**CD Lab**

**Session 1 – Lab 2 (09/12/2020)**

Parthivi Choubey CSE – B - 5th semester

180905456 Roll. no. - 60

**Question 1**

**Code**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <ctype.h>

struct token

{

char lexeme[64];

unsigned int row,col;

char type[20];

};

static int row=1,col=1;

char buf[2048];

char specialsymbols[] = {'?',';',':',',','.'};

const char \*keywords[] = {"const", "bool", "char", "int","float", "double","unsigned", "return", "for", "while", "do", "switch","if", "else","case", "break","printf", "continue"};

char arithmeticsymbols[] = {'\*','%'};

int isKeyword(char \*word)

{

for(int i=0;i<sizeof(keywords)/sizeof(char\*);i++)

{

if(strcmp(word,keywords[i])==0)

return 1;

}

return 0;

}

int charBelongsTo(int c, char \*arr)

{

int len=0;

if(arr==specialsymbols)

len=sizeof(specialsymbols)/sizeof(char);

else if(arr==arithmeticsymbols)

len=sizeof(arithmeticsymbols)/sizeof(char);

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

{

if(c==arr[i])

return 1;

}

return 0;

}

void fillToken(struct token \*t, char c, int row, int col, char \*type)

{

t->lexeme[0]=c; t->lexeme[1]='\0';

t->row=row; t->col=col;

strcpy(t->type,type);

}

void newLine()

{

row++; col=1;

}

struct token getNextToken(FILE \*f)

{

int c;

struct token tkn=

{

.row=-1

};

int gotToken=0;

while(!gotToken && (c = fgetc(f))!=EOF)

{

if(charBelongsTo(c,specialsymbols))

{

fillToken(&tkn,c,row,col,"SS");

gotToken=1;

++col;

}

else if(charBelongsTo(c,arithmeticsymbols))

{

fillToken(&tkn,c,row,col,"ARITHMETIC OPERATOR");

gotToken=1;

++col;

}

else if(c=='(')

{

fillToken(&tkn,c,row,col,"LB");

gotToken=1;

++col;

}

else if(c==')')

{

fillToken(&tkn,c,row,col,"RB");

gotToken=1;

++col;

}

else if(c=='{')

{

fillToken(&tkn,c,row,col,"LC");

gotToken=1;

++col;

}

else if(c=='}')

{

fillToken(&tkn,c,row,col,"RC");

gotToken=1;

++col;

}

else if(c=='[')

{

fillToken(&tkn,c,row,col,"LS");

gotToken=1;

++col;

}

else if(c==']')

{

fillToken(&tkn,c,row,col,"RS");

gotToken=1;

++col;

}

else if(c=='+')

{

int d=fgetc(f);

if(d!='+')

{

fillToken(&tkn,c,row,col,"ARITHMETIC OPERATOR");

gotToken=1;

++col;

fseek(f,-1,SEEK\_CUR); //go back 1 step \*

}

else

{

fillToken(&tkn,c,row,col,"UNARY OPERATOR");

strcpy(tkn.lexeme,"++");

gotToken=1;

col+=2; //skip next as it is already included

}

}

else if(c=='-')

{

int d=fgetc(f);

if(d!='-')

{

fillToken(&tkn,c,row,col,"ARITHMETIC OPERATOR");

gotToken=1;

++col;

fseek(f,-1,SEEK\_CUR); //go back 1 step \*

}

else

{

fillToken(&tkn,c,row,col,"UNARY OPERATOR");

strcpy(tkn.lexeme,"--");

gotToken=1;

col+=2; //skip next as it is already included

}

}

else if(c=='=')

{

int d=fgetc(f);

if(d!='=')

{

fillToken(&tkn,c,row,col,"ASSIGNMENT OPERATOR");

gotToken=1;

++col;

fseek(f,-1,SEEK\_CUR); //go back 1 step \*

}

else

{

fillToken(&tkn,c,row,col,"RELATIONAL OPERATOR");

strcpy(tkn.lexeme,"==");

gotToken=1;

col+=2; //skip next as it is already included

}

}

else if(isdigit(c))

{

tkn.row=row;

tkn.col=col++;

tkn.lexeme[0]=c;

int k=1;

while((c=fgetc(f))!=EOF && isdigit(c))

{

tkn.lexeme[k++]=c;

col++;

}

tkn.lexeme[k]='\0';

strcpy(tkn.type,"NUMBER");

gotToken=1;

fseek(f,-1,SEEK\_CUR); //go back 1 step \*

}

else if(c == '#')

{

while((c = fgetc(f)) != EOF && c != '\n');

newLine();

}

else if(c=='\n')

{

newLine();

c = fgetc(f);

if(c == '#')

{

while((c = fgetc(f)) != EOF && c != '\n');

newLine();

}

else if(c != EOF)

fseek(f, -1, SEEK\_CUR);

}

else if(isspace(c))

++col;

else if(isalpha(c)||c=='\_')

{

tkn.row=row;

tkn.col=col++;

tkn.lexeme[0]=c;

int k=1;

while((c=fgetc(f))!= EOF && isalnum(c))

{

tkn.lexeme[k++]=c;

++col;

}

tkn.lexeme[k]='\0';

if(isKeyword(tkn.lexeme))

strcpy(tkn.type,"KEYWORD");

else

strcpy(tkn.type,"IDENTIFIER");

gotToken=1;

fseek(f,-1,SEEK\_CUR);

}

else if(c=='/')

{

int d=fgetc(f);

++col;

if(d=='/')

{

while((c=fgetc(f))!= EOF && c!='\n')

++col;

if(c=='\n')

newLine();

}

else if(d=='\*')

{

do

{

if(d=='\n')

newLine();

while((c==fgetc(f))!= EOF && c!='\*')

{

++col;

if(c=='\n')

newLine();

}

++col;

}while((d==fgetc(f))!= EOF && d!='/' && (++col));

++col;

}

else

{

fillToken(&tkn,c,row,--col,"ARITHMETIC OPERATOR");

gotToken=1;

fseek(f,-1,SEEK\_CUR);

}

}

else if(c == '"')

{

tkn.row = row;

tkn.col = col;

strcpy(tkn.type, "STRING LITERAL");

int k = 1;

tkn.lexeme[0] = '"';

while((c = fgetc(f)) != EOF && c != '"')

{

tkn.lexeme[k++] = c;

++col;

}

tkn.lexeme[k] = '"';

gotToken = 1;

}

else if(c == '<' || c == '>' || c == '!')

{

fillToken(&tkn, c, row, col, "RELATIONAL OPERATOR");

++col;

int d = fgetc(f);

if(d == '=')

{

++col;

strcat(tkn.lexeme, "=");

}

else

{

if(c == '!')

strcpy(tkn.type, "LOGICAL OPERATOR");

fseek(f, -1, SEEK\_CUR);

}

gotToken = 1;

}

else if(c == '&' || c == '|')

{

int d = fgetc(f);

if(c == d)

{

tkn.lexeme[0] = tkn.lexeme[1] = c; tkn.lexeme[2] = '\0';

tkn.row = row;

tkn.col = col;

++col;

gotToken = 1;

strcpy(tkn.type, "LOGICAL OPERATOR");

}

else

{

tkn.lexeme[0] = c; tkn.lexeme[1] = '\0';

tkn.row = row;

tkn.col = col;

++col;

gotToken = 1;

strcpy(tkn.type, "BITWISE OPERATOR");

fseek(f, -1, SEEK\_CUR);

}

++col;

}

else

++col;

}

return tkn;

}

int main()

{

printf("Enter file name: ");

char input[256];

scanf("%s",input);

FILE \*f = fopen(input, "r");

if (f == NULL)

{

printf("Cannot open file\n");

exit(0);

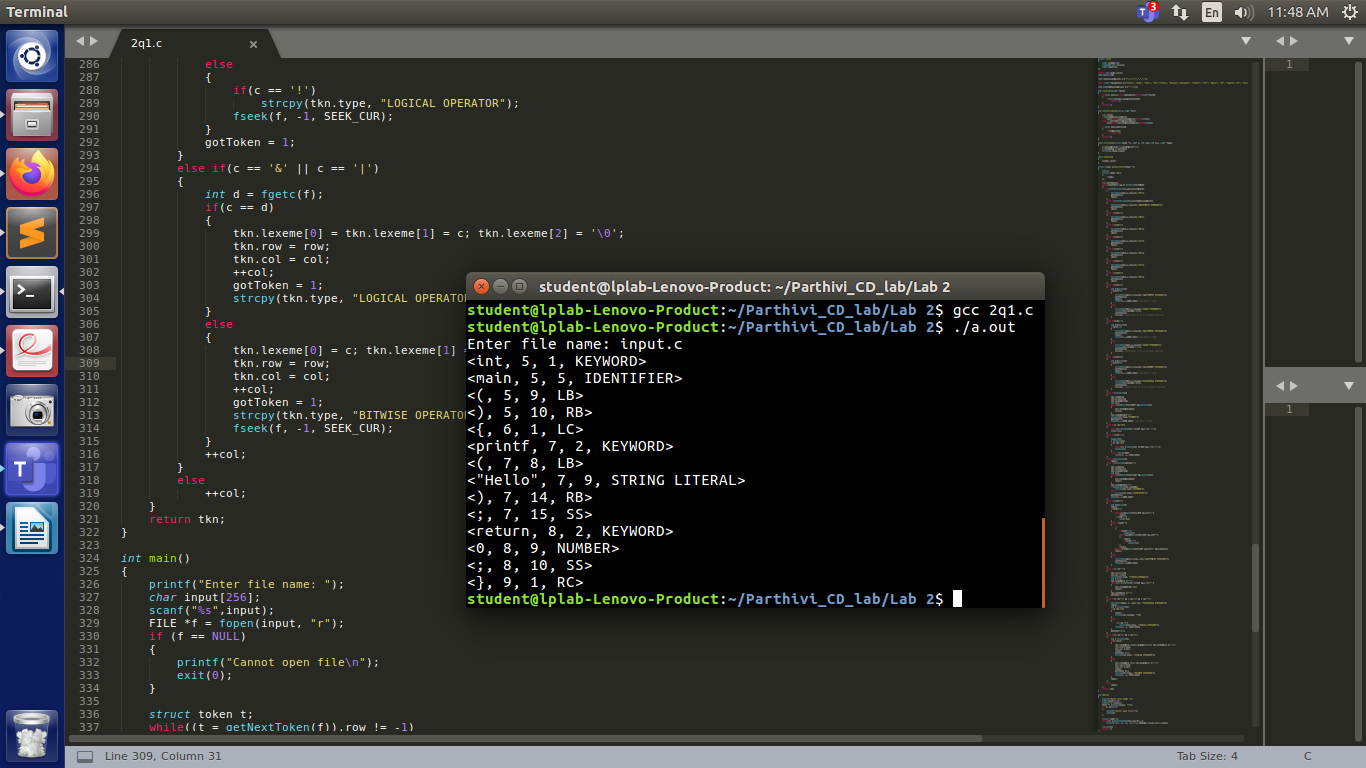
}

struct token t;

while((t = getNextToken(f)).row != -1)

printf("<%s, %d, %d, %s>\n",t.lexeme,t.row,t.col,t.type);

fclose(f);

 return 0;

}

**Output**