**Compiler Design Lab**

**Assignment-1**

**Implementation of Lexical Analyzer**

**Name : Vikraman S**

**Reg No. : 185001195**

**Code:**

#include<stdio.h> //Preprocessor

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

int operator(char op1,char op2) //Function to find whether it is operator or not

{

int flag=1;

if(op1=='<')

{

if(op2=='=')

printf("RELOP ");

else

printf("RELOP ");

}

else if(op1=='>')

{

if(op2=='=')

printf("RELOP ");

else

printf("RELOP ");

}

else if(op1=='=')

{

if(op2=='=')

printf("Equal to ");

else

printf("ASSIGN ");

}

else if(op1=='!')

{

if(op2=='=')

printf("Not Equal to ");

else

printf("LOGOP ");

}

else if(op1=='+' && op2==' ')

printf("ARIOP ");

else if(op1=='-' && op2==' ')

printf("ARIOP ");

else if(op1=='\*' && op2==' ')

printf("ARIOP ");

else if(op1=='/' && op2==' ')

printf("ARIOP ");

else if(op1=='&' && op2=='&')

printf("LOGOP ");

else if(op1=='|' && op2=='|')

printf("LOGOP ");

else

flag=0;

return flag;

}

int identifier(char \*x) ////Function to find whether it is indentifier or not

{

if(isalpha(x[0]))

{

printf("ID ");

return 1;

}

else

return 0;

}

int constant(char \*x) //Function to find whether it is constant or not

{

if(isdigit(x[0]))

{

printf("NUMCONST ");

return 1;

}

else

return 0;

}

int comment(char x,char y) //Function to find whether it is comment or not

{

if(x=='/' && y=='/')

{

printf("Single line comment ");

return 1;

}

else if(x=='/' && y=='\*')

{

printf("Multiple line comment ");

return 1;

}

else

return 0;

}

int keywords(char \*str) //Function to find whether it is keyword or not

{

if(str[0]=='#')

{

printf("Preprocessor ");

return 1;

}

else if (!strcmp(str, "if") || !strcmp(str, "else") || !strcmp(str, "while") || !strcmp(str, "do") || !strcmp(str, "break") || !strcmp(str, "continue") || !strcmp(str, "int")

|| !strcmp(str, "double") || !strcmp(str, "float") || !strcmp(str, "return") || !strcmp(str, "char") || !strcmp(str, "case") || !strcmp(str, "char")

|| !strcmp(str, "sizeof") || !strcmp(str, "long") || !strcmp(str, "short") || !strcmp(str, "typedef") || !strcmp(str, "switch") || !strcmp(str, "unsigned")

|| !strcmp(str, "void") || !strcmp(str, "static") || !strcmp(str, "struct") || !strcmp(str, "goto"))

{

printf("KW ");

return 1;

}

else

{

int i,j,len=strlen(str);

for(i=0;i<len;i++)

{

if(str[i]=='(')

{

for(j=i+1;j<len;j++)

{

if(str[j]==')')

{

printf("FC ");

return 1;

}

}

printf("SP ");

return 1;

}

else if(str[i]=='(' || str[i]==')' || str[i]=='{' || str[i]=='}' || str[i]=='[' || str[i]==']' || str[i]==';' || str[i]==',')

{

printf("SP ");

return 1;

}

}

return 0;

}

}

void main() //main function

{

FILE \*fp,\*fp1;

char s[100];

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

printf("\nLex Input :\n");

while(!feof(fp))

{

fscanf(fp," %[^\n]",s);

printf("\t%s\n",s);

}

fclose(fp);

printf("\n\nLex Output :\n");

char ch;

int i;

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

printf("\t");

while(!feof(fp))

{

char st[100];

fscanf(fp," %[^\n]",st);

fp1=fopen("temp.txt","w");

fprintf(fp1,"%s",st);

fclose(fp1);

fp1=fopen("temp.txt","r");

while(!feof(fp1))

{

char str[100];

fscanf(fp1,"%s",str);

for(i=0;i<100;i++)

{

if(keywords(str))

break;

else if(comment(str[i],str[i+1]))

i++;

else if(operator(str[i],str[i+1]))

i+=2;

else if(identifier(str))

break;

else if(constant(str))

break;

else

continue;

}

}

printf("\n\t");

fclose(fp1);

}

remove("temp.txt");

fclose(fp);

}

**Output:**

