[GitHub](https://github.com/dorismoisuc/FLCD)

Documentation for Lexical Scanner

The following classes are present:

**SymbolTable:**

A hash table is used for the symbol table, the required operations: add and search are

implemented.

For the hash function Sum of ASCII codes of chars is used. It adds the values of ascii codes and

the result is the sum % the size of the hash table.

For the collision resolution method linear probing is used. If the position is used, the element is

put in the next empty space.

* add method: adds a given key in the symbol table

True, if the key is added

False, if the key is not added

* search method: searches a given key in the symbol table

the position of the key

-1, if the key is not found

**Specification**

A hash map is used, where the key = token, and the value is a unique code for the token, starting with 0 for identifier, 1 for constant and it is being incremented for each separator, operator and reserved word.

**PIF**

Also a hash map is used, key = code of token, defined in the specification class, value = position in the symbol table. 0 – identifiers, 1 – constants and -1 for separators, operators and reserved words.

**LexicalScanner**

The algorithm splits each line of the source code into tokens and for each token: if it’s a constant or an identifier it looks up for it’s position in the symbol table. If it is a separator, operator, or reserved word, the position will be -1.

The token and the position will be appended into a string builder for the output file. The lexical error will also be displayed, with the error, and the corresponding line.

Everything will be added to the PIF with the corresponding code from the Specification class and the position in symbol table if it’s a constant or identifier, or -1 if it’s a reserved word, operator or separator.