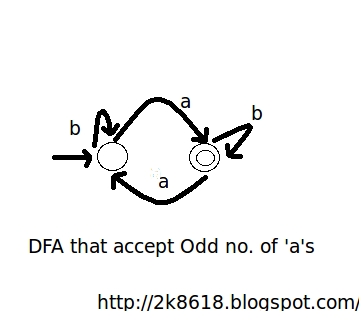
* [Implement finite automata: Odd number of a](http://2k8618.blogspot.com/2011/03/implement-finite-automata-in-lex-odd-no.html)



program:  
  
%{   
  
%}   
  
reg (b|ab\*a)\*ab\*    
  
%%   
  
{reg}      printf("%s Accepted",yytext);   
.\*     printf("%s Not Accepted",yytext);   
  
  
%%   
  
main()   
{   
yylex();   
return 0;   
}   
  
Input:x.c   
a   
ab   
aab   
aba   
abba   
  
Output:   
students@ccflab-desktop:~$ lex b.l   
students@ccflab-desktop:~$ gcc lex.yy.c -lfl   
students@ccflab-desktop:~$ ./a.out<x.c   
a Accepted   
ab Accepted   
aab Not Accepted   
aba Not Accepted   
abba Not Accepted   
students@ccflab-desktop:~$

…………………………………………………………………………………………………………….

**NUMBER EACH LINE - LEX PROGRAM - COMPILER DESIGN**

**Program**:   
  
%{  
int lineno =1;  
%}  
  
line .\*\n  
  
%%  
{line} {printf("%5d %s", lineno++,yytext);}  
  
%%  
main()  
{  
yylex();  
return 0;  
}  
  
  
**Output**:   
input: compilerdesign.txt  
  
Design of a Lexical Analyzer using Finite Automation  
Design of lexical analyzer using LEX  
Design of recursive descent and LL (1) parsers  
Implementation of Operator precedence Parsing  
Design of parser for arithmetic expressions using YACC  
Design of a simple type checker  
Generation of IC for arithmetic expressions  
Simple code optimization strategies  
Design of a code generator   
Writing a simple Compiler   
                    www.2k8618.blogspot.com  
                     
                     
nn@linuxmint ~ $ lex l1.l  
nn@linuxmint ~ $ gcc lex.yy.c -lfl  
nn@linuxmint ~ $ ./a.out <compilerdesign.txt  
    1 Design of a Lexical Analyzer using Finite Automation  
    2 Design of lexical analyzer using LEX  
    3 Design of recursive descent and LL (1) parsers  
    4 Implementation of Operator precedence Parsing  
    5 Design of parser for arithmetic expressions using YACC  
    6 Design of a simple type checker  
    7 Generation of IC for arithmetic expressions  
    8 Simple code optimization strategies  
    9 Design of a code generator   
   10 Writing a simple Compiler   
   11                     www.2k8618.blogspot.com

…………………………………………………………………………………………………………………………………………………………..

**COMMENT REMOVAL - LEX PROGRAM - COMPILER DESIGN**

**Program**:   
  
%{  
     
%}  
  
comment1    \/\\*(.|\n)\*\\*\/  
comment2    \/\/.\*  
  
%%  
  
{comment1}    ;  
{comment2}    ;  
.|\n        ECHO;  
  
%%  
main()  
{  
    yylex();  
    return 0;  
}  
  
**Output**:  
  
( input file: c1.c  
  
#include<stdio.h>  
main ()  
{  
    int i,n,fact=1;  
    printf("Enter the number: ");  
    scanf("%d",&n);            //inputing the number  
    for(i=1;i<=n;i++)        /\* finding factorial \*/  
    {  
        fact = fact\*i;  
    }  
    printf("Factorial=%d\n",fact);  
  
}  
)  
  
  
nn@linuxmint ~ $ lex l2.lex  
nn@linuxmint ~ $ gcc lex.yy.c -lfl  
nn@linuxmint ~ $ ./a.out <c1.c  
#include<stdio.h>  
main ()  
{  
    int i,n,fact=1;  
    printf("Enter the number: ");  
    scanf("%d",&n);              
    for(i=1;i<=n;i++)          
    {  
        fact = fact\*i;  
    }  
    printf("Factorial=%d\n",fact);  
  
}  
nn@linuxmint ~ $

………………………………………………………………………………………………………………………………………………

**REPLACE SEQUENCES OF WHITE SPACES BY A SINGLE BLANK CHARACTER - LEX PROGRAM - COMPILER DESIGN**

**Elimination of Multiple Spaces ,Tabs & Emptylines**

**Program**:  
  
%{  
      
%}  
  
space [ \t]  
emptyline \n  
%%  
  
{space}+ printf(" ");  
{emptyline}+ printf("\n");  
. ECHO;  
  
%%  
  
main()  
{  
  
    yylex();  
}  
**Output**:  
  
**// input: sum.c**  
  
#include<stdio.h>  
main()  
{  
  
  
//program to add two numbers  
int a,      b,c;  
scanf("%d%d",&a,&b);  
c=a+b;/\* finding sum &  
printing\*/  
printf("sum=%d",c);  
}  
  
  
  
nn@linuxmint ~ $ lex mw.l  
nn@linuxmint ~ $ gcc lex.yy.c -ll  
nn@linuxmint ~ $ ./a.out<sum.c  
#include<stdio.h>  
main()  
{  
//program to add two numbers  
int a, b,c;  
scanf("%d%d",&a,&b);  
c=a+b;/\* finding sum &  
printing\*/  
printf("sum=%d",c);  
}  
nn@linuxmint ~ $

…………………………………………………………………………………………………………………………………………………….

**COUNT THE NUMBER OF OCCURRENCES OF A PARTICULAR WORD OR LETTER - LEX PROGRAM - COMPILER DESIGN**

**Program**:  
  
%{  
    int icount=0,factcount=0;  
%}  
  
%%  
  
fact    factcount++;  
i         icount++;  
(.|\n)    ;  
  
%%  
main()  
{  
    yylex();  
    printf("Count of \"fact\"= %d \nCount of letter 'i' = %d\n",factcount,icount);     
    return 0;  
}  
  
**Output**:  
  
  
nn@linuxmint ~ $ lex l4.l  
nn@linuxmint ~ $ gcc lex.yy.c -ll  
nn@linuxmint ~ $ ./a.out<c1.c  
Count of "fact"= 5   
Count of letter 'i' = 17  
nn@linuxmint ~ $   
  
  
// **Input file: c1.c**  
#include<stdio.h>   
main ()   
{   
    int i,n,fact=1;   
    printf("Enter the number: ");   
    scanf("%d",&n);            //inputing the number   
    for(i=1;i<=n;i++)        /\* finding factorial \*/   
    {   
        fact = fact\*i;   
    }   
    printf("Factorial=%d\n",fact);   
  
}

…………………………………………………………………………………………………………………………………………………….

**REMOVE ALL THE OCCURRENCES OF A WORD OR LETTER - LEX PROGRAM - COMPILER DESIGN**

**Program**:  
  
%{  
     
  
%}  
  
%%  
"printf"    ;  
(.|\n)        ECHO;  
%%  
main()  
{  
yylex();  
return 0;  
}  
  
**Output**:  
( input file: c1.c  
#include<stdio.h>  
main ()  
{  
    int i,n,fact=1;  
    printf("Enter the number: ");  
    scanf("%d",&n);            //inputing the number  
    for(i=1;i<=n;i++)        /\* finding factorial \*/  
    {  
        fact = fact\*i;  
    }  
    printf("Factorial=%d\n",fact);  
  
}  
)  
  
  
nn@linuxmint ~ $ lex l5.lex  
nn@linuxmint ~ $ gcc lex.yy.c -lfl  
nn@linuxmint ~ $ ./a.out <c1.c  
#include<stdio.h>  
main ()  
{  
    int i,n,fact=1;  
    ("Enter the number: ");  
    scanf("%d",&n);            //inputing the number  
    for(i=1;i<=n;i++)        /\* finding factorial \*/  
    {  
        fact = fact\*i;  
    }  
    ("Factorial=%d\n",fact);  
  
}  
  
  
nn@linuxmint ~ $

………………………………………………………………

**COUNT THE NUMBER OF LINES,WORDS AND CHARACTERS - LEX PROGRAM - COMPILER DESIGN**

**Program**:  
%{  
      
    int lnno=0,wordno=0,charno=0;  
%}  
  
word [.\* .\*\t]  
eol [\n]  
  
%%  
{word}    {wordno++; charno+=yyleng;}  
{eol}     {charno++;lnno++;wordno++;}  
.    {charno++;}  
%%  
main()  
{  
    yylex();  
    printf("Line number= %d\n",lnno);  
    printf("Word number= %d\n",wordno);  
    printf("Character number= %d\n",charno);  
    return 0;  
}  
  
**Output**:  
  
nn@linuxmint ~ $ lex l6.l  
nn@linuxmint ~ $ gcc lex.yy.c -lfl  
nn@linuxmint ~ $ ./a.out <c1.c  
Line number= 13  
Word number= 43  
Character number= 232  
nn@linuxmint ~ $   
( input file : c1.c  
  
#include<stdio.h>  
main ()  
{  
    int i,n,fact=1;  
    printf("Enter the number: ");  
    scanf("%d",&n);            //inputing the number  
    for(i=1;i<=n;i++)        /\* finding factorial \*/  
    {  
        fact = fact\*i;  
    }  
    printf("Factorial=%d\n",fact);  
  
}  
)

…………………………………………………….

**SELECT LINES STARTING WITH 'M' - LEX PROGRAM - COMPILER DESIGN**

**Program**:  
  
%{  
%}  
reg1 ^m.\*\n  
  
%%  
{reg1} {ECHO;}  
. ;  
%%  
**Output**:  
  
( input file : c1.c  
  
#include<stdio.h>  
main ()  
{  
    int i,n,fact=1;  
    printf("Enter the number: ");  
    scanf("%d",&n);            //inputing the number  
    for(i=1;i<=n;i++)        /\* finding factorial \*/  
    {  
        fact = fact\*i;  
            }  
    printf("Factorial=%d\n",fact);  
}  
  
)  
  
  
nn@linuxmint ~ $ lex l7.lex  
nn@linuxmint ~ $ gcc lex.yy.c -lfl  
nn@linuxmint ~ $ ./a.out <c1.c  
  
main ()  
  
  
  
  
  
  
  
  
  
  
  
nn@linuxmint ~ $

………………………………………………………………

**COUNT THE NUMBER OF LINES ENDING WITH "COM" - LEX PROGRAM - COMPILER DESIGN**

**Program:**  
  
//**Lex file: com.l**  
  
%{  
int count=0;  
%}  
DIGIT [0-9]  
ALPHA [a-zA-Z]  
%%  
({ALPHA}|{DIGIT})\*com {count++;}  
%%  
  
main()  
{  
    yylex();  
    printf("Count= %d\n",count);  
    return 0;  
      
}  
  
**Output**:  
  
n@linuxmint ~ $ lex com.l  
nn@linuxmint ~ $ gcc lex.yy.c -ll  
nn@linuxmint ~ $ ./a.out<com.txt  
www.2k8618.blogspot.  
www.2k8cs.tk  
www.google.  
www.gmail.  
  
Count= 3  
nn@linuxmint ~ $

……………………………………………………….

**COUNT THE VOWELS - LEX PROGRAM**

**Program:**  
  
// **lex file: vw.l**  
  
%{  
int count=0;  
%}  
  
%%  
  
[aeiouAEIOU] {count++;ECHO;}  
  
%%  
main()  
{  
    yylex();  
    printf("\nNumber of vowels= %d\n",count);  
    return 0;  
}  
  
**Output:**  
  
nn@linuxmint ~ $ lex vw.l  
nn@linuxmint ~ $ gcc lex.yy.c -ll  
nn@linuxmint ~ $ ./a.out<vw.txt  
www.2k8618.blogspot.com  
www.2k8cs.tk  
www.google.com  
www.gmail.com  
  
  
Number of vowels= 10  
nn@linuxmint ~ $

……………………………………….

**CONVERT LOWERCASE TO UPPERCASE & REVERSE - LEX PROGAM - COMPILER DESIGN**

**CHANGE CASE - LEX - PROGRAM**

**Program:**  
  
// **Lex file: cap.l**  
  
  
lower [a-z]  
CAPS  [A-Z]  
space    [ \t\n]  
  
%%  
{lower}         {printf("%c",yytext[0]- 32);}  
{CAPS}        {printf("%c",yytext[0]+ 32);}  
{space}        ECHO;  
.                    ECHO;  
%%  
  
main()  
{  
    yylex();  
      
}  
  
**Output:**  
nn@linuxmint ~ $ lex cap.l  
nn@linuxmint ~ $ gcc lex.yy.c -ll  
nn@linuxmint ~ $ ./a.out<tst.txt  
WWW.2K8618.BLOGSPOT.COM  
sanjana    jamsheena chaithanya neethu  
GOVINDAPRASAD VIPIN ADARSH SHIVIN  
baby brinda kavya helen  
SALMAN TINU RICHARD  SIBIN   
SHIVIN laji NABEEL  
www.2k8cse.cu.cc  
nn@linuxmint ~ $   
  
// **tst.txt**  
  
www.2k8618.blogspot.com  
SANJANA    JAMSHEENA CHAITHANYA NEETHU  
govindaprasad vipin adarsh shivin  
BABY BRINDA KAVYA HELEN  
salman tinu richard  sibin   
shivin LAJI nabeel  
WWW.2K8CSE.CU.CC

…………………………………………………………….

**COUNT THE POSITIVE NUMBERS, NEGATIVE NUMBERS & FRACTIONS - LEX PROGRAM - COMPILER DESIGN**

**Aim:**  
Write a lex program to count the number of Positive numbers, Negative numbers & Fractions.  
  
**Program:**  
  
// **lex file: a.l**  
  
%{  
    int postiveno=0;  
    int negtiveno=0;  
    int positivefractions=0;  
    int negativefractions=0;  
%}  
  
DIGIT [0-9]  
%%  
  
\+?{DIGIT}+                              postiveno++;  
-{DIGIT}+                                  negtiveno++;  
  
\+?{DIGIT}\*\.{DIGIT}+            positivefractions++;  
-{DIGIT}\*\.{DIGIT}+                negativefractions++;  
. ;      
%%  
  
main()  
{  
    yylex();  
    printf("\nNo. of positive numbers: %d",postiveno);  
    printf("\nNo. of Negative numbers: %d",negtiveno);  
    printf("\nNo. of Positive fractions: %d",positivefractions);  
    printf("\nNo. of Negative fractions: %d\n",negativefractions);  
}  
  
**Output:**  
  
nn@linuxmint ~ $ lex a.l  
nn@linuxmint ~ $ gcc lex.yy.c -ll  
nn@linuxmint ~ $ ./a.out<a.txt  
  
  
No. of positive numbers: 2  
No. of Negative numbers: 3  
No. of Positive fractions: 4  
No. of Negative fractions: 5  
nn@linuxmint ~ $  
  
// **Input file: a.txt**   
+12,-123,1.1,-1.1,12,-2,-3,2.1,3.2,5.1,-5.5,-6.1,-7.7,-8.8

………………………………………………………………………………………

**CONVERT DECIMAL NUMBER TO HEXADECIMAL NUMBER IN A FILE - LEX PROGRAM - COMPILER DESIGN**

**Program:**  
  
  
// **lex file: conv.l**  
  
%{  
    #include<stdio.h>  
    int num,r,digit=0,count,pcount=0,i;  
    char a[20];  
%}  
DIGIT [0-9]  
  
%%  
  
{DIGIT}+ {    num=atoi(yytext);  
        while(num!=0)  
        {  
            r=num%16;  
            digit='0'+r;  
            if(digit>'9')  
            digit+=7;  
            a[count++]=digit;  
            num=num/16;  
        }  
        for(i=count-1;i>=pcount;--i)  
                printf("%c",a[i]);  
                pcount=count;  
        }  
.|\n    ECHO;  
          
%%  
main()  
{  
    yylex();  
    return 0;  
}        
  
**Output:**  
nn@linuxmint ~ $ lex conv.l  
nn@linuxmint ~ $ gcc lex.yy.c -lfl  
nn@linuxmint ~ $ ./a.out<c.txt  
DEEPAK A  
F HUNAIF  
NAVITHA 10  
15 RAJINA   
ABID 18  
1A SANITHA  
nn@linuxmint ~ $  
  
  
// **c.txt**  
  
DEEPAK 10  
15 HUNAIF  
NAVITHA 16  
21 RAJINA   
ABID 24  
26 SANITHA

………………………………………………………

**SELECT LINES ENDING WITH 'COM' - LEX PROGRAM - COMPILER DESIGN**

**Program:**  
  
// Lex file: com.l  
  
%{  
int count=0;  
%}  
  
%%  
.\*com\n {count++;ECHO;}  
. ;  
  
%%  
  
main()  
{  
    yylex();  
    printf("\nCount= %d\n",count);  
    return 0;  
      
}  
  
**Output:**  
nn@linuxmint ~ $ lex com.l  
nn@linuxmint ~ $ gcc lex.yy.c -ll  
nn@linuxmint ~ $ ./a.out<com.txt  
  
www.google.com  
www.yahoo.com  
www.2k8618.blogspot.com  
  
  
  
  
Count= 3  
nn@linuxmint ~ $  
  
// **com.txt**  
  
www.2k8cs.tk  
www.google.com  
www.yahoo.com  
www.2k8618.blogspot.com  
www.2k8cse.cu.cc

……………………………………………………..

**COUNT OF WORDS STARTING WITH A - LEX PROGRAM - COMPILER DESIGN**

**Program:**  
  
// **Lex file: aa.l**  
  
%{  
        int count=0;  
%}  
alpha    [a-zA-Z]  
digit      [0-9]  
space    [ \t\n]  
start      ^a  
%%  
  
{start}                                           {count++;}  
{space}(a|A)({alpha}|{digit})\*    {count++;}  
.                                                      ;  
  
%%  
  
main()  
{  
    yylex();  
    printf("count= %d\n",count);  
}  
**Output:**  
  
nn@linuxmint ~ $ lex aa.l  
nn@linuxmint ~ $ gcc lex.yy.c -ll  
nn@linuxmint ~ $ ./a.out<tst.txt  
  
count= 6  
nn@linuxmint ~ $   
  
  
  
// **tst.txt**  
  
afsal ARIFA aaa www.2k8618.blogspot.com  
aiswarya saranya    sooraj  
arun reshmi  
a www.2k8cse.cu.cc