

EXP 1

LEXICAL ANALYZER

AIM : To implement the Lexical Analyzer using C language

PROCEDURE:

- Include necessary modules and header files, `stdio.h`, `stdlib.h`, `string.h`, `stdbool.h`
- Define function `isValidKeyword` that will check for the valid keywords in the string argument passed to it.
- Define another function, `isValidInteger` that will check for integers in the string argument that is passed with the function call, returns true if the string is integer
- Define function `isRealNumber` with `bool` return type that checks if the string argument is a real number and then return true.
- Define `isValidOperator` with `bool` as the return type that returns true when the string argument is an operator.
- Define `isValidDelimiter` with `Boolean` as the return type that returns true when the string argument passes is a delimiter (, - / < >)
- Define a function named `substring` that takes a string, a starting index and an ending index and returns part of the string in between these indices.
- Define `detectToken` function that takes a string as argument and splits it into multiple parts and use the above created functions to find and print tokens.
- Call the `detectToken` function in the main function and pass it the string (the line of the code) that needs to be lexically analysed.

CODE:

```
#include <stdbool.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

bool isValidDelimiter(char ch) {

    if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||

        ch == '/' || ch == ',' || ch == ';' || ch == '>' ||

        ch == '<' || ch == '=' || ch == '(' || ch == ')')

        return (true);

        return (false);

}

bool isValidOperator(char ch){

    if (ch == '+' || ch == '-' || ch == '*' ||

        ch == '/' || ch == '>' || ch == '<' ||

        ch == '=')

        return (true);

        return (false);

}

// Returns 'true' if the string is a VALID IDENTIFIER.

bool isValidIdentifier(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' || isValidDelimiter(str[0]) == true)

        return (false);

        return (true);

}
```

```

bool isValidKeyword(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);
}

bool isValidInteger(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);
}

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);
}

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);
}

void detectTokens(char* str) {
    int left = 0, right = 0;

    int length = strlen(str);

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

        if (isValidDelimiter(str[right]) == true && left == right) {
            if (isValidOperator(str[right]) == true)
                printf("Valid operator : '%c'\n", str[right]);

            right++;

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

            if (isValidKeyword(subStr) == true)
                printf("Valid keyword : '%s'\n", subStr);

            else if (isValidInteger(subStr) == true)

```

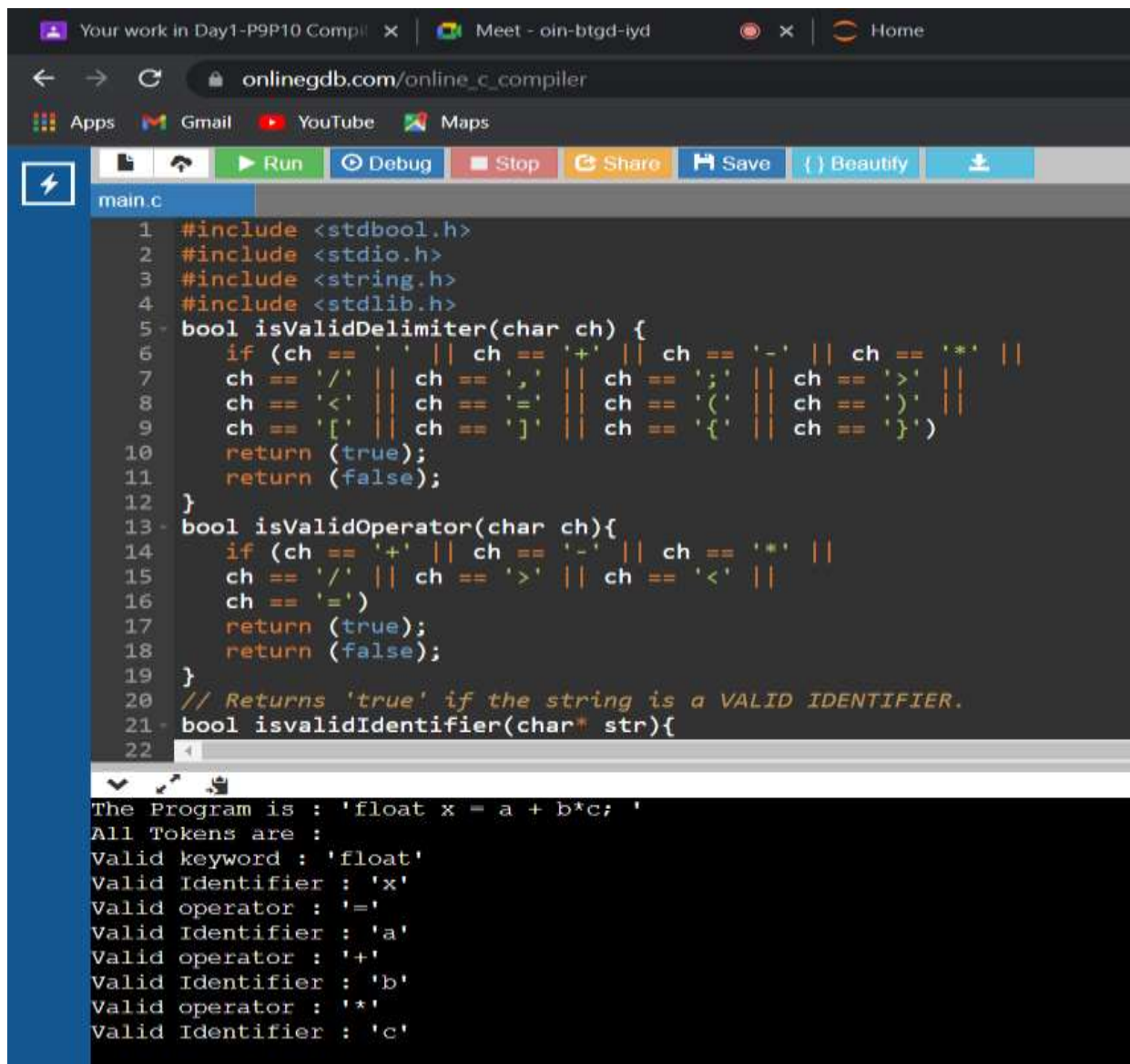
```

        printf("Valid Integer : '%s'\n", subStr);
    else if (isRealNumber(subStr) == true)
        printf("Real Number : '%s'\n", subStr);
    else if (isValidIdentifier(subStr) == true
        && isValidDelimiter(str[right - 1]) == false)
        printf("Valid Identifier : '%s'\n", subStr);
    else if (isValidIdentifier(subStr) == false
        && isValidDelimiter(str[right - 1]) == false)
        printf("Invalid Identifier : '%s'\n", subStr);
    left = right;
}
}
return;
}
int main(){
    char str[100] = "float x = a + b*c; ";
    printf("The Program is : '%s' \n", str);
    printf("All Tokens are : \n");
    detectTokens(str);
    return (0);
}

```

RESULT: The Lexical Analyzer problem has been implemented.

OUTPUT:



The screenshot shows a web browser window with the URL `onlinegdb.com/online_c_compiler`. The browser tabs include "Your work in Day1-P9P10 Compil...", "Meet - oin-btgd-lyd", and "Home". The browser's address bar and navigation buttons are visible. Below the browser window is a toolbar with buttons for "Run", "Debug", "Stop", "Share", "Save", "Beautify", and a download icon. The main editor area, titled "main.c", contains the following C code:

```
1 #include <stdbool.h>
2 #include <stdio.h>
3 #include <string.h>
4 #include <stdlib.h>
5 bool isValidDelimiter(char ch) {
6     if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
7         ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
8         ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
9         ch == '[' || ch == ']' || ch == '{' || ch == '}')
10        return (true);
11    return (false);
12 }
13 bool isValidOperator(char ch){
14     if (ch == '+' || ch == '-' || ch == '*' ||
15         ch == '/' || ch == '>' || ch == '<' ||
16         ch == '=')
17        return (true);
18    return (false);
19 }
20 // Returns 'true' if the string is a VALID IDENTIFIER.
21 bool isValidIdentifier(char* str){
22
```

Below the code editor, the output of the program is displayed in a black box with white text:

```
The Program is : 'float x = a + b*c; '
All Tokens are :
Valid keyword : 'float'
Valid Identifier : 'x'
Valid operator : '='
Valid Identifier : 'a'
Valid operator : '+'
Valid Identifier : 'b'
Valid operator : '*'
Valid Identifier : 'c'
```

RA1911003010387

CHITRALEKHA.CH