

Experiments\PostfixEvaluation.c

```
1  #include <stdio.h>
2  #include <string.h>
3  #include <ctype.h>
4  #include <math.h>
5
6  #define max 100
7
8  float st[max];
9  int top = -1;
10
11 void push(float st[], float val);
12 float pop(float st[]);
13 float EvaluatePostfix(char exp[]);
14
15 int main() {
16     float val;
17     char exp[100];
18
19     printf("\t\tEvaluating single digit postfix\n\nEnter a postfix expression: ");
20     fgets(exp, sizeof(exp), stdin);
21     // Remove the newline character if present
22     exp[strcspn(exp, "\n")] = 0;
23
24     val = EvaluatePostfix(exp);
25     printf("\nResult: %f\n", val);
26
27     return 0;
28 }
29
30 float EvaluatePostfix(char exp[]) {
31     int i = 0;
32     float op1, op2, value;
33
34     while (exp[i] != '\0') {
35         // If the character is a digit, push it to the stack
36         if (isdigit(exp[i])) {
37             push(st, (float)(exp[i] - '0'));
38         }
39         // Skip spaces in the expression
40         else if (exp[i] == ' ') {
41             i++;
42             continue;
43         }
44         // If it's an operator, pop two operands from the stack and perform the operation
45         else {
46             if (top < 1) {
47                 printf("Error: Insufficient operands for operator '%c'\n", exp[i]);
48                 return -1;
49             }
50             op2 = pop(st);
51             op1 = pop(st);
52             value = EvaluatePostfixOperator(exp[i], op1, op2);
53             push(st, value);
54             i++;
55         }
56     }
57     return pop(st);
58 }
```

```

49     }
50     op1 = pop(st);
51     op2 = pop(st);
52
53     switch (exp[i]) {
54         case '+':
55             value = op2 + op1;
56             break;
57         case '-':
58             value = op2 - op1;
59             break;
60         case '*':
61             value = op2 * op1;
62             break;
63         case '/':
64             if (op1 != 0) {
65                 value = op2 / op1;
66             } else {
67                 printf("Error: Division by zero\n");
68                 return -1;
69             }
70             break;
71         case '%':
72             value = fmod(op2, op1);
73             break;
74         default:
75             printf("Unexpected character: %c\n", exp[i]);
76             return -1;
77     }
78     push(st, value);
79 }
80 i++;
81 }
82 // Final result should be the only value left in the stack
83 if (top != 0) {
84     printf("Error: The postfix expression is invalid\n");
85     return -1;
86 }
87 return pop(st);
88 }
89
90 void push(float st[], float val) {
91     if (top == max - 1) {
92         printf("\nStack Overflow\n");
93     } else {
94         top++;
95         st[top] = val;
96     }
97 }
98

```

```
99 float pop(float st[]) {
100     if (top == -1) {
101         printf("\nStack Underflow\n");
102         return -1;
103     } else {
104         float val = st[top];
105         top--;
106         return val;
107     }
108 }
109
110
111 /*
112 OUTPUT:-
113
114         Evaluating single digit postfix
115
116 Enter a postfix expression: 231*+9-
117
118 Result: -4.000000
119
120 */
```