

## **DATA STRUCTURES AND ALGORITHMS LAB ASSIGNMENT – 4**

### **Student Details**

- Name: Pratyush Deo Singh
  - Registration Number: 25BCE5101
  - Course / Branch: B Tech CSE
  - Semester: 2<sup>nd</sup>
  - Subject: Data Structures and Algorithms
  - Faculty Name: Dr. Malini A
-

## Program 1: Infix to Postfix Conversion using Stack

### Aim

To write a C program to convert an infix expression into postfix expression using stack.

### Program Code

```
C 1_infix_to_postfix.c > ...
1  #include <stdio.h>
2  #include <ctype.h>
3
4  char stack[50];
5  int top = -1;
6
7  void push(char x) {
8      stack[++top] = x;
9  }
10
11 char pop() {
12     return stack[top--];
13 }
14
15 int priority(char x) {
16     if(x == '+' || x == '-') return 1;
17     if(x == '*' || x == '/') return 2;
18     return 0;
19 }
20
21 int main() {
22     char infix[50], postfix[50];
23     int i = 0, k = 0;
24     printf("Enter infix expression: ");
25     scanf("%s", infix);
26
27     while(infix[i]) {
28         if(isalnum(infix[i]))
29             postfix[k++] = infix[i];
30         else if(infix[i] == '(')
31             push(infix[i]);
32         else if(infix[i] == ')') {
33             while(stack[top] != '(')
34                 postfix[k++] = pop();
35             pop();
36         } else {
37             while(top != -1 && priority(stack[top]) >= priority(infix[i]))
38                 postfix[k++] = pop();
39             push(infix[i]);
40         }
41         i++;
42     }
43
44     while(top != -1)
45         postfix[k++] = pop();
46
47     postfix[k] = '\0';
48     printf("Postfix expression: %s", postfix);
49     return 0;
50 }
Ln 51, Col 1 Spaces: 4 UTF-8 CRLF {} C Win32
```

```
C 1_infix_to_postfix.c > ...
21 int main() {
27     while(infix[i]) {
31         push(infix[i]);
32     } else if(infix[i] == ')') {
33         while(stack[top] != '(')
34             postfix[k++] = pop();
35         pop();
36     } else {
37         while(top != -1 && priority(stack[top]) >= priority(infix[i]))
38             postfix[k++] = pop();
39         push(infix[i]);
40     }
41     i++;
42 }
43
44 while(top != -1)
45     postfix[k++] = pop();
46
47 postfix[k] = '\0';
48 printf("Postfix expression: %s", postfix);
49 return 0;
50 }
51
Ln 51, Col 1 Spaces: 4 UTF-8 CRLF {} C Win32
```

### Output

```
Enter infix expression: (A+B)*C
Postfix expression: AB+C*
```

## Program 2: Infix to Prefix Conversion using Stack

### Aim

To write a C program to convert an infix expression into prefix expression using stack.

### Program Code

```
C 2_infix_to_prefix.c > ...
1  #include <stdio.h>
2  #include <string.h>
3  #include <ctype.h>
4
5  char stack[50];
6  int top = -1;
7
8  void push(char x) {
9      stack[++top] = x;
10 }
11
12 char pop() {
13     return stack[top--];
14 }
15
16 int priority(char x) {
17     if(x == '+' || x == '-') return 1;
18     if(x == '*' || x == '/') return 2;
19     return 0;
20 }
21
22 int main() {
23     char infix[50], prefix[50], temp[50];
24     int i, k = 0;
25
26     printf("Enter infix expression: ");
27     scanf("%s", infix);
28
29     strrev(infix);
30
```

```
C 2_infix_to_prefix.c > ...
22 int main() {
30
31     for(i = 0; infix[i]; i++) {
32         if(infix[i] == '(')
33             infix[i] = ')';
34         else if(infix[i] == ')')
35             infix[i] = '(';
36     }
37
38     i = 0;
39     while(infix[i]) {
40         if(isalnum(infix[i]))
41             temp[k++] = infix[i];
42         else if(infix[i] == '(')
43             push(infix[i]);
44         else if(infix[i] == ')') {
45             while(stack[top] != '(')
46                 temp[k++] = pop();
47             pop();
48         } else {
49             while(top != -1 && priority(stack[top]) > priority(infix[i]))
50                 temp[k++] = pop();
51             push(infix[i]);
52         }
53         i++;
54     }
55
56     while(top != -1)
57         temp[k++] = pop();
58
59     temp[k] = '\0';
60     strrev(temp);
61     strcpy(prefix, temp);
62
63     printf("Prefix expression: %s", prefix);
64     return 0;
65 }
66
```

### Output

```
Enter infix expression: (A+B)*C
Prefix expression:  *+ABC
```

## Program 3: Postfix Expression Evaluation

### Aim

To write a C program to evaluate a postfix expression using stack.

### Program Code

```
C 3_postfix_evaluation.c > ...
1  #include <stdio.h>
2  #include <ctype.h>
3
4  int stack[50];
5  int top = -1;
6
7  void push(int x) {
8      stack[++top] = x;
9  }
10
11 int pop() {
12     return stack[top--];
13 }
14
15 int main() {
16     char postfix[50];
17     int i, a, b;
18
19     printf("Enter postfix expression: ");
20     scanf("%s", postfix);
21
22     for(i = 0; postfix[i]; i++) {
23         if(isdigit(postfix[i]))
24             push(postfix[i] - '0');
25         else {
26             b = pop();
27             a = pop();
28             if(postfix[i] == '+') push(a + b);
29             if(postfix[i] == '-') push(a - b);
30             if(postfix[i] == '*') push(a * b);
31             if(postfix[i] == '/') push(a / b);
32         }
33     }
34
35     printf("Result = %d", pop());
36     return 0;
37 }
38
```

### Output

```
Enter postfix expression: 23*54*+9-
Result = 17
```

## Program 4: Prefix Expression Evaluation

### Aim

To write a C program to evaluate a prefix expression using stack.

### Program Code

```
C 4_prefix_evaluation.c > ...
1  #include <stdio.h>
2  #include <ctype.h>
3  #include <string.h>
4
5  int stack[50];
6  int top = -1;
7
8  void push(int x) {
9      stack[++top] = x;
10 }
11
12 int pop() {
13     return stack[top--];
14 }
15
16 int main() {
17     char prefix[50];
18     int i, a, b;
19
20     printf("Enter prefix expression: ");
21     scanf("%s", prefix);
22
23     for(i = strlen(prefix) - 1; i >= 0; i--) {
24         if(isdigit(prefix[i]))
25             push(prefix[i] - '0');
26         else {
27             a = pop();
28             b = pop();
29             if(prefix[i] == '+') push(a + b);
30             if(prefix[i] == '-') push(a - b);
31             if(prefix[i] == '*') push(a * b);
32             if(prefix[i] == '/') push(a / b);
33         }
34     }
35
36     printf("Result = %d", pop());
37     return 0;
38 }
39
40
```

### Output

```
Enter prefix expression: -+7*45+20
Result = 25
```

### Result

All stack-based expression conversion and evaluation programs were successfully implemented and executed.