



## Assignment 1: Expression Evaluation using Stack

Sarim Baig

Due: Tuesday, February 5, 2013 before class

An arithmetic expression is composed of operands (numbers), arithmetic operators (+, /, -, \*), and the left and right parenthesis '(' and ')'. While evaluating an arithmetic expression the operator precedence follows the well-known 'BODMAS' rule – i.e. Brackets, Division, Multiplication, Addition and then Subtraction.

Normally, when we write an expression (such as the following) the operator is always between the two operands (considering that we have only binary operators, which take two operands). This way of writing an expression is called *infix* notation.

Following are examples of arithmetic expressions in infix notation:

$$(2-1) + (6/2) * 3$$

$$3 + (2-5) * 4$$

$$3 + 4/2 - (4 + (12/3 + 2))$$

In *infix* notation, expressions are easy to read by humans but difficult to process by computers. In order to make them easy to evaluate by computers they are converted into *postfix* notation, where each operator follows its two operands, which in turn can be expressions in *postfix* or just numbers. For example, the first expression given above can be written in postfix notation as follows:

$$2\ 1 -\ 6\ 2 / \ 3\ * \ +$$

To see what this means for evaluation the following colored depiction might help. In each case the red part is the part to be evaluated – and  $a_1$ ,  $a_2$ ,  $a_3$ , and  $a_4$  are the answers obtained at each step:

$$\mathbf{2\ 1 -\ 6\ 2 / \ 3\ * \ +}$$

$$\mathbf{a_1\ 6\ 2 / \ 3\ * \ +}$$

$$\mathbf{a_1\ a_2\ 3\ * \ +}$$

$$\mathbf{a_1\ a_3 \ +}$$

$$\mathbf{a_4}$$

Here  $a_1=1$ ,  $a_2=3$ ,  $a_3=9$ , and  $a_4=10$ , which is the final answer. (Obviously you will not store  $a_1$ ,  $a_2$  etcetera, but the actual numbers).

Similarly, the other two expressions in *postfix* notation are:

$2\ 5 - 4 * 3 +$

$3\ 4\ 2 / + 12\ 3 / 2 + 4 + -$

Your job in this assignment is to write a C++ program which will receive as input an arithmetic expression I, which is a string in the *infix* notation, and do the following:

- i) Implement the template based stack we studied in class, and use it to:
- ii) Convert I into the equivalent string, P, in *postfix* notation.
- iii) Evaluate the postfix notation P, and report the answer.

For this purpose you will write two functions:

string convertToPostfix(string I);

double evaluatePostfix(string P);

**Your program must read ten infix expressions from a file called ‘exp10.txt’ – line by line (there will be one expression per line) and output the ten answers on the screen. For evaluation we will replace your file with our own.**

### ALGORITHMS

For your convenience we’ve provided below the two algorithms you will need to implement, to accomplish these tasks. Read these carefully and consult me if there are any problems:

#### **(1) Conversion to Postfix**

Define a stack

Go through each character in the input string I

If it is a number, append it to output string P

If it is left parenthesis, push it to the stack

If it is operator  $*+/-$  then

    If the stack is empty push it to the stack

    If the stack is not empty then start a loop:

        If the top of the stack has higher precedence

        Then pop and append to output string P

        Else break

    Push to the stack

If it is right parenthesis, then

    While stack not empty and top not equal to left parenthesis

        Pop from stack and append to output string P

    Finally, pop out the left brace.

## (2) Evaluation of Postfix expression

Start with an empty stack.  
Scan input string P from left to right  
    If an operand is found  
        push it onto the stack  
    ELSE If an operator is found  
        Pop the stack and call the value A  
        Pop the stack and call the value B  
        Evaluate **B op A** using the operator just found.  
        Push the resulting value onto the stack  
End-While  
Pop the stack (this is the final value)

**Note:** There will be absolutely NO EXTENSION in the deadline. It is your responsibility to discuss your problems (if any) with me, and get the work done in time.

**Good Luck**