Sashrika Surya
RA1911027010092
Section: N2
Lab Batch: 1

Compiler Design Lab
Experiment 1: Lexical Analysis

Aim: To study and implement a method to perform Lexical Analysis

Theory:
Lexical analysis is the first phase of a compiler. It takes modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.
If the lexical analyzer finds a token invalid, it generates an error. The lexical analyzer works closely with the syntax analyzer. It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands.

Working:
1. Tokenization
2. Removal of white space
3. Removal of comments
4. Generating errors by providing row numbers and column numbers

Algorithm:
1. Read the input file which contains demo C++ program
2. Split the code line by line
3. Check each word in line against a set of predefined tokens
4. Print them line by line

Code:

```
import re

keyword =
['break','case','char','const','countinue','deafult','do','int','else','enum','extern','float','for','
goto','if','long','register','return','short','signed','sizeof','static','switch','typedef','union','u
nsigned','void','volatile','while']
built_in_functions = ['clrscr()','printf(','scanf(','getch()','main()']
operators = ['+','-','*','/','%','==','!
=','>','<','>=','<=','&&','||','!','&','|','^','~','>>','<<','=','+=','-=','*=']
specialsymbol = ['@','#','$','_','!']
```

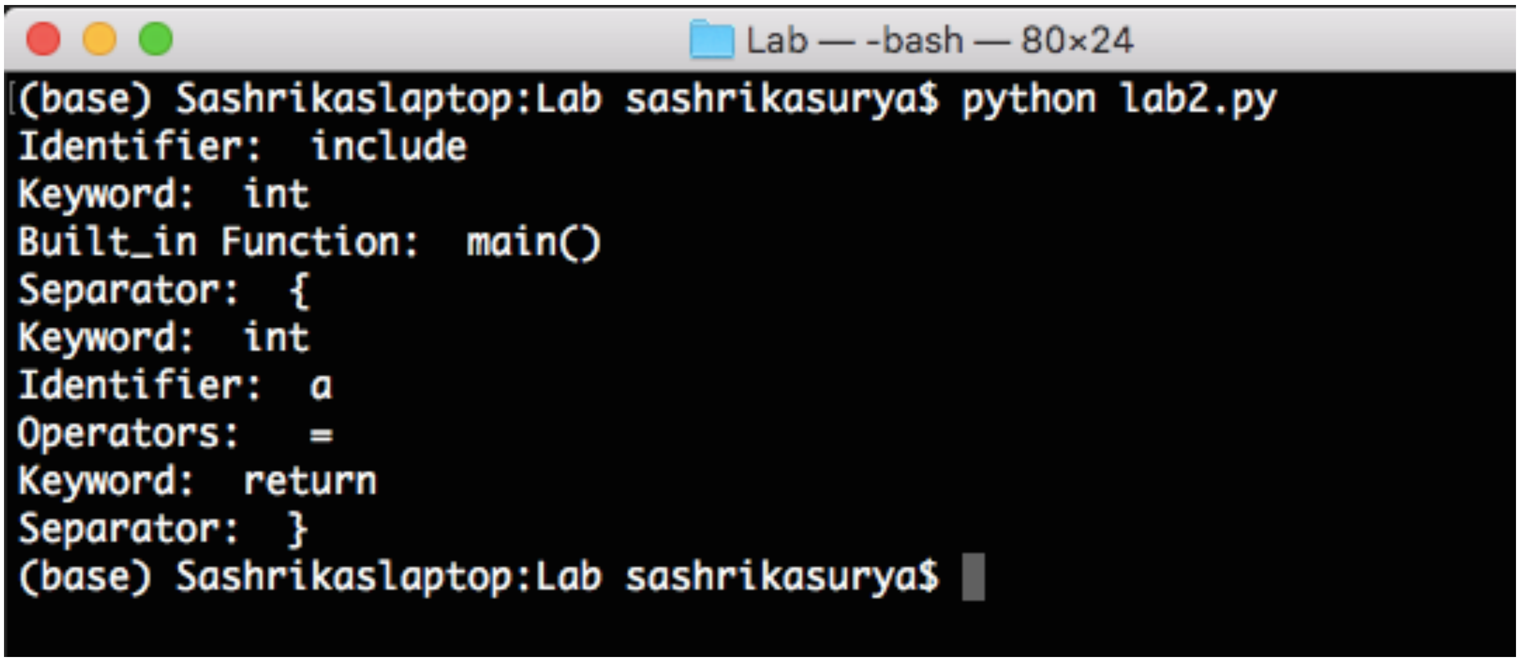
```
separator = [',',':',';','\n','\t','{','}','(',')','[',']']


file = open('lexical.txt','r+')
contents = file.read()
splitCode = contents.split()
length = len(splitCode)
for i in range(0,length):
    if splitCode[i] in keyword:
        print("Keyword: ",splitCode[i])
        continue
    if splitCode[i] in operators:
        print("Operators:  ",splitCode[i])
        continue
    if splitCode[i] in specialsymbol:
        print("Special Operator: ",splitCode[i])
        continue
    if splitCode[i] in built_in_functions:
        print("Built_in Function: ",splitCode[i])
        continue
    if splitCode[i] in separator:
        print("Separator: ",splitCode[i])
        continue
    if re.match(r'(#include*).*', splitCode[i]):
        print ("Header File: ", splitCode[i])
        continue
    if re.match(r'^[-+]?[0-9]+$',splitCode[i]):
        print("Numerals: ",splitCode[i])
        continue
    if re.match(r"^[^\d\W]\w*\Z", splitCode[i]):
        print("Identifier: ",splitCode[i])
```

Demo Text File:

```
include <stdio.h>
int main()
{
    int a;
    a = 10;
    return 0;
 }
```

Output:



```
(base) Sashrikaslaptop:Lab sashrikasurya$ python lab2.py
Identifier:  include
Keyword:  int
Built_in Function:  main()
Separator:  {
Keyword:  int
Identifier:  a
Operators:   =
Keyword:  return
Separator:  }
(base) Sashrikaslaptop:Lab sashrikasurya$ 
```

Result:

Hence the given aim was achieved as lexical analyser was studied and implemented.