Tecnológico de Monterrey

Campus Ciudad de México

E1 Syntax highlighter

Arturo Utrilla Hernández **A01174331**

Daniel Esteban Hernández García **A01652966**

Rodrigo Martínez Vallejo **A00573055**

Ximena Silva Bárcena **A01785518**

Prof. Sergio Ruiz Loza

Implementation of Computational Methods

Grupo **641**

Tuesday, June 10th, 2025

| Brief report |

The following measurements were taken running the program on all 5 source .cpp files containing different lengths in lines:

- **Measure 1**: 7 ms
- **Measure 2**: 3 ms
- **Measure 3**: 3 ms
- **Measure 4**: 3 ms

Average execution time: ***5.25 ms***

The lexer operates using a recursive algorithm that processes each character or token in the file. Since each recursive call processes one unit and advances through the input, the base complexity per file is O(n), where n is the number of lines (or tokens) in the file.

The total complexity of the program can be determined based on two key factors:

- **N**: the number of source files to process
- **L**: the average number of lines (or tokens) per file

Since each of the N files is processed independently and sequentially, and each file takes O(L) time to process, the overall time complexity of the algorithm is: **O(N × L)**

To test this complexity analysis, more measurements were taken by doubling and tripling the number of files processed:

- **10 files**:
  - Measures: 7 ms, 10 ms, 10 ms, 11 ms
  - Average: ***9.5 ms***
- **15 files**:
  - Measures: 15 ms, 16 ms, 17 ms, 15 ms
  - Average: ***15.75 ms***

This matches the observed execution behavior, where total processing time increases linearly with the size and number of input files.

| Ethical Reflection |

Arturo Utrilla Hernández:

The development of a lexical parser that tokenizes C++ code may appear to be a purely technical achievement, but it carries ethical implications, especially considering its potential applications. Such a tool can be used in compilers, code analysis systems, educational platforms, or even AI-based code generation and review systems. On the positive side, it can support accessibility to programming, help detect vulnerabilities early, and promote cleaner, more maintainable code.

Daniel Esteban Hernández García:

Lexers are a very important tool in the world of technology. They are widely used in areas like web scrapers, parsers, and natural language processing to tokenize human language for AI, for example. But there is one field that especially gets my attention: security tools. The purpose of the lexer we developed was just to recognize tokens from the C++ language, but lexers can also be used to scan code for malicious intent and help identify potential threats. If a lexer made for this task is poorly designed, it could leave big organizations vulnerable to malware attacks. In the best-case scenario, nothing happens, but what if the target is a government system or something related to public health? The damage could be serious, even life-threatening. That’s why we, as future professionals in tech, must learn to build reliable, high-quality tools. Not just lexers, but software in general. It's also our ethical duty to think beyond the code and consider how our work might affect others, both positively and negatively.

Rodrigo Martínez Vallejo:

Upon reflecting on the question, I firmly believe that any technology I develop as a future computer scientist carries ethical implications. On one hand, I must consider the potential uses of what I create. I recall a talk given by a software engineer who worked on major machine learning projects, including some for the U.S. military. While he didn't specify his exact role, he mentioned he had been working with distressing video material that affected him mentally. This example highlights how technology can be weaponized or used for destructive and immoral purposes. On the other hand, the technology developed in this project serves two clear ethical purposes: first, as a didactic tool (for education and learning), and second, as a purely functional solution to a real-world problem. This duality reminds me that, while technology can be misused, it can also empower, educate, and improve lives. As a developer, I have a responsibility to consider the societal impact of my work and strive to create systems that align with ethical principles. Ultimately, technology itself is neutral, but its application is not. The ethical burden falls on us, the creators, to ensure that what we build contributes positively to society.

Ximena Silva Bárcena:

The development of lexical analysis technology, such as this C++ lexer, serves as a fundamental component in automated code processing systems that carry significant societal implications. While lexical analyzers democratize software development by making programming tools more accessible and efficient, they simultaneously raise concerns about potential job displacement for junior developers whose routine coding tasks could be automated. The technology's ability to process multiple files in milliseconds enables rapid analysis that could accelerate software development cycles, yet this same efficiency may contribute to a skills gap where fundamental programming knowledge becomes less valued in favor of higher-level system design. Moreover, sophisticated code analysis tools built upon lexical analyzers introduce critical ethical questions regarding intellectual property and privacy in software development. The capacity to automatically parse and analyze vast codebases raises concerns about unauthorized analysis of proprietary code, potential copyright violations, and the perpetuation of algorithmic biases present in training data. As these systems become more prevalent, they create dependencies that could be exploited maliciously, potentially introducing security vulnerabilities across software systems that rely on automated parsing technologies, thus highlighting the need for careful consideration of both the benefits and risks inherent in advancing code analysis capabilities.