

30/9/20

Lab Program - 2.

IBM19CS090

Mohammed Ibrahim Rahil.

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

Code:

```
#include <stdio.h>
#include <string.h>
#include <process.h>
int F(char symbol)
{
    switch (symbol)
    {
        case '+':
        case '-': return 3;
        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 expression\n");
```

```
scanf("%s", infix);
```

```
infix_postfix(infix, postfix);
```

```
printf("The postfix exp is\n");
```

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

```
getch();
```

```
}
```

```

Postfix.c
Saved

1 #include<stdio.h>
2 #include<string.h>
3 int f(char sym)
4 {
5     switch(sym)
6     {
7         case '+':
8         case '-':return 2;
9         case '*':
10        case '/':return 4;
11        case '^':
12        case '$':return 5;
13        case '(':return 0;
14        case '#':return -1;
15        default:return 8;
16    }
17}
18int g(char sym)
19{
20    switch(sym)
21    {
22        case '+':
23        case '-':return 1;
24        case '*':
25        case '/':return 3;
26        case '^':
27        case '$':return 6;
28        case '(':return 9;
29        case ')':return 0;
30        default:return 7;
31    }
32}
33void infix_postfix(char infix[],char postfix[])
34{
35    int top,i,j;
36    char s[30],sym;
37    top=-1;
38    s[++top]='#';
39    j=0;
40    for(i=0;i<strlen(infix);i++)
41    {
42        sym=infix[i];
43        while(f(s[top])>g(sym))
44        {
45            postfix[j]=s[top--];
46            j++;
47        }
48        if(f(s[top])!=g(sym))
49            s[++top]=sym;
50        else
51            top--;
52    }
53    while(s[top]!='#')
54        postfix[j++]=s[top--];
55    postfix[j]='\0';
56}
57void main()
58{
59    char infix[20];
60    char postfix[20];
61    printf("Enter the Valid Infix Expression\n");
62    scanf("%s",infix);
63    infix_postfix(infix,postfix);
64    printf("The Postfix Expression is ");
65    printf("\n%s\n",postfix);
66}

```

× Terminal



Enter the Valid Infix Expression

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

The Postfix Expression is

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

× Terminal



```
Enter the Valid Infix Expression
(a+(b-c)*d)
The Postfix Expression is
abc-d*+
```

× Terminal



```
Enter the Valid Infix Expression
(a+b)*(d-f)
The Postfix Expression is
ab+df-*
```