

**Ex No: 1**

**Date:**

## **IMPLEMENT CODE TO RECOGNIZE TOKENS IN C**

### **AIM:**

To implement the program to identify C keywords, identifiers, operators, end statements like [], {} using C tool.

### **ALGORITHM:**

1. Identify the basic tokens in c such as keywords, numbers, variables, etc.
2. Declare the required header files.
3. Get the input from the user as a string and it is passed to a function for processing.
4. The functions are written separately for each token and the result is returned in the form of bool either true or false to the main computation function.
5. Functions are issymbol() for checking basic symbols such as () etc , isoperator() to check for operators like +, -, \*, / , isidentifier() to check for variables like a,b, iskeyword() to check the 32 keywords like while etc., isInteger() to check for numbers in combinations of 0-9, isnumber() to check for digits and substring().
6. Declare a function detecttokens() that is used for string manipulation and Iteration then the result is returned from the functions to the main. If it's an invalid Identifier error must be printed.
7. Declare main function get the input from the user and pass to detecttokens() function.

### **PROGRAM:**

```
#include<stdio.h>
#include<string.h>
int main()
{
    char str[100]={0};
    printf("Enter the statement :");
    scanf("%s",&str);

    char str2[100];
    strcpy(str2,str);

    const char
    chara[100]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z',' ','=','+','-','*','/'};
```

```

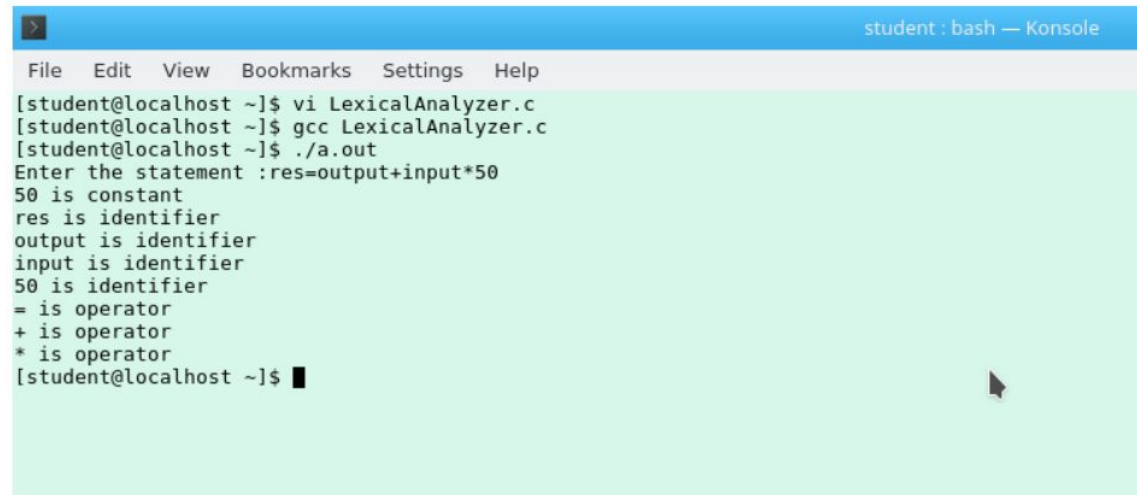
char *token3;
token3=strtok(str2,chara);
while(token3!=NULL)
{
printf("%s is constant\n",token3);
token3=strtok(NULL,chara);
}

const char deli[50]={' ','=','+','-','*','/'};
char *token;
token = strtok(str,deli);
while(token!=NULL)
{
printf("%s is identifier\n",token);
token=strtok(NULL,deli);
}
const char
alpha[100]={'a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z','A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z','0','1','2','3','4','5','6','7','8','9'};

char *token2;
token2=strtok(str2,alpha);
while(token2!=NULL)
{
printf("%s is operator\n",token2);
token2=strtok(NULL,alpha);
}

return 0;
}

```

**OUTPUT:**

The screenshot shows a terminal window titled "student : bash — Konsole". The terminal has a menu bar with "File", "Edit", "View", "Bookmarks", "Settings", and "Help". The command history shows the user editing "LexicalAnalyzer.c", compiling it with "gcc", and running the executable "a.out". The program prompts for a statement, and the user enters "res=output+input\*50". The program then outputs the tokens and their categories: "50 is constant", "res is identifier", "output is identifier", "input is identifier", "50 is identifier", "=", "+", and "\*".

```
student : bash — Konsole
File Edit View Bookmarks Settings Help
[student@localhost ~]$ vi LexicalAnalyzer.c
[student@localhost ~]$ gcc LexicalAnalyzer.c
[student@localhost ~]$ ./a.out
Enter the statement :res=output+input*50
50 is constant
res is identifier
output is identifier
input is identifier
50 is identifier
= is operator
+ is operator
* is operator
[student@localhost ~]$
```

**RESULT**