SymbolTable Class

Github Link:

https://github.com/917-SzaboBalazs/FLCD/tree/main/lab2

The `SymbolTable` class is an implementation of a simple hash table that allows storing key-value pairs.

Node Class

The `Node` class represents a linked list node, used to store key-value pairs within the hash table.

Attributes:

- `_key`: The key associated with the node.
- `_value`: The value associated with the node.
- `_next`: A reference to the next node in the linked list.

SymbolTable Class

The `SymbolTable` class represents the hash table.

Attributes:

- `_table`: An array used to store linked lists (buckets) of key-value pairs.
- `_size`: The current number of key-value pairs stored in the hash table.
- `_capacity`: The total capacity of the hash table.

Methods:

`__init__(self, capacity)` The constructor method initializes an empty hash table with a given capacity.

- `capacity`: The initial capacity of the hash table.

`__hash(self, key)` A private method that computes the hash value for a given key.

`insert(self, key, value)` Inserts a key-value pair into the hash table.

- `key`: The key to be inserted.
- `value`: The value associated with the key.

`find(self, key)` Finds the value associated with a given key in the hash table.

- `key`: The key to search for.

`remove(self, key)` Removes a key-value pair from the hash table based on the given key.

- `key`: The key to be removed.

`size(self)` Returns the current number of key-value pairs in the hash table.

`capacity(self)` Returns the total capacity of the hash table.

Usage

To use the `SymbolTable` class, you can create an instance of it and then perform operations like inserting, finding, or removing key-value pairs using the provided methods.

Example:

“‘python symbol_table = SymbolTable(100) symbol_table.insert(“key1”, “value1”) value = symbol_table.find(“key1”) symbol_table.remove(“key1”) size = symbol_table.size() capacity = symbol_table.capacity()