```
/*Implement stack as an ADT. Use this ADT to
perform expression conversion (Infix  Prefix). */

#include <iostream>
#include <stack>
using namespace std;

// Function to convert Infix expression to Prefix expression
string InfixToPrefix(string infix) {
    stack<char> s;
    string prefix = "";

    // Iterate through each character of the infix expression
    for (int i = 0; i < infix.length(); i++) {
        // If the current character is an operand, add it to the prefix expression
        if (isalnum(infix[i]))
            prefix += infix[i];
        // If the current character is an operator
        else {
            // If the operator stack is not empty and the top of the stack is not a
left parenthesis,
            // pop operators from the stack and add them to the prefix expression
until
            // the top of the stack is a left parenthesis or the stack is empty
            while (!s.empty() && s.top() != '(' && s.top() != ')') {
                prefix += s.top();
                s.pop();
            }
            // Push the current operator onto the stack
            s.push(infix[i]);
        }
    }

    // Pop any remaining operators from the stack and add them to the prefix
expression
    while (!s.empty()) {
        prefix += s.top();
        s.pop();
    }

    return prefix;
}

int main() {
    string infix = "A*B+C*D";
    cout << "Infix: " << infix << endl;
    cout << "Prefix: " << InfixToPrefix(infix) << endl;
    return 0;
}
```