

```

1  | #include <stdio.h>
2  | #include <ctype.h>
3  | #define SIZE 50
4
5  | char stack[SIZE];
6  | int top = -1;
7
8  | void push(char element) {
9  |     stack[++top] = element;
10 | }
11
12 | char pop() {
13 |     return stack[top--];
14 | }
15
16 | int pr(char symbol) {
17 |     if (symbol == '^') {
18 |         return 3;
19 |     } else if (symbol == '*' || symbol == '/') {
20 |         return 2;
21 |     } else if (symbol == '+' || symbol == '-') {
22 |         return 1;
23 |     } else {
24 |         return 0;
25 |     }
26 | }
27
28 | int main() {
29 |     char infix[50], postfix[50], ch, element;
30 |     int i = 0, k = 0;
31 |     printf("Enter the Infix expression: ");
32 |     scanf("%49s", infix);
33
34 |     push('#');
35 |     while ((ch = infix[i++]) != '\0') {
36 |         if (ch == '(') {
37 |             push(ch);
38 |         } else if (isdigit((unsigned char)ch)) {
39 |             postfix[k++] = ch;
40 |         } else if (ch == ')') {
41 |             while (stack[top] != '(')
42 |                 postfix[k++] = pop();
43 |             element = pop();
44 |             (void)element;
45 |         } else {
46 |             while (stack[top] != '(' && pr(stack[top]) >= pr(ch))
47 |                 postfix[k++] = pop();
48 |             push(ch);
49 |         }
50 |     }
51
52 |     while (stack[top] != '#')
53 |         postfix[k++] = pop();
54
55 |     postfix[k] = '\0';
56 |     printf("\nPostfix = %s\n", postfix);
57 |     return 0;
58 | }
59

```

Enter the Infix expression: A+(B*C-(D/E^F)*G)*H

Postfix = ABC*DEF^/G*-H**

Process returned 0 (0x0) execution time : 71.107 s

Press any key to continue.

06/10

WAP. to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), minus, * multiply and / divide.

```
#include <stdio.h>
#include <ctype.h>
#define size 50
```

```
char stack[size];
int top = -1;
```

void push (char element)

```
{
    stack[++top] = element;
}
```

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

```
int pr (char symbol) {
    if (symbol == '^')
        return 3;
```

```
else if (symbol == '*' || symbol == '/')
    return 2;
```

```
else if (symbol == '+' || symbol == '-')
    return 1;
```

```
else
```

```
return 0;
```

```

int main() {
    char infix[50], postfix[50], ch;
    int i=0, k=0;
    printf("Enter the expression");
    while ((ch = infix[i++]) != '\0') {
        if (ch == '(') {
            push(ch);
        }
        else if (isalnum(unsigned char) ch) {
            postfix[k++] = ch;
        }
        else if (ch == ')') {
            while (stack[top] != '(') {
                postfix[k++] = pop();
                element = pop();
                (void) element;
            }
        }
        else {
            while (stack[top] != '(' && pr(stack[top]) > pr(ch)) {
                postfix[k++] = pop();
            }
            push(ch);
        }
    }
    while (stack[top] != '#') {
        postfix[k++] = pop();
    }
    postfix[k] = '\0';
    printf("\n Postfix = %s", postfix);
    return 0;
}

```

output
Enter the Infix
Postfix = A

output

Enter the Infix expression: (A+(B*C-CD/E^F)*G)*H
Postfix = ABC*DEF^/G*-H*+