Infix to postfix expression conversion program

BY

DR. RASHMI NAVEEN RAJ, I&CT DEPT.,

rashmi.naveen@manipal.edu

LECTURE-8, OCT 23 (10.30PM TO 12.30PM), 2021

Algorithm to convert infix expression to postfix

1. Scan the infix string from left to right. Assume the operand is single digit .

2. If the scanned character is an operand, then copy to postfix string

else

// If the scanned character is an operator, then

if it is a right parenthesis then

- POP till you find first left parenthesis
- //…

- else if the precedence of the scanned operator is > then the precedence of the operator on top of the stack, then PUSH the scanned operator
- Else
- POP until you get an operator with precedence less than the scanned operator
- precedence

- Repeat the steps till you get a NULL character in the infix string
- POP all the operators to the postfix string

Infix to postfix expression conversion program

```
enum precedence{ lparen, rparen, plus, minus, times, divide,
                 mod, eos, operand};
```

(lparen = 0, rparen = 1, plus = 2, minus = 3, times = 4, divide = 5, eos = 8)

```
icp[] = {20, 19, 12, 12, 13, 13, 13, 0};
isp[] = {0, 19, 12, 12, 13, 13, 13, 0}
```

// icp[plus] = icp[2] = 12

precedence of '+'

```
precedence get_token(char c)
{
    switch(c)
    {
    case '(' : return lparen;
    case ')' : return rparen;
    case '+' : return plus;
```

icp : incoming precedence

isp : in stack precedence

Infix to postfix expression conversion program

```
    case '-' : return minus;
    case '*' : "     times;
    case '/' : "     divide;
    case '%' : "     mod;
    case '#' : return cos;
    default  : return operand;
  }
}
```

```
class stack{    int top;
                char a[50];
                int maxsize;

public:         stack();
                char pop();
                void push(char);
                —
                —
                friend void in2postfix(char []);

};
```

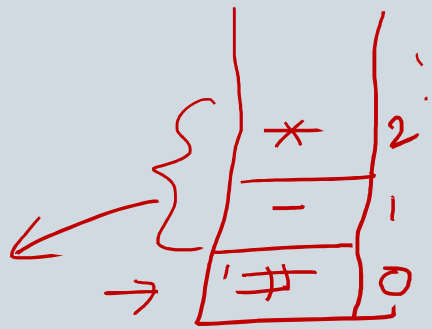
```
Void in2postfix(char infix[])
{
  precedence temp;
  int i=0, j=0;
  char postfix[20];
  Stack S;
  S.push('#'); // eos operator

  while(infix[i] != '\0')
  {
    temp = get_token(infix[i]);
    if(temp == operand)
        postfix[j++] = infix[i];
    else if(temp == rparen)
    {
      while(S.a[S.top] != '(')
      {
          postfix[j++] = S.pop();
      }
      S.pop(); // to pop '('
    }
    else
    {
      while(icp[temp] <=
            isp[get_token(S.a[S.top])])
      {
        postfix[j++] = S.pop();
      }
      S.push(infix[i]);
    }
    i++;
  }
```

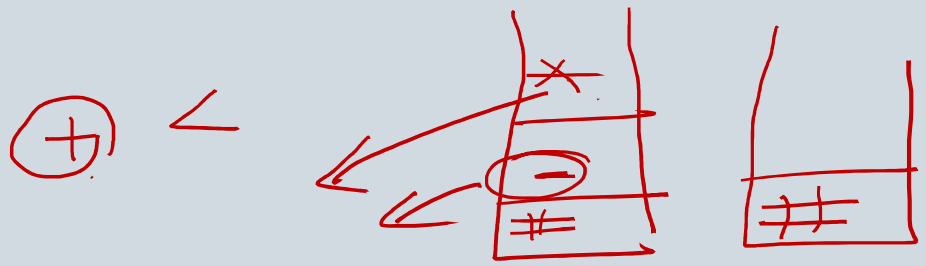
```
while(s.top>0) // . . .
        postfix[j++]=s.pop();
postfix[j]='\0';
cout<<"postfix expression is:"<<postfix;
}

int main()
{
        ? 1. Read the infix expression from user
          2. Call the fn in2post(infix);
          0

}
```



'\0'



Infix to Prefix conversion

Sl. No.	Infix
1	a+b*c-d
2	a*b+c/d
3	a*(b+c)/d
4	(a*(b+c))/d
5	a/b^c+d*e-f*g
6	(a+b)*c/d-e
7	(a+b)*(c-d)/(e+f)
8	**(a+b)*(c-d)/((e-f)*(g+h))**