1BM19CS098

Page No. :
Date : / 201

30/09/2020

Week 3 - Converting Infix to Postfix

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

Algorithm:

Step 1: Stack precedence function F:

```
switch(symbol)
  {
     case '+':
     case '-': return 2;
     case '*':
     case '/': return 4;
     case '^':
     case '$': return 5;
     ~~case '#'~~
     case '(': return 0;
     case '#': return -1;
     default: return 8;
  }
```

Step 2: Input precedence function G:

```
switch(symbol)
```

RISH

```
{
  case'+':
  case'-': return 1;
  case'*':
  case'/': return 3;
  case'^':
  case'$: return 6;
  case'(': return 9;
  case')': return 0;
 default: return 7;
  }
}
```

~~void int~~ Step 3: Method to implement conversion from infix to postfix (infix_postfix):

```
        top= -1;
        S[++top]='#'
        j=0;
      for(i=0; i<strlen(infix); i++)
        {
          symbol= infix[i];
          while (stackprecedence >Input precedence)
            {
               postfix[j] = S[top--]; // Pop from stack
                j++;
            }
        if (stack precedence != Input precedence)
```

```
    S[++top] = symbol        //Push symbol to stack
    else
    top--;            //Pop symbol without storing.
}
while(S[top]!='#')
    postfix[j++]=S[top--];
postfix[j]='\0';
}
```

Step 4 : Main function :
Enter infix expression and store in infix[]

Step 5 : Call infix_postfix :
infix_postfix(infix, postfix);

Step6 : Print the converted postfix expression.