GitHub repo: https://github.com/ciprianturcu/University-Projects/tree/main/Semester5/Formal%20Languages%20and%20Compiler%20Design/lab3/SymbolTable/src

Documentation:

The Symbol Table is composed as a Hash Table. Both the Symbol Table and Hash Table are generic implementations such that they could be used for integer, string, or any other types. The Hash Table uses open addressing with linear probing to solve common collision problems and uses a rehash when the load gets over a threshold therefore, we are not constrained to a specific capacity. From the implementation of the Hash Table the Symbol Table's elements are stored as only keys representing the given symbol and the index in which they are placed by the hash function as the position.

Operations:

Hash Table :

- hashFunction(K key) – computes the position in the hash table in which the given key should be placed
- rehashTable() – rehashes the entire table to increase capacity
- insertNode(K key) – insert a key into the hash table and return its position when the operation is successful
- getPosition(K key) – get the index in the hash table where the key is located
- getByposition(int position) – get the key from the hash table at the given position
- getCapacity() – get table capacity.

Symbol Table:

- getHashTable () – get the table
- add() – add a symbol to the table and return its position
- getPosition(K symbol) – return the position in the table of the symbol
- getByPosition(int position) – return the symbol from the given position