Lab5. Infix-to-Postfix conversion & Evaluation

Problem: 스택을 이용하여, 수식 연산을 수행하는 문제. Infix form의 수식을 입력받고, postfix form으로 conversion한후 4칙연산으로 계산한다.

- 입력데이터 (lab5.txt):

3+ 4*6
8/2+ 6*2
4+ 5+ 6-3*9
9-3+ (8-6/2)

(Step):

1. 데이터 화일에서 입력 받은 수식(infix form)을 화면에 출력한다. (Echo data)
2. infix-to-postfix algorithm에 의해 conversion한 후 postfix form을 출력한다. (postfix)
3. postfix evaluation algorithm을 이용하여 최종값을 출력한다. (result)

- **알고리즘1 (Infix to postfix conversion)**

```
function postconv (char buffer[]) {

   while ((token = buffer[i]) != '\0') {
         if ((token == operand)) {      // if (token >= '0' && token <= '9') //
               print token;    Postfix[j++] = token;      // Eval위해 저장
         }
         else if (token == '(')    push(token);
         else if (token == ')') {
               while (stack[top] != '(') {
                     popdata = pop(); print popdata;    Postfix [j++] = popdata;
               }
               popdata = pop();    // '(' 버림
         }
         else { // if operator
              while (Prec(token) <= Prec(stack[top]))      {//if operator
                 popdata = pop();      print popdata;    Postfix [j++] = popdata;
               }
               push(token);
          }
         i++;
     }} // end of while
    while (top != 0) {    //stack에 남은 것 모두 처리
         popdata = pop();            Postfix [j++] = popdata;
     }
    Postfix [j++] = '\0'        // end of line mark
} // end of function postconv
```

- 연산자 우선순위

과제용 연산자	순위
)	3
*, /	2
+, -	1
(	0

```
int Prec(char op)   { // 연산자 우선순위 반환
        switch (op) {
            case '(': case ')': return 0;
            case '+': case '-': return 1;
            case '*': case '/': return 2;
            }
        return -1;
}
```

- **알고리즘2 (evaluation of Postfix form)**

```
Function Posteval (char postbuffer[]) {
    while ((token= postbuffer[i]) != '\0') { // repeat until end of string
          if (token == operand)    push(token-'0');      //operand -> 0~9사이
          else {
                        pn2 = pop();      pn1 = pop();
             switch (token) {
                        case '+':    result = pn1 + pn2; break;
                        case '-':    result = pn1 - pn2; break;
                        case '*':    result = pn1 * pn2; break;
                        case '/':    result = pn1 / pn2; break;
                                default: print("ERROR in expression"); break;
             }//end of switch
           }
          i++;        push(result);
        }//end of while

      finaldata = pop();
      print "The final result is ", finaldata;
```

(조건)

* infix form은 9미만의 십진수를 infix형태로 표현한 것을 이용한다. (ex. 3 + 2)
* 사칙연산만을 사용한다. (ex. + , -, /, *)
* Operator는 "(", "-", "+", "/", "*", ")" 로 한정한다.

● 실행 화면


Infix expression : 3+4*6 The Postfix conversion: 346*+
The final result : 27
Infix expression : 8/2+6*2 The Postfix conversion: 82/62*+
The final result : 16
Infix expression : 4+5+6-3*9 The Postfix conversion: 45+6+39*-
The final result : -12
Infix expression : 9-3+(8-6/2) The Postfix conversion: 93-862/-+
The final result : 11