

Scanner

Goian Tudor – 933/2

<https://github.com/VaruTudor/Formal-Languages-and-Compiler-Design>

Requirement - Implement the Symbol Table (ST) as the specified data structure, with the corresponding operations

Implementation

- I decided to use a Hash Table as the data structure for the Symbol Table. The main advantage of hash tables over other data structures is speed. The access time of an element is on average $O(1)$, therefore lookup could be performed very fast.
- For each entry in the ST a [Deque](#) (which under the hood is a doubly-linked list) will be used. It ensures that there will be no conflicts – when elements hash to the same position, they will be part of the same deque.
- I use modular hashing (the hash function is simply $h(k) = k \bmod m$ for some m - size. The value k is an integer hash code generated from the key. If m is a where for strings I add the ASCII ([American Standard Code for Information Interchange](#)) codes of each composing character.
- Supported operations
 - add(key)
input: key – the token (string)
output: getPosition(key)
 - remove(key)
input: key - the token (string)
return: -

- contains(key)
 - input: key - the token (string)
 - output: true/false
- getPosition(key)
 - input: key - the token (string)
 - output: (listPosition, listIndex) – listPosition is the position in the outer list and listIndex is the position in the current deque

Tokenizing

For a line, character by character, check if the current one is an operator, separator, string or belongs in the ST (meaning it is constant or identifier) .

Scanning

Line by line performs tokenizing, adds each identifier or constant to the ST and the keywords, separators and operators /w (-1,-1). For the constants the code will be “const” and for identifiers “id”.