

Exp-04

Implementation of Syntax Analysis: Infix, postfix, prefix notations

NAME: Ayush Jindal
REG NO: RA1911003010308
SECTION : E1

AIM:

Implementation of Postfix, Prefix, Infix notation

Source Code:

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

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

```
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()

else:

    while stack and stack[-1] != '(' and PRI[ch] <= PRI[stack[-1]]:
        output += stack.pop()

    stack.append(ch)


while stack:
    output += stack.pop()

print(f'POSTFIX: {output}')

return output


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()
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)


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]}')


return exp_stack[-1]

expres = input("INPUT THE EXPRESSION: ")

pre = infix_to_prefix(expres)

pos = infix_to_postfix(expres)

```



OnlineGDB beta

online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

out • FAQ • Blog • Terms of Use • Contact Us

• GDB Tutorial • Credits • Privacy

© 2016 - 2022 GDB Online

main.py

8'''

9OPERATORS = set(['+', '-', '*', '/', '(', ')'])

10

11PRI = {'+': 1, '-': 1, '*': 2, '/': 2}

12

13

14

15

16def infix_to_postfix(formula):

17 stack = []

18

19 output = ''

20

21 for ch in formula:

22

23 if ch not in OPERATORS:

24

25 output += ch

26

27 elif ch == '(':

28

29 stack.append('(')

30

31 elif ch == ')':


32

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

34 output += stack.pop()

35

36 stack.pop()



OnlineGDB beta

online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

out • FAQ • Blog • Terms of Use • Contact Us

• GDB Tutorial • Credits • Privacy

© 2016 - 2022 GDB Online

main.py

96

97 op_stack.append(ch)

98

99 # Leftover

100

101 while op_stack:

102 op = op_stack.pop()

103

104 a = exp_stack.pop()

105

106 b = exp_stack.pop()

107

108 exp_stack.append(op + b + a)

109

110 print(f'PREFIX: {exp_stack[-1]}')

111

input

INPUT THE EXPRESSION: a+b/c

PREFIX: +a/bc

POSTFIX: abc/+

...

...Program finished with exit code 0

Press ENTER to exit console.

RESULT:

Successfully implemented postfix, prefix, infix notations.