IMPLEMENTATION OF LEXICAL ANALYSER ALONG WITH SPECIFICATION OF TOKENS

RA1811003010344

VIKASH MISHRA

**COMPILER DESIGN LAB - 1**

AIM - Implementation of lexical analyzer with specification of tokens

ALGORITHM -

1 We made an 2d array named ‘Keyword’ which stores the keywords and such as ‘void’,’float’,’int’ ,etc.

2 In the main function we declared char operators array and numbers array which will be used to identify the given input in the token

3 We open the text file which contains the input , and thus check for the pointer of the file , if we are unable to point towards the doc we print ‘error’ and exit

4 using the while loop with in nested for loop , we check for individual token as and compare it to our keyword , operator and identifier array which we made earlier and print them accordingly

CODE -

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<ctype.h>

int isKeyword(char buffer[]){

char keyword[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(strcmp(keyword[i], buffer) == 0){

flag = 1;

break;

}

}

return flag;

}

int main(){

char ch, buffer[15], operators[] = "+-\*/%=" , number[]="1234567890" ;

FILE \*fp;

int i,j=0;

fp = fopen("text.txt","r");

if(fp == NULL){

printf("error - pointer error , file opening \n");

exit(0);

}

while((ch = fgetc(fp)) != EOF){

/// operator

for(i = 0; i < 6; ++i){

if(ch == operators[i])

printf("%c is operator\n", ch);

}

// number

for(i = 0; i < 6; ++i){

if(ch == number[i])

printf("%c is number\n", ch);

}

if(isalnum(ch)){

buffer[j++] = ch;

}

else if((ch == ' ' || ch == '\n') && (j != 0)){

buffer[j] = '\0';

j = 0;

if(isKeyword(buffer) == 1)

printf("%s is keyword\n", buffer);

else

printf("%s is identifier\n", buffer);

}

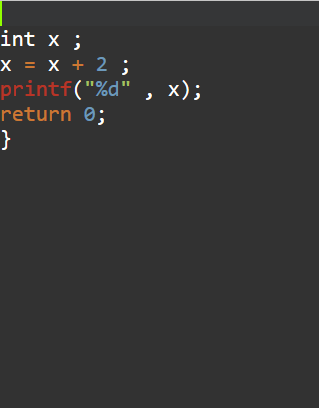
}

fclose(fp);

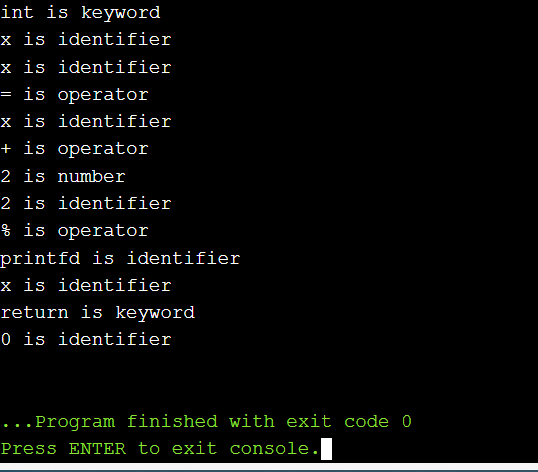
return 0;

}

INPUT -



OUTPUT -



RESULT - The lexical analyzer was implemented and is working successfully.