

```

#include <iostream>

#include <stack>

using namespace std;

//FUNCTION TO CHECK AND RETURN THE PRECEDENCE OF THE OPERATOR
int precedence(char c)
{
    if (c == '^')
    {
        return 3;
    }
    else if (c == '*' || c == '/')
    {
        return 2;
    }
    else if (c == '+' || c == '-')
    {
        return 1;
    }
    else
    {
        return -1;
    }
}

void infixToPost(string s)

```

```

{
    stack<char> stack;

    stack.push('{');

    int len = s.length();

    string str2;

    for (int i = 0; i < len; i++)
    {
        if ((s[i] >= 'a' and s[i] <= 'z') or (s[i] >= 'A' and s[i] <= 'Z'))
        {
            str2 += s[i];
        }

        else if (s[i] == '(')
        {
            stack.push('(');
        }

        //THIS DEALS WITH THE ENCOUNTER OF BRACKETS IN THE EXPRESSION

        else if (s[i] == ')')
        {
            while (stack.top() != '{' and stack.top() != '(')
            {
                char c = stack.top();

                stack.pop();

                str2 += c;
            }
        }
    }
}

```

```

    if (stack.top() == '(')
    {
        char c = stack.top();
        stack.pop();
    }
}

//IF THE PRECEDENCE OF THE NEW OPERATOR SCANNED IS LESS THAN THAT OF PRESENT
IN THE STACK

//THEN THIS CONDITION INVOKES

else
{
    while (stack.top() != '{' and precedence(s[i]) <= precedence(stack.top()))
    {
        char oper = stack.top();
        stack.pop();
        str2 += oper;
    }
    stack.push(s[i]);
}

}

//DEALING WITH REMAINING PART OF THE OPERATORS IN STACK

while (stack.top() != '{')
{
    char c = stack.top();
    stack.pop();
    str2 += c;
}

```

```
}
```

```
cout << "The postfix expression is: " << str2 << endl;
```

```
}
```

```
int main()
```

```
{
```

```
    string expression;
```

```
    cin >> expression;
```

```
    infixToPost(expression);
```

```
    return 0;
```

```
}
```

```
// (A+B)*(C+D)
```

```
// A+B*C+D
```

The screenshot shows a C++ program in a code editor and its output in a console window. The code implements an infix-to-postfix conversion algorithm using a stack. It processes the expression "A+B\*C+D" and outputs "ABC\*+D+".

```
68         }
69         stack.push(s[i]);
70     }
71 }
72
73 //DEALING WITH REMAINING PART OF THE OPERATORS IN STACK
74 while (stack.top() != '{')
75 {
76     char c = stack.top();
77     stack.pop();
78     str2 += c;
79 }
80
81 cout << "The postfix expression is: " << str2 << endl;
```

Input

```
A+B*C+D
The postfix expression is: ABC*+D+

..Program finished with exit code 0
Press ENTER to exit console.
```

