

Department of Computer Science and Engineering

S.G.Shivanirudh , 185001146, Semester VI

26 February 2021

UCS1602 - Compiler Design

Exercise 4: Recursive Descent Parser using C

Objective:

Write a program in C to construct Recursive Descent Parser for the following grammar which is for arithmetic expression involving + and *. Check the Grammar for left recursion and convert into suitable for this parser. Write recursive functions for every non-terminal. Call the function for start symbol of the Grammar in main(). Extend this parser to include division, subtraction and parenthesis operators

Code:

```
1 #include<stdio.h>
2 #include<string.h>
3 #include<stdlib.h>
4
```

```

5 void tab(int val){
6     while(val--){
7         printf("\t");
8     }
9 int F(char *, int *, int);
10 int Tprime(char *, int *, int);
11 int T(char *, int *, int);
12 int Eprime(char *, int *, int);
13 int E(char *, int *, int);
14
15 int main(){
16
17     char *str = (char*)calloc(100, sizeof(char));
18     printf("\nEnter string to parse: ");
19     scanf(" %s", str);
20     strcat(str, "$");
21     int look_ahead = 0;
22
23     printf("-----\n");
24     printf("Enter E\n");
25     E(str, &look_ahead, 1);
26     printf("Exit E\n");
27     printf("-----\n");
28
29     if (str[look_ahead] == '$')
30         printf("\nSuccess");
31     else
32         printf("\nFailure: %c at position %d not expected. \n", str[look_ahead], look_ahead);
33 }
34
35 int F(char *str, int *look_ahead, int level)
36 {
37     if (str[*look_ahead] == 'i')
38     {
39         tab(level);
40         printf("F: i matched\n");
41         (*look_ahead)++;
42     }
43     else if (str[*look_ahead] == '(')
44     {
45         tab(level);
46         printf("F: ( matched\n");
47         (*look_ahead)++;
48         E(str, look_ahead, level + 1);

```

```

49         if (str[*look_ahead] == ')')
50         {
51             tab(level);
52             printf("F: ) matched\n");
53             (*look_ahead)++;
54         }
55     }
56 }
57
58 int Tprime(char *str, int *look_ahead, int level){
59     if(str[*look_ahead] == '*'){
60         tab(level);
61         printf("T': * matched\n");
62         (*look_ahead)++;
63
64         tab(level);
65         printf("-----\n");
66         tab(level);
67         printf("Enter F\n");
68         F(str, look_ahead, level + 1);
69         tab(level);
70         printf("Exit F\n");
71         tab(level);
72         printf("-----\n");
73
74         tab(level);
75         printf("-----\n");
76         tab(level);
77         printf("Enter T'\n");
78         Tprime(str, look_ahead, level + 1);
79         tab(level);
80         printf("Exit T'\n");
81         tab(level);
82         printf("-----\n");
83     }
84     else if(str[*look_ahead] == '/'){
85         tab(level);
86         printf("T': / matched\n");
87         (*look_ahead)++;
88
89         tab(level);
90         printf("-----\n");
91         tab(level);
92         printf("Enter F\n");
93     }

```

```

94         F(str, look_ahead, level + 1);
95         tab(level);
96         printf("Exit F\n");
97         tab(level);
98         printf("-----\n");
99
100        tab(level);
101        printf("-----\n");
102        tab(level);
103        printf("Enter T'\n");
104        Tprime(str, look_ahead, level + 1);
105        tab(level);
106        printf("Exit T'\n");
107        tab(level);
108        printf("-----\n");
109
110    }
111 }
112
113 int T(char *str, int *look_ahead, int level){
114     tab(level);
115     printf("-----\n");
116     tab(level);
117     printf("Enter F\n");
118     F(str, look_ahead, level + 1);
119     tab(level);
120     printf("Exit F\n");
121     tab(level);
122     printf("-----\n");
123
124     tab(level);
125     printf("-----\n");
126     tab(level);
127     printf("Enter T'\n");
128     Tprime(str, look_ahead, level + 1);
129     tab(level);
130     printf("Exit T'\n");
131     tab(level);
132     printf("-----\n");
133 }
134 int Eprime(char *str, int *look_ahead, int level){
135
136     if(str[*look_ahead] == '+'){
137         tab(level);
138         printf("E': + matched\n");

```

```

139         (*look_ahead)++;
140
141         tab(level);
142         printf("-----\n");
143         tab(level);
144         printf("Enter T\n");
145         T(str, look_ahead, level + 1);
146         tab(level);
147         printf("Exit T\n");
148         tab(level);
149         printf("-----\n");
150
151         tab(level);
152         printf("-----\n");
153         tab(level);
154         printf("Enter E'\n");
155         Eprime(str, look_ahead, level + 1);
156         tab(level);
157         printf("Exit E'\n");
158         tab(level);
159         printf("-----\n");
160     }
161     else if(str[*look_ahead] == '-'){
162         tab(level);
163         printf("E': - matched\n");
164         (*look_ahead)++;
165
166         tab(level);
167         printf("-----\n");
168         tab(level);
169         printf("Enter T\n");
170         T(str, look_ahead, level + 1);
171         tab(level);
172         printf("Exit T\n");
173         tab(level);
174         printf("-----\n");
175
176         tab(level);
177         printf("-----\n");
178         tab(level);
179         printf("Enter E'\n");
180         Eprime(str, look_ahead, level + 1);
181         tab(level);
182         printf("Exit E'\n");
183         tab(level);

```

```

184         printf("-----\n");
185     }
186 }
187
188 int E(char *str, int *look_ahead, int level){
189     tab(level);
190     printf("-----\n");
191     tab(level);
192     printf("Enter T\n");
193     T(str, look_ahead, level+1);
194     tab(level);
195     printf("Exit T\n");
196     tab(level);
197     printf("-----\n");
198
199     tab(level);
200     printf("-----\n");
201     tab(level);
202     printf("Enter E'\n");
203     Eprime(str, look_ahead, level+1);
204     tab(level);
205     printf("Exit E'\n");
206     tab(level);
207     printf("-----\n");
208 }

```

Output:

Success scenario:

```
Enter string to parse: i+i/i*i-i
-----
Enter E
-----
Enter T
-----
Enter F
      F: i matched
Exit F
-----
Enter T'
Exit T'
-----
Exit T
-----
Enter E'
      E': + matched
-----
Enter T
-----
Enter F
      F: i matched
Exit F
-----
Enter T'
      T': / matched
-----
Enter F
      F: i matched
Exit F
-----
Enter T'
      T': * matched
-----
```

```

Enter F
    F: i matched
Exit F
-----
Enter T'
Exit T'
-----
Exit T'
-----
Exit T
-----
Enter E'
    E': - matched
    Enter T
        Enter F
            F: i matched
        Exit F
        -----
        Enter T'
        Exit T'
        -----
    Exit T
    -----
    Enter E'
    Exit E'
    -----
Exit E'
-----
Exit E'
-----

```

```

Exit E
-----
Success%

```


Failure scenario:

```
Enter string to parse: i+i/(i)i
-----
Enter E
-----
Enter T
-----
Enter F
F: i matched
Exit F
-----
Enter T'
Exit T'
-----
Exit T
-----
Enter E'
E': + matched
-----
Enter T
-----
Enter F
F: i matched
Exit F
-----
Enter T'
T': / matched
-----
Enter F
F: ( matched
-----
Enter T
-----
Enter F
F: i matched
Exit F
-----
```

```

-----
Enter T'
Exit T'
-----
Exit T
-----
Enter E'
Exit E'
-----
F: ) matched
Exit F
-----
Enter T'
Exit T'
-----
Exit T'
-----
Exit T
-----
Enter E'
Exit E'
-----
Exit E'
-----
Exit E
-----
Failure: i at position 7 not expected.

```

Learning Outcomes:

- Understood the basic working of recursive descent parser.
 - Learnt how to use left-recursion-eliminated grammar to write code for recursive descent parser.
-