

## LAB 2

→ WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands & the binary operators +, -, \* & /.

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#include <process.h>
```

```
int F (char Symbol)
```

```
{ switch (Symbol).
```

```
{ case '+' :
```

```
case '-' : return 2;
```

```
case '*' :
```

```
case '/' : return 4;
```

```
case '^' :
```

```
case '$' : return 5;
```

```
case '(' : return 0;
```

```
case '#' : return -1;
```

```
default : return 8;
```

```
}
```

```
} int G (char Symbol).
```

```
{ switch (Symbol)
```

```
{ case '+' :
```

```
case '-' : return 1;
```

```
case '*' :
```

```
case '/' : return 3;
```

```
case '^' :
```

```
case '$' : return 6;
```

```
case '(' : return 9;
```

```
case '#' : return 0;
```

```
default : return 7;
```

```
}
```

```
}
```

void infix-postfix (char infix [], char postfix [])

{ int top, i, j;

char s[30], symbol;

top = -1;

s[++top] = '#';

j = 0;

for (i = 0; i < strlen(infix); i++)

{ symbol = infix[i];

while (F(s[top]) > G(symbol))

{ postfix[j] = s[top--];  
j++;

}

if (F(s[top]) != G(symbol))

s[++top] = symbol;

else

top--;

}

while (s[top] != '#')

{ postfix[j++] = s[top--];

}  
postfix[j] = '\\0';

}

void main()

{ char infix[20];

char postfix[20];

printf("enter the valid infix expn \\n");

scanf("%s", infix);

infix-postfix(infix, postfix);

printf("the postfix expn is \\n");

printf("%s \\n", postfix);

~~getch~~

getch();

}

```
1  #include<stdio.h>
2  #include<string.h>
3  #include<process.h>
4  int F(char symbol)
5  {
6      switch(symbol)
7      {
8          case '+':
9          case '-':return 2;
10         case '*':
11         case '/':return 4;
12         case '^':
13         case '$':return 5;
14         case '(':return 0;
15         case '#':return -1;
16         default :return 8 ;
17     }
18 }
19 int G(char symbol)
20 {
21     switch (symbol)
22     {
23         case '+':
24         case '-':return 1;
25         case '*':
26         case '/':return 3;
27         case '^':
28         case '$':return 6;
29         case '(':return 9;
30         case ')':return 0;
31         default :return 7 ;
32     }
33 }
34 void infix_postfix(char infix[],char postfix[])
35 {
36     int top,i,j;
37     char s[30],symbol;
38     top=-1;
39     s[++top]='#';
40     j=0;
41     for(i=0;i<strlen(infix);i++)
42     {
43         symbol=infix[i];
44         while (F(s[top])>G(symbol))
```



```
29         case '(':return 9;
30         case ')':return 0;
31         default :return 7 ;
32     }
33 }
34 void infix_postfix(char infix[],char postfix[])
35 {
36     int top,i,j;
37     char s[30],symbol;
38     top=-1;
39     s[++top]='#';
40     j=0;
41     for(i=0;i<strlen(infix);i++)
42     {
43         symbol=infix[i];
44         while(F(s[top])>G(symbol))
45         {
46             postfix[j]=s[top--];
47             j++;
48         }
49         if(F(s[top])!=G(symbol))
50             s[++top]=symbol;
51         else
52             top--;
53     }
54     while (s[top]!='#')
55     {
56         postfix [j++]=s[top--];
57     }
58     postfix[j]='\0';
59 }
60
61 void main()
62 {
63     char infix [20];
64     char postfix[20];
65     printf("Enter the valid infix expression:\n");
66     scanf("%s",infix);
67     infix_postfix(infix,postfix);
68     printf("postfix expression is :\n");
69     printf("%s",postfix);
70
71 }
72
```



■ "C:\Users\MEGHA\Documents\MITHUN\practice\ds lab 2.exe"

Enter the valid infix expression:

$a+b*(c^d-e)^{(f+g*h)}-i$

postfix expression is :

$abcd^e-fgh*+^{*}+i-$

Process returned 17 (0x11)      execution time : 42.077 s

Press any key to continue.