

18CSC304J/ Compiler Design

Submitted By:- ANANNYA P. NEOG (RA1911003010367)

Exp-1: Implementation of Lexical Analyzer

Aim:- Implement the code for Lexical Analyzer using C or C++ or Python or Java

Code:-

```
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// Checks and returns 'true' if the character is a DELIMITER.
bool isDelimiter(char ch)
{
    if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
        ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
        ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
        ch == '[' || ch == ']' || ch == '{' || ch == '}')
        return (true);
    return (false);
}

// Checks and returns 'true' if the character is an OPERATOR.
bool isOperator(char ch)
{
    if (ch == '+' || ch == '-' || ch == '*' ||
        ch == '/' || ch == '>' || ch == '<' ||
        ch == '=')
        return (true);
    return (false);
}

// Checks and returns 'true' if the string is a VALID IDENTIFIER.
bool validIdentifier(char* str)
{
    if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||
        str[0] == '3' || str[0] == '4' || str[0] == '5' ||
        str[0] == '6' || str[0] == '7' || str[0] == '8' ||
        str[0] == '9' || isDelimiter(str[0]) == true)
        return (false);
    return (true);
}

// Checks and returns 'true' if the string is a KEYWORD.
bool isKeyword(char* str)
{

```

```

if (!strcmp(str, "if") || !strcmp(str, "else") ||
    !strcmp(str, "while") || !strcmp(str, "do") ||
    !strcmp(str, "break") ||
    !strcmp(str, "continue") || !strcmp(str, "int")
    || !strcmp(str, "double") || !strcmp(str, "float")
    || !strcmp(str, "return") || !strcmp(str, "char")
    || !strcmp(str, "case") || !strcmp(str, "char")
    || !strcmp(str, "sizeof") || !strcmp(str, "long")
    || !strcmp(str, "short") || !strcmp(str, "typedef")
    || !strcmp(str, "switch") || !strcmp(str, "unsigned")
    || !strcmp(str, "void") || !strcmp(str, "static")
    || !strcmp(str, "struct") || !strcmp(str, "goto"))
    return (true);
return (false);
}

```

// Checks and returns 'true' if the string is an INTEGER.

```

bool isInteger(char* str)
{
    int i, len = strlen(str);

    if (len == 0)
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] != '5'
            && str[i] != '6' && str[i] != '7' && str[i] != '8'
            && str[i] != '9' || (str[i] == '-' && i > 0))
            return (false);
    }
    return (true);
}

```

// Checks and returns 'true' if the string is a REAL NUMBER.

```

bool isRealNumber(char* str)
{
    int i, len = strlen(str);
    bool hasDecimal = false;

    if (len == 0)
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] != '5'
            && str[i] != '6' && str[i] != '7' && str[i] != '8'
            && str[i] != '9' && str[i] != '.' ||
            (str[i] == '-' && i > 0))
            return (false);
        if (str[i] == '.')
            hasDecimal = true;
    }
}

```

```

    }
    return (hasDecimal);
}

```

// Extracts the SUBSTRING.

```

char* subString(char* str, int left, int right)
{
    int i;
    char* subStr = (char*)malloc(
        sizeof(char) * (right - left + 2));

    for (i = left; i <= right; i++)
        subStr[i - left] = str[i];
    subStr[right - left + 1] = '\0';
    return (subStr);
}

```

// Parsing the input STRING.

```

void parse(char* str)
{
    int left = 0, right = 0;
    int len = strlen(str);

    while (right <= len && left <= right) {
        if (isDelimiter(str[right]) == false)
            right++;

        if (isDelimiter(str[right]) == true && left == right) {
            if (isOperator(str[right]) == true)
                printf("%c' IS AN OPERATOR\n", str[right]);

            right++;
            left = right;
        } else if (isDelimiter(str[right]) == true && left != right
            || (right == len && left != right)) {
            char* subStr = subString(str, left, right - 1);

            if (isKeyword(subStr) == true)
                printf("%s' IS A KEYWORD\n", subStr);

            else if (isInteger(subStr) == true)
                printf("%s' IS AN INTEGER\n", subStr);

            else if (isRealNumber(subStr) == true)
                printf("%s' IS A REAL NUMBER\n", subStr);

            else if (validIdentifier(subStr) == true
                && isDelimiter(str[right - 1]) == false)
                printf("%s' IS A VALID IDENTIFIER\n", subStr);

            else if (validIdentifier(subStr) == false
                && isDelimiter(str[right - 1]) == false)

```

```

        printf("%s' IS NOT A VALID IDENTIFIER\n", subStr);
        left = right;
    }
}
return;
}

// DRIVER FUNCTION
int main()
{
    // maximum length of string is 100 here
    char str[100] = "int a = b + 1 * c - 2.2; ";

    parse(str); // calling the parse function

    return (0);
}

```

The screenshot shows a C compiler IDE with a dark theme. The top toolbar includes buttons for Run, Debug, Stop, Share, Save, and Beautify. The language is set to C. The code in the editor is as follows:

```

1 #include <stdbool.h>
2 #include <stdio.h>
3 #include <string.h>
4 #include <stdlib.h>
5
6 // Checks and returns 'true' if the character is a DELIMITER.
7 bool isDelimiter(char ch)
8 {
9     if (ch == '+' || ch == '-' || ch == '*' || ch == '/' ||
10         ch == '<' || ch == '>' || ch == '=' || ch == '[' || ch == ']' ||
11         ch == '{' || ch == '}')
12         return (true);
13     return (false);
14 }
15
16 // Checks and returns 'true' if the character is an OPERATOR.
17 bool isOperator(char ch)
18 {
19     if (ch == '+' || ch == '-' || ch == '*' ||
20         ch == '/' || ch == '>' || ch == '<' ||
21         ch == '=')
22         return (true);
23     return (false);
24 }
25
26 // Checks and returns 'true' if the string is a VALID IDENTIFIER.
27 bool validIdentifier(char* str)
28 {
29     if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||
30         str[0] == '3' || str[0] == '4' || str[0] == '5' ||
31         str[0] == '6' || str[0] == '7' || str[0] == '8' ||
32         str[0] == '9' || isDelimiter(str[0]) == true)
33         return (false);
34     return (true);
35 }
36

```

```
mpiler
main.c
37
38 // Checks and returns 'true' if the string is a KEYWORD.
39 bool isKeyword(char* str)
40 {
41     if (!strcmp(str, "if") || !strcmp(str, "else") ||
42         !strcmp(str, "while") || !strcmp(str, "do") ||
43         !strcmp(str, "break") ||
44         !strcmp(str, "continue") || !strcmp(str, "int")
45         || !strcmp(str, "double") || !strcmp(str, "float")
46         || !strcmp(str, "return") || !strcmp(str, "char")
47         || !strcmp(str, "case") || !strcmp(str, "char")
48         || !strcmp(str, "sizeof") || !strcmp(str, "long")
49         || !strcmp(str, "short") || !strcmp(str, "typedef")
50         || !strcmp(str, "switch") || !strcmp(str, "unsigned")
51         || !strcmp(str, "void") || !strcmp(str, "static")
52         || !strcmp(str, "struct") || !strcmp(str, "goto"))
53         return (true);
54     return (false);
55 }
56
57 // Checks and returns 'true' if the string is an INTEGER.
58 bool isInteger(char* str)
59 {
60     int i, len = strlen(str);
61
62     if (len == 0)
63         return (false);
64     for (i = 0; i < len; i++) {
65         if (str[i] != '0' && str[i] != '1' && str[i] != '2'
66             && str[i] != '3' && str[i] != '4' && str[i] != '5'
67             && str[i] != '6' && str[i] != '7' && str[i] != '8'
68             && str[i] != '9' || (str[i] == '-' && i > 0))
69             return (false);
70     }
71     return (true);
72 }
73
```

```
mpiler
main.c
74 // Checks and returns 'true' if the string is a REAL NUMBER.
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 // Extracts the SUBSTRING.
96 char* subString(char* str, int left, int right)
97 {
98     int i;
99     char* subStr = (char*)malloc(
100         sizeof(char) * (right - left + 2));
101
102     for (i = left; i <= right; i++)
103         subStr[i - left] = str[i];
104     subStr[right - left + 1] = '\0';
105     return (subStr);
106 }
107
```

```
mpiler
main.c
107
108 // Parsing the input STRING.
109 void parse(char* str)
110 {
111     int left = 0, right = 0;
112     int len = strlen(str);
113
114     while (right <= len && left <= right) {
115         if (isDelimiter(str[right]) == false)
116             right++;
117
118         if (isDelimiter(str[right]) == true && left == right) {
119             if (isOperator(str[right]) == true)
120                 printf("%c' IS AN OPERATOR\n", str[right]);
121
122             right++;
123             left = right;
124         } else if (isDelimiter(str[right]) == true && left != right)
125             || (right == len && left != right)) {
126             char* subStr = subString(str, left, right - 1);
127
128             if (isKeyword(subStr) == true)
129                 printf("%s' IS A KEYWORD\n", subStr);
130
131             else if (isInteger(subStr) == true)
132                 printf("%s' IS AN INTEGER\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 // DRIVER FUNCTION
151 int main()
152 {
153     // maximum length of string is 100 here
154     char str[100] = "int a = b + 1 * c - 2.2; ";
155
156     parse(str); // calling the parse function
157
158     return (0);
159 }
```

Output:-

```
input
'int' IS A KEYWORD
'a' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'b' IS A VALID IDENTIFIER
'+' IS AN OPERATOR
'1' IS AN INTEGER
'*' IS AN OPERATOR
'c' IS A VALID IDENTIFIER
'-' IS AN OPERATOR
'2.2' IS A REAL NUMBER

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