

LAB ASSIGNMENT

Name – Akshit Jain

Reg. No – 21BRS1088

Question – Create lexical analyzer using lex language

> Reading the first and follow c code to find different tokens

Code

```
%{
int COMMENT=0;
}%
identifier [a-zA-Z][a-zA-Z0-9]*
%%
#.* {printf ("\n %s is a Preprocessor Directive",yytext);}
int |
float |
main |
if |
else |
printf |
scanf |
for |
char |
getch |
while {printf ("\n %s is a Keyword",yytext);}
"/*" {COMMENT=1;}
"*/*" {COMMENT=0;}
{identifier}\( {if(!COMMENT) printf ("\n Function:\t %s",yytext);}
\{ {if(!COMMENT) printf ("\n Block Begins");}
\} {if(!COMMENT) printf ("\n Block Ends");}
{identifier}(\[[0-9]*\])? {if(!COMMENT) printf ("\n %s is an Identifier",yytext);}
\".*" {if(!COMMENT) printf ("\n %s is a String",yytext);}
[0-9]+ {if(!COMMENT) printf ("\n %s is a Number",yytext);}
\(\)|\)? {if(!COMMENT) printf ("\t");ECHO;printf ("\n");}
\ ( ECHO;
= {if(!COMMENT) printf ("\n%s is an Assmt oprtr",yytext);}
\<= |
\>= |
\< |
== {if(!COMMENT) printf ("\n %s is a Rel. Operator",yytext);}
.\n
%%
int main(int argc, char **argv)
{
if(argc>1)
{
FILE *file;
file=fopen(argv[1],"r");
```

```

if(!file)
{
printf("\n Could not open the file: %s",argv[1]);
exit(0);
}
yyin=file;
}
yylex();
printf("\n\n");
return 0;
}
int yywrap()
{
return 0;
}

```

OUTPUT

The screenshot shows a code editor with two files open: `firststandfollow.c` and `21brs1088_lexicalanalyzer.l`. The `21brs1088_lexicalanalyzer.l` file contains the following code:

```

1  %lex
2  %option noyywrap
3  %include "stdio.h"
4  %include "string.h"
5  %include "stdlib.h"
6  %include "unistd.h"
7  %include "sys/types.h"
8  %include "fcntl.h"
9  %include "sys/stat.h"
10 %include "sys/time.h"
11 %include "sys/unistd.h"
12 %include "sys/wait.h"
13 %include "sys/socket.h"
14 %include "sys/poll.h"
15 %include "sys/select.h"
16 %include "sys/mman.h"
17 %include "sys/shm.h"
18 %include "sys/sem.h"
19 %include "sys/signal.h"
20 %include "sys/uio.h"
21 %include "sys/xattr.h"
22 %include "sys/zfs.h"
23 %include "sys/zfs_ioctl.h"
24 %include "sys/zfs_mount.h"
25 %include "sys/zfs_util.h"
26 %include "sys/zfs_vfsops.h"
27 %include "sys/zfs_vnops.h"
28 %include "sys/zfs_zfsops.h"
29 %include "sys/zfs_zfsops.h"
30 %include "sys/zfs_zfsops.h"
31 %include "sys/zfs_zfsops.h"
32 %include "sys/zfs_zfsops.h"
33 %include "sys/zfs_zfsops.h"
34 %include "sys/zfs_zfsops.h"
35 %include "sys/zfs_zfsops.h"
36 %include "sys/zfs_zfsops.h"
37 %include "sys/zfs_zfsops.h"
38 %include "sys/zfs_zfsops.h"
39 %include "sys/zfs_zfsops.h"
40 %include "sys/zfs_zfsops.h"
41 %include "sys/zfs_zfsops.h"
42 %include "sys/zfs_zfsops.h"
43 %include "sys/zfs_zfsops.h"
44 %include "sys/zfs_zfsops.h"
45 %include "sys/zfs_zfsops.h"
46 %include "sys/zfs_zfsops.h"
47 %include "sys/zfs_zfsops.h"
48 %include "sys/zfs_zfsops.h"
49 %include "sys/zfs_zfsops.h"
50 %include "sys/zfs_zfsops.h"
51 %include "sys/zfs_zfsops.h"
52 %include "sys/zfs_zfsops.h"
53 %include "sys/zfs_zfsops.h"
54 %include "sys/zfs_zfsops.h"
55 %include "sys/zfs_zfsops.h"

```

The output of the lexical analyzer is shown in a terminal window, displaying the following tokens and their classifications:

```

%includestdio.h> is a Preprocessor Directive
#includestring.h> is a Preprocessor Directive
int is a Keyword
i is an Identifier
j is an Identifier
l is an Identifier
m is an Identifier
n is an Identifier
= is an Asmnt oprtr
0 is a Number
o is an Identifier
p is an Identifier
nv is an Identifier
z is an Identifier
= is an Asmnt oprtr
0 is a Number
x is an Identifier
= is an Asmnt oprtr
0 is a Number
char is a Keyword
str[10] is an Identifier
temp is an Identifier
temp[10] is an Identifier
temp[20] is an Identifier
ptr is an Identifier
struct is an Identifier
nread is an Identifier

```

```
student@614: ~/Desktop
File Edit View Search Terminal Help
-> $pt
student@614:~/Desktop$ lex 21brs1088_lexicalanalyzer.l
student@614:~/Desktop$ cc lex.yy.c
student@614:~/Desktop$ ./a.out firstandfollow.c

#include<stdio.h> is a Preprocessor Directive
#include<string.h> is a Preprocessor Directive
int is a Keyword
i is an Identifier
j is an Identifier
l is an Identifier
m is an Identifier
n is an Identifier
= is an Assmt oprtr
0 is a Number
o is an Identifier
p is an Identifier
nv is an Identifier
z is an Identifier
= is an Assmt oprtr
0 is a Number
x is an Identifier
= is an Assmt oprtr
0 is a Number
char is a Keyword
str[10] is an Identifier
temp is an Identifier
temp2[10] is an Identifier
temp3[20] is an Identifier
ptr is an Identifier
struct is an Identifier
nrod is an Identifier
```