Data Structures & Algorithms

DSA – Infix to Postfix Conversion

Instructor: Mahfuz Hasan Reza

Learn With Mahfuz

Let's Code Together

Infix **Prefix** **Postfix**

$$5 + 2 \qquad + \; 5 \; 2 \qquad 5 \; 2 \; +$$

operator operand

Operator Precedence and Associativity

Operator	Precedence	Associativity
()	HIGH ↑	
^		Right to Left
* / %		Left to Right
+ -	LOW	Left to Right

Learn With Mahfuz
Let's Code Together

Rules

- Operands are written **directly** to the postfix expression.
- Operator:
    1. Empty Stack -> Push
    2. High Precedence -> Push
    3. Low Precedence -> Pop, Check
    4. Same Precedence:

       Left to Right Associativity -> Pop, Check

       Right to Left Associativity -> Push
    5. (-> Push it onto the stack
    6.) -> Pop from the stack until find (

Infix to Postfix using Stack

a + (b - c * d * e ^ f) + g * h / i ^ j

Current char	Stack	postfix Expression
a		a
+	+	a
(	+(	a
b	+(	ab
-	+(-	ab
c	+(-	abc
*	+(-*	abc
d	+(-*	abcd
*	+(-*	abcd*
e	+(-*	abcd*e
^	+(-*^	abcd*e

Pre	Ass
( )	
^	R→L
* / %	} L→R
+ -	} L→R

operator

precc.↑ → push

precc.↓ → pop, check

prec. same

Ass L→R : pop,push

Ass R→L : push

Current char	Stack	Postfix Exp.
f	+(-*^	abcd*ef
)	+	abcd*ef^*-
+	+	abcd*ef^*-+
g	+	abcd*ef^*-+g
*	+*	abcd*ef^*-+g
h	+*	abcd*ef^*-+gh
/	+/	abcd*ef^*-+gh*
i	+/	abcd*ef^*-+gh*i
^	+/^	abcd*ef^*-+gh*i
j	+/^	abcd*ef^*-+gh*ij
		abcd*ef^*-+gh*ij^/+

Infix to Postfix using Stack

a – b / (c – d) * e

Current Char	Stack	Postfix Exp
a		a
–	–	a
b	–	ab
/	/ –	ab
(	(/ –	ab
c	〃	abc
–	– (/ –	abc
d	〃	abcd
)	/ –	abcd–
*	* –	abcd–/
e	〃	abcd–/e
		abcd–/e*–

Infix to Postfix using Stack

a – (b * (c - d * e))

Current Char	Stack	Postfix Exp.
a		a
–	–	a
(	–(	a
b	–(	ab
*	–(*	ab
(	–(*(	ab
c	–(*(	abc
–	–(*(–	abc
d	–(*(–	abcd
*	–(*(–*	abcd
e	–(*(–*	abcde
)	–(*	abcde*–
)	–	abcde*–*
		abcde*–*–

THANK YOU!

Learn With Mahfuz

Let's Code Together