



Dash



All



Articles



Videos



Problems



Quiz

<< Prev

Next >>

Infix To Prefix

To convert an infix to postfix expression refer to this article [Infix to Postfix](#) article. We use the same to convert Infix to Prefix.

- Step 1: Reverse the infix expression i.e $A+B*C$ will become $C*B+A$. Note while reversing each '(' will become ')' and each ')' becomes '('.
- Step 2: Obtain the "nearly" postfix expression of the modified expression i.e $CB*A+$.
- Step 3: Reverse the postfix expression. Hence in our example prefix is $+A*BC$.

Note that for Step 2, we don't use the postfix algorithm as it is. There is a minor change in the algorithm. We have to pop all the operators from the stack which are **greater than or equal to** in precedence than that of the scanned operator. But here, we have to pop all the operators from the stack which are **greater** in precedence than that of the scanned operator. Only in the case of '^' operator, we pop operators from the stack which are **greater than or equal to** in precedence.

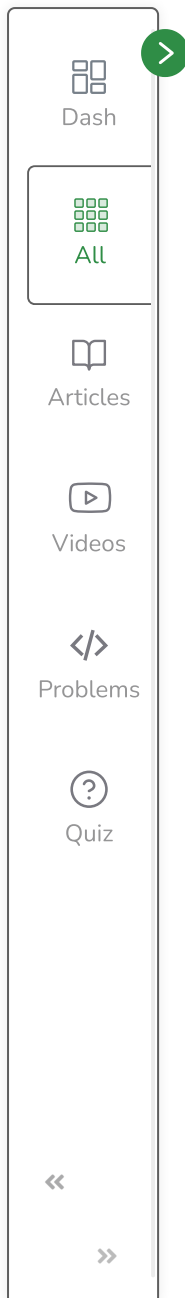
Below is the C++ and Java implementation of the algorithm.

C++

Java

```
// JAVA program to convert infix to prefix
import java.util.*;

class GFG
{
```



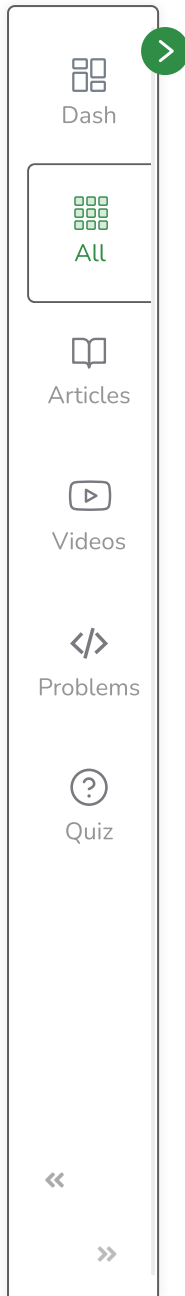
```
static boolean isalpha(char c)
{
    if (c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z')
    {
        return true;
    }
    return false;
}
```

```
static boolean isdigit(char c)
{
    if (c >= '0' && c <= '9')
    {
        return true;
    }
    return false;
}
```

```
static boolean isOperator(char c)
{
    return (!isalpha(c) && !isdigit(c));
}
```

```
static int getPriority(char C)
{
    if (C == '-' || C == '+')
        return 1;
    else if (C == '*' || C == '/')
        return 2;
```





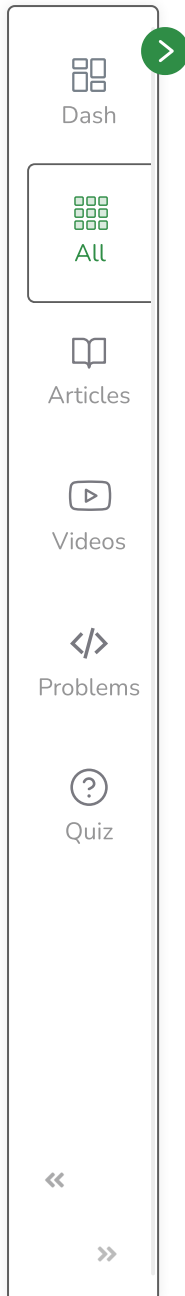
```
else if (C == '^')
    return 3;

return 0;
}

// Reverse the letters of the word
static String reverse(char str[], int start, int end)
{
    // Temporary variable to store character
    char temp;
    while (start < end)
    {
        // Swapping the first and last character
        temp = str[start];
        str[start] = str[end];
        str[end] = temp;
        start++;
        end--;
    }
    return String.valueOf(str);
}

static String infixToPostfix(char[] infix1)
{
    System.out.println(infix1);
    String infix = '(' + String.valueOf(infix1) + ')';
```





```
int l = infix.length();
Stack<Character> char_stack = new Stack<>();
String output="";

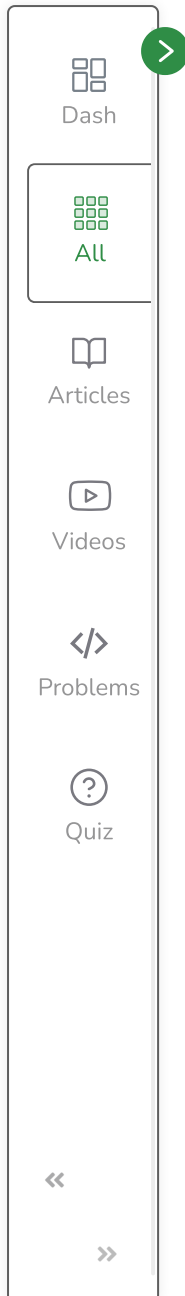
for (int i = 0; i < l; i++)
{

    // If the scanned character is an
    // operand, add it to output.
    if (isalpha(infix.charAt(i)) || isdigit(infix.charAt(i)))
        output += infix.charAt(i);

    // If the scanned character is an
    // '(', push it to the stack.
    else if (infix.charAt(i) == '(')
        char_stack.add('(');

    // If the scanned character is an
    // ')', pop and output from the stack
    // until an '(' is encountered.
    else if (infix.charAt(i) == ')')
    {
        while (char_stack.peek() != '(')
        {
            output += char_stack.peek();
            char_stack.pop();
        }
    }
}
```





```
// Remove '(' from the stack
char_stack.pop();
}

// Operator found
else {
    if (isOperator(char_stack.peek()))
    {
        while ((getPriority(infix.charAt(i)) <
                    getPriority(char_stack.peek()))
            || (getPriority(infix.charAt(i)) <=
                    getPriority(char_stack.peek())
                    && infix.charAt(i) == '^'))
        {
            output += char_stack.peek();
            char_stack.pop();
        }

        // Push current Operator on stack
        char_stack.add(infix.charAt(i));
    }
}
while(!char_stack.empty()){
    output += char_stack.pop();
}
return output;
}
```





Dash



All



Articles



Videos



Problems



Quiz

Get 90% Refund!

Courses

Tutorials

Jobs

Practice

Contests



«

»

```
static String infixToPrefix(char[] infix)
{
    /*
     * Reverse String Replace ( with ) and vice versa Get Postfix Reverse Postfix *
     */
    int l = infix.length;

    // Reverse infix
    String infix1 = reverse(infix, 0, l - 1);
    infix = infix1.toCharArray();

    // Replace ( with ) and vice versa
    for (int i = 0; i < l; i++)
    {
        if (infix[i] == '(')
        {
            infix[i] = ')';
            i++;
        }
        else if (infix[i] == ')')
        {
            infix[i] = '(';
            i++;
        }
    }

    String prefix = infixToPostfix(infix);
}
```



P



Dash



All



Articles



Videos



Problems



Quiz



```
// Reverse postfix
prefix = reverse(prefix.toCharArray(), 0, l-1);

return prefix;
}

// Driver code
public static void main(String[] args)
{
    String s = ("x+y*z/w+u");
    System.out.print(infixToPrefix(s.toCharArray()));
}
}
```

Output

```
++x/*yzwu
```

Time Complexity: $O(n)$

Stack operations like push() and pop() are performed in constant time. Since we scan all the characters in the expression once the complexity is linear in time i.e $O(n)$

Auxiliary Space: $O(n)$ due to recursive stack space

[Mark as Read](#)

 Report An Issue

If you are facing any issue on this page. Please let us know.

