EXPERIMENT - I
LEXICAL ANALYSIS

August 26, 2020

ADITHYA D RAJAGOPAL
ROLL NO : 9
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM

AIM

To design and implement a lexical analyzer for given language using Python. The lexical analyzer should ignore redundant spaces, tabs and new line.

THEORY

The very first phase of a compiler deals with lexical analysis. A lexical analyser, also known as scanner, converts the high level input program into a sequence of tokens. A lexical token is a sequence of characters which is treated as a unit in the grammar of the programming languages.
The common types of tokens include:

1. **Keyword:** A keyword is a word reserved by a programming language having a special meaning.

2. **Identifier:** It is a user-defined name used to uniquely identify a program element. It can be a class, method, variable, namespace etc.

3. **Operator:** It is a symbol that tells the compiler or interpreter to perform specific mathematical, relational or logical operation and produce final result.

4. **Separator:** Separators are used to separate one programming element from the other.

5. **Literal:** A literal is a notation for representing a fixed value and do not change during the course of execution of the program.

ALGORITHM

Algorithm 1 Algorithm for Lexical Analyser

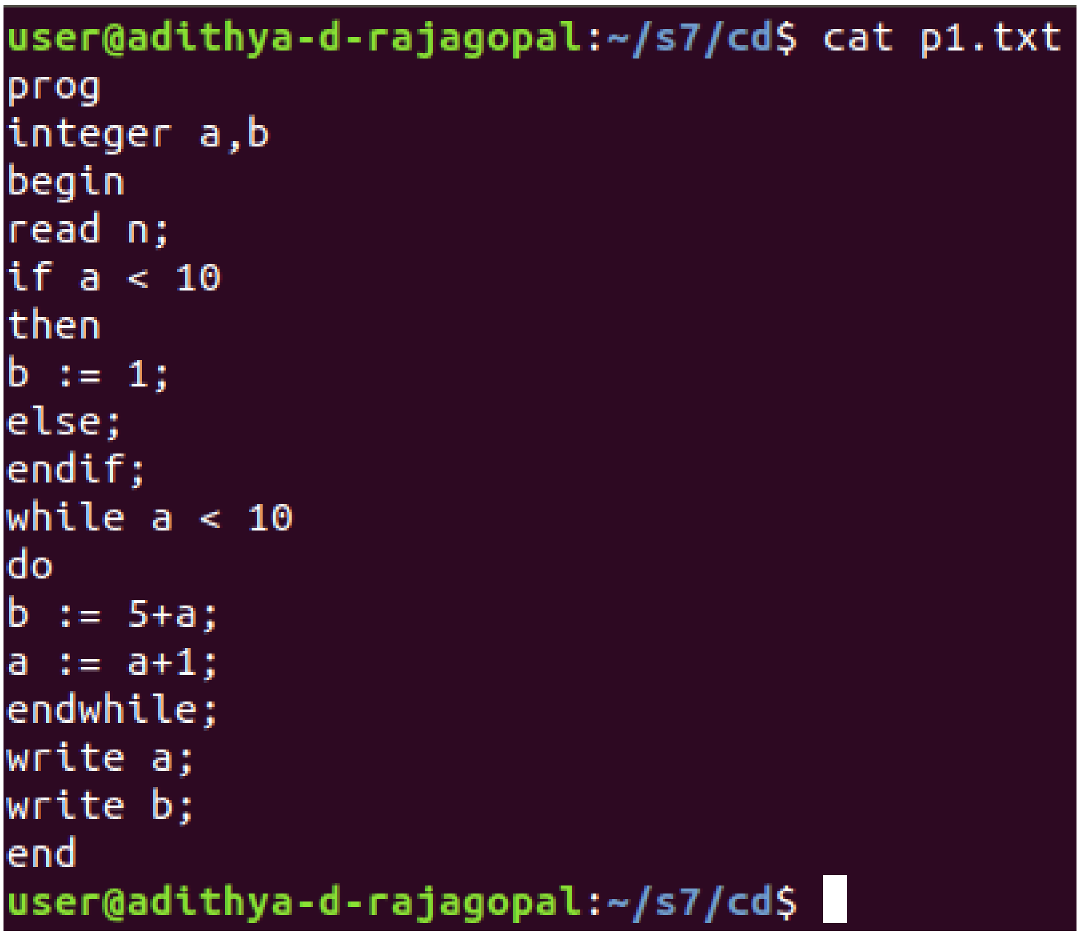
```
 1: Start
 2: Read the input file.
 3: Read from the file word by word.
 4: Split the word into meaningful tokens using delimiters.
 5: Read the tokens one by one.
 6: if Token is a keyword then
 7:     print <Token, Keyword>
 8: else if Token is an identifier then
 9:     print <Token, Identifier>
10: else if Token is an operator then
11:     print <Token, Operator>
12: else if Token is a separator then
13:     print <Token, Separator>
14: else if Token is a literal then
15:     print <Token, Literal>
16: end if
17: Stop
```

SOURCE CODE

```
def isSpecialSymbol(token):
        symbol=("~","!","@","#","$","&",
        "(","&",")","[","]","{","}",":",
        "?",".","|",",","%","+","-","*",
        "/","\"","\\","=","<",">")
        for i in token:
                if i in symbol:
                        return True
        return False

def isOperator(token):
        op=("+","-","*","/","%","^","<",">",":=","&","|","!")
        if token in op:
                return True
        return False

def isKeyword(token):
        keywords=("prog","begin","end","read","write",
        "integer","string","character","float",
        "if","else","then","endif","while","do","endwhile")
        if token in keywords:
                return True
        return False

def isIdentifier(token):
        id=("1","2","3","4","5","6","7","8","9","0","_")
        if token[0] not in id:
                if(not isSpecialSymbol(token)):
                        return True
        return False

def isLiteral(token):
        num=("1","2","3","4","5","6","7","8","9","0")
        count=0
        for i in token:
                if i not in num:
                        if i!="." and count==0:
                                return False
                        count=1
        return True

def isSymbol(token):
        if token==",":
                return True
        return False

def mixed(token):
        op=("+","-","*","/","%","^","<",">",":=","&","|","!",",")
        s=[]
        for o in op:
                if o in token:
                        t=token.split(o)
                        if token[0]==o:
                                s.append(o)
                        for id in t:
                                s.append(id)
                                s.append(o)
                        if(token[len(token)-1]!=o):
                                s.pop()
        scan(s)

def scan(program):
        for line in program:
                token=line.split()
                for i in range(len(token)):
                        if token[i][len(token[i])-1]==';':
                                token[i]=token[i][:-1]
                        if(isKeyword(token[i])):
```

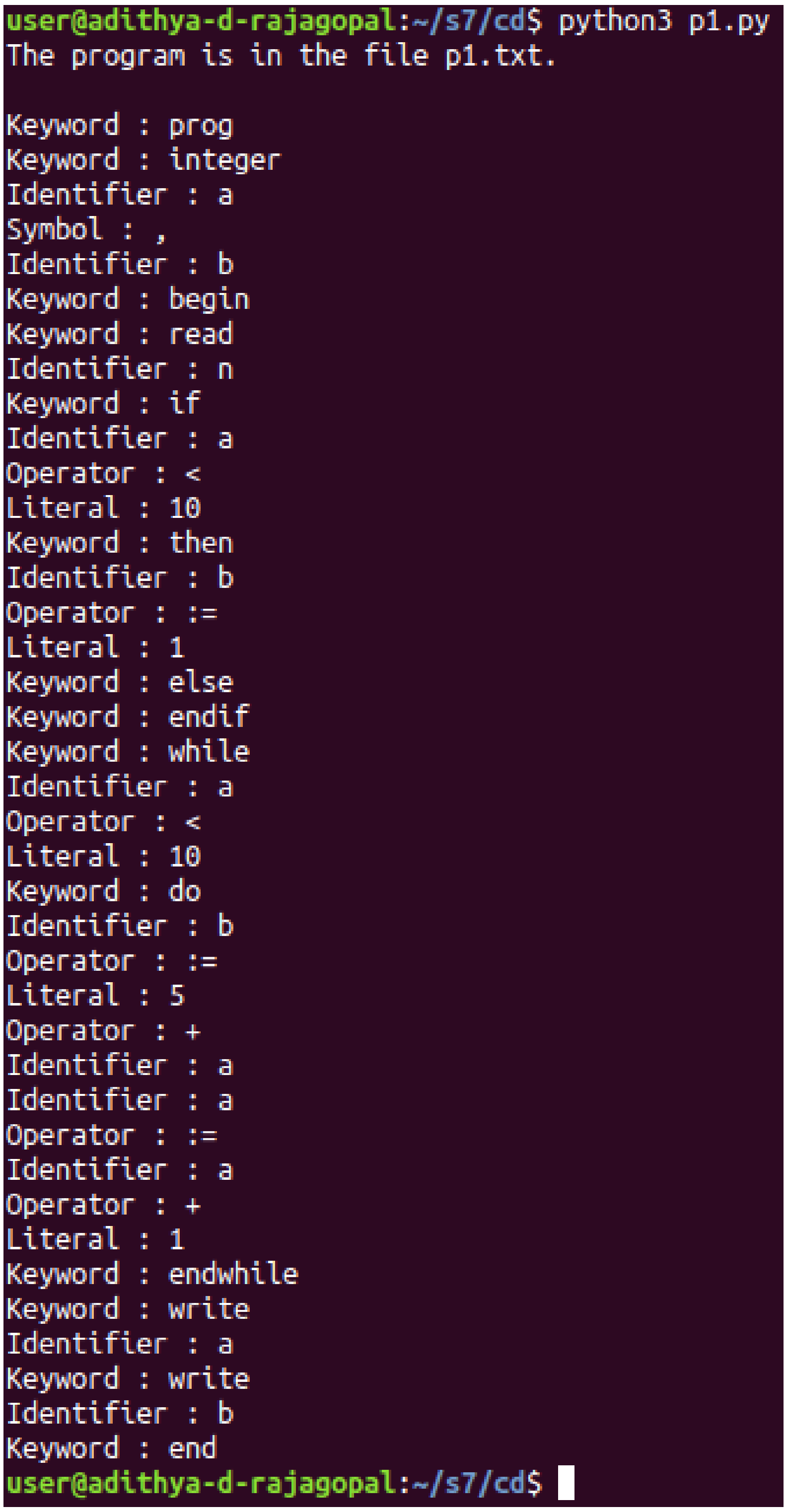
```
                print("Keyword : "+token[i])
            elif(isIdentifier(token[i])):
                print("Identifier : "+token[i])
            elif(isLiteral(token[i])):
                print("Literal : "+token[i])
            elif(isOperator(token[i])):
                print("Operator : "+token[i])
            elif(isSymbol(token[i])):
                print("Symbol : "+token[i])
            else:
                mixed(token[i])

program=[]
print("The program is in the file p1.txt.\n")
f=open("p1.txt", "r")
for line in f:
    program.append(line)
    if line=="end\n":
        break
f.close()
scan(program)
```

SAMPLE INPUT



```
user@adithya-d-rajagopal:~/s7/cd$ cat p1.txt
prog
integer a,b
begin
read n;
if a < 10
then
b := 1;
else;
endif;
while a < 10
do
b := 5+a;
a := a+1;
endwhile;
write a;
write b;
end
user@adithya-d-rajagopal:~/s7/cd$
```

SAMPLE OUTPUT



```
user@adithya-d-rajagopal:~/s7/cd$ python3 p1.py
The program is in the file p1.txt.

Keyword : prog
Keyword : integer
Identifier : a
Symbol : ,
Identifier : b
Keyword : begin
Keyword : read
Identifier : n
Keyword : if
Identifier : a
Operator : <
Literal : 10
Keyword : then
Identifier : b
Operator : :=
Literal : 1
Keyword : else
Keyword : endif
Keyword : while
Identifier : a
Operator : <
Literal : 10
Keyword : do
Identifier : b
Operator : :=
Literal : 5
Operator : +
Identifier : a
Identifier : a
Operator : :=
Identifier : a
Operator : +
Literal : 1
Keyword : endwhile
Keyword : write
Identifier : a
Keyword : write
Identifier : b
Keyword : end
user@adithya-d-rajagopal:~/s7/cd$
```

RESULT

A lexical analyzer has been designed and implemented using Python and the outputs were verified.