

# Formal Languages and Compiler Design

## Second laboratory – Documentation

### Symbol Table Implementation

Link to Git: <https://github.com/albcristi/formal-languages-and-compiler-design>

Data structure chosen for implementing the Symbol Table: Hash Table

About the Hash Table:

- m size of the table with m a prime number
- hashing function: sum of the ASCII characters of the key modulo m
- collision solution: we store a linked list at each position of the hash table, so that we can add a new node when we have a collision.
- each node contains the identifier that is stored in the linked list, the previous and the next node. The index is the position of that node in the linked list
- the search method will return the position a pair containing <hashing value of identifier, index of corresponding node> if the element is found in the hash table or the pair <-1, -1> otherwise
- the add method adds a new element in the hash table and returns a pair representing the following: <hash value of the added element, position in the linked list of the newly created node>

About the Symbol Table:

- stores data using a hash table
- supports two methods: **search** and **add** that take as argument a string value

### Implemented Classes

Node
+ identifier: string + index: int + nextNode: Node + prevNode: Node
+Node(identifier: string, index: int, node: Node) + getIdentifier(): string + getIndex(): int + getNextNode(): Node + getPrevNode(): Node + setIdentifier(ident: string) + setIndex(index: int) + setPrevNode(node: Node) + setNextNode(node: Node)

MyHashTable
- elements: Node[] - m: int
+MyHashTable(m: int) - hashingFunction(identifier: string): int + addElement(identifier: string): Pair<int, int> + searchElement(identifier: string): Pair<int, int>

SymbolTable
- hashTable: MyHashTable
+SymbolTable(m: int) + add(identifier: string): Pair<int, int> + search(identifier: string): Pair<int, int>

## Class Diagram

