

Software Metrics Analyzer

SPL-1



About

Title :

Software Metrics Analyzer

By :

Nafiz Mahmud Fardin

BSSE-1528

Supervisor :

Dr.Mohammad Shoyaib

Description

This project is a **Software Metrics Analyzer** designed to evaluate C++ source files by analyzing their structure and content. It calculates various metrics, including size metrics (e.g., lines of code), comment metrics (e.g., comment density), and token metrics (e.g., counts of keywords, identifiers, operators, and delimiters). The tool supports processing multiple files, provides detailed insights into code quality and complexity, and visualizes the results for easy interpretation



Motivation

The motivation for this project stems from the growing need for efficient tools to assess code quality, maintainability, and complexity in software development. By analyzing key metrics such as line counts, token distributions, and comment density, developers can identify potential issues, optimize their code, and ensure adherence to best practices

Aim

This project aims to simplify the process of code evaluation, promote cleaner and more maintainable codebases, and provide valuable insights to both beginners and professionals striving to improve their programming skills and project quality.

Features

*Metrics analyzed (e.g.,
keywords, operators, lines
of code, comments).*

Multi-file support.

User-friendly visualization.

Tools

And

Languages

1.C++

2.VS Code

3.Libraries(e.g.
Regex,algorithm,filesystem,
iomanip etc)

—

Methodology

- 1. File Reading*
- 2. Tokenization*
- 3. Metrics Calculation*
- 4. Visualization*

DIFFICULTIES-

1. Tokenization Challenges

- Handling edge cases in tokenization, such as nested comments, string literals, or escaped characters.
- Differentiating between similar tokens like operators and delimiters.

2. Multi-Line Comments

- Accurately identifying and counting multi-line comments without errors.

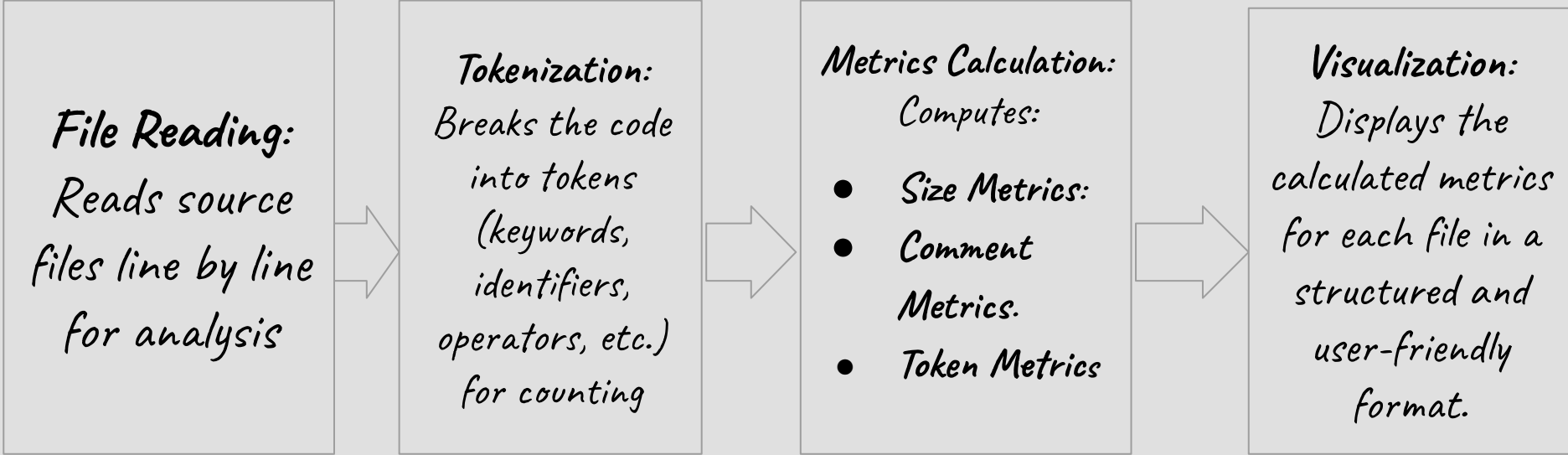
3. File Handling

- Reading and processing large files efficiently.
- Managing file paths and ensuring compatibility across operating systems.

4. Metrics Accuracy

- Ensuring the calculated metrics, such as line counts and token counts, are precise.
- Dealing with complex code structures that may affect metrics.

***File Reading:**
Reads source
files line by line
for analysis*



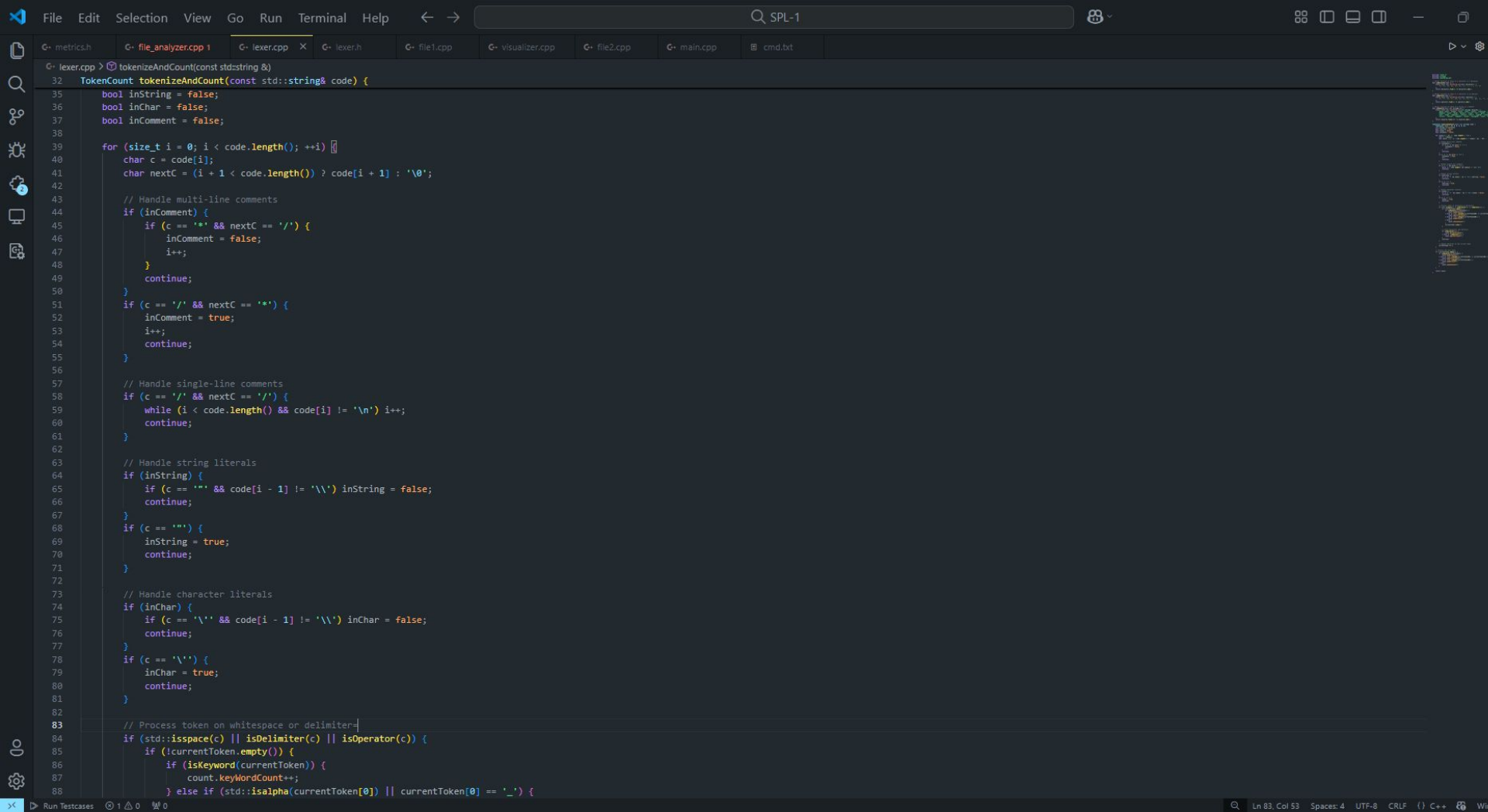
***Tokenization:**
Breaks the code
into tokens
(keywords,
identifiers,
operators, etc.)
for counting*

***Metrics Calculation:**
Computes:*

- *Size Metrics:*
- *Comment Metrics.*
- *Token Metrics*

***Visualization:**
Displays the
calculated metrics
for each file in a
structured and
user-friendly
format.*

Workflow



Main Entry to the Programme-

```
main.cpp > main(int, char * [])
1  #include "metrics.h"
2  #include "lexer.h"
3  #include <iostream>
4  #include <filesystem>
5  #include<fstream>
6  #include<string>
7  #include<numeric>
8  using namespace std;
9
10 int main(int argc, char* argv[])
11 {
12     if(argc<2)
13     {
14         cout << "Usage : " << argv[0] << " <file1.cpp> [file2.cpp]... " << endl;
15         return 1;
16     }
17     vector<FileMetrics> allMetrics;
18     for(int i=1;i<argc;i++)
19     {
20         string filename = argv[i];
21         if(!std::filesystem::exists(filename))
22         {
23             cerr << "File not found:" << filename << endl;
24             continue;
25         }
26         vector<string> lines = readFileLines(filename);
27         std::string code = accumulate(lines.begin(), lines.end(), std::string("\n"));
28         TokenCount metrics2 = tokenizeAndCount(code);
29         //TokenCount metrics2 = tokenizeAndCount(filename);
30         sizeMetrics metrics = calculateMetrics(lines);
31         CommentMetrics metrics1 = calculateComments(lines);
32         allMetrics.push_back({filename,metrics,metrics1,metrics2});
33     }
34     visualizemetrics(allMetrics);
35     return 0;
36
37 }
```

```
C:\Users\fardi\SPL-1>g++ -std=c++17 main.cpp file_analyzer.cpp visualizer.cpp lexer.cpp -o metrics_analyzer
```

```
C:\Users\fardi\SPL-1>metrics_analyzer.exe file1.cpp file2.cpp
```

```
CODE SIZE METRICS-
```

FileName	TotalLines	CodeLines	BlankLines	TotalCommentLines	Function	MultiCommentLines	Comment Ratio
file1.cpp	8	8	0	0	1	0	0.00%
file2.cpp	15	15	0	0	4	0	0.00%

```
TOKEN METRICS-
```

FileName	keyWords	Identifiers	Operators	Numbers	Delimiters	Unknowns
file1.cpp	6	6	5	5	13	0
file2.cpp	9	17	12	4	28	1

```
C:\Users\fardi\SPL-1>|
```

Future Plans for the Project:

1. **Support for Multiple Programming Languages:** Extend the analyzer to support languages like Python, Java, and JavaScript.
2. **Advanced Metrics:** Add deeper analysis, such as cyclomatic complexity, maintainability index, and Halstead metrics.
3. **Graphical User Interface (GUI):** Build an intuitive GUI for easier interaction and visualization of metrics.
4. **Real-Time Analysis:** Enable real-time code analysis as users write or edit their code.
5. **Integration with IDEs:** Create plugins for popular IDEs like VS Code or IntelliJ to provide metrics within the development environment.

GitHub LINK-

**[https://github.com/nafizfardin28/SPL-1.
git](https://github.com/nafizfardin28/SPL-1.git)**

THANK YOU!