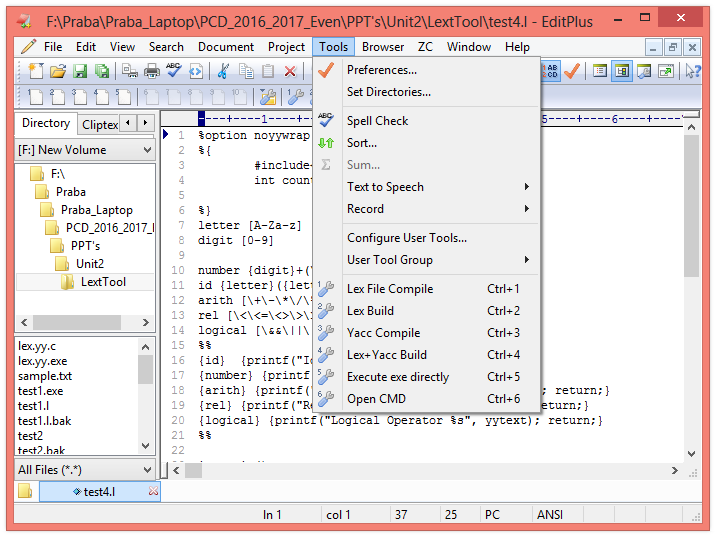
LEX – Tool for Lexical analyzer

Lex environment can be linux or windows. If windows, install Flex tool – IDE – create new lex file with an extension test1.l. Lef File Compile – Lex Build – Execute exe direclly.

If linux, lex and yacc tools are built in, follow the steps to execute

* lex test1.l
* cc lex.yy.c
* ./.aout

Snapshot Flex Tool



**Example 1: Counting the identifier tokens**

%{ #include<stdio.h>

int count=0;

%}

letter [A-Za-z]

digit [0-9]

id {letter}({letter}|{digit})\*

%%

{id} {count++; printf("Count = %d", count);}

\+|\-|\\*|\/|% {printf("%s arithmetic operators",yytext);}

%%

int main()

{

int i, n;

printf("Enter the number \n");

scanf("%d", &n);

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

{

if(i<n)

{

printf("Enter the string \n");

yylex();

}

else

break;

}

printf("Number of identifiers = %d", count);

return 0;

}

**Example 2 – Program for identifying and printing the lexemes and their types**

%{

#include<stdio.h>

int count=0;

%}

letter [A-Za-z]

digit [0-9]

number {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

id {letter}({letter}|{digit})\*

arith [\+\-\\*\/\%]

rel [\<\<=\<>\>\>=\=]

logical [\&&\||\!]

%%

{id} {printf("Identifier %s", yytext); return;}

{number} {printf("Number %s", yytext); return;}

{arith} {printf("Arithmetic Operator %s", yytext); return;}

{rel} {printf("Relationl Operator %s", yytext); return;}

{logical} {printf("Logical Operator %s", yytext); return;}

%%

int main()

{

int i, n;

n=1;

while(n==1)

{

printf("\n Enter the input \t");

yylex();

printf("\n Do you want to continue \t ");

scanf("%d",&n);

}

return 0;

}

**Example 3 – Program to return the relevant tokens**

%{

#include<stdio.h>

#include<conio.h>

#include <y.tab.h>

int count=0;

#define ID 1

#define NUM 2

#define REL 3

#define LT 4

%}

letter [A-Za-z]

digit [0-9]

number {digit}+(\.{digit}+)?(E[+-]?{digit}+)?

id {letter}({letter}|{digit})\*

arith [\+\-\\*\/\%]

rel [\<\<=\<>\>\>=\=]

logical [\&&\||\!]

%%

{id} {return ID;}

{number} {return NUM;}

{arith} {printf("Arithmetic Operator %s", yytext); return;}

{rel} { yyval=LT; return REL;}

{logical} {printf("Logical Operator %s", yytext); return;}

%%

int main()

{

switch(yylex())

{

case 1:

printf("Identifier");

break;

case 2:

printf("Number");

break;

case 3:

printf("Relational Operator");

if(yyval==4)

printf("< Operator");

break;

default:

break;

}

getch();

return 0;

}