# Lexical Analyzer

## Project No 1

Kevin Lobo, Samantha Arburola, Víctor Chaves

Semester I 2017
Compilers and Interpreters
Computer Engineering

April 5, 2017

TEC | Tecnológico de Costa Rica

# Scanner

The lexical analysis of a source code consists of reading said source code and converting the characters in it into a sequence of valid tokens for a specific language (previously defined). In the case of this project, this scanner will analyze programs written in the C programming language, specifically the C99 standard, and said scanner will make use of the tool flex for generating said scanner, as said tool allows to quickly define a scanner by making use of regular expressions to catch all possible valid combinations of strings that form valid tokens.

# Tokens

The tokens produced by the scanner are as follows:

+ **Keyword:** Reserved words that are specific of the C language and by themselves have a meaning.

+ **Type:** The type of data that will be used in the context, like in a variable declaration. These are by definition keywords, but are treated as a different token for ease of using a different color in the output.

+ **Identifier:** A valid name for denoting an object, such as variables, functions, members of structures, enumerations, user types, and others.

+ **Constant:** As its name implies, these are values which are constant, such as numbers (both integers and floats) to give an example.

# Tokens

+ **StringLiteral:** A different type of constant, string literals are a sequence of characters (chars) of length zero or above, enclosed by double-quotes.

+ **Operator:** Specific punctuators that are used mostly in mathematical operations, such as the plus sign or the minus sign. As previously mentioned, these are still classified as punctuators, but for ease of coloring them differently, they are treated as their own classification.

+ **Punctuator:** Specific symbols or sequence of symbols which hold independent syntactic and semantic meaning, and completely dependent of context to mean something.

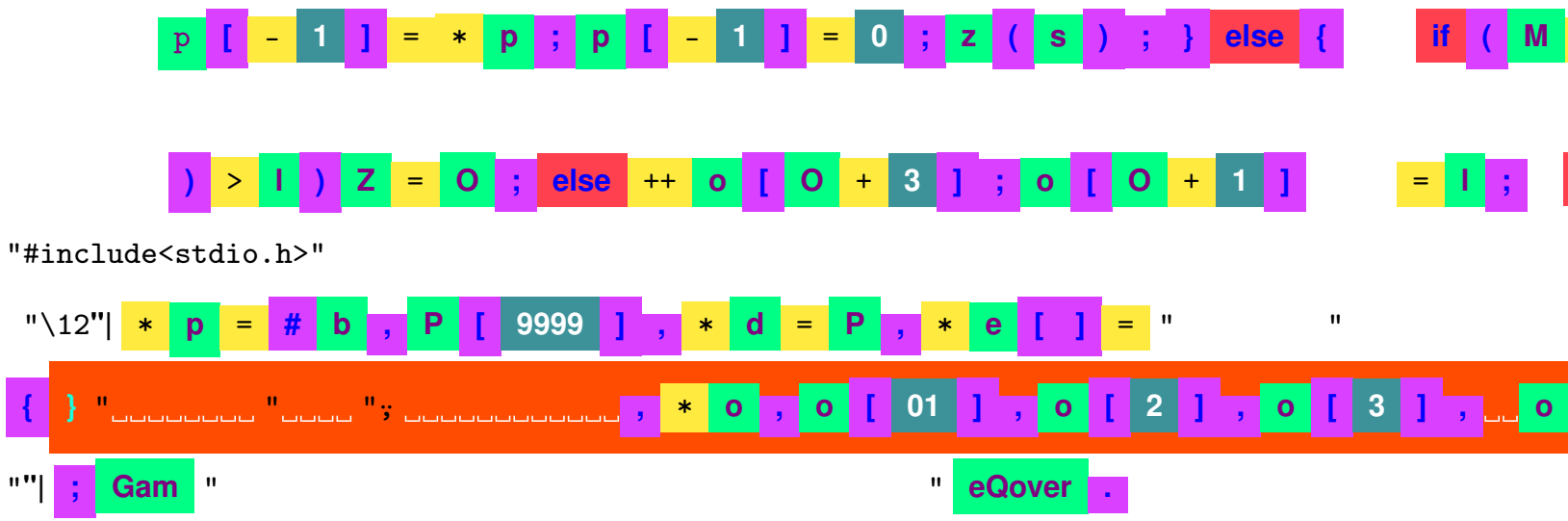
+ **Error:** A created token, this type is used to denote any sort of lexical error that may appear on the source code.

# Tokens

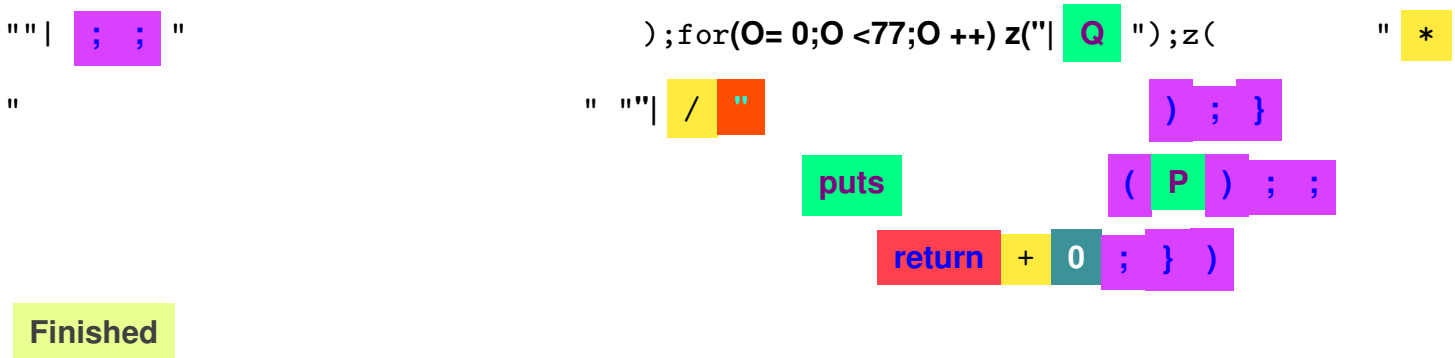

TOKENS
Keyword
Type
Identifier
Constant
StringLiteral
Operator
Punctuator
Error

# test09

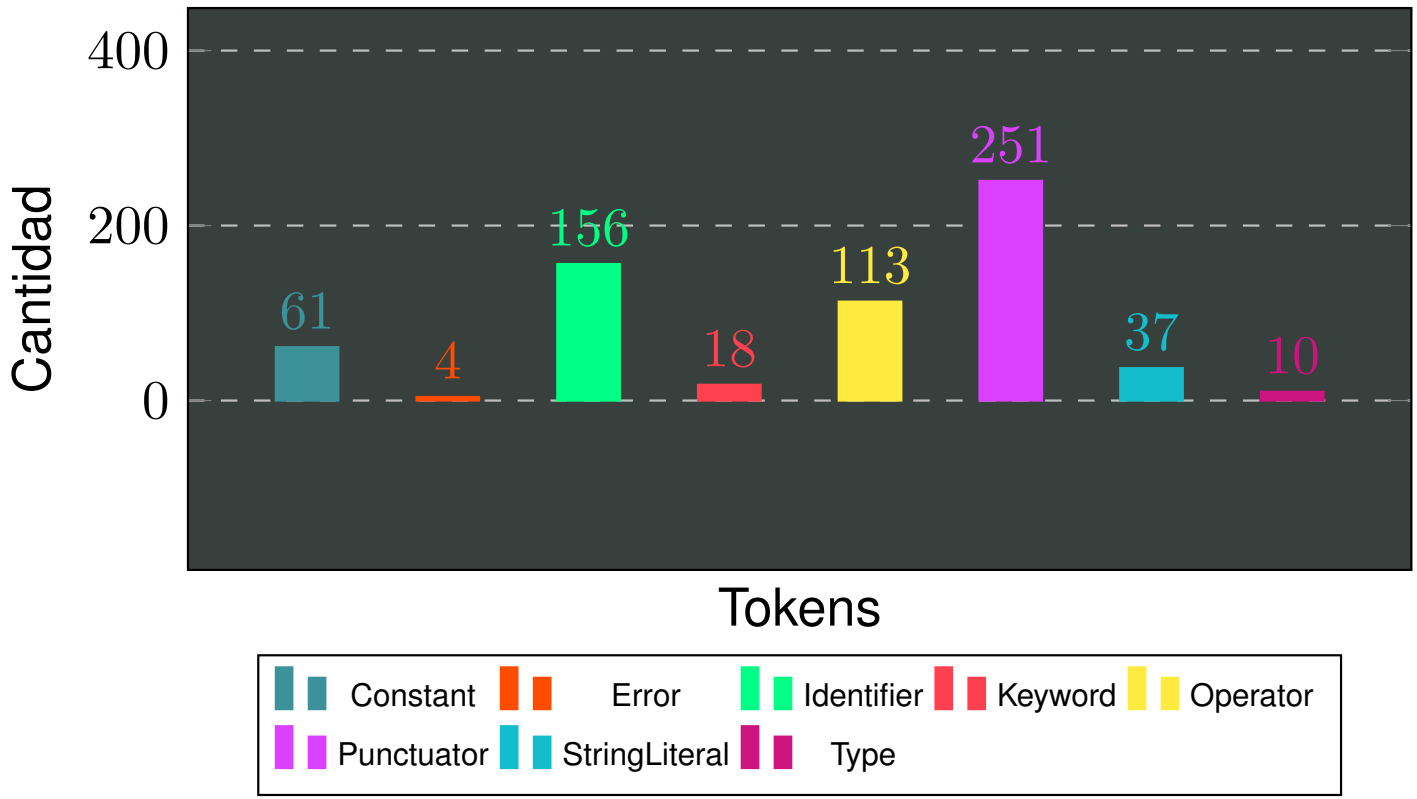
# test09


p [ - 1 ] = * p ; p [ - 1 ] = 0 ; z ( s ) ; } else { if ( M
) > I ) Z = O ; else ++ o [ O + 3 ] ; o [ O + 1 ] = I ;
"#include<stdio.h>"
"\12"| * p = # b , P [ 9999 ] , * d = P , * e [ ] = " "
{ } "␣␣␣␣␣␣␣␣␣" "␣␣␣␣␣";␣␣␣␣␣␣␣␣␣␣␣␣␣␣, * o , o [ 01 ] , o [ 2 ] , o [ 3 ] , ␣␣ o
""| ; Gam " " eQover .

# test09



# Histogram


400
200
0
Cantidad
61
4
156
18
113
251
37
10
Tokens
Constant
Error
Identifier
Keyword
Operator
Punctuator
StringLiteral
Type

# Pie Chart

- Constant 9.38%
- Error 0.62%
- Identifier 24.00%
- Keyword 2.77%
- Operator 17.38%
- Punctuator 38.62%
- String Literal 5.69%
- Type 1.54%