

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CD Practical 5

Aim : To implement Lexical Analyzer in LEX Tool for the input file.

Test file :

```
#include <stdio.h>
#include <conio.h>

void main()
{
    // Declaration of variable
    int a, b = 1000, c, i = 10;
    char x, y;
    char a = 'x';
    float p = 10.2, q = 20.5;
    scanf("%d %d", &a, &b);
    /*
    Addition of Two number
    */
    c = a + b;
    printf("Sum:%d", c);

    // Comment1
    if (a > b)
    {
        printf("a is max");
    }
    else
    {
        printf("b is max");
    }
}
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CD Practical 5

```
}  
a = b++ + c++;  
a += b;  
b = c && a;  
  
// print 1 to 100  
  
for (i = 1; i < 100; i++)  
{  
    printf("%d", i);  
}  
}
```

Code :

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

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CD Practical 5

```
%s\n", yytext); valid++;}
```

```
int|char|float|void|main|if|else|for|else[ ]if|for|scanf|printf  
{ printf("Keywords: %s\n", yytext); valid++;}
```

```
"/*"([^\*]|\\*+([^\*/]))*\*+ "/"      {   }
```

```
"//".*      {   }
```

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

```
\"([^\\""]|\\\.)*\"      { printf("String: %s\n", yytext);  
valid++;}
```

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

```
\+|\+|\-|\-|\+|=|\+|=|\*|=|\*|=|\/=|\%=|\&|=|\|=|\^|=|\!|=|\|=|\leq|\>=|\&&|\||\|  
{ printf("Operators: %s\n", yytext); valid++; }
```

```
\#.*      { printf("Header: %s\n", yytext);  
valid++;}
```

```
\'([^\\"']|\\\.)*\'      { printf("String: %s\n", yytext);  
valid++;}
```

```
\n      {   }
```

Name - Yash Lakhtariya

Enrollment number - 21162101012

Branch - CBA Batch - 71

CD Practical 5

```
[ \t]+                { }
```



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



```
%%
```



```
int yywrap() {
    return 1;
}
```



```
int main() {
    printf("\n");
    yyin = fopen("p5.c", "r");
    yylex();
    printf("Valid Tokens: %d\n", valid);
    return 0;
}
```

Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 71
CD Practical 5

Output :

[illegible]

The image is a composite of two screenshots. The left screenshot shows a code editor with a C++ program. The program includes `<stdio.h>` and defines a variable `valid=0`. It contains several `printf` statements and conditional logic to validate input. The right screenshot shows the output of the program in a terminal window. The output lists various tokens categorized by type: Operators, Operators/Brackets, Identifier, Keywords, and Valid Tokens. The categories and their counts are: Operators: ++, Operators/Brackets: ;, Identifier: a, Operators: +=, Identifier: b, Operators/Brackets: ;, Identifier: b, Operators/Brackets: =, Identifier: c, Operators: &&, Identifier: a, Operators/Brackets: ;, Keywords: for, Operators/Brackets: (, Identifier: i, Operators/Brackets: =, Integer: 1, Operators/Brackets: ;, Identifier: 1, Operators/Brackets: ;, Identifier: 1, Operators/Brackets: <, Integer: 100, Operators/Brackets: ;, Identifier: i, Operators: ++, Operators/Brackets:), Operators/Brackets: {, Operators/Brackets: {, Keywords: printf, Operators/Brackets: (, String: "%n", Operators/Brackets: }, Identifier: i, Operators/Brackets:), Operators/Brackets: ;, Operators/Brackets: }, Operators/Brackets: }, Operators/Brackets: }, Valid Tokens: 125.