

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CD Practical 4

### Write Lex Program to

- 1) Identify integer, Float and Exponential numbers
- 2) Identify Single and Multiline comments in C program
- 3) Identify valid tokens in given statement :

```
scanf("%d %d",&a,&b);  
printf("%d %d",a,b);
```

### Code :

```
%{  
#include <stdio.h>  
%}  
  
%%  
  
[+-]?[0-9]+                { printf("Integer: %s\n", yytext); }  
  
[+-]?[0-9]*\.[0-9]+?  { printf("Float: %s\n", yytext); }  
  
[+-]?[0-9]*\.[0-9]+([eE][+-]?[0-9]+)? { printf("Exponential:  
%s\n", yytext); }  
  
"/*"([^\*]|\\*+[^*/])*\*+/"    { printf("Comment: %s\n",  
yytext); }  
  
"//".*                    { printf("Comment: %s\n",  
yytext); }
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CD Practical 4

```
[a-zA-Z_][a-zA-Z0-9_]*      { printf("Valid Token: %s\n",
yytext); }

\"([^\\""]|\\.)*\"          { printf("Valid Token: %s\n",
yytext); }

[\\(\\)\\[\\]\\{\\}\\+\\-\\*\\/\\=\\>\\<\\!\\&\\|\\%\\^\\;\\,\\.\\.\\? ] { printf("Valid
Token: %s\n", yytext); }

\" { printf("Valid Token: %s\n", yytext); }

' { printf("Valid Token: %s\n", yytext); }

\\n { }

[ \\t]+ { }

. { printf("Unrecognized Character: %s\n", yytext); }

%%

int yywrap() {
    return 1;
}

int main() {
    printf("\n");
    yyin = fopen("p4", "r");
    yylex();
}
```

Name - Yash Lakhtariya  
Enrollment number - 21162101012  
Branch - CBA      Batch - 71  
CD Practical 4

```
    return 0;  
}
```

Output :

The screenshot shows a code editor on the left and a terminal on the right. The code editor displays a C program for a lexical analyzer using flex. The program includes `<stdio.h>` and defines several regular expressions for tokens: integers, floats, exponentials, comments, identifiers, and literals. It uses `printf` to output the type of each token. The terminal shows the command to compile the program with `flex p4.l` and `gcc lex.yy.c`, followed by the execution of the resulting `a.out` binary. The output lists various valid tokens: `scanf`, `(`, `"d %d"`, `,`, `&`, `a`, `;`, `b`, `)`, `printf`, `(`, `"d %d"`, `,`, `a`, `,`, `b`, `)`, and `;`.

```
1 2 4 13 A ~/D/s/c/p4_p5 07:45 AM 6% 0.13GB 54°C 49°C 40% 30 98%  
p4.l u x p5.l u  
1 %{  
2 #include <stdio.h>  
3 %}  
4  
5 %%  
6  
7 [+]?[0-9]+ { printf("Integer: %s\n", yytext); }  
8  
9 [+]?[0-9]*\.[0-9]+? { printf("Float: %s\n", yytext); }  
10  
11 [+]?[0-9]*\.[0-9]+([eE][+-]?[0-9]+)? { printf("Exponential: %s\n", yytext); }  
12  
13 /*{([^\n]|\\n|\\t|\\\"|\\'|\\\\\\\\)*\\\" { printf("Comment: %s\n", yytext); }  
14  
15 /*{([^\n]|\\n|\\t|\\\"|\\'|\\\\\\\\)*\\\" { printf("Comment: %s\n", yytext); }  
16  
17 [a-zA-Z_][a-zA-Z0-9_]* { printf("Valid Token: %s\n", yytext); }  
18  
19 \"([^\n]|\\n|\\t|\\\"|\\'|\\\\\\\\)*\" { printf("Valid Token: %s\n", yytext); }  
20  
21 \\([^\n]|\\n|\\t|\\\"|\\'|\\\\\\\\)*\\ { printf("Valid Token: %s\n", yytext); }  
22  
23 \\\\ { printf("Valid Token: %s\n", yytext); }  
24  
25 \\' { printf("Valid Token: %s\n", yytext); }  
26  
27 \\\\n { }  
28  
29 [ \\\\t]+ { }  
30  
31 . { printf("Unrecognized Character: %s\n", yytext); }  
32  
33 %%  
34  
35 int yywrap() {  
36     return 1;  
37 }  
38
```

```
ysl ~ 07:44 cd "/home/ysl/Documents/sem7practicals/CD/p4_p5/" && flex p4.l &&  
gcc lex.yy.c && ./a.out  
  
Valid Token: scanf  
Valid Token: (  
Valid Token: "d %d"  
Valid Token: ,  
Valid Token: &  
Valid Token: a  
Valid Token: ;  
Valid Token: b  
Valid Token: )  
Valid Token: ;  
Valid Token: printf  
Valid Token: (  
Valid Token: "d %d"  
Valid Token: ,  
Valid Token: a  
Valid Token: ,  
Valid Token: b  
Valid Token: )  
Valid Token: ;  
  
ysl ~/p4_p5 main ? v14.2.1 07:44
```