Symbol table – implementation using hash table

GitHub link - https://github.com/marianguceanu/FLCD

I chose a hash table as a data structure for the symbol table because of its time performance.

The classes I chose to use in this implementation (which is in C#) are as follows:

User defined class “Node” that takes one type as a generic, for key, and it also contains the fields “key” of the type specified and “value” of type string, considering that this class will be used to parse strings after all (and also for ease of implementation).

User defined class “HashTable” that takes a type as a generic, I did some tests for “string” as a type and also for “int” to make sure it works at least with these base types.

For the implementation of the actual symbol table I will choose to make a single table that should contain both constants and identifiers.

The hash table itself has all the operations we would need such as “Add”, “Get”, “GetKey”, “Remove”, “Clear”, “Contains” and “Count”. It also uses a hash function that is also using C#’s internal hash function for storing constants and identifiers so that we also get the maximum efficiency and randomization from the hash function.