

# **18CSC304J/ Compiler Design**

**Submitted By:- ANANNYA P. NEOG (RA1911003010367)**

## **Exp-5: FIRST AND FOLLOW computation**

**Aim:-** To write a program to perform FIRST AND FOLLOW computation

### **Code:-**

```
import sys
sys.setrecursionlimit(60)

def first(string):
    #print("first({})".format(string))
    first_ = set()
    if string in non_terminals:
        alternatives = productions_dict[string]

        for alternative in alternatives:
            first_2 = first(alternative)
            first_ = first_ | first_2

    elif string in terminals:
        first_ = {string}

    elif string==" or string=="@':
        first_ = {'@'}

    else:
        first_2 = first(string[0])
        if '@' in first_2:
            i = 1
            while '@' in first_2:
                #print("inside while")

                first_ = first_ | (first_2 - {'@'})
                #print('string[i:]=', string[i:])
                if string[i:] in terminals:
                    first_ = first_ | {string[i:]}
                    break
                elif string[i:] == "':
                    first_ = first_ | {'@'}
                    break
                first_2 = first(string[i:])
                first_ = first_ | first_2 - {'@'}
                i += 1
        else:
            first_ = first_ | first_2

    #print("returning for first({})".format(string),first_)
    return first_
```

```

def follow(nT):
    #print("inside follow({})".format(nT))
    follow_ = set()
    #print("FOLLOW", FOLLOW)
    prods = productions_dict.items()
    if nT==starting_symbol:
        follow_ = follow_ | {'$'}
    for nt,rhs in prods:
        #print("nt to rhs", nt,rhs)
        for alt in rhs:
            for char in alt:
                if char==nT:
                    following_str = alt[alt.index(char) + 1:]
                    if following_str=="":
                        if nt==nT:
                            continue
                        else:
                            follow_ = follow_ | follow(nt)
                    else:
                        follow_2 = first(following_str)
                        if '@' in follow_2:
                            follow_ = follow_ | follow_2-{'@'}
                            follow_ = follow_ | follow(nt)
                        else:
                            follow_ = follow_ | follow_2
    #print("returning for follow({})".format(nT),follow_)
    return follow_

```

```

no_of_terminals=int(input("Enter no. of terminals: "))

```

```

terminals = []

```

```

print("Enter the terminals :")

```

```

for _ in range(no_of_terminals):
    terminals.append(input())

```

```

no_of_non_terminals=int(input("Enter no. of non terminals: "))

```

```

non_terminals = []

```

```

print("Enter the non terminals :")

```

```

for _ in range(no_of_non_terminals):
    non_terminals.append(input())

```

```

starting_symbol = input("Enter the starting symbol: ")

```

```

no_of_productions = int(input("Enter no of productions: "))

```

```

productions = []

```

```

print("Enter the productions:")
for _ in range(no_of_productions):
    productions.append(input())

#print("terminals", terminals)

#print("non terminals", non_terminals)

#print("productions", productions)

productions_dict = {}

for nT in non_terminals:
    productions_dict[nT] = []

#print("productions_dict", productions_dict)

for production in productions:
    nonterm_to_prod = production.split("->")
    alternatives = nonterm_to_prod[1].split("/")
    for alternative in alternatives:
        productions_dict[nonterm_to_prod[0]].append(alternative)

#print("productions_dict", productions_dict)

#print("nonterm_to_prod", nonterm_to_prod)
#print("alternatives", alternatives)

FIRST = {}
FOLLOW = {}

for non_terminal in non_terminals:
    FIRST[non_terminal] = set()

for non_terminal in non_terminals:
    FOLLOW[non_terminal] = set()

#print("FIRST", FIRST)

for non_terminal in non_terminals:
    FIRST[non_terminal] = FIRST[non_terminal] | first(non_terminal)

#print("FIRST", FIRST)

FOLLOW[starting_symbol] = FOLLOW[starting_symbol] | {'$'}
for non_terminal in non_terminals:
    FOLLOW[non_terminal] = FOLLOW[non_terminal] | follow(non_terminal)

#print("FOLLOW", FOLLOW)

```

```
print("{: ^20}{: ^20}{: ^20}".format('Non Terminals','First','Follow'))
```

```
for non_terminal in non_terminals:
```

```
print("{: ^20}{: ^20}{: ^20}".format(non_terminal,str(FIRST[non_terminal]),str(FOLLOW[non_terminal])))
```

```
main.py
1 import sys
2 sys.setrecursionlimit(60)
3
4 def first(string):
5     #print("first({})".format(string))
6     first_ = set()
7     if string in non_terminals:
8         alternatives = productions_dict[string]
9
10        for alternative in alternatives:
11            first_2 = first(alternative)
12            first_ = first_ | first_2
13
14    elif string in terminals:
15        first_ = {string}
16
17    elif string==' ' or string=='@':
18        first_ = {'@'}
19
20    else:
21        first_2 = first(string[0])
22        if '@' in first_2:
23            i = 1
24            while '@' in first_2:
25                #print("inside while")
26
27                first_ = first_ | (first_2 - {'@'})
28                #print('string[i:]=', string[i:])
29                if string[i:] in terminals:
30                    first_ = first_ | {string[i:]}
31                    break
32                elif string[i:] == '':
33                    first_ = first_ | {'@'}
34                    break
35                first_2 = first(string[i:])
36                first_ = first_ | first_2 - {'@'}
37                i += 1
38    else:
39        first_ = first_ | first_2
40
41    #print("returning for first({})".format(string),first_)
42    return first_
43
44
45
```

```
main.py
46 def follow(nT):
47     #print("inside follow({})".format(nT))
48     follow_ = set()
49     #print("FOLLOW", FOLLOW)
50     prods = productions_dict.items()
51     if nT==starting_symbol:
52         follow_ = follow_ | {'$'}
53     for nt,rhs in prods:
54         #print("nt to rhs", nt,rhs)
55         for alt in rhs:
56             for char in alt:
57                 if char==nT:
58                     following_str = alt[alt.index(char) + 1:]
59                     if following_str=='':
60                         if nt==nT:
61                             continue
62                         else:
63                             follow_ = follow_ | follow(nt)
64                     else:
65                         follow_2 = first(following_str)
66                         if '@' in follow_2:
67                             follow_ = follow_ | follow_2 - {'@'}
68                             follow_ = follow_ | follow(nt)
69                         else:
70                             follow_ = follow_ | follow_2
71     #print("returning for follow({})".format(nT),follow_)
72     return follow_
73
74
75
76
77
78 no_of_terminals=int(input("Enter no. of terminals: "))
79
80 terminals = []
81
82 print("Enter the terminals :")
83 for _ in range(no_of_terminals):
84     terminals.append(input())
```

```
main.py
85
86 no_of_non_terminals=int(input("Enter no. of non terminals: "))
87
88 non_terminals = []
89
90 print("Enter the non terminals :")
91 for _ in range(no_of_non_terminals):
92     non_terminals.append(input())
93
94 starting_symbol = input("Enter the starting symbol: ")
95
96 no_of_productions = int(input("Enter no of productions: "))
97
98 productions = []
99
100 print("Enter the productions:")
101 for _ in range(no_of_productions):
102     productions.append(input())
103
104 #print("terminals", terminals)
105
106 #print("non terminals", non_terminals)
107
108 #print("productions",productions)
109
110
111
112 productions_dict = {}
113
114 for nT in non_terminals:
115     productions_dict[nT] = []
116
117
```

```
main.py
118 #print("productions_dict",productions_dict)
119
120 for production in productions:
121     nonterm_to_prod = production.split("->")
122     alternatives = nonterm_to_prod[1].split("/")
123     for alternative in alternatives:
124         productions_dict[nonterm_to_prod[0]].append(alternative)
125
126 #print("productions_dict",productions_dict)
127
128 #print("nonterm_to_prod",nonterm_to_prod)
129 #print("alternatives",alternatives)
130
131
132 FIRST = {}
133 FOLLOW = {}
134
135 for non_terminal in non_terminals:
136     FIRST[non_terminal] = set()
137
138 for non_terminal in non_terminals:
139     FOLLOW[non_terminal] = set()
140
141 #print("FIRST",FIRST)
142
143 for non_terminal in non_terminals:
144     FIRST[non_terminal] = FIRST[non_terminal] | first(non_terminal)
145
146 #print("FIRST",FIRST)
147
148 FOLLOW[starting_symbol] = FOLLOW[starting_symbol] | {'$'}
149 for non_terminal in non_terminals:
150     FOLLOW[non_terminal] = FOLLOW[non_terminal] | follow(non_terminal)
151
152 #print("FOLLOW", FOLLOW)
153
154
155 print("{: ^20}{: ^20}{: ^20}".format('Non Terminals','First','Follow'))
156 for non_terminal in non_terminals:
157     print("{: ^20}{: ^20}{: ^20}".format(non_terminal,str(FIRST[non_terminal]),str(FOLLOW[non_terminal])))
```

(Kindly refer the output screenshot for the inputs)

### Input 1:

2  
x  
y  
  
3  
X  
Y  
Z  
  
X  
4  
X->YxYy  
X->ZyZx  
Y->@  
Z->@

### Output 1:

```
input
Enter no. of terminals: 2
Enter the terminals :
x
y
Enter no. of non terminals: 3
Enter the non terminals :
X
Y
Z
Enter the starting symbol: X
Enter no of productions: 4
Enter the productions:
X->YxYy
X->ZyZx
Y->@
Z->@
Non Terminals      First      Follow
    X              {'y', 'x'}  {'$'}
    Y              {'@'}      {'x'}
    Z              {'@'}      {'y'}

...Program finished with exit code 0
Press ENTER to exit console.
```

## Input 2:

5  
+  
\*  
a  
(  
)

5  
E  
B  
T  
Y  
F

E  
5  
E->TB  
B->+TB/@  
T->FY  
Y->\*FY/@  
F->a/(E)

## Output 2:

```
input
Enter no. of terminals: 5
Enter the terminals :
+
*
a
(
)
Enter no. of non terminals: 5
Enter the non terminals :
E
B
T
Y
F
Enter the starting symbol: E
Enter no of productions: 5
Enter the productions:
E->TB
B->+TB/@
T->FY
Y->*FY/@
F->a/(E)

Non Terminals      First      Follow
E      {'a', '('}      {'$', ')'}
B      {'+', '@'}      {'$', ')'}
T      {'a', '('}      {'+', '$', ')'}
Y      {'*', '@'}      {'+', '$', ')'}
F      {'a', '('}      {'+', '$', '*', ')'}

...Program finished with exit code 0
Press ENTER to exit console.
```

**Result:-** The FIRST and FOLLOW sets of non-terminals of a grammar were found successfully.