

Assignment 2

On

Lexical Analysis

CSE-0302 Summer 2021

Rebecca Sultana
Dept. of CSE
State University of Bangladesh (SUB)
Dhaka, Bangladesh
rebecca.sultana.riya@gmail.com

Abstract—To write a program that reads any simple program as source and separates out the valid tokens from the source program.

I. INTRODUCTION

As it is known that Lexical Analysis is the first phase of compiler also known as scanner. It converts the input program into a sequence of Tokens.

II. AIMS

A C program consists of various tokens and a token is either a keyword, identifiers, operators, separators, parenthesis, numbers, symbol etc..

- 1) Keywords: Examples- for, while, if etc.
- 2) Identifier Examples- Variable name, function name etc.
- 3) Operators: Examples- '+', '++', '-' etc.
- 4) Separators: Examples- ',', ' ', ';' etc

III. OUTPUT

Type the line of code below:

unkn 100.o5]

'unkn' Is a valid identifier

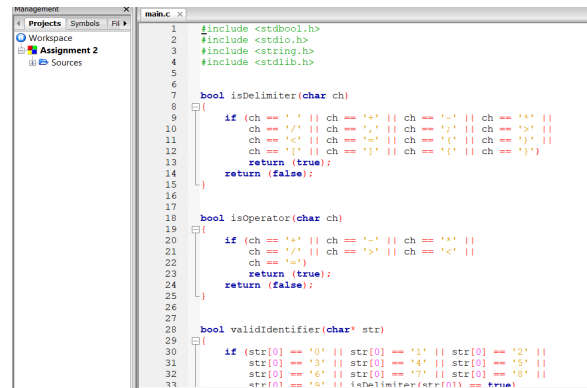
'100.o5 Is not a valid identifier

IV. CONCLUSION

C program to detect tokens in a C program.

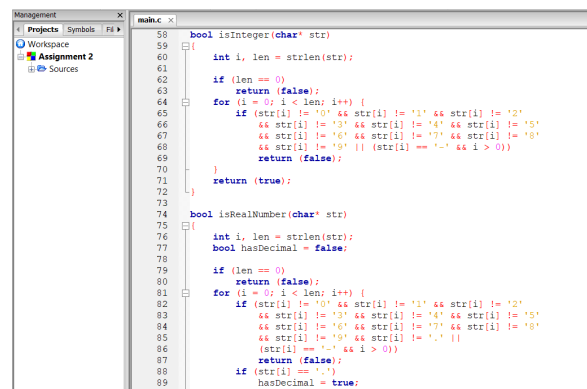
V. CODE

Below is a C program to print all the keywords, literals, valid identifiers, invalid identifiers, integer number, real number in a given C program:



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

Fig. 1.



```
58 bool isInteger(char* str)
59 {
60     int i, len = strlen(str);
61     if (len == 0)
62         return (false);
63     for (i = 0; i < len; i++) {
64         if (str[i] != '0' && str[i] != '1' && str[i] != '2' &&
65             str[i] != '3' && str[i] != '4' && str[i] != '5' &&
66             str[i] != '6' && str[i] != '7' && str[i] != '8' &&
67             str[i] != '9' || (str[i] == '-' && i > 0))
68             return (false);
69     }
70     return (true);
71 }
72
73 bool isRealNumber(char* str)
74 {
75     int i, len = strlen(str);
76     bool hasDecimal = false;
77     if (len == 0)
78         return (false);
79     for (i = 0; i < len; i++) {
80         if (str[i] != '0' && str[i] != '1' && str[i] != '2' &&
81             str[i] != '3' && str[i] != '4' && str[i] != '5' &&
82             str[i] != '6' && str[i] != '7' && str[i] != '8' &&
83             str[i] != '9' && str[i] != '.' ||
84             (str[i] == '-' && i > 0))
85             return (false);
86         if (str[i] == '.' && hasDecimal == true)
87             return (false);
88         if (str[i] == '.')
89             hasDecimal = true;
90     }
91 }
```

Fig. 2.

```

58 bool isInteger(char* str)
59 {
60     int i, len = strlen(str);
61     if (len == 0)
62         return (false);
63     for (i = 0; i < len; i++) {
64         if (str[i] != '0' && str[i] != '1' && str[i] != '2'
65             && str[i] != '3' && str[i] != '4' && str[i] != '5'
66             && str[i] != '6' && str[i] != '7' && str[i] != '8'
67             && str[i] != '9' || (str[i] == '-' && i > 0))
68             return (false);
69     }
70     return (true);
71 }
72
73 bool isRealNumber(char* str)
74 {
75     int i, len = strlen(str);
76     bool hasDecimal = false;
77     if (len == 0)
78         return (false);
79     for (i = 0; i < len; i++) {
80         if (str[i] != '0' && str[i] != '1' && str[i] != '2'
81             && str[i] != '3' && str[i] != '4' && str[i] != '5'
82             && str[i] != '6' && str[i] != '7' && str[i] != '8'
83             && str[i] != '9' && str[i] != '.' ||
84             (str[i] == '-' && i > 0))
85             return (false);
86         if (str[i] == '-' && i > 0)
87             hasDecimal = true;
88     }
89     return (true);
90 }

```

Fig. 3.

```

94 // Extracts the SUBSTRING
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
101     for (i = left; i <= right; i++)
102         subStr[i - left] = str[i];
103     subStr[right - left + 1] = '\0';
104     return (subStr);
105 }
106
107 // Parsing the input STRING.
108 void parse(char* str)
109 {
110     int left = 0, right = 0;
111     int len = strlen(str);
112
113     while (right <= len && left <= right) {
114         if (isDelimiter(str[right]) == false)
115             right++;
116
117         if (isDelimiter(str[right]) == true && left == right) {
118             if (isOperator(str[right]) == true)
119                 printf("%c is an operator\n", str[right]);
120
121             right++;
122             left = right;
123         }
124     }
125 }

```

Fig. 4.

```

126 else if (isDelimiter(str[right]) == true && left != right
127     || (right == len && left != right)) {
128     char* subStr = subString(str, left, right - 1);
129     printf("%s is a keyword\n", subStr);
130
131     else if (isInteger(subStr) == true)
132         printf("%s is an integer number\n", subStr);
133
134     else if (isRealNumber(subStr) == true)
135         printf("%s is a real number\n", subStr);
136
137     else if (validIdentifier(subStr) == true
138         && isDelimiter(str[right - 1]) == false)
139         printf("%s is a valid identifier\n", subStr);
140
141     else if (validIdentifier(subStr) == false
142         && isDelimiter(str[right - 1]) == false)
143         printf("%s is not a valid identifier\n", subStr);
144     left = right;
145 }
146 }
147 return;
148 }
149
150 int main()
151 {
152     //length of string is 100
153     char str[200];
154     printf("Type the line of code below: \n");
155     scanf("%[^\n]", str);
156     parse(str);
157     return (0);
158 }
159
160
161
162
163

```

Fig. 5.