

Syntax Parsing in Programming Languages:

Parsing expressions is a key step in many compilers and language processors. When a language's syntax requires parsing mathematical or logical expressions, converting between infix and postfix notation ensures that expressions are evaluated correctly. Accept an infix expression and show the expression in postfix form.

Program

```
#include <iostream>
#include <string>
using namespace std;
const int MAX_SIZE = 50;
class Stack {
private:
    char arr[MAX_SIZE];
    int top;
    int isOperator(char ch) {
        if (ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '%' || ch == '^') {
            return 1;
        }
        return 0; }
    int isOperand(char ch) {
        if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z') || (ch >= '0' && ch <= '9')) {
            return 1;
        }
        return 0; }
    int precedence(char op) {
        if (op == '+' || op == '-')
            return 1;

        if (op == '*' || op == '/' || op == '%')
            return 2;
        if (op == '^')
            return 3;
        return 0; }
public:
    Stack() {
        top = -1;
    }
    int isEmpty() {
        return (top == -1) ? 1 : 0;
    }
    int isFull() {
```

```

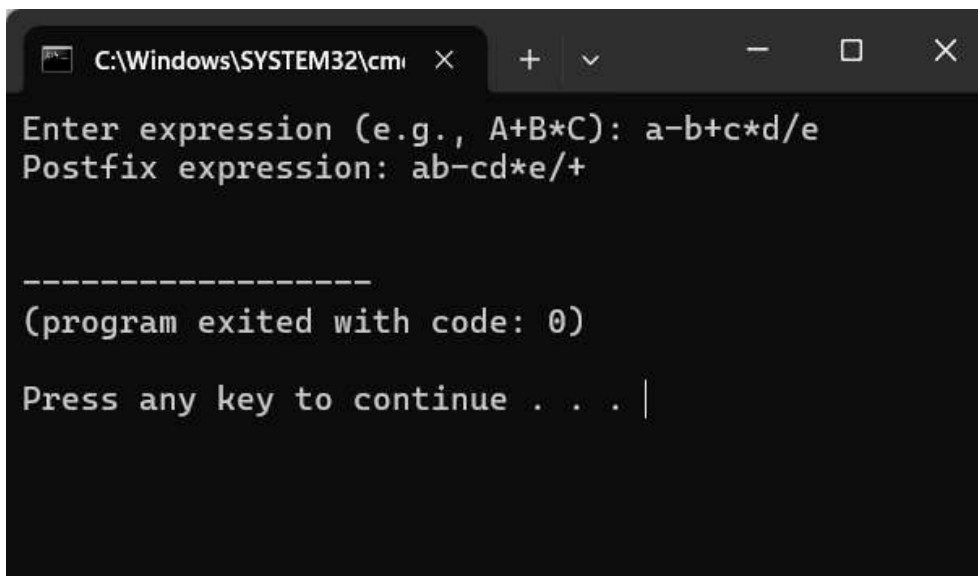
        return (top == MAX_SIZE - 1) ? 1 : 0;    }
void push(char ch) {
    if (isFull()) {
        cout << "Stack overflow" << endl;
        return;
    }
    arr[++top] = ch;
}
char pop() {
    if (isEmpty()) {
        return '\0';
    }
    return arr[top--];    }
char peek() {
    if (isEmpty()) { return '\0';    }
    return arr[top];    }
void infixToPostfix(const string& infix) {
    string postfix_output = "";
    for (char ch : infix) {
        if (isOperand(ch)) {
            postfix_output += ch;
        } else if (ch == '(') {
            push(ch);
        } else if (ch == ')') {
            while (isEmpty() == 0 && peek() != '(') {
                postfix_output += pop();
            }
            if (isEmpty() == 0 && peek() == '(') {
                pop();    }
        } else if (isOperator(ch)) {
            while (isEmpty() == 0 && peek() != '(' &&
                precedence(ch) <= precedence(peek())) {
                postfix_output += pop();    }
            push(ch);
        }
    }
    while (isEmpty() == 0) {
        postfix_output += pop();
    }

    cout << "Postfix expression: " << postfix_output << endl;
}
};

```

```
int main() {  
    string expression;  
    cout << "Enter expression (e.g., A+B*C): ";  
    cin >> expression;  
  
    Stack s;  
    s.infixToPostfix(expression);  
    return 0;  
}
```

Output:



```
C:\Windows\SYSTEM32\cmd.exe x + v - □ ×  
Enter expression (e.g., A+B*C): a-b+c*d/e  
Postfix expression: ab-cd*e/+  
  
-----  
(program exited with code: 0)  
Press any key to continue . . . |
```