Intermediate code generation – Postfix, Prefix

RA181100010344

VIKASH MISHRA

COMPILER DESIGN LAB 10

AIM - To convert the infix expression into prefix and postfix notation accordingly

ALGOTIHM -

We made an stack which inputs the value of each element in the expression.

For infix to postfix ,according to the precedence rule if any bracket it found it will push it into the stack ,and if we find any close bracket accordingly , we pop out and push the stack. When an element is encountered it will check with the precedence rule and pop and push the element in the stack according to the rule. The reverse logic is applied from conversion of infix to postfix.

CODE -

OPERATORS = set(['+', '-', '\*', '/', '(', ')'])

PRI = {'+':1, '-':1, '\*':2, '/':2}

# POSTFIX

def infix\_to\_postfix(formula):

stack = []

output = ''

for ch in formula:

if ch not in OPERATORS:

output += ch

elif ch == '(':

stack.append('(')

elif ch == ')':

while stack and stack[-1] != '(':

output += stack.pop()

stack.pop() # pop '('

else:

while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:

output += stack.pop()

stack.append(ch)

# leftover

while stack:

output += stack.pop()

print(f'POSTFIX: {output}')

return output

# PREFIX

def infix\_to\_prefix(formula):

op\_stack = []

exp\_stack = []

for ch in formula:

if not ch in OPERATORS:

exp\_stack.append(ch)

elif ch == '(':

op\_stack.append(ch)

elif ch == ')':

while op\_stack[-1] != '(':

op = op\_stack.pop()

a = exp\_stack.pop()

b = exp\_stack.pop()

exp\_stack.append( op+b+a )

op\_stack.pop() # pop '('

else:

while op\_stack and op\_stack[-1] != '(' and PRI[ch] <= PRI[op\_stack[-1]]:

op = op\_stack.pop()

a = exp\_stack.pop()

b = exp\_stack.pop()

exp\_stack.append( op+b+a )

op\_stack.append(ch)

# leftover

while op\_stack:

op = op\_stack.pop()

a = exp\_stack.pop()

b = exp\_stack.pop()

exp\_stack.append( op+b+a )

print(f'PREFIX: {exp\_stack[-1]}\n')

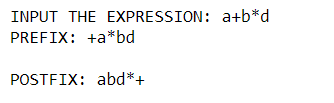
return exp\_stack[-1]

expres = input("INPUT THE EXPRESSION: ")

pre = infix\_to\_prefix(expres)

pos = infix\_to\_postfix(expres)

OUTPUT -



RESULT - Intermediate code generator for prefix and postfix was implemented successfully.