	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology **Department of** **Artificial Intelligence and Data Science**

Name: Siddhesh Dilip Khairnar		
Class: SY	Division: B	Roll No: 272028
Semester: III	Academic Year: 2022-2023	
Subject Name & Code: Data Structure, ADUA21202		
Title of Assignment: Implement stack for expression conversion (infix to postfix)		

Assignment No.- 6

PAGE NO.:
DATE: / /

Ds Assignment 6

Name: Siddhesh Khairnar
Rollno: 272028

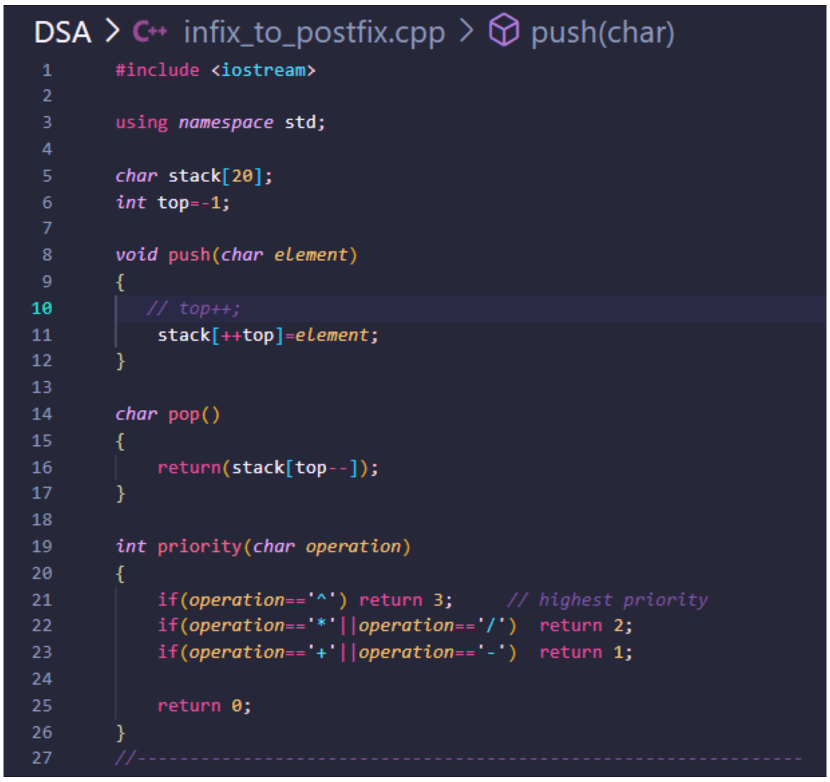
Aim: Implement stack for expression conversion (infix to postfix)

Theory:
Infix expression: The expression of the form an operator b(a+b). when an operator is in - between every pair of operand.
Postfix expression: The expression of the form a b operator (ab+). when an operator is followed by every pair of operand.
STACK: → Stack is a linear data structure which follow a particula order in which the operation are performed. The order may be LIFO (last in First out) or FILO. There are many real life example of a Stack. consider an example of plate stacked over one another in the canteen. The plate which is at top is the first one to be removed. that the plate which has been placed at the bottommost position remain in the stack for the longest periode of time. so, it can be simply seen to follo LIFO/FILO order.

Conclusion: Thus we have successfully implemented stack for expressio conversion (infix to postfix

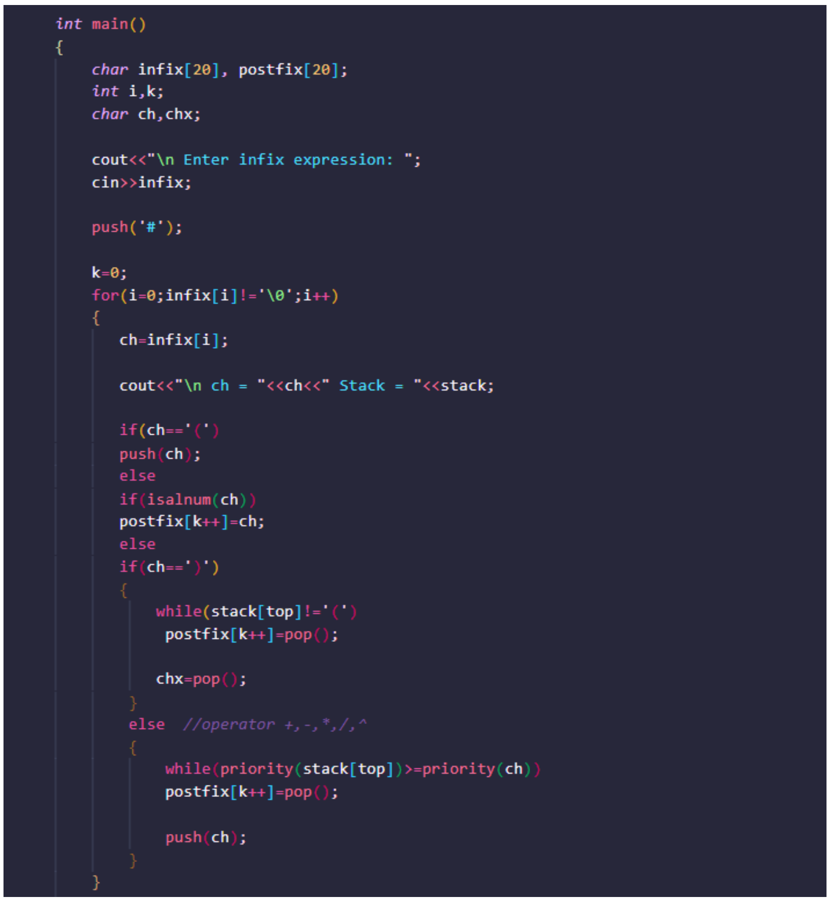
©
Ashwin

Program:

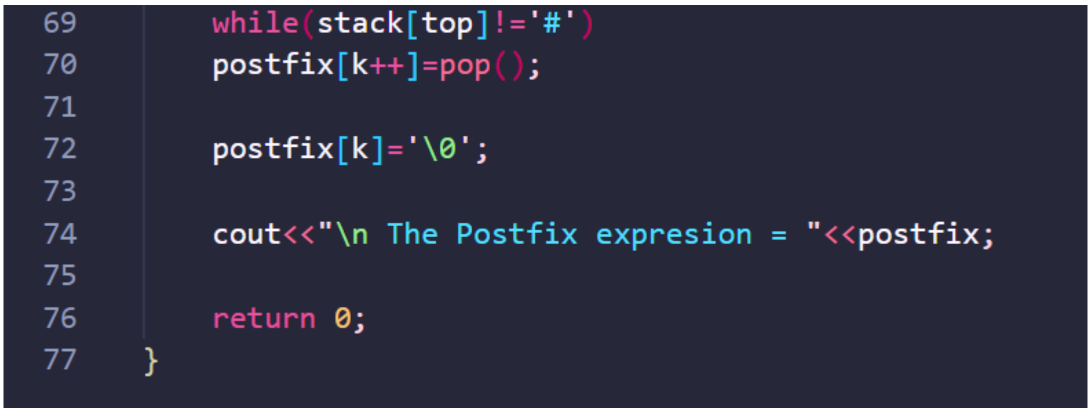


DSA > infix_to_postfix.cpp > push(char)

```cpp
#include <iostream>

using namespace std;

char stack[20];
int top=-1;

void push(char element)
{
    // top++;
    stack[++top]=element;
}

char pop()
{
    return(stack[top--]);
}

int priority(char operation)
{
    if(operation=='^') return 3;     // highest priority
    if(operation=='*'||operation=='/')  return 2;
    if(operation=='+'||operation=='-')  return 1;

    return 0;
}
//------------------------------------------------------------------
```

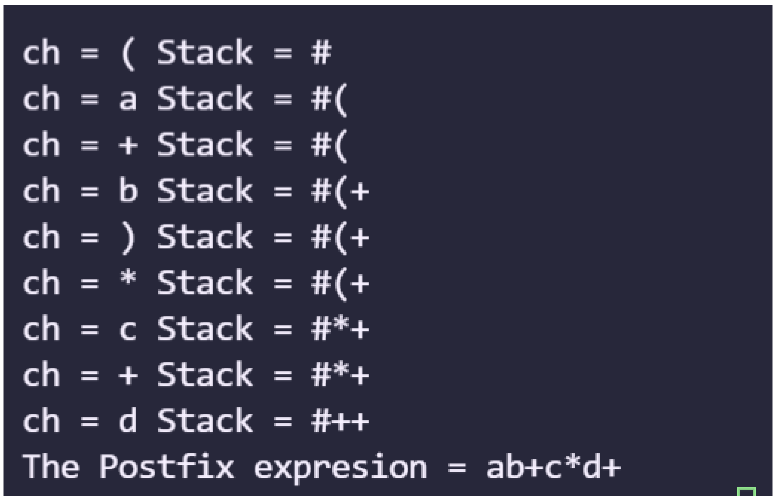


```cpp
int main()
{
    char infix[20], postfix[20];
    int i,k;
    char ch,chx;

    cout<<"\n Enter infix expression: ";
    cin>>infix;

    push('#');

    k=0;
    for(i=0;infix[i]!='\0';i++)
    {
       ch=infix[i];

       cout<<"\n ch = "<<ch<<" Stack = "<<stack;

       if(ch=='(')
       push(ch);
       else
       if(isalnum(ch))
       postfix[k++]=ch;
       else
       if(ch==')')
       {
          while(stack[top]!='(')
           postfix[k++]=pop();

          chx=pop();
        }
        else  //operator +,-,*,/,^
        {
           while(priority(stack[top])>=priority(ch))
           postfix[k++]=pop();

           push(ch);
        }
    }
```



```
    while(stack[top]!='#')
    postfix[k++]=pop();

    postfix[k]='\0';

    cout<<"\n The Postfix expresion = "<<postfix;

    return 0;
}
```

Output:



```
ch = ( Stack = #
ch = a Stack = #(
ch = + Stack = #(
ch = b Stack = #(+
ch = ) Stack = #(+
ch = * Stack = #(+
ch = c Stack = #*+
ch = + Stack = #*+
ch = d Stack = #++
The Postfix expresion = ab+c*d+
```