

Arithmetic Expression Conversion Using stack

Conversion of Infix Expression into Postfix Expression

Algorithm: Infix-to-Postfix (Q, P)

Here Q is an arithmetic expression in infix notation and this algorithm generates the postfix expression P using stack.

1. Scan the infix expression Q from left to right.
2. Initialize an empty stack.
3. Repeat step 4 to 5 until all characters in Q are scanned.
4. If the scanned character is an operand, add it to P.
5. If the scanned character is an operator Φ , then
 - (a) If stack is empty, push Φ to the stack.
 - (b) Otherwise repeatedly pop from stack and add to P each operator which has the same or higher precedence than Φ .
 - (c) Push Φ to the stack.

6. If scanned character is a left parenthesis “(“, then push it to stack.
7. If scanned character is a right parenthesis “)”, then
 - (a) Repeatedly pop from stack and add to P each operator until “(“ is encountered.
 - (b) Remove “(“ from stack.
8. If all the characters are scanned and stack is not empty, then
 - (a) Repeatedly pop the stack and add to P each operator until the stack is empty.
9. Exit.

Example: Q: $5 * (6 + 2) - 12 / 4$ and P: ?

Infix Expression Q	Stack	Postfix Expression P
5		5
*	*	5
(* (5
6	* (5, 6
+	* (+	5, 6
2	* (+	5, 6, 2
)	*	5, 6, 2, +
-	-	5, 6, 2, +, *
12	-	5, 6, 2, +, *, 12
/	- /	5, 6, 2, +, *, 12
4	- /	5, 6, 2, +, *, 12, 4
	-	5, 6, 2, +, *, 12, 4, /
		5, 6, 2, +, *, 12, 4, /, -

Postfix Expression P : 5, 6, 2, +, *, 12, 4, /, -

Example: Q: $A * ((B + C) - D) / E$ and P: ?

Infix Expression Q	Stack	Postfix Expression P
A		A
*	*	A
(* (A
(* ((A
B	* ((A B
+	* ((+	A B
C	* ((+	A B C
)	* (A B C +
-	* (-	A B C +
D	* (-	A B C + D
)	*	A B C + D -
/	/	A B C + D - *
E	/	A B C + D - * E
		A B C + D - * E /

Postfix Expression P : A B C + D - * E /

Postfix Expression Evaluation

Algorithm: Postfix-Evaluation (P, Value)

Here P is an arithmetic expression in postfix notation and this algorithm finds the value of this expression using stack.

1. Scan the postfix expression P from left to right.
2. Initialize an empty stack.
3. Repeat step 4 to 5 until all characters in P are scanned.
4. If the scanned character is an operand, push it to the stack.
5. If the scanned character is an operator Φ , then
 - (a) Remove two top elements of stack where A is the top element and B is the next-to-top element.
 - (b) Evaluate $T = B \Phi A$ and push T to the stack.
6. Pop the stack and assign the top element of the stack to Value.
7. Exit

Example: Postfix : 5, 6, 2, +, *, 12, 4, /, - and Value: ?

Postfix Expression Q	Stack
5	5
6	5, 6
2	5, 6, 2
+	5, 8
*	40
12	40, 12
4	40, 12, 4
/	40, 3
-	37

Value: 37

Example: Postfix : **456*+** and Value: ?

Postfix Expression Q	Stack
4	4
5	4, 5
6	4, 5, 6
*	4, 30
+	34

Value: 34

END!!!!