



Evaluation of Expression



Evaluation of Expression

- An expression is a collection of operators and operands that represents a specific value.

Expression Types

- Based on the operator position, expressions are divided into 3 types. They are as follows...

- 1. Infix Expression**
- 2. Postfix Expression**
- 3. Prefix Expression**



Infix Expression

- In infix expression, operator is used in between operands

Ex: $a+b$

Postfix Expression

- In postfix expression, operator is used after operands.

Ex: $ab+$

Prefix Expression

- In prefix expression, operator is used before operands.

Ex : $+ab$



Expression Conversion



- Any expression can be represented using three types of expressions (Infix, Postfix and Prefix)
- To convert any Infix expression into Postfix or Prefix expression we can use the following procedure :
 - ✓ Find all the operators in the given Infix Expression.
 - ✓ Find the order of operators evaluated according to their Operator precedence.
 - ✓ Convert each operator into required type of expression (Postfix or Prefix) in the same order.



Example: (Infix to Postfix)



- Consider the following Expression

$$D = A + B * C$$

- **Step 1:** The Operators in the given Infix Expression
: = , + , *
- **Step 2:** The Order of Operators according to their preference : * , + , =
- **Step 3:** Now, convert the first operator * ----- $D = A + B C *$
- **Step 4:** Convert the next operator + ----- $D = A B C * +$
- **Step 5:** Convert the next operator = ----- $D A B C * + =$
- Finally after conversion we get $D A B C * + =$



Steps to convert Infix to Postfix using Stack



- Read all the symbols one by one from left to right in the given Infix Expression.
- If the reading symbol is operand, then directly print it to the result (Output).
- If the reading symbol is left parenthesis '(', then Push it on to the Stack.
- If the reading symbol is right parenthesis ')', then Pop all the contents of stack until respective left parenthesis is popped and print each popped symbol to the result.
- If the reading symbol is operator (+ , - , * , / etc.), then Push it on to the Stack.
- However, first pop the operators which are already on the stack that have higher or equal precedence than current operator and print them to the result