

**Ex. No. 3A**

26-07-2017

## CONVERSION OF INFIX TO POSTFIX EXPRESSION

### Question:

Develop a C++ program to convert an infix expression to postfix expression using stack.

### Algorithm:

1. Start.
2. Scan the Infix expression from left to right for tokens (Operators, Operands & Parentheses) and perform the steps 2 to 5 for each token in the Expression.
3. If token is operand, Append it in postfix expression.
4. If token is left parentheses “(”, push it in stack.
5. If token is an operator,
  - Pop all the operators which are of higher or equal precedence then the incoming token and append them (in the same order) to the output Expression.
  - After popping out all such operators, push the new token on stack.
6. If “)” right parentheses are found,
  - Pop all the operators from the Stack and append them to Output String, till you encounter the Opening Parenthesis “(“.
  - Pop the left parenthesis but don't append it to the output string (Postfix notation does not have brackets).
7. When all tokens of Infix expression have been scanned. Pop all the elements from the stack and append them to the Output String.
8. The Output string is the Corresponding Postfix Notation and display the postfix expression.
9. End.

### Program:

```
#include <iostream>

using namespace std;

char stk[10];

int top=-1;

char pop()
{
    char data;
    if(top==-1)
        return -1;
    else
    {
        data=stk[top];
        top--;
        return data;
    }
}

void push(char x)
{
    top=top+1;
    stk[top]=x;}
```

```
int priority(char x)
{
    if(x=='(')
        return 0;
    else if(x=='+'||x=='-')
        return 1;
    else if(x=='*'||x=='/')
        return 2;
    else
        return -1;
}
```

```
int main()
{
    char x;
    char xep[10];
    cout<<"\n Enter infix expression: ";
    char *e;
    cin>>xep;
    e=xep;
    cout<<"\n Postfix expression: ";
```

```
while(*e!='\0')
{
    if(isalnum(*e))
        cout<<*e;
    else if(*e=='(')
        push(*e);
    else if(*e==')')
    {
        while((x=pop())!='(')
            cout<<x;
    }
    else
    {
        while(priority(stk[top])>=priority(*e))
            cout<<pop();
        push(*e);
    }
    e++;
}
while(top!=-1)
{
    cout<<pop(); }}
```

## Output:

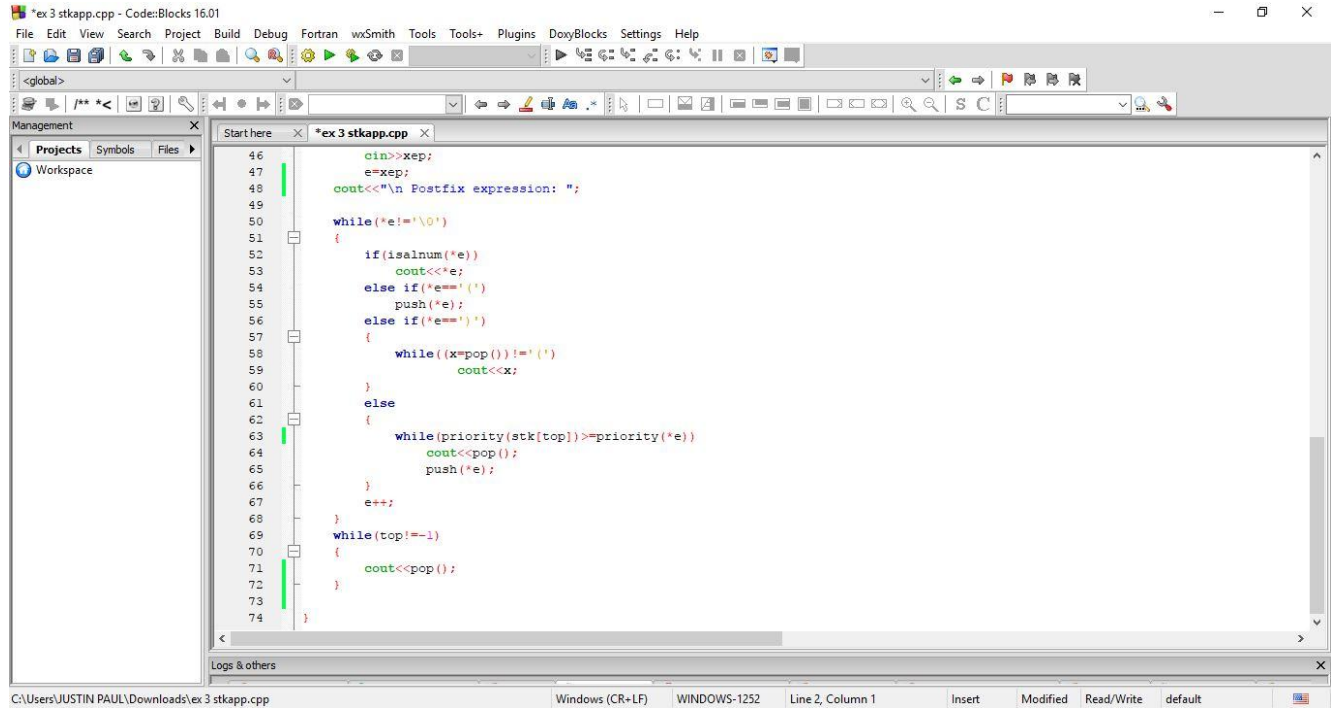
The screenshot shows a C++ IDE with the file `*ex 3 stkapp.cpp` open. The code defines a stack structure and functions for pushing and popping elements. The stack is implemented as an array `stk` of size 10, with a `top` pointer initialized to -1. The `pop()` function returns the top element and decrements `top`. The `push(char x)` function increments `top` and adds the character `x` to the stack. The `priority(char x)` function is partially visible at the bottom of the code block.

```
1 //Program to convert infix expression to postfix expression
2 #include <iostream>
3
4 using namespace std;
5
6 char stk[10];
7 int top=-1;
8
9 char pop()
10 {
11     char data;
12     if(top==0)
13         return -1;
14     else
15     {
16         data=stk[top];
17         top--;
18         return data;
19     }
20 }
21
22 void push(char x)
23 {
24     top=top+1;
25     stk[top]=x;
26 }
27
28 int priority(char x)
29 {
```

The screenshot shows the continuation of the C++ program. The `priority(char x)` function is fully defined, returning the precedence of operators: 0 for '(', 1 for '+', '-', 2 for '\*', '/', and -1 for all other characters. The `main()` function prompts the user to enter an infix expression, reads it into `xep`, and then enters a loop to process the expression character by character, pushing operators onto the stack and printing operands.

```
28 int priority(char x)
29 {
30     if(x=='(')
31         return 0;
32     else if(x=='+'||x=='-')
33         return 1;
34     else if(x=='*'||x=='/')
35         return 2;
36     else
37         return -1;
38 }
39
40 int main()
41 {
42     char x;
43     char xep[10];
44     cout<<"\n Enter infix expression: ";
45     char *e;
46     cin>>xep;
47     e=xep;
48     cout<<"\n Postfix expression: ";
49
50     while(*e!='\0')
51     {
52         if(isalnum(*e))
53             cout<<*e;
54         else if(*e=='(')
55             push(*e);
56         else if(*e==')')
```

# DATA STRUCTURES LAB



```
46     cin>>xep;
47     e=xep;
48     cout<<"\n Postfix expression: ";
49
50     while(*e!='\0')
51     {
52         if(isalnum(*e))
53             cout<<*e;
54         else if(*e=='(')
55             push(*e);
56         else if(*e=='(')
57         {
58             while((x=pop())!='(')
59                 cout<<x;
60         }
61         else
62         {
63             while(priority(stk[top])>=priority(*e))
64                 cout<<pop();
65             push(*e);
66         }
67         e++;
68     }
69     while(top!=-1)
70     {
71         cout<<pop();
72     }
73
74 }
```



```
"C:\Users\JUSTIN PAUL\Downloads\ex 3 stkapp.exe"

Enter infix expression: A+B*C-D

Postfix expression: ABC*+D-
Process returned 0 (0x0)   execution time : 8.626 s
```

## VIDEO URL:

<https://youtu.be/CdtwCYs8ID0>

## RESULT:

The program to convert infix expression to postfix expression using stack is implemented successfully and the output is verified.