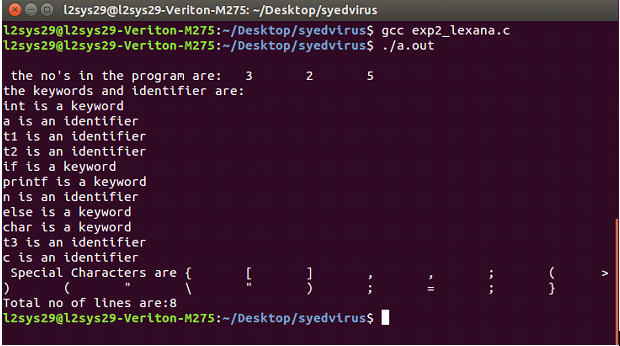
**PRACTICAL- 1**

Develop a lexical analyzer to recognize few patterns in C. (Ex. identifiers, constants, comments, operators etc.).

SOURCE CODE:-

#include<string.h>   
#include<ctype.h>   
#include<stdio.h>   
#include<stdlib.h>   
void keyword(char str[10])   
{   
if(strcmp("for",str)==0||strcmp("while",str)==0||strcmp("do",str)==0||strcmp("int",str)==0||strcmp("float",str)==0||strcmp("char",str)==0||strcmp("double",str)==0||strcmp("printf",str)==0||strcmp("switch",str)==0||strcmp("case",str)==0)   
  printf("\n%s is a keyword",str);   
else   
  printf("\n%s is an identifier",str);   
}   
void main()   
{   
FILE \*f1,\*f2,\*f3;   
char c,str[10],st1[10];   
int num[100],lineno=0,tokenvalue=0,i=0,j=0,k=0;   
f1=fopen("input","r");   
f2=fopen("identifier","w");   
f3=fopen("specialchar","w");   
while((c=getc(f1))!=EOF)   
{   
  if(isdigit(c))   
  {   
  tokenvalue=c-'0';   
  c=getc(f1);   
  while(isdigit(c))   
  {   
   tokenvalue\*=10+c-'0';   
   c=getc(f1);   
  }   
  num[i++]=tokenvalue;   
  ungetc(c,f1);   
  }   
  else   
  if(isalpha(c))   
  {   
   putc(c,f2);   
   c=getc(f1);   
   while(isdigit(c)||isalpha(c)||c=='\_'||c=='$')   
   {   
    putc(c,f2);   
    c=getc(f1);   
   }   
   putc(' ',f2);   
   ungetc(c,f1);   
  }   
  else   
  if(c==' '||c=='\t')   
  printf(" ");   
  else   
  if(c=='\n')   
  lineno++;   
  else   
  putc(c,f3);   
}   
fclose(f2);   
fclose(f3);   
fclose(f1);   
printf("\n the no's in the program are:");   
for(j=0;j<i;j++)   
  printf("\t%d",num[j]);   
printf("\n");   
f2=fopen("identifier","r");   
k=0;   
printf("the keywords and identifier are:");   
while((c=getc(f2))!=EOF)   
if(c!=' ')   
str[k++]=c;   
else   
{   
  str[k]='\0';   
  keyword(str);   
  k=0;   
}   
fclose(f2);   
f3=fopen("specialchar","r");   
printf("\n Special Characters are");   
while((c=getc(f3))!=EOF)   
printf("\t%c",c);   
printf("\n");   
fclose(f3);   
printf("Total no of lines are:%d",lineno);   
}

OUTPUT:-



**PRACTICAL –2**

Design a lexical analyzer for given language and the lexical analyzer should ignore redundant spaces, tabs and new lines.

SOURCE CODE:-

#include<string.h>   
#include<ctype.h>   
#include<stdio.h>   
void keyword(char str[10])   
{   
if(strcmp("for",str)==0||strcmp("while",str)==0||strcmp("do",str)==0|| strcmp("int",str)==0||strcmp("float",str)==0||strcmp("char",str)==0||strcmp("double",str)==0||   
strcmp("static",str)==0||strcmp("switch",str)==0||strcmp("case",str)==0)   
printf("\n%s is a keyword",str);   
else   
printf("\n%s is an identifier",str);   
}   
main()   
{   
FILE \*f1,\*f2,\*f3;   
char c,str[10],st1[10];   
int num[100],lineno=0,tokenvalue=0,i=0,j=0,k=0;   
printf("\nEnter the c program");/\*gets(st1);\*/   
f1=fopen("input","w");   
while((c=getchar())!=EOF)   
putc(c,f1);   
fclose(f1);   
f1=fopen("input","r");   
f2=fopen("identifier","w");   
f3=fopen("specialchar","w");   
while((c=getc(f1))!=EOF){   
if(isdigit(c))   
{   
tokenvalue=c-'0';   
c=getc(f1);   
while(isdigit(c)){   
tokenvalue\*=10+c-'0';   
c=getc(f1);   
}   
num[i++]=tokenvalue;   
ungetc(c,f1);   
}   
else if(isalpha(c))   
{   
putc(c,f2);   
c=getc(f1);   
while(isdigit(c)||isalpha(c)||c=='\_'||c=='$')   
{   
putc(c,f2);   
c=getc(f1);   
}   
putc(' ',f2);   
ungetc(c,f1);   
}   
else if(c==' '||c=='\t')   
printf(" ");   
else   
if(c=='\n')   
lineno++;   
else   
putc(c,f3);   
}   
fclose(f2);   
fclose(f3);   
fclose(f1);   
printf("\nThe no's in the program are");   
for(j=0;j<i;j++)   
printf("%d",num[j]);   
printf("\n");   
f2=fopen("identifier","r");   
k=0;   
printf("The keywords and identifiersare:");   
while((c=getc(f2))!=EOF){   
if(c!=' ')   
str[k++]=c;   
else   
{   
str[k]='\0';   
keyword(str);   
k=0;   
}   
}   
fclose(f2);   
f3=fopen("specialchar","r");   
printf("\nSpecial characters are");   
while((c=getc(f3))!=EOF)   
printf("%c",c);   
printf("\n");   
fclose(f3);   
printf("Total no. of lines are:%d",lineno);   
}

OUTPUT:-

**Input:**   
Enter Program $ for termination:   
{   
int a[3],t1,t2;   
t1=2; a[0]=1; a[1]=2; a[t1]=3;   
t2=-(a[2]+t1\*6)/(a[2]-t1);   
if t2>5 then   
print(t2);   
else {   
int t3;   
t3=99;   
t2=-25;   
print(-t1+t2\*t3); /\* this is a comment on 2 lines \*/   
} endif   
}   
(cntrl+z)

**Output:**

**Variables :** a[3] t1 t2 t3   
**Operator :** - + \* / >   
**Constants :** 2 1 3 6 5 99 -25   
**Keywords :** int if then else endif   
**Special Symbols :** , ; ( ) { }    
**Comments :** this is a comment on 2 lines