LEXICAL ANALYZER

AIM: TO IMPLEMENT LEXICAL ANALYZER IN C LANGUAGE TO DETECT keywords,operators,valid identifier,invalid identifier.

ALGORITHM:

```
·  Create a text file with source code in it.

· if(source file==NULL)

return “file cannot be opened”.

· while(!feof(f))

Read the  characters from source code.

Divides the given program into valid tokens.

If(!strcmp(word,key word)

printf("%s- key word\n",word);

else if(!strcmp(word,operator)

printf("%s- operator",word);

else

printf(“%s-identifier”,word);

· Remove white space characters from the given program.

· Remove comments.

· It generates lexical errors.
```

Source Code(C++):

Main.cpp: -

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

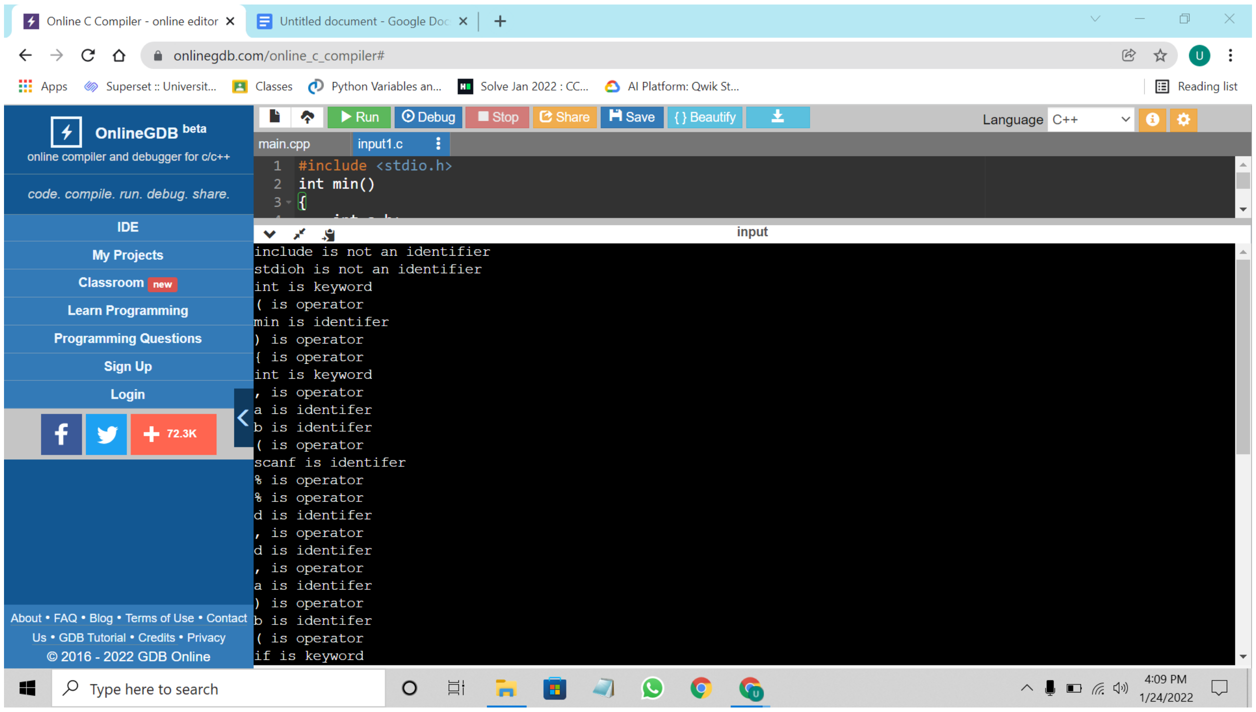
```
#include<bits/stdc++.h>
using namespace std;
int isKeyword(string buffer){
    char keywords[32][10] = {"auto", "break", "case", "char", "const", "continue", "default",
"do", "double", "else", "enum", "extern", "float", "for", "goto", "if", "int", "long", "register",
"return", "short", "signed", "sizeof", "static", "struct", "switch", "typedef", "union",
"unsigned", "void", "volatile", "while"};
    int i, flag = 0;
    for (i = 0; i < 32; ++i){
        if (keywords[i] == buffer){
            flag = 1;
            break;
        }
    }
    return flag;
}
bool isOperator(char ch){
    char operators[] = "+-*/%=(){},";
    for (int i = 0; i < 11; ++i){
        if (ch == operators[i]){
            return true;
        }
    }
    return false;
}
void Check(string &buffer){
    if(buffer.size() != 0){
        if (isKeyword(buffer) == 1){
            cout << buffer <<  " is keyword" << endl;
        }
        else{
            if(buffer == "include" || buffer =="stdioh" || buffer == "define" || buffer == ""){
                cout << buffer << " is not an identifier" << endl;
            }else{
                cout << buffer << " is identifer" << endl;
            }
        }
        buffer = "";
    }
}
```

```
int main()
{
    char ch;
    string buffer = "";
    FILE *fp;
    fp = fopen("input1.c", "r");
    if (fp == NULL){
        printf("error while opening the file\n");
        exit(0);
    }
    while ((ch = fgetc(fp)) != EOF){
        if(isOperator(ch)){
            cout << ch << " is operator" << endl;
            Check(buffer);
        }
        if(isalnum(ch)){
            buffer += ch;
        }
        else if ((ch == ' ' || ch == '\n') && (buffer.size() != 0)){
            Check(buffer);
        }
    }
    fclose(fp);
    return 0;
}
```

Input1.c:-

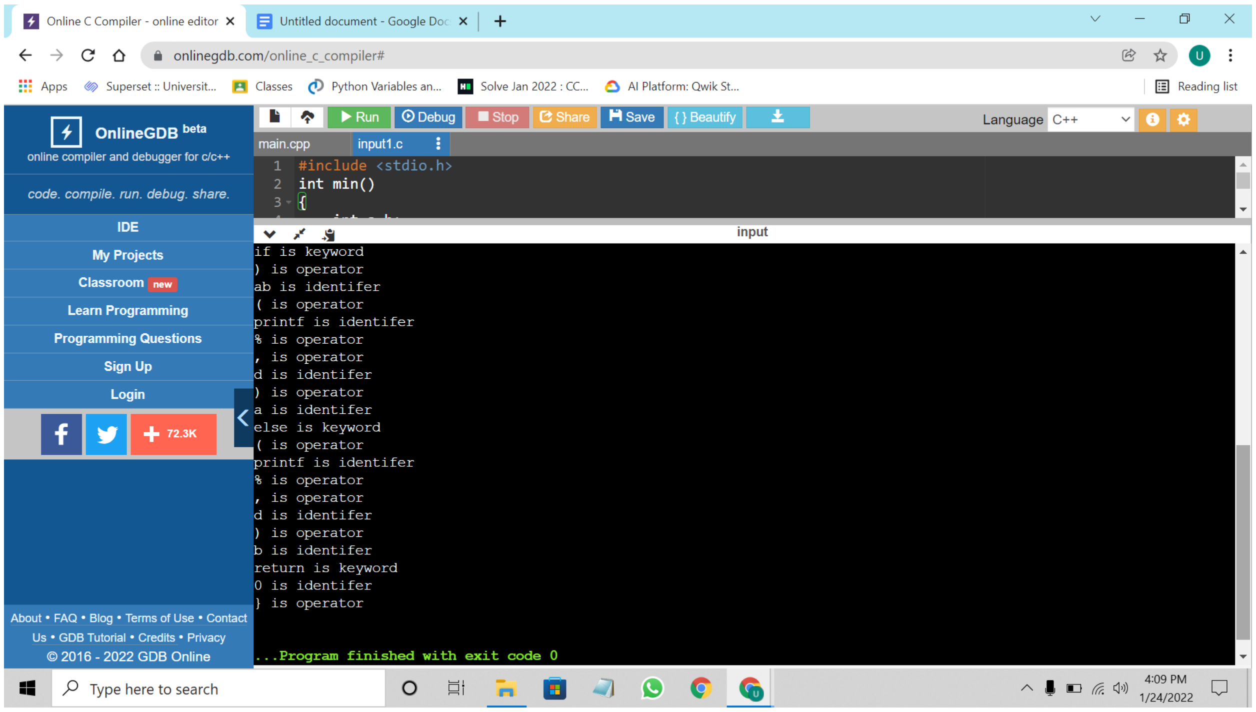
```
#include <stdio.h>
int min()
{
    int a,b;
    scanf("%d%d",&a,&b);
    if(a<b)
    printf("%d",a);
    else
    printf("%d",b);
    return 0;
}
```

Output:



```
#include <stdio.h>
int min()
{
```

```
include is not an identifier
stdioh is not an identifier
int is keyword
( is operator
min is identifer
) is operator
{ is operator
int is keyword
, is operator
a is identifer
b is identifer
( is operator
scanf is identifer
% is operator
% is operator
d is identifer
, is operator
d is identifer
, is operator
a is identifer
) is operator
b is identifer
( is operator
if is keyword
```



```
#include <stdio.h>
int min()
{
```

```
if is keyword
) is operator
ab is identifer
( is operator
printf is identifer
% is operator
, is operator
d is identifer
) is operator
a is identifer
else is keyword
( is operator
printf is identifer
% is operator
, is operator
d is identifer
) is operator
b is identifer
return is keyword
0 is identifer
} is operator


...Program finished with exit code 0
```

Result:

The implementation of lexical analyzer in the c program was compiled and executed successfully.