COMPILER DESIGN

EXP-1(A) LEXICAL ANALYSER

Tamojit Sarkar

RA1811027010034

CSE-BD Sec-I2

Aim: To perform lexical analysis on C/C++ code

Language Used:

Python

Procedure:

1. Create a python file
2. Create function for checking all the keywords, operators and identifiers and run through the file with C/C++ language code.
3. Check whether it mentions all the keywords, operators and identifiers present in the code.
4. Display the output of the file by detecting and printing each and every part of the code.
5. Make the analyzer in such a way that it can classify all the keywords, identifiers and operators.

Code:

```
special_char = [',', ';', '(', ')', '[', ']', '{', '}','\n','\t']
operators = ['<','>','+','-','*','/','%','^','?','=','#','&']
keywords = ['void','int', 'float', 'double', 'char', 'main', 'return','if','for','long','double','return','sizeof','while','
file = open("C:/Users/H/Documents/code.txt",'r')
a = [item.split() for item in file]
bucket = []
for item in a:
    for p in item:
        bucket.append(p)
def check_if_alpha(item):
    if item.isalpha():
        return True
    else:
        return False

def check_if_operators(item):
    if item in operators:
        return True
    else:
        return False

def check_if_special(item):
    if item in special_char:
        return True
    else:
        return False
def check_if_alnum(item):
    if item.isalnum():
        return True
    else:
        return False
def check_if_num(item):
    if item.isnumeric():
        return True
    else:
        return False
def lexi(item):
    if check_if_alpha(item):
        if item in keywords:
            print('{} is a keyword'.format(item))
        else:
            print('{} is an identifier'.format(item))
    elif check_if_alnum(item):
        print('{} is a identifier'.format(item))
    elif check_if_num(item):
        print('{} is a number'.format(item))
    elif check_if_operators(item):
         print('{} is an operator'.format(item))
    elif check_if_special(item):
         print('{} is a special character'.format(item))
    else:
        if len(item)>1:
            if item[:5]=='scanf' or item[:6]=='printf':
                print('It is {} statement'.format(item[:5]))
            elif item[-1] in special_char and item[-2] in special_char:
                print('{} is a special character'.format(item[-1]))
                print('{} is a special character'.format(item[-2]))
                item = item[:-2]
                lexi(item)
            elif item[0]=='<' and item[-1]=='>':
                item = item[1:-1]
                print('{} is header file'.format(item))
            elif item[-2:]=='\n' or item[-2:]=='\t':
                print('{} is a special character'.format(item[-2:]))
                item = item[:-2]
                lexi(item)
            elif item[0]=='#':
                print('{} is a special character'.format(item[0]))
                item = item[1:]
                lexi(item)
            elif item[-1] in special_char:
                print('{} is a special character'.format(item[-1]))
                item = item[:-1]
                lexi(item)
            elif item[-1] in operators:
                print('{} is a operator'.format(item[-1]))
                item = item[:-1]
                lexi(item)
            else:
                print("Can't Analyse")
for item in bucket:
    lexi(item)
```

Output:

```
# is a special character
include is a keyword
stdio.h is header file
int is a keyword
) is a special character
( is a special character
main is a keyword
{ is a special character
int is a keyword
, is a special character
number1 is a identifier
, is a special character
number2 is a identifier
; is a special character
sum is an identifier
It is scanf statement
sum is an identifier
= is an operator
number1 is a identifier
+ is an operator
; is a special character
number2 is a identifier
It is print statement
return is a keyword
; is a special character
0 is a identifier
} is a special character
```

C/C++ Code File:

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

    int number1, number2, sum;
    scanf("%d%d",&number1,&number2);
    sum = number1 + number2;
    printf("%d",sum);
    return 0;
}
```

Conclusion: Lexical Analyser of C/C++ code is being written in python.