

## **Compiler Design Lab**

### **Week 1 - B**

Aim: To Implement lexica analyzer in c++ language to detect tokens to identify keywords,operators,identifiers

Algorithm :

- 1) Input the source program as a file.
- 2) Now open the file and start iterating from the first character to the End of the file.
- 3) Now if you come across any operators then print that particular character is a operator.And check the current string is a keyword or Identifier or nor.
- 4) If you find a character is a alphabet or number add it to the current string.
- 5) If you come across any character like space or '\n' or '\t' and your current string size is greater than 0. Repeat step -3.
- 6) This process should be continued until we reach the end of the file

Source Code :

Language : C++

```
#include<bits/stdc++.h>
using namespace std;

bool checkOperator(char ch){
    char operators[] = "+-*/%={},,";
    for (int i = 0; i < 11; ++i){
        if (ch == operators[i]){
            return true;
        }
    }
    return false;
}

int checkKeyWord(string program){
    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] == program){
            flag = 1;
            break;
        }
    }
    return flag;
}

void Check(string &program){
    if(program.size() != 0){
        if (checkKeyWord(program) == 1){
            cout << program << " is keyword" << endl;
        }
        else{
            if(program == "include" || program == "stdioh" || program == "define" || program ==
"stdlibh" || program == "stringh" || program == "ctypeh" || program == "bitsstdch"){
                cout << program << " is not an identifier" << endl;
            }else{
                cout << program << " is identifier" << endl;
            }
        }
        program = "";
    }
}

int main(){
    char ch;
    string program = "";
    FILE *file;
    file = fopen("input.c", "r");
    if (file == NULL){
        printf("Error while opening file ...\n");
        exit(0);
    }
    while ((ch = fgetc(file)) != EOF){
        if(checkOperator(ch)){
            cout << ch << " is operator" << endl;
        }
    }
}
```

```
        Check(program);
    }
    if(isalnum(ch)){
        program += ch;
    }
    else if ((ch == ' ' || ch == '\n') && (program.size() != 0)){
        Check(program);
    }
}
fclose(file);
cout << endl;
return 0;
}
```

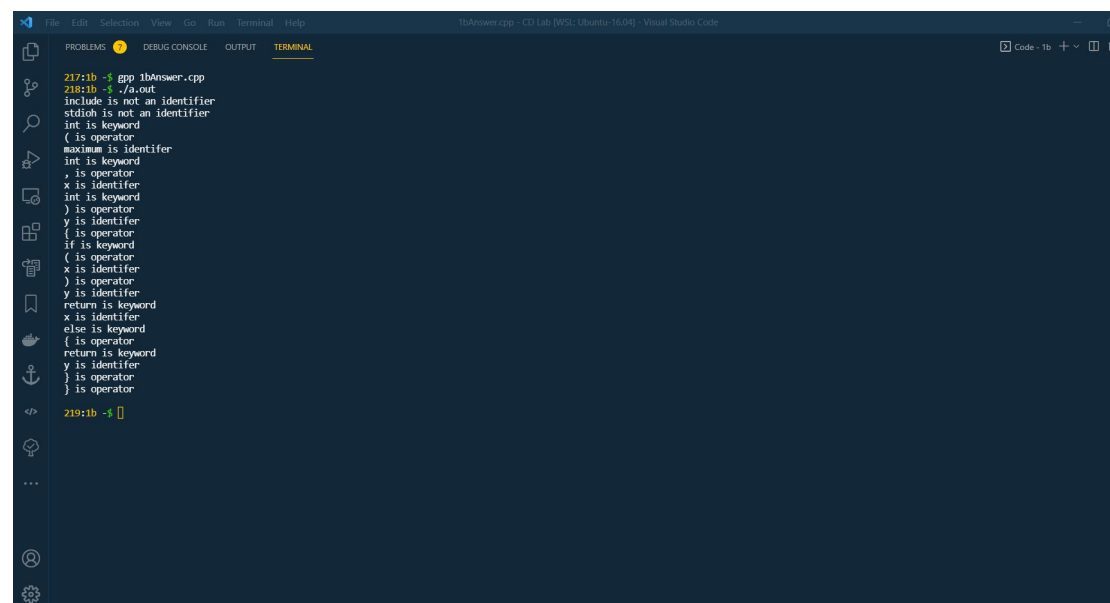
Inputs :

1) File name : input1.c

Code :

```
#include <stdio.h>
int maximum(int x, int y) {
    if (x > y)
        return x;
    else {
        return y;
    }
}
```

Output:



```
217:1b -> gpp 1bAnswer.cpp
218:1b -> ./a.out
include is not an identifier
stdioh is not an identifier
int is keyword
( is operator
maximum is identifier
int is keyword
, is operator
x is identifier
int is keyword
) is operator
y is identifier
{ is operator
if is keyword
( is operator
x is identifier
) is operator
y is identifier
return is keyword
x is identifier
else is keyword
{ is operator
return is keyword
y is identifier
} is operator
}
```

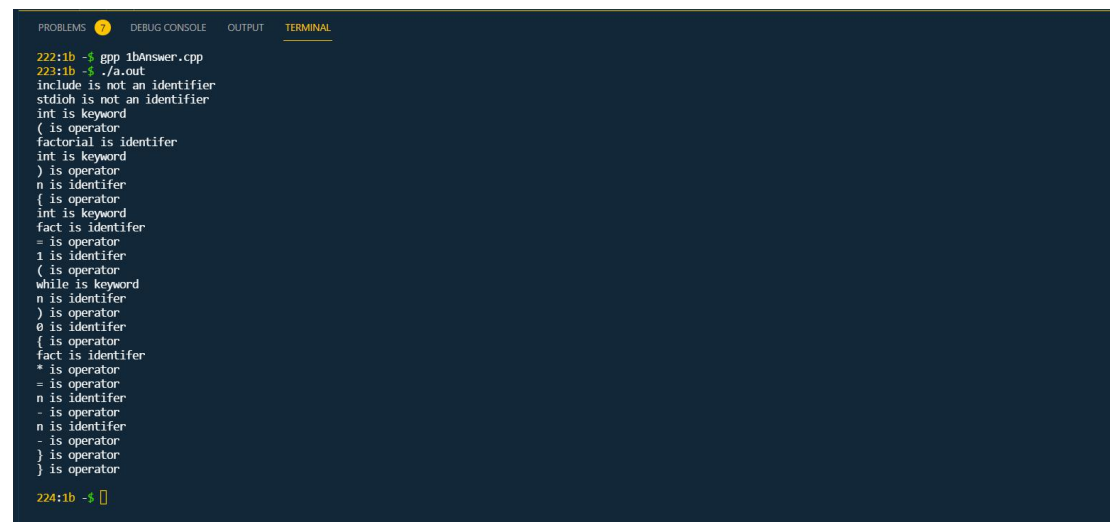
2)filename : input.c

Code :

```
#include <stdio.h>

int factorial(int n){
    int fact = 1;
    while(n > 0){
        fact *= n;
        n--;
    }
}
```

Output:



```
PROBLEMS 17 DEBUG CONSOLE OUTPUT TERMINAL
222:1b - $ gpp 1bAnswer.cpp
223:1b - $ ./a.out
include is not an identifier
stdioh is not an identifier
int is keyword
( is operator
factorial is identifier
int is keyword
) is operator
n is identifier
{ is operator
int is keyword
fact is identifier
= is operator
1 is identifier
( is operator
while is keyword
n is identifier
) is operator
0 is identifier
{ is operator
fact is identifier
* is operator
= is operator
n is identifier
- is operator
n is identifier
- is operator
} is operator
} is operator

224:1b - $ []
```