Government College of Engineering, Jalgaon
Department of Computer Engineering
EXPERIMENT NO:- 2

Name of Student:- Ishwari Rajiv Narkhede.

Batch :-B4

PRN :- 1841034

Date of Performance: - **Date of completion:-**

TITLE: - Program for Calculating the Size of Program. (Line of Code Measure)

AIM: - To measure the size of program in Line of Code Measure.

REQUIRED S/W: - C or C++, PC

THEORY:-

<u>Traditional Code measures</u>: -

The most common measure of source code program length is number of lines of code but some lines of code are different from other. e.g. many program easier to read it line of code are being use to estimate programming effort then blank, line implements a difficult algorithm similarly comment line improve program understandability & they centrally require some effort as the code itself many different scheme have been proposed for counting lines we must take grate case to clarify what we may take in counting and how we are counting it in particular we must explain how each of following is handled:

- Blank Space
- Comment line
- Data declaration
- Lines

That contains several separate instructions

We recommend that the number of comment lines can be measured & recorded

There are two major types of LOC measures:

Physical SLOC (LOC): Specific definitions of these two measures vary, but the most common definition of physical SLOC is a count of lines in the text of the program's source code including comment lines. Blank lines are also included unless the lines of code in a section consist of more than 25% blank lines. In this case blank lines in excess of 25% are not counted toward lines of code.

Logical SLOC (LLOC): Logical SLOC attempts to measure the number of executable "statements", but their specific definitions are tied to specific computer languages (one simple logical SLOC measure for C-like programming languages is the number of statement-terminating semicolons). It is much easier to create tools that measure physical SLOC, and physical SLOC definitions are easier to explain. However, physical SLOC measures are sensitive to logically irrelevant formatting and style conventions, while logical SLOC is less sensitive to formatting and style conventions. However, SLOC measures are often stated without giving their definition, and logical SLOC can often be significantly different from physical SLOC.

Total length (loc):- NCLOC + CLOC some useful indirect measures as follows:

Ratio= CLOC/LOC

This measures the density of the commented line of code.

Note: NCLOC- Non-Comment Line of Code.

CLOC-Comment Line of Code.

*** Dealing with No-textual code or External code:-**

There is another accept of language dependence that causes a program object oriented development also suggest new measure length found that a count of object.

Predicting Length: - Since source code is relatively easy to analysis and length is a well studied attribute that is required by many prediction models. It is desirable to predict length as early as possible in lines and always having benefit good; stable length measure

For calculating line of code we have to first find the Statement type and the statements and then we can measure the line of code of the program.

For Example:

Executable, non executable: 202, 0

Declarative: 19

Compiler directives, Comments: 1, 2

Blank lines: 4

Finally we have developed the program for calculating the line of code of the given program or the software.

CONCLUSION:-

Hence , we learned how to measure the size of program in Line of Code Measure.

Page :
Date : / /

LOC count:-
example:-
1) Write a program addition of two numbers.

```
#include<stdio.h>
int main
{
    int a=10;
    int b=10;
    int c;

    c=a+b          //Calculating sum of no's.
    printf("%d+%d = %d", a,b,c);
    return 0;
}
```

physical lines = 10
logical lines = 8
comment lines = 1
Blank line = 1

2) Consider this snippet of C code as an example of the ambiguity encountered when determining SLOC:

```
for (i=0;i<100;i++)
 printf("hello");    /* How many lines of
                        code is this? */
```

In this example, we have:-

1 = physical line of code (LOC)

Page :
Date : / /

2 = logical lines of code (LLOC)
1 = comment line

3) example:-

```
/* Now how many lines of code is this? */
for(i=0;i<100;i++)
{
    printf("hello");
}
```

In this example we have,

physical lines of code (LOC) = 4
logical lines of code (LLOC) = 2
Comment line = 1.

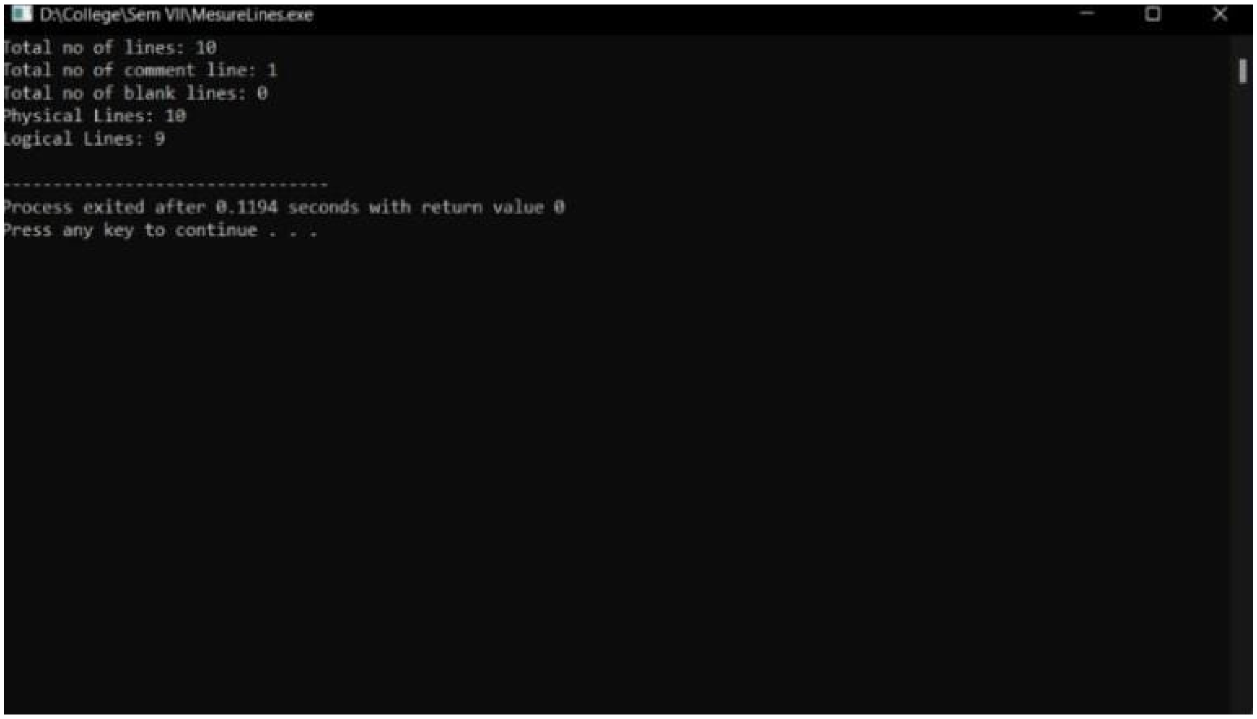
PROGRAM :-

```
 #include <stdio.h>
int main()
{
int line_count = 0, n_o_c_l = 0, n_o_n_b_l = 0, n_o_b_l = 0, n_e_c = 0;
FILE *fp1;
char ch;
char * arrr = "program.c";
fp1 = fopen(arrr, "r");
while ((ch = fgetc(fp1))!= EOF)
{
if (ch == '\n')
{
line_count++;
}
if (ch == '\n')
{
if ((ch = fgetc(fp1)) == '\n')
{fseek(fp1, -1, 1);
n_o_b_l++;
}
}
if (ch == ';')
{
if ((ch = fgetc(fp1)) == '\n')
{
fseek(fp1, -1, 1);
n_e_c++;
}
}
}
fseek(fp1, 0, 0);
while ((ch = fgetc(fp1))!= EOF)
{
if (ch == '/')
{
if ((ch = fgetc(fp1)) == '/')
{
n_o_c_l++;
}
}
}
printf("Total no of lines: %d\n", line_count);
printf("Total no of comment line: %d\n", n_o_c_l);
printf("Total no of blank lines: %d\n", n_o_b_l);
printf("Physical Lines: %d\n", line_count-n_o_b_l);
printf("Logical Lines: %d\n",line_count-n_o_b_l-n_o_c_l);
return 0;}
```

File:

```
#include<stdio.h>
int main()
{
int a=40;
int b=80;
int c;
c=a+b;//calculation
printf("%d+%d=%d",a,b,c);
return 0;
}
```

OUTPUT :-



D:\College\Sem VII\MesureLines.exe

```
Total no of lines: 10
Total no of comment line: 1
Total no of blank lines: 0
Physical Lines: 10
Logical Lines: 9

--------------------------------
Process exited after 0.1194 seconds with return value 0
Press any key to continue . . .
```