Lecture-12

Infix to Postfix Conversion

```
1       #include<stdio.h>
2       char stack[20];
3       int top = -1;
4       void push(char x)
5       {
6           stack[++top] = x;
7       }
8
9       char pop()
10      {
11          if(top == -1)
12              return -1;
13          else
14              return stack[top--];
15      }
16
17      int priority(char x)
18      {
19          if(x == '(')
20              return 0;
21          if(x == '+' || x == '-')
22              return 1;
23          if(x == '*' || x == '/')
24              return 2;
25      }
26
27      main()
28      {
29          char exp[20];
30          char *e, x;
31          printf("Enter the expression :: ");
32          scanf("%s",exp);
33          e = exp;
34          while(*e != '\0')
35          {
36              if(isalnum(*e))
37                  printf("%c",*e);
38              else if(*e == '(')
39                  push(*e);
40              else if(*e == ')')
41              {
```

```
42                 while((x = pop()) != '(')
43                     printf("%c", x);
44             }
45             else
46             {
47                 while(priority(stack[top]) >= priority(*e))
48                     printf("%c",pop());
49                 push(*e);
50             }
51             e++;
52         }
53         while(top != -1)
54         {
55             printf("%c",pop());
56         }
57     }
```

OUTPUT:

```
Enter the expression :: a+b*c
abc*+

Enter the expression :: (a+b)*c+(d-a)
ab+c*da-+
```

Evaluate POSTFIX Expression Using Stack

```
1       #include<stdio.h>
2       int stack[20];
3       int top = -1;
4        void push(int x)
5       {
6               stack[++top] = x;
7       }
8
9       int pop()
10      {
11              return stack[top--];
12      }
13
14      int main()
15      {
16              char exp[20];
17              char *e;
18              int n1,n2,n3,num;
19              printf("Enter the expression :: ");
20              scanf("%s",exp);
21              e = exp;
22              while(*e != '\0')
23              {
24                      if(isdigit(*e))
```

```
25              {
26                      num = *e - 48;
27                      push(num);
28              }
29              else
30              {
31                      n1 = pop();
32                      n2 = pop();
33                      switch(*e)
34                      {
35                              case '+':
36                              {
37                                      n3 = n1 + n2;
38                  break;
39                              }
40                              case '-':
41                              {
42                                      n3 = n2 - n1;
43                                      break;
44                              }
45                              case '*':
46                              {
47                                      n3 = n1 * n2;
48                                      break;
49                              }
```

```
50                              case '/':
51                              {
52                                      n3 = n2 / n1;
53                                      break;
54                              }
55                      }
56                      push(n3);
57              }
58              e++;
59          }
60          printf("\nThe result of expression %s  =  %d\n\n",exp,pop());
61          return 0;
62
63      }
64
```

Output:

```
Enter the expression :: 245+*

The result of expression 245+*  =  18
```