

# Lexical Analyzer Implementation Documentation

Lab 2 & 3















Alexandra-Mirela Sandor, 936/1













Link to Github: <https://github.com/916-Sandor-Alexandra/LFTC/>

## Symbol Table:

The data structure chosen for the implementation is a **Hash Table**.

The hash table is represented as an array of linked lists and in this manner any collisions are handled by separate chaining. For hash code computation, I have used polynomial rolling hash function.

HashTable		
 	m	int
 	HashTable(int)	
 	contains(String)	boolean
 	getPositionInTable(String)	SimpleEntry<Integer, Integer>
 	hash_function(String)	int
 	put(String)	SimpleEntry<Integer, Integer>
 	table	ArrayList<LinkedList<String>>

SymbolTable		
 	table	HashTable
 	m	int
 	SymbolTable(int)	
 	addToken(String)	SimpleEntry<Integer, Integer>
 	getPosition(String)	SimpleEntry<Integer, Integer>
 	toString()	String

## Lexical Analyzer:

- Input: **pr1.in, pr2.in, pr3.in, pr3err.in, token.in** (the 4 programs from Lab1b)
- Output: **PIF.out, symtab.out** (text files) + message in the console ("Lexically correct"/ "Lexical error" + details)

The scanner contains two instances of Symbol Table, one for the constants and one for the identifiers.

## Token Classifier:

This is a tool in the scanner used to split the defined keywords, operators and separators in the token.in file in 3 corresponding arrays and check if a received input is a valid token.

### PIF (Program Internal Form):

The PIF is defined by a list of a pairs. A pair consists of a string (token) and an entry that represents the position in the symbol table if the token is an identifier or a constant or a (-1, -1) mapping if the token is not in the Symbol Table i.e. it is a keyword/operator/separator.

Analyzer		
identifierTable	SymbolTable	
constantTable	SymbolTable	
tokenClassifier	TokenClassifier	
Analyzer(int)		
lineToTokens(String)		void
scanProgram(String)		void
writeInSymbolTable()		void
PIF		PIF
errors		List<String>

TokenClassifier		
keywords		List<String>
operators		List<String>
separators		List<String>
TOKEN_FILE		String
TokenClassifier()		
isConstant(String)		boolean
isIdentifier(String)		boolean
isKeyword(String)		boolean
isOperator(String)		boolean
isSeparator(String)		boolean
readAndStoreTokens(String)		void

PIF		
PIF		List<SimpleEntry<String, SimpleEntry<Integer, Integer>>>
PIF()		
put(SimpleEntry<String, SimpleEntry<Integer, Integer>>)		void
toString()		String
writeInPIF()		void