

Implementation of Lexical analyzers for a C program

Name- Husna Qasim

Reg. No- RA1911031010138

Sec- CSE-IT[L2]

AIM: To write a program for lexical analyzer which takes a C file as the input file and converts the content as count of tokens.

ALGORITHM:

1. Read the C program file
2. Create lists of keywords, constants, operators, special symbols
3. Read each line in the file, split the words in each line
4. If the word is in any of the above lists, append it to a separate list and repeat this step till the last line of the C program
5. Print the tokens and their respective counts in the C program

CODE:

```

1 #include <stdbool.h>
2 #include <stdio.h>
3 #include <string.h>
4 #include <stdlib.h>
5 int delimiter = 0;
6 int oper = 0;
7 int identi = 0;
8 int key = 0;
9 int INTEGER = 0;
10 int realnumber = 0;
11 int ivi = 0;
12 bool isDelimiter(char ch)
13 {
14     if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
15         ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
16         ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
17         ch == '[' || ch == ']' || ch == '{' || ch == '}{')
18         return (true);
19     delimiter++;
20 }
21 return (false);
22 }
23 bool isOperator(char ch)
24 {
25     if (ch == '+' || ch == '-' || ch == '*' ||
26         ch == '/' || ch == '>' || ch == '<' ||
27         ch == '='){
28         return (true);
29         oper++;
30     }
31     return (false);
32 }
33 bool validIdentifier(char* str)
34 {
35     if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||

```

```

35     if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||
36         str[0] == '3' || str[0] == '4' || str[0] == '5' ||
37         str[0] == '6' || str[0] == '7' || str[0] == '8' ||
38         str[0] == '9' || isDelimiter(str[0]) == true)
39         return (false);
40     return (true);
41 }
42 bool isKeyword(char* str)
43 {
44     if (!strcmp(str, "if") || !strcmp(str, "else") ||
45         !strcmp(str, "while") || !strcmp(str, "do") ||
46         !strcmp(str, "break") ||
47         !strcmp(str, "continue") || !strcmp(str, "int")
48         || !strcmp(str, "double") || !strcmp(str, "float")
49         || !strcmp(str, "return") || !strcmp(str, "char")
50         || !strcmp(str, "case") || !strcmp(str, "char")
51         || !strcmp(str, "sizeof") || !strcmp(str, "long")
52         || !strcmp(str, "short") || !strcmp(str, "typedef")
53         || !strcmp(str, "switch") || !strcmp(str, "unsigned")
54         || !strcmp(str, "void") || !strcmp(str, "static")
55         || !strcmp(str, "struct") || !strcmp(str, "goto"))
56         return (true);
57     return (false);
58 }
59 bool isInteger(char* str)
60 {
61     int i, len = strlen(str);
62
63     if (len == 0)
64         return (false);
65     for (i = 0; i < len; i++) {
66         if (str[i] != '0' && str[i] != '1' && str[i] != '2'
67             && str[i] != '3' && str[i] != '4' && str[i] != '5'
68             && str[i] != '6' && str[i] != '7' && str[i] != '8'
69             && str[i] != '9' || (str[i] == '-' && i > 0))
70             return (false);

```

```

67         && str[i] != '5' && str[i] != '4' && str[i] != '3'
68         && str[i] != '6' && str[i] != '7' && str[i] != '8'
69         && str[i] != '9' || (str[i] == '-' && i > 0))
70         return (false);
71     }
72     return (true);
73 }
74
75 bool isRealNumber(char* str)
76 {
77     int i, len = strlen(str);
78     bool hasDecimal = false;
79
80     if (len == 0)
81         return (false);
82     for (i = 0; i < len; i++) {
83         if (str[i] != '0' && str[i] != '1' && str[i] != '2'
84             && str[i] != '3' && str[i] != '4' && str[i] != '5'
85             && str[i] != '6' && str[i] != '7' && str[i] != '8'
86             && str[i] != '9' && str[i] != '.' ||
87             (str[i] == '-' && i > 0))
88             return (false);
89         if (str[i] == '.')
90             hasDecimal = true;
91     }
92     return (hasDecimal);
93 }
94
95 char* subString(char* str, int left, int right)
96 {
97     int i;
98     char* subStr = (char*)malloc(
99         sizeof(char) * (right - left + 2));
100     for (i = left; i <= right; i++)
101         subStr[i - left] = str[i];
102     subStr[right - left + 1] = '\0';

```

```

100     for (i = left; i <= right; i++)
101         subStr[i - left] = str[i];
102     subStr[right - left + 1] = '\0';
103     return (subStr);
104 }
105 void parse(char* str)
106 {
107     int left = 0, right = 0;
108     int len = strlen(str);
109
110     while (right <= len && left <= right) {
111         if (isDelimiter(str[right]) == false)
112             right++;
113
114         if (isDelimiter(str[right]) == true && left == right) {
115             if (isOperator(str[right]) == true){
116                 printf("%c IS AN OPERATOR\n", str[right]);
117                 oper++;
118             }
119
120             right++;
121             left = right;
122         } else if (isDelimiter(str[right]) == true && left != right
123             || (right == len && left != right)) {
124             char* subStr = substring(str, left, right - 1);
125
126             if (isKeyword(subStr) == true){
127                 printf("%s IS A KEYWORD\n", subStr);
128                 key++;
129             }
130
131             else if (isInteger(subStr) == true){
132                 printf("%s IS AN INTEGER\n", subStr);
133                 INTEGER++;
134             }
135

```

```

130
131         else if (isInteger(subStr) == true){
132             printf("%s IS AN INTEGER\n", subStr);
133             INTEGER++;
134         }
135
136         else if (isRealNumber(subStr) == true){
137             printf("%s IS A REAL NUMBER\n", subStr);
138             realnumber++;
139         }
140
141         else if (validIdentifier(subStr) == true
142             && isDelimiter(str[right - 1]) == false){
143             identi++;
144             printf("%s IS A VALID IDENTIFIER\n", subStr);
145         }
146
147         else if (validIdentifier(subStr) == false
148             && isDelimiter(str[right - 1]) == false){
149             printf("%s IS NOT A VALID IDENTIFIER\n", subStr);
150             ivi++;
151         }
152         left = right;
153     }
154 }
155 return;
156 }
157 int main()
158 {
159     char str[100] = "int x = 5 + y; ";
160     parse(str);
161     printf("number of delimiter = %d \n", delimiter);
162     printf("number of operator = %d \n", oper);
163     printf("number of identifier = %d \n", identi);
164     printf("number of keyword = %d \n", key);
165     printf("number of integer = %d \n", INTEGER);
166     printf("number of real number = %d \n", realnumber);
167     printf("number of valid identifier = %d \n", identi);
168     printf("number of invalid identifier = %d \n", ivi);
169 }

```

```

135
136     else if (isRealNumber(subStr) == true){
137         printf("%s' IS A REAL NUMBER\n", subStr);
138         realnumber++;
139     }
140
141     else if (validIdentifier(subStr) == true
142             && isDelimiter(str[right - 1]) == false){
143         identi++;
144         printf("%s' IS A VALID IDENTIFIER\n", subStr);
145     }
146
147     else if (validIdentifier(subStr) == false
148             && isDelimiter(str[right - 1]) == false){
149         printf("%s' IS NOT A VALID IDENTIFIER\n", subStr);
150         ivi++;
151     }
152     left = right;
153 }
154 }
155 return;
156 }
157 int main()
158 {
159     char str[100] = "int x = 5 + y; ";
160     parse(str);
161     printf("number of delimiter = %d \n", delimiter);
162     printf("number of operator = %d \n", oper);
163     printf("number of identifier = %d \n", identi);
164     printf("number of KEYWORD = %d \n", key);
165     printf("number of INTEGER = %d \n", INTEGER);
166     printf("number of realnumber = %d \n", realnumber);
167     printf("number of invalid identifier = %d \n", ivi);
168     return (0);
169 }
170

```

OUTPUT—

```
'int' IS A KEYWORD
'x' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'5' IS AN INTEGER
'+' IS AN OPERATOR
'y' IS A VALID IDENTIFIER
number of delimiter = 0
number of operator = 2
number of identifier = 2
number of KEYWORD = 1
number of INTEGER = 1
number of realnumber = 0
number of invalid identifier = 0

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

RESULT-

The given program has been successfully executed.