https://github.com/PeluzaVerde/LFTC/tree/main

Documentation

My Symbol table uses one hash table with separate chaining. The hash table is a list of hashnodes, which are a singly linked list, each hashnode having a 'next' in the case of hashing to the same position. The symbol table saves the hashnode, key-value at the position indicated by the hashing of the key .The hash function I used is from the class Objects, a built in function, then modulo of the capacity of the hash table.

Hashnode-

K Key, V Value, Hashnode<K,V> next

K is the key of the node

V is the value of the node

Next is a pointer to the next node.

Hash table-

List of hashnodes, capacity, size.

The list of hashnodes are the buckets of the hash table, each is a singly linked list.

Capacity is the number of buckets of the hash table

Size is the current number of hashnodes.

Int Hash(Key K). Hashes the key to a position in the Symbol Table.

V add(Key K, Value V). Adds the key and value in the form of a hashnode. Returns the the old value if it exists, or null otherwise.

V get(Key K). Returns the value at the given Key.

Remove (K key). Removes a hashnode from the hash table.

Symbol table-

Hashtable of string and integer for key and value.

Integer Add(String Key, int value). Adds the symbol and returns the value.

Remove(String Key). Removes a symbol from the symbol table.

Int get(String key). returns the value of the key from the symbol table.

Bool contains(String key). Returns true if the symbol is in the Symbol table. False otherwise