Symbol table implementation (as a hash table)

Github link: https://github.com/MihaiCatalin120/Compiler

The symbol table is constructed around a dynamic capacity and a fixed load factor, the capacity changing whenever the load factor is surpassed. The hash function used for the table is the predefined python hash function for a string modulo the capacity of the table.

Add

Takes as a parameter the identifier or constant and returns its position (pair of integers) from the table after adding it. Before anything, it checks with the find function if the entry is already in the table and return its position if so, otherwise it will check if the load factor is surpassed, resizing the table if so, then it applies the hash function to the token and stores it in the table based on the hash and return the index where the value is stored.

Find

Takes as parameter the identifier/constant to be searched and returns its position from the table or the pair (-1, -1) if it is not found. The function applies the hash function to the token then it returns the position corresponding to the resulting key.

InitMap

Returns a map of structure (string -> list) with size equal to current capacity

Resize

Checks if the ratio between the table size and the current capacity exceeds the load factor. If so, capacity is doubled and a new map is generated based on the new capacity and all the old elements are added into the new map.