

Experiment 04\infix_to_postfix.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <ctype.h>
4 #include <string.h>
5
6 #define MAX 100
7
8 char stack[MAX];
9 int top = -1;
10
11 void push(char x) {
12     if (top == MAX - 1)
13         return;
14     stack[++top] = x;
15 }
16
17 char pop() {
18     if (top == -1)
19         return -1;
20     return stack[top--];
21 }
22
23 int priority(char x) {
24     if (x == '(') return 0;
25     if (x == '+' || x == '-') return 1;
26     if (x == '*' || x == '/') return 2;
27     return -1;
28 }
29
30 void infixToPostfix(char* infix, char* postfix) {
31     int i = 0, j = 0;
32     char x;
33     push('(');
34     strcat(infix, ")");
35     while (infix[i]) {
36         if (isalnum(infix[i])) {
37             postfix[j++] = infix[i];
38         } else if (infix[i] == '(') {
39             push(infix[i]);
40         } else if (infix[i] == ')') {
41             while ((x = pop()) != '(') {
42                 postfix[j++] = x;
43             }
44         } else {
45             while (priority(stack[top]) >= priority(infix[i])) {
46                 postfix[j++] = pop();
47             }
48             push(infix[i]);
49         }
50     }
51 }
```

```
49         }
50         i++;
51     }
52     postfix[j] = '\0';
53 }
54
55 int evalPostfix(char* postfix) {
56     int stackVal[MAX];
57     int topVal = -1;
58     int i = 0;
59     while (postfix[i]) {
60         if (isdigit(postfix[i])) {
61             stackVal[++topVal] = postfix[i] - '0';
62         } else {
63             int val2 = stackVal[topVal--];
64             int val1 = stackVal[topVal--];
65             switch(postfix[i]) {
66                 case '+': stackVal[++topVal] = val1 + val2; break;
67                 case '-': stackVal[++topVal] = val1 - val2; break;
68                 case '*': stackVal[++topVal] = val1 * val2; break;
69                 case '/': stackVal[++topVal] = val1 / val2; break;
70             }
71         }
72         i++;
73     }
74     return stackVal[topVal];
75 }
76
77 int main() {
78     char infix[MAX], postfix[MAX];
79     printf("Enter infix expression (single digit operands): ");
80     scanf("%s", infix);
81     infixToPostfix(infix, postfix);
82     printf("Postfix expression: %s\n", postfix);
83     int result = evalPostfix(postfix);
84     printf("Evaluation result: %d\n", result);
85     return 0;
86 }
87 }
```