One of the applications of Stack is in the conversion of arithmetic expressions in high-level programming languages into machine readable form. As our computer system can only understand and work on a binary language, it assumes that an arithmetic operation can take place in two operands only e.g., **A+B, C\*D,D/A** etc. But in our usual form an arithmetic expression may consist of more than one operator and two operands e.g. **(A+B)\*C(D/(J+D))**.

These complex arithmetic operations can be converted into polish notation using stacks which then can be executed in two operands and an operator form.

Infix Expression

It follows the scheme of **<operand><operator><operand>** i.e. an <operator> is preceded and succeeded by an <operand>. Such an expression is termed infix expression. E.g., **A+B**

Postfix Expression

It follows the scheme of **<operand><operand><operator>** i.e. an <operator> is succeeded by both the <operand>. E.g., **AB+**

Algorithm to convert Infix To Postfix

Let, X is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression Y.

1. Push “(“onto Stack, and add “)” to the end of X.
2. Scan X from left to right and repeat Step 3 to 6 for each element of X until the Stack is empty.
3. If an operand is encountered, add it to Y.
4. If a left parenthesis is encountered, push it onto Stack.
5. If an operator is encountered ,then:
   1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) which has the same precedence as or higher precedence than operator.
   2. Add operator to Stack.  
      [End of If]
6. If a right parenthesis is encountered ,then:
   1. Repeatedly pop from Stack and add to Y each operator (on the top of Stack) until a left parenthesis is encountered.
   2. Remove the left Parenthesis.  
      [End of If]  
      [End of If]
7. END.