

Prg:- C program to find factorial of n using Recursion.

#include <stdio.h>

~~#include <conio.h>~~

int fact (int n)

{

if (n == 0) return 1;

return n \* fact (n-1);

void main ()

{

int n;

~~clrscr~~

printf ("Enter the value of n\n");

scanf ("%d", &n);


printf ("The factorial of %d = %d\n", n, fact (n));

}

IBM1903090

RAHIL



factorial.c 

Saved



```
1 #include<stdio.h>
2 int fact(int n)
3 {
4     if(n==0)
5         return 1;
6     return n*fact(n-1);
7 }
8 void main()
9 {
10     int n;
11     printf("Enter the value of n:");
12     scanf("%d",&n);
13     printf("\n%d! = %d",n,fact(n));
14 }
```

× Terminal



Enter the value of n:5

5! = 120

Process finished.

C-Program: (For Evaluation of Postfix)

IBM19CS090  
RAHIL

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

```
#include <math.h>
```

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

```
double compute ( char symbol, double op1, double op2 )
```

```
{  
    switch (symbol)
```

```
{
```

```
    case '+': return op1 + op2;
```

```
    case '-': return op1 - op2;
```

```
    case '*': return op1 * op2;
```

```
    case '/': return op1 / op2;
```

case '\$':

case '^': return pow (op1, op2);

}

}

void main ()

{ double s[20];

double res;

double op1, op2;

int top, i;

char postfix [20], symbol;

pf ("Enter Postfix exprsn: \n");

sf (" . / \* ^ ", &postfix);

top = -1;

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

{

symbol = postfix [i];

if (isdigit (symbol))

s[++top] = symbol - '0';

else

{ op2 = s[top--];

op1 = s[top--];

res = compute (symbol, op1, op2);

s[++top] = res;

} }

res = s[top--];

pf ("result = %f \n", res);

getch ();

}

IBM19CS090

CS090

RAHIL



```
1 #include<stdio.h>
2 #include<math.h>
3 #include<string.h>
4 double eval(char sym,double op1,double op2)
5 {
6     switch(sym)
7     {
8         case '+':return (op1+op2);
9         case '-':return (op1-op2);
10        case '*':return (op1*op2);
11        case '/':return (op1/op2);
12        case '^':
13        case '$':return pow(op1,op2);
14    }
15 }
16 void main()
17 {
18     double s[20],res,op1,op2;
19     int top,i;
20     char pf[20],sym;
21     printf("Enter the Postfix Expression:");
22     scanf("%s",pf);
23     top=-1;
24     for(i=0;i<strlen(pf);i++)
25     {
26         sym=pf[i];
27         if(isdigit(sym))
28             s[++top]=sym-'0';
29         else
30         {
31             op2=s[top--];
32             op1=s[top--];
33             res=eval(sym,op1,op2);
34             s[++top]=res;
35         }
36     }
37     res=s[top--];
38     printf("\nEvaluated Result=%f",res);
39 }
```

× Terminal



Enter the Postfix Expression:123+\*321-+\*

Evaluated Result=20.000000

```

prefix.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 5;
11        case '#':return -1;
12        default:return 6;
13    }
14}
15int g(char sym)
16{
17    switch(sym)
18    {
19        case '+':
20        case '-':return 3;
21        case '*':
22        case '/':return 4;
23        default:return 7;
24    }
25}
26void rev(char a[30]) {
27
28    int i, j;
29    char temp[100];
30    for (i = strlen(a)-1,j=0;i+1!=0;--i,++j)
31    {
32        temp[j] = a[i];
33    }
34    temp[j] = '\0';
35    strcpy(a, temp);
36}
37
38void infix_prefix(char infix[],char prefix[])
39{
40    int top,i,j;
41    char s[30],sym;
42    top=-1;
43    s[++top]='#';
44    j=0;
45    rev(infix);
46    for(i=0;i<strlen(infix);i++)
47    {
48        sym=infix[i];
49        while(f(s[top])>g(sym))
50        {
51            prefix[j]=s[top--];
52            j++;
53        }
54        if(f(s[top])!=g(sym))
55            s[++top]=sym;
56        else
57            top--;
58    }
59    while(s[top]!='#')
60        prefix[j++]=s[top--];
61    prefix[j]='\0';
62    rev(prefix);
63}
64void main()
65{
66    char infix[20];
67    char prefix[20];
68    printf("Enter the Valid Infix Expression\n");
69    scanf("%s",infix);
70    infix_prefix(infix,prefix);
71    printf("The Prefix Expression is ");
72    printf("\n%s\n",prefix);
73}

```



× Terminal



Enter the Valid Infix Expression

a+b\*c+d

The Prefix Expression is

++a\*bcd

× Terminal



Enter the Valid Infix Expression

a/b-c/d

The Prefix Expression is

-/ab/cd

Process finished.

× Terminal



Enter the Valid Infix Expression

a+b

The Prefix Expression is

+ab