

**International School**

**Software Measurements & Analysis**

**CMU-CS 462 JIS**

**Project Report**

**Last update: May , 2023**

**Length – LOC tool**

**Submitted by:**

**Minh, Vo Duc**

**Dung, Nguyen Thi Xuan**

**Ty, Nguyen Van**

**Tan, Nguyen Van**

**Approved by:**

**MSc Man, Nguyen Duc**

**PROJECT INFORMATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project acronym** | LOC | | |
| **Project Title** | Length – LOC Tool | | |
| **Start Date** | May 10, 2023 | **End Date** | May 31, 2023 |
| **Lead Institution** | International School, Duy Tan University | | |
| **Project Mentor** | MSc Man, Nguyen Duc | | |
| **Project Leader & Contact Details** | Minh, Vo Duc  Email: [voducminh0611@gmail.com](mailto:voducminh0611@gmail.com)  Tel: 0935119003 | | |
| **Partner Organization** | Duy Tan University | | |
| **Project Web URL** |  | | |
| **Team members** | Minh, Vo Duc  (MSSV: 26211236188) | minhvoduc878@gmail.com | 0935119003 |
| Dung, Nguyen Thi Xuan  (MSSV: 26201141758 | xuandung0704@gmail.com | 0934993489 |
| Ty, Nguyen Van  (MSSV: 26211236049) | nguyenvanty1812@gmai.com | 0396696223 |
| Tan, Nguyen Van  (MSSV:26211236051) | vantan.nguyen1812@gmail.com | 0979550389 |

|  |  |
| --- | --- |
| Name | Task |
| Minh, Vo Duc | **- Document :** the algorithm and UI, basic theory  **- Code :** frm\_Main |
| Dung, Nguyen Thi Xuan | **- Document :** the algorithm, basic theory, testcase  **- Code:** frm\_Main, frm\_Menu, frm\_Home |
| Ty, Nguyen Van | **- Code:** frm\_Menu, frm\_Main  **- Document:** testcase, basic theory |
| Tan, Nguyen Van | **- Code:** frm\_About Us, frm\_Home  **- Document:** basic theory, testcase |

Table of Contents

[I. The theory of measurement: 4](#_Toc136409356)

[1. Definition: 4](#_Toc136409357)

[2. Measurement methods: 5](#_Toc136409358)

[a. Physical LOC (Lines of Code) 5](#_Toc136409359)

[b. Logical SLOC (also known as LLOC) 6](#_Toc136409360)

[c. Executable SLOC 6](#_Toc136409361)

[3. Advantages of LOC Metrics: 6](#_Toc136409362)

[a. Scope for automation of counting: 7](#_Toc136409363)

[b. Estimating project size and complexity: 7](#_Toc136409364)

[c. Measuring productivity: 7](#_Toc136409365)

[d. Evaluating software quality: 7](#_Toc136409366)

[e. Identifying potential bugs: 8](#_Toc136409367)

[f. Compliance with industry standards: 8](#_Toc136409368)

[4. Disadvantages of LOC Metrics: 8](#_Toc136409369)

[a. Limited scope: 8](#_Toc136409370)

[b. Difference in languages: 8](#_Toc136409371)

[c. Narrow perspective: 9](#_Toc136409372)

[d. Quality concerns: 9](#_Toc136409373)

[e. Lack of user-friendliness: 9](#_Toc136409374)

[5. Variations of LOC: 9](#_Toc136409375)

[6. REFERENCES: 10](#_Toc136409376)

[II. Demo: 10](#_Toc136409377)

[1. Tool: 10](#_Toc136409378)

[a. UI: 10](#_Toc136409379)

[b. Algorithm: 10](#_Toc136409380)

[2. Testcase 10](#_Toc136409381)

[3. Work breakdown table: 13](#_Toc136409382)

I. The theory of measurement:

Source lines of code (SLOC), also known as lines of code (LOC), is a software metric used to measure the size of a software program by counting the number of lines in the text of the program's source code. SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or maintainability once the software is produced. According to Boehm et al’s survey on cost estimation approaches, size metric is used as an essential input for most of cost estimation models. For example, COCOMO, SLIM, SEER-SEM, and PriceS all use SLOC; Checkpoint and other functionality-based models use function points or other functional sizing units as size input.

**1. Definition:**

- Line of code (LOC) refers to the number of individual lines of programming code in a software application or program.

- It is a common metric used by software developers and managers to measure the size, complexity, and productivity of software development projects.

- NCLOC and CLOC are measurement metrics used in software development to count the lines of code in a program.

+ NCLOC: non-commented source line of code or effective lines of code (ELOC).

+ CLOC: commented source line of code

- By measuring NCLOC and CLOC separately we can define:

Total length (LOC) = NCLOC + CLOC

- The units of LOC are:

+ KLOC- Thousand lines of code

+ NLOC- Non comment lines of code

+ KDSI- Thousands of delivered source instruction

- LOC can be used to estimate project timelines, allocate resources, and assess the quality of the code. It is also used in compliance with industry standards and regulations.

- While it is not a perfect measure of software development progress or quality, it remains a widely used metric in the industry.

**2. Measurement methods:**

- There are two main types of LOC metrics: Physical LOC and Logical LOC. Physical LOC counts all lines of code, including comments and blank lines, while Logical LOC counts only the lines of code that contribute to the functionality of the software. Other variations of LOC metrics include Executable LOC, which counts only the lines of code that are compiled and executed by the computer, and Source Lines of Code per Function Point (SLOC/FP), which measures the amount of code required to implement a single function point in a software system.

a. Physical LOC (Lines of Code)

- A measure of the number of lines of code in a software system.

- It counts all lines of code, including comments and blank lines, and is often used as a metric to estimate the size and complexity of a software project.

- However, it is important to note that SLOC alone is not a comprehensive measure of software quality or productivity, and should be used in conjunction with other metrics and analysis techniques.

b. Logical SLOC (also known as LLOC)

- A measure of SLOC. It counts the number of statements or logical lines of code in a program, which includes all executable statements, declarations, and comments.

- Logical SLOC is used to measure the size and complexity of a software system and is often used for estimating development effort, testing effort, and maintenance effort.

- However, it does not take into account the physical structure of the code or the number of lines actually written, which can vary depending on coding style and formatting.

c. Executable SLOC

- It counts the number of lines of code that contain executable instructions, such as statements and expressions that perform specific operations or calculations.

- This measure is often used to estimate the size and complexity of a software system, as well as to track progress during development and testing.

- However, it may not provide a complete picture of the code's overall structure and readability, as it does not include comments or other non-executable code.

**3. Advantages of LOC Metrics:**

*"Measuring software productivity by lines of code is like measuring progress on an airplane by how much it weighs."- Bill Gates*

a. Scope for automation of counting:

- LOC is a physical entity, manual counting effort can be easily eliminated by automating the counting process. Small utilities may be developed for counting the LOC in a program.

b. Estimating project size and complexity:

- LOC can be used as a measure of the size and complexity of a software project. This information is useful for project planning, scheduling, and resource allocation.

c. Measuring productivity:

- By tracking the number of lines of code produced by developers over time, managers can assess their productivity and identify areas for improvement.

d. Evaluating software quality:

- The number of lines of code can be an indicator of software quality. For example, if a program has a large number of lines of code but produces very little output, it may be an indication that the code is inefficient or poorly written.

e. Identifying potential bugs:

- Large programs with many lines of code are more likely to have bugs than smaller programs. By counting the number of lines of code, developers can identify areas that may need more testing or debugging.

f. Compliance with industry standards:

- Some industries have standards that require software projects to meet certain size or complexity criteria. Counting the number of lines of code can help ensure compliance with these standards.

**4. Disadvantages of LOC Metrics:**

a. Limited scope:

- LOC only measures the size of the program’s code and does not take into account other important aspects of the software development process, such as requirements, design, testing, or documentation.

b. Difference in languages:

- Counting lines of code is not always an accurate way to measure the complexity or quality of software. Different programming languages and coding styles can result in vastly different line counts for similar functionality. Consider two applications that provide the same functionality one is written in Java and the other is written in C#:

+ The number of function points would be exactly the same, but aspects of the application would be different.

+ The lines of code needed to develop the application would certainly not be the same. As a consequence, the amount of effort required to develop the application would be different (hours per function point).

+ Unlike Lines of Code, the number of Function Points will remain constant

c. Narrow perspective:

- LOC characterizes only one aspect of size, namely length, and does not account for functionality or complexity. Therefore, a program with a smaller LOC may be more complex and harder to maintain than a program with a larger LOC.

d. Quality concerns:

- Using LOC as a sole metric for evaluating the productivity of programmers may encourage bad programming practices, such as code duplication or excessive comments, which can result in bloated and less maintainable code.

e. Lack of user-friendliness:

- LOC is a technical metric that may not be easily understandable for non-technical users, making it challenging to communicate the program’s size and complexity to stakeholders outside of the development team.

Overall, while LOC is a valuable tool for measuring the size of software programs, it is important to consider its limitations and complement it with other metrics to ensure a more comprehensive and accurate evaluation of the software development process.

**5. Variations of LOC:**

- Count of physical lines including blank lines.

- Count of all lines except blank lines and comments.

- Count of all statements except comments (statements taking more than one line count as only one line).

- Count of all lines except blank lines, comments, declarations and headings.

- Count of only executable statements, not including exception conditions.

**6. REFERENCES:**

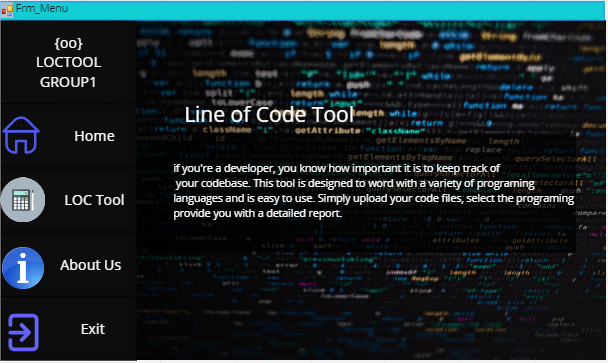
* https://tutorials.freshersnow.com/software-engineering/size-oriented-metrics/
* <https://en.wikipedia.org/wiki/Source_lines_of_code#:~:text=Source%20lines%20of%20code%20(SLOC,of%20the%20program's%20source%20code>.
* https://en.wikiversity.org/wiki/Software\_metrics\_and\_measurement

II. Demo:

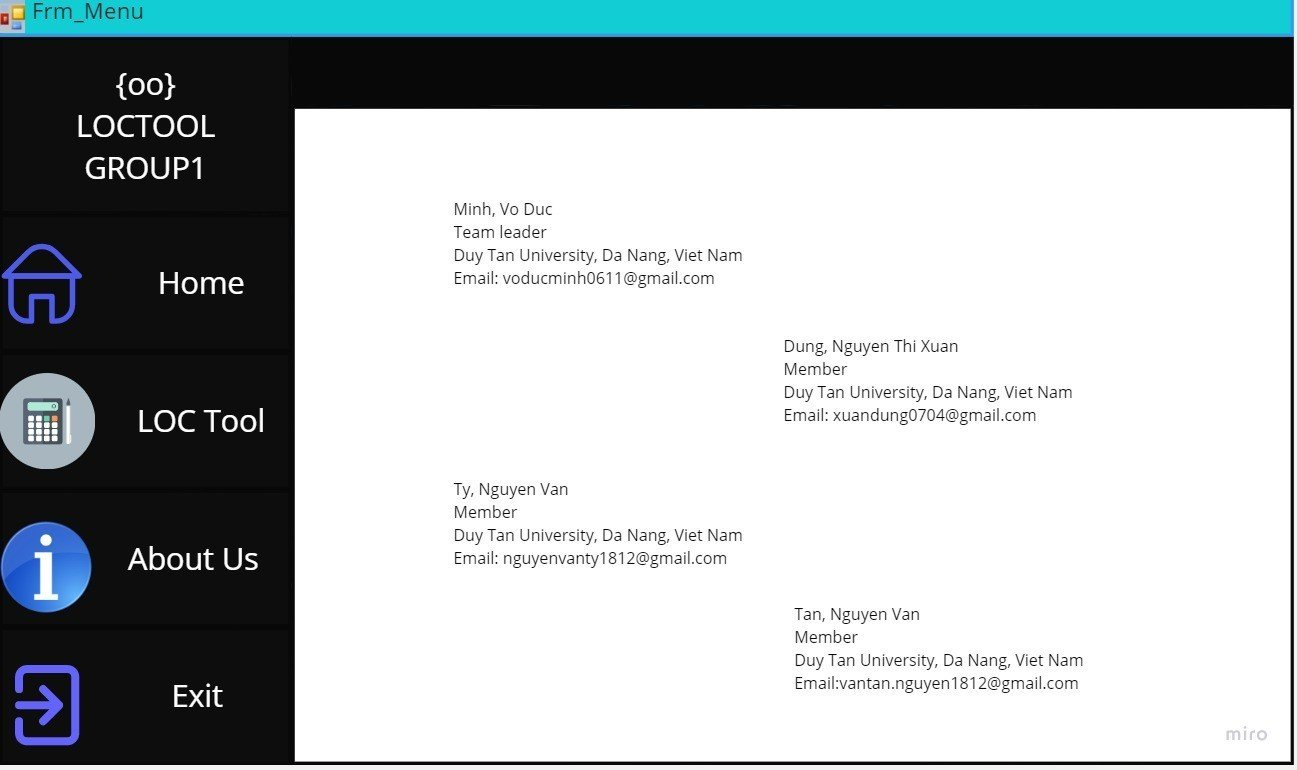
**1. Tool:**

a. UI:

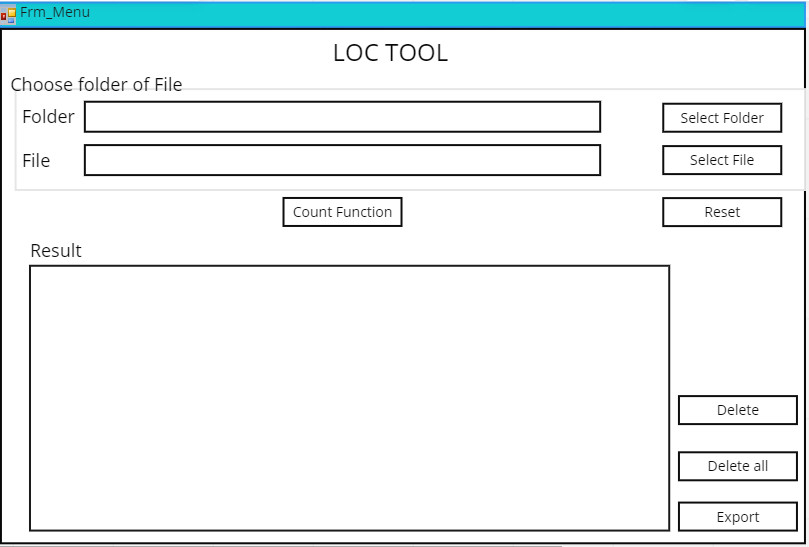
- Main Window



- Member Information



* Form\_LOCtool



b. Algorithm:

- Input: A code file

- Output: file name, LOC, Blank, Comment, Directory file

- Idea :

+ Create a function to check if it is a code file or not

+ If true, it will create a count function

- Algorithm :

+ B1: IsSourceFile function is a function to check if it is ile code or not? by convention .cs .cpp .js .java

+ B2: With the function to start counting

Give a variable associated with the path and create counters like blank, comment and the lines variable that counts the number of lines of code

Using StreamReader to read the entire code file to check if "" is encountered, the variable blank ++ and if // or /\* is encountered, then comment ++

Then display on ListView

+ B3: similar to reading folder

**2. Testcase**

|  |  |
| --- | --- |
| No | Function Name |
| 1 | Select Folder |
| 2 | Select File |
| 3 | Count Line in Folder |
| 4 | Count Line in File |
| 5 | Reset |
| 6 | Delete |
| 7 | Delete All |
| 8 | Export |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Case Description** | **Pre – Condition** | **Step** | **Excepted Output** | **Actual Output** | **Result** |
| TC01 | Select the folder containing c# files | Open the system | 1. Click LOC Tool button  2. Click Select Folder button  3. Select folder | Open folder successfully, show folder path on textbox | Open folder successfully, show folder path on textbox | Pass |
| TC02 | Select the folder containing c++ files | Open the system | 1. Click LOC Tool button  2. Click Select Folder button  3. Select folder | Open folder successfully, show folder path on textbox | Open folder successfully, show folder path on textbox | Pass |
| TC03 | Select the folder containing java files | Open the system | 1. Click LOC Tool button  2. Click Select Folder button  3. Select folder | Open folder successfully, show folder path on textbox | Open folder successfully, show folder path on textbox | Pass |
| TC04 | Invalid folder selection | Open the system | 1. Click LOC Tool button  2. Click Select Folder button | Open folder successfully, show folder path on textbox | Open folder successfully, show folder path on textbox | Pass |
| TC05 | Select file c# | Open the system | 1. Click LOC Tool button  2. Click Select File button  3. Select file | Open file successfully, show file path on textbox | Open file successfully, show file path on textbox | Pass |
| TC06 | Select file c++ | Open the system | 1. Click LOC Tool button  2. Click Select File button  3. Select file | Open file successfully, show file path on textbox | Open file successfully, show file path on textbox | Pass |
| TC07 | Select file java | Open the system | 1. Click LOC Tool button  2. Click Select File button  3. Select file | Open file successfully, show file path on textbox | Open file successfully, show file path on textbox | Pass |
| TC08 | Invalid file selection | Open the system | 1. Click LOC Tool button  2. Click Select File button  3. Select file | Open file successfully, show file path on textbox | Open file successfully, show file path on textbox | Pass |
| TC09 | Counting c#, c++ and java code files | Select Folder or File | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button | Success, display the result to listview | Success, display the result to listview | Pass |
| TC10 | Count lines of code with comments | Select Folder or File | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button | Success, display the result to listview | Success, display the result to listview | Pass |
| TC11 | Count lines of code without comments | Select Folder or File | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button | Success, display the result to listview | Success, display the result to listview | Pass |
| TC12 | Count lines of code with blank lines | Select Folder or File | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button | Success, display the result to listview | Success, display the result to listview | Pass |
| TC13 | Count lines of code without blank lines | Select Folder or File | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button | Success, display the result to listview | Success, display the result to listview | Pass |
| TC14 | Count lines of code contains only comments | Select Folder or File | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button | Success, display the result to listview | Success, display the result to listview | Pass |
| TC15 | Count lines of code contains only blank lines | Select Folder or File | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button | Success, display the result to listview | Success, display the result to listview | Pass |
| TC16 | Count lines of code without selecting a file or folder | Select Folder or File | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button | Failed, the screen displays asking to select folder or file first | Success, display the result to listview | Failed |
| TC17 | Count the number of lines of code in the file is not valid | Select Folder or File | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button | Failed, the display asks to select another folder or file | Program not working | Failed |
| TC18 | Reset data | Select Folder or File | 1. Click LOC Tool button  2. Click Reset button | Success | Success | Pass |
| TC19 | Delete item in listview | Have data on listview | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button  5. Click item  6. Click Delete button | Successful, the selected item has been deleted | Successful, the selected item has been deleted | Pass |
| TC20 | Delete item in empty listview | Have data on listview | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button  5. Click item  6. Click Delete button | Successful, the selected item has been deleted | Successful, the selected item has been deleted | Pass |
| TC21 | Delete multiple items in empty list view | Have data on listview | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button  5. Click item  6. Click Delete button | Successful, the selected item has been deleted | Failed, only the first selected item can be deleted | Fail |
| TC22 | Delete all item in listview | Have data on listview | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button  5. Click item  6. Click Delete button | Successful, the all item has been deleted. | Successful, the all item has been deleted. | Pass |
| TC24 | Delete all item in empty listview | Have data on listview | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Counting Function button  5. Click item  6. Click Delete button | Successful, the all item has been deleted. | Successful, the all item has been deleted. | Pass |
| TC25 | Export data | Have data on listview | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Export button | Successful | Successful | Pass |
| TC26 | Export data when item is deleted | Have data on listview | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Export button | Successful | Successful | Pass |
| TC27 | Export data when istview is empty | Have data on listview | 1. Click LOC Tool button  2. Click Select Folder or Select File button  3. Select folder or file  4. Click Export button | Successful | Successful | Pass |

**3. Work breakdown table:**

|  |  |  |
| --- | --- | --- |
| Task |  | Undertaker |
| Làm tài liệu |  |  |
|  | Mục lí thuyết cơ bản | Dung |
|  | Mục tool (UI, thuật toán) | Dung, Minh, Tân, Tỵ |
|  | Các testcase | Tân, Dung, Tỵ |
| Code |  |  |
|  | Tạo giao diện và code cho các form\_main, form\_Member | Dung |
|  | Code form LOC tool | Minh, Dung |