull() {
   if (top == stackSize - 1) return 1;
   else return 0;
}
// 스택이 비어있는지 확인하는 함수
int isEmpty() {
   if (top == -1) return 1;
   else return 0;
}
// 스택에 push 하는 함수
void push(int item) {
   ++top;
   stack[top] = item;
}
// 스택에서 pop하는 함수
int pop() {
   return (stack[top--]);
}
// top의 값을 바꾸지 않고 마지막 요소를 리턴하는 함수
int lastElement() {
          return (stack[top-1]);
}
// 스택 정보를 보여주는 함수
void displayStack()
{
   int sp;
   if (isEmpty()) cout << "Stack is empty!" << endl;
   else {
      sp = top;
      while (sp != -1) {
         cout << stack[sp] << " ";
         sp--;
      }
      cout << endl;
   }
}

// operator의 우선순위를 출력하는 함수. 우선순위가 높을수록 큰 정수를 반환한다.
int exp_operator(int x)
{
       char op[]={'\0','(',')','+','-','*','/','%'};
       int i;

       for(i = 1; i < 8; i++)
            if(op[i]==(char)x)        return i;
       return 0;
}

// infix 식을 postfix로 변환하는 함수
string infix_to_postfix(string line) {
          string postfix = "";
          create_stack();
          int len = line.length();
          for(int i=0; i < len; i++) {
```

```
		if(line[i] == '(') {
			// ( 를 만나면 push 합니다.
			push(line[i]);
		}
		else if(line[i] == ')') {
			// ) 를 만나면 ( 를 만날때까지 push 하고 ( 는 버려줍니다.
			char temp = (char)pop();
			while(temp != '(' && !isEmpty()) {
				postfix += temp;
				temp = pop();
			}

		}
		else if(line[i] == '+' || line[i] == '-' || line[i] == '*' || line[i] == '/' || line[i] == '%') {

			if(isEmpty()) {
				// 만약 operator를 받았을때 stack 이 비어있다면 무조건 push 합니다.
				push(line[i]);
			}
			else if(exp_operator(line[i]) > exp_operator(lastElement())) {
				// stack의 마지막 operator의 우선순위보다 지금 operator의 우선순위가
				높다면 stack의 마지막 operator를 postfix 변수에 추가해줍니다.
				char item = (char)pop();
				postfix += item;
				push(line[i]);
			}
			else {
				// 우선순위가 낮다면 그냥 push 해줍니다.
				push(line[i]);
			}
		}
		else {  // 숫자는 바로 postfix 변수에 추가해줍니다.
			postfix += line[i];
		}
	}
	while(!isEmpty()) {
		// 마지막으로 item이 빌때까지 pop를 하면서 postfix 변수에 추가해줍니다.
		char item = (char)pop();
		postfix += item;
	}
	return postfix;
}

// Postfix 식을 계산하는 함수
int postfix_evaluation(string postfix) {
	int len = postfix.length();
	create_stack();
	for(int i = 0; i < len; i++) {
		if(postfix[i] >= '0' && postfix[i] <= '9') {
			// 만약 숫자라면 int 형으로 변환해서 push 합니다.
			int item = postfix[i] - '0';
			push(item);
		}
		else {
			// 숫자가 아니라면 operator에 따라 계산을 진행해줍니다.
			int a = pop();
			int b = pop();
			switch(postfix[i]) {
				case '+': push(b+a); break;
```

```
                                case '-': push(b-a); break;
                                case '*': push(b*a); break;
                                case '/': push(b/a); break;
                        }
                }
        }
        // 반복하면 마지막 요소가 계산 값이 됩니다.
        return pop();
}
```