
A1-Implementation of Lexical Analyzer using C

Sneha Sriram Kannan 185001157

03-02-2021

1 Code

```
1 #include <stdio.h>
2
3 #include <string.h>
4
5 #include <ctype.h>
6
7 int substr(char str1[], char str2[]);
8 int checkFunction(char str[]);
9 int checkKeyword(char str[]);
10 int doubleop(char a, char b);
11 int doublelogicalop(char a, char b);
12 int LexicalAnalyzer();
13
14 int main() {
15     LexicalAnalyzer();
16     return 0;
17 }
18
19 int LexicalAnalyzer() {
20     char file[10][128];
21
22     FILE * fd = fopen("code.txt", "r");
23     int i = 0, doubleopIndex;
24     //reading code from a file and storing in an array
25     while (fgets(file[i], sizeof(file[i]), fd))
26         i++;
27     int multi = 0;
28
29     printf("=====\n");
30     printf("Code to analyze:\n");
31     printf("=====\n");
32     for (int j = 0; j < i; j++) {
33         printf("%s", file[j]);
34     }
35     printf("\n\n");
36     printf("=====\n");
37     printf("Output of Lexical Analyzer\n");
38     printf("=====\n");
39     //traversing through rows of the file
40     for (int j = 0; j < i; j++) {
41         if (checkFunction(file[j]) == 1) //if it is a function, FC is the only output
42         {
43             printf("FC\n");
44             continue;
45         }
46         for (int k = 0; k < strlen(file[j]); k++) //traversing through characters in a row of
            the file
```

```

47 {
48     //checking for end of a multiline comment
49     if (multi == 1) {
50         if ((strlen(file[j]) - 1) && file[j][k] == '*' && file[j][k + 1] == '/') {
51             printf("ENDMULTILINECOMMENT ");
52             multi = 0;
53             break;
54         }
55     }
56     //not part of multi line comment
57     else {
58         //checking for start of comments
59         if (k != (strlen(file[j]) - 1) && file[j][k] == '/' && file[j][k + 1] == '/') {
60             printf("SINGLELINECOMMENT ");
61             break;
62         } else if ((strlen(file[j]) - 1) && file[j][k] == '/' && file[j][k + 1] == '*')
63     {
64         printf("MULTILINECOMMENT ");
65         multi = 1;
66         break;
67     }
68     //checking for operators
69     else if (file[j][k] == '+' || file[j][k] == '-' || file[j][k] == '*' || file[j][
k] == '/' || file[j][k] == '%')
70     printf("ARITHOP ");
71     else if (file[j][k] == '!')
72     printf("LOGICALOP ");
73     else if (k != (strlen(file[j]) - 1) && doublelogicalop(file[j][k], file[j][k +
1]) == 1)
74     printf("LOGICALOP ");
75     //checking for seperators
76     else if (file[j][k] == '{' || file[j][k] == '}' || file[j][k] == ';' || file[j][
k] == ',' || file[j][k] == ')') || file[j][k] == '('
77     printf("SP ");
78     //checking for operators
79     else if (k != (strlen(file[j]) - 1) && doubleop(file[j][k], file[j][k + 1]) ==
80     1) {
81         //checking if it is an operator with 2 symbols (<>, <= etc)
82         if (!(isalpha(file[j][k + 1]) || isdigit(file[j][k + 1])))
83             k++;
84         continue;
85     } else if (file[j][k] == '=')
86     printf("ASSIGN ");
87     //searching for numbers
88     else if (isdigit(file[j][k]) && k != (strlen(file[j]) - 1)) {
89         while ((k + 1) != (strlen(file[j]) - 1)) {
90             if (isdigit(file[j][k + 1]))
91                 k++;
92             else if (file[j][k + 1] == '.')
93                 k++;
94             else
95                 break;
96         }
97         printf("NUMCONST ");
98     }
99     //space
100     else if (file[j][k] == ' ')
101         continue;
102     else if (file[j][k] == '\\') {
103         while (k != (strlen(file[j]) - 1)) {
104             k++;
105             if (file[j][k] == '\\') {
106                 printf("CHARCONST ");
107                 break;
108             }
109         }
110     }
111     } else if (file[j][k] == '\\') {
112         while (k != (strlen(file[j]) - 1)) {
113

```

```

114         k++;
115         if (file[j][k] == '\\') {
116             printf("STRCONST ");
117             break;
118         }
119     }
120 }
121 //extracting a string checking for keywords and id
122 else if (isalpha(file[j][k])) {
123     char substring[200];
124     int subIndex = 0;
125     substring[subIndex++] = file[j][k];
126     while ((k + 1) != (strlen(file[j]) - 1)) {
127         if (isalpha(file[j][k + 1])) {
128             substring[subIndex++] = file[j][k + 1];
129             k++;
130         } else
131             break;
132     }
133     substring[subIndex++] = '\\n';
134     substring[subIndex] = '\\0';
135     if (checkKeyword(substring) == 1)
136         printf("KW ");
137     else
138         printf("ID ");
139 }
140 }
141 }
142 printf("\\n");
143 }
144 return 0;
145 }
146
147 int doublelogicalop(char a, char b) //checks if ab is a logical operator
148 {
149     if ((a == b && b == '&') || (a == b && b == '|'))
150         return 1;
151     return 0;
152 }
153
154 int doubleop(char a, char b) //checks if ab is a relational operator
155 {
156     if (a == '<') {
157         if (b == '>')
158             printf("NE ");
159         else if (b == '=')
160             printf("LE ");
161         else
162             printf("LT ");
163         return 1;
164     }
165     if (a == '>') {
166         if (b == '=')
167             printf("GE ");
168         else
169             printf("GT ");
170         return 1;
171     }
172     if (a == '=' && b == '=') {
173         printf("EQ ");
174         return 1;
175     }
176     return 0;
177 }
178
179 int checkFunction(char str[]) //checks if str is a function
180 {
181     int i = 0;
182     int open = 0, close = 0;
183     char funcname[200];
184     int subIndex = 0;
185     while (!isalpha(str[i]))

```

```

186     i++;
187     while (i < strlen(str)) {
188         if (str[i] == ' ')
189             i++;
190         if (isalpha(str[i]))
191             funcname[subIndex++] = str[i++];
192         else if (str[i] == '(') {
193             open = 1;
194             i++;
195             break;
196         } else
197             return 0;
198     }
199     funcname[subIndex++] = '\n';
200     funcname[subIndex] = '\0';
201     if (checkKeyword(funcname))
202         return 0;
203     while (i < strlen(str) && open == 1) {
204         if (str[i++] == ')')
205             return 1;
206     }
207     return 0;
208 }
209
210 int checkKeyword(char str[]) //checks if str is a keyword
211 {
212     FILE * fd = fopen("keywords.txt", "r");
213     char filestr[20];
214     while (fgets(filestr, 60, fd) != NULL) {
215         if (strcmp(str, filestr) == 0) {
216             return 1;
217         }
218     }
219     return 0;
220 }
221
222 int substr(char str1[], char str2[]) //checks if str1 is a substring of str2
223 {
224     int i, j = 0;
225     while (i < strlen(str1) && j < strlen(str2)) {
226         if (str1[i] == str2[j]) {
227             i++;
228             j++;
229         } else {
230             j++;
231         }
232     }
233     if (i == strlen(str1))
234         return 1;
235     else
236         return 0;
237 }

```

2 Output Screenshot

```
> clang-7 -pthread -lm -o main main.c
> ./main
=====
Code to analyze:
=====
/* This is a multi
line
comment*/
hello(){
    printf("Hello world");
}
main()
{
    int a=10;b=20;
    char c='a';
    float x=2.34;
    hello();
    if(a>b) //check
    printf("a is greater");
    else
    printf("b is greater");
}

=====
Output of Lexical Analyzer
=====
MULTILINECOMMENT

ENDMULTILINECOMMENT
FC
FC
SP
FC
SP
KW ID ASSIGN NUMCONST SP ID ASSIGN NUMCONST SP
KW ID ASSIGN CHARCONST SP
KW ID ASSIGN NUMCONST SP
FC
KW SP ID GT ID SP SINGLELINECOMMENT
FC
KW
FC
SP
```

Figure 1: Lexical Analyzer Output