



Tecnológico de Monterrey

Syntax Highlighter (C++, C, JS)

This is a syntax-highlighter was made for the class Implementation of Computational Methods, using FLEX (fast lexical analyzer generator) and C++.

Author: José Carlos Martínez Núñez | A01639664.

Table of Contents

- Syntax Highlighter (C++, C, JS)
 - Table of Contents
 - Lexical Categories
 - Color Schemes
 - Dracula
 - Rainbow
 - Installation
 - Usage/Examples
 - The implemented algorithms and their execution time
 - Screenshots
 - Time Complexity
 - How does FLEX work?
 - Other Algorithms
 - Ethical Implications that this type of technology could have on society
 - References

Lexical Categories

The Lexical Categories supported by this program are:

- Preprocessor Keywords

- Reserved Words
- Types
- Operators
- Booleans
- Grouping Characters
- Multi Line Comments
- Single Line Comments
- Strings
- Function Names
- Class Identifiers
- Identifiers
- Package Names

These are all matched with their respective regular expressions in the **Lexer.l** file.














Color Schemes

The syntax-highlighter supports two color schemes for each lexical category:

Dracula

Token	Color
Preprocessor Keywords	■ #ff79c6
Reserved Words	■ #ff79c6
Types	■ #ff79c6
Operators	■ #ff79c6
Booleans	■ #bd93f9
Grouping Characters	■ #ffb86c
Multi Line Comments	■ #6272a4
Single Line Comments	■ #6272a4
Strings	■ #f1fa8c
Function Names	■ #50fa7b
Class Identifiers	■ #8be9fd
Identifiers	■ #f8f8f2
Package Names	■ #f1fa8c

Rainbow

Token	Color
Preprocessor Keywords	 #00e9b0
Reserved Words	 #feb300
Types	 #306cc9
Operators	 #00ff00
Booleans	 #18b646
Grouping Characters	 #e8002e
Multi Line Comments	 #81800c
Single Line Comments	 #ffff00
Strings	 #00aeae
Function Names	 #65ecb1
Class Identifiers	 #e4f4df
Identifiers	 #ba00ff
Package Names	 #903a47

Installation

```
# Generate Lexical Analyzer
flex Lexer.l
# Compile Generated Analyzer with the main.cpp
g++ -std=c++17 Lexer.cpp main.cpp -o "syntax-highlighter"
```

Usage/Examples

```
./syntax-highlighter FILE
```

The output file will be saved in **FILE.html**.

The implemented algorithms and their execution time

Once the program opens a file, the lexical analyzer will loop through each character, once a match is found the resulting string will be HTML encoded, once again looping through each matched

character and replacing every HTML specific symbol (<, >, &, etc.) with their respective alternatives. Then depending on the selected color scheme it'll assign a color to the specific matched string and write it to the output file.

The output of the program is an html document with the lexical categories in their respective colors. The console output will be the execution time of the program measured in milliseconds.

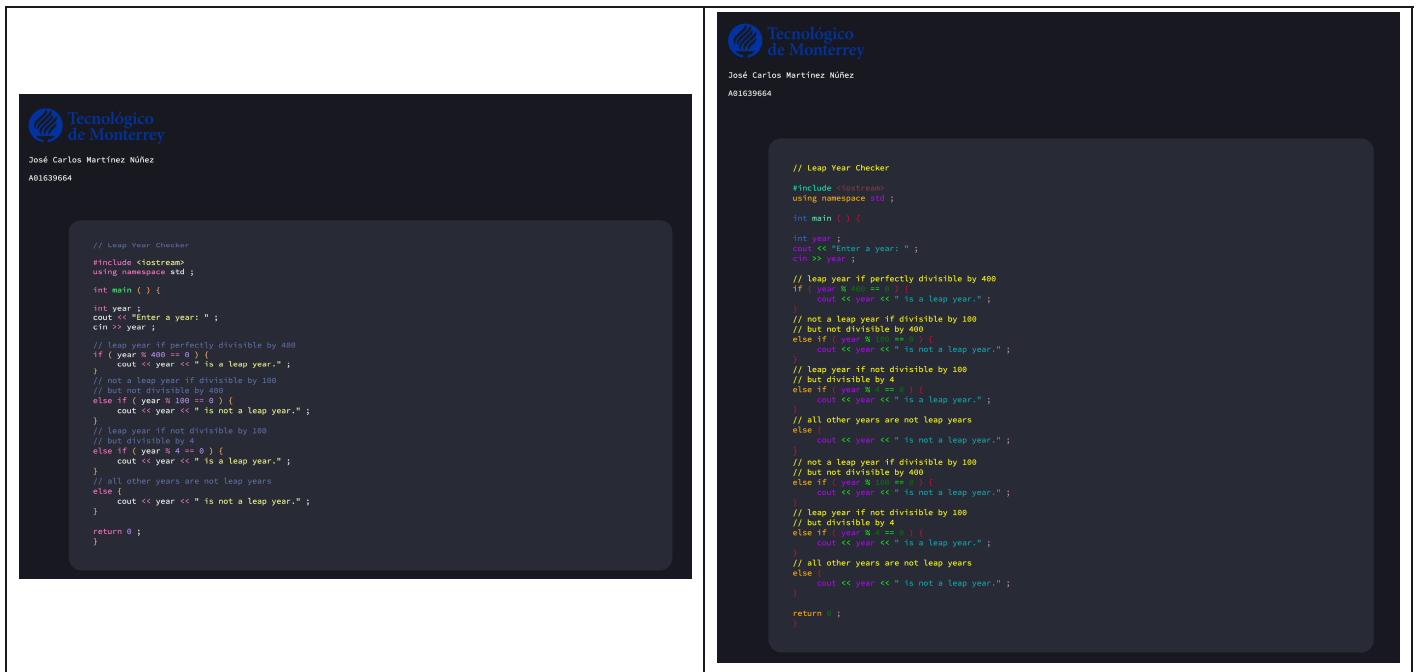
Note: The execution time of the program will vary depending on your computer specs.

Screenshots

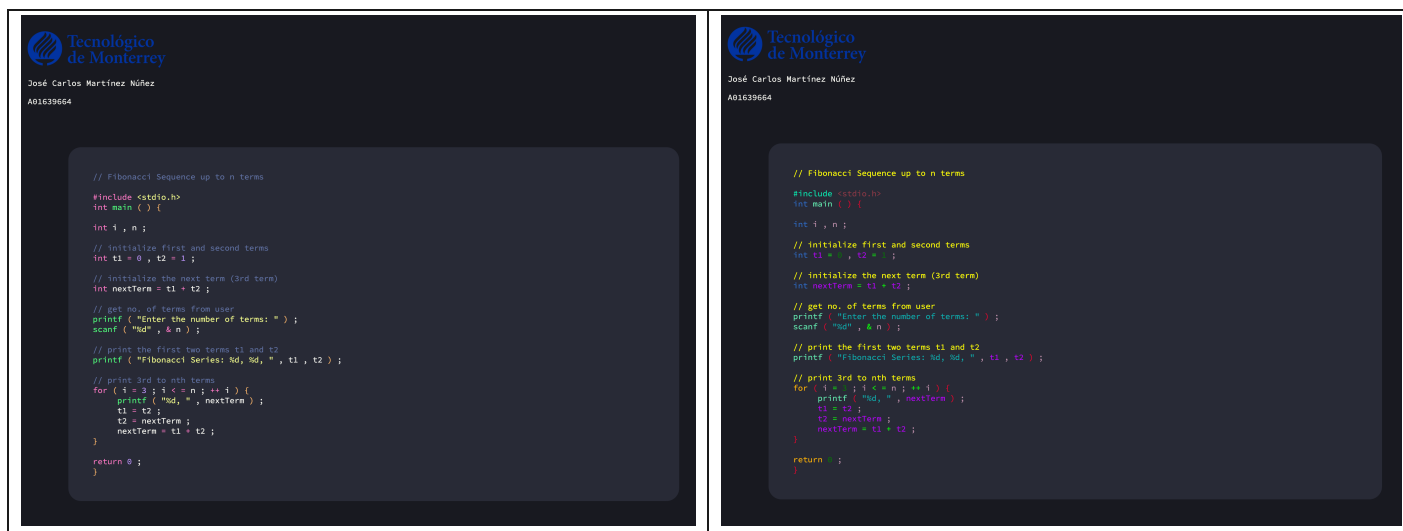
The following example files are located in the **examples** folder.



“example.js” Execution Time 2 milliseconds



“example.cpp” Execution Time 1 milliseconds



“example.c” Execution Time 1 milliseconds

Time Complexity

How does FLEX work?

Flex (fast lexical analyzer generator) is a free and open-source software alternative to lex. It is a computer program that generates lexical analyzers (also known as “scanners” or “lexers”).

A Flex lexical analyzer usually has time complexity $O(n)$ in the length of the input. That is, it performs a constant number of operations for each input symbol.

Other Algorithms

Apart from the code generated by the lexical analyzer once a token is matched we run another $O(n)$ algorithm to remove any HTML special characters.

This means that the total time complexity of the whole program is $O(n^2)$.

Nevertheless, as seen in the last section the program runs fairly fast.

Ethical Implications that this type of technology could have on society

This type of technology has many uses apart from creating syntax highlighters, technologies like Flex allow us to create lexical analyzers that can be used for all types of purposes from creating our own interpreters for other programming languages to automating lots of processes that require identification of tokens. One example of how these technologies can help society is analyzing laws to fix ambiguous wording or in the research field to analyze dead languages from our past so that

understand more about other civilizations. Performing all these tasks with the help of computers will greatly improve the amount of time it'll take to do this manually. It is incredibly important that technologies like these are used for good and not personal gain.

References

- Funchal, G. (2011, April 14). Most efficient way to escape XML/HTML in C++ string?. Stack Overflow. <https://stackoverflow.com/a/5665377>
- Levine, John R.; Mason, Tony; Brown, Doug (1992). *lex & yacc* (2nd ed.). O'Reilly. p. 279. ISBN 1-56592-000-7. A freely available version of lex is flex.