https://github.com/917wiszniewskibianca/lftc

Documentation

I chose to implement the ‘SymbolTable’ as a hash map due to the fact that it allows for :

* Fast Lookup and Retrieval: Hash tables provide fast and efficient key-based lookups. Retrieving data from a hash table typically takes O(1) time on average, making it one of the fastest data structures for this purpose.
* Predictable Performance: Hash tables offer consistent and predictable performance for operations like adding, checking existence, and retrieving data. The time complexity for these operations is independent of the size of the symbol table
* Easy Implementation: Hash tables are relatively straightforward to implement and use.
* Scalability: Hash tables can be easily resized to accommodate more data without significantly affecting their performance.
* Well-Suited for Symbol Tables: Symbol tables, which typically involve key-value pairs, can benefit from the simplicity and efficiency of hash table implementations, especially when the number of entries is relatively large.

The `SymbolTable` class manages three hash tables for identifiers, integer constants, and string constants.

Methods:

- `\_\_init\_\_(self, size)`: Initializes a new SymbolTable with a given size.

- `add\_identifier(self, name)`: Adds an identifier to the symbol table and returns its position.

- `add\_int\_constant(self, constant)`: Adds an integer constant to the symbol table and returns its position.

- `add\_string\_constant(self, constant)`: Adds a string constant to the symbol table and returns its position.

- `has\_identifier(self, name)`: Checks if the symbol table contains a given identifier.

- `has\_int\_constant(self, constant)`: Checks if the symbol table contains a given integer constant.

- `has\_string\_constant(self, constant)`: Checks if the symbol table contains a given string constant.

- `get\_position\_identifier(self, name)`: Gets the position of an identifier in the symbol table.

- `get\_position\_int\_constant(self, constant)`: Gets the position of an integer constant in the symbol table.

- `get\_position\_string\_constant(self, constant)`: Gets the position of a string constant in the symbol table.

- `to\_string(self)`: Returns a string representation of the entire symbol table.

HashTable Documentation:

The `HashTable` class represents a generic hash table.

Methods:

- `\_\_init\_\_(self, size)`: Initializes a new hash table with a given size.

- `hash(self, key)`: Computes the hash value for a given key (either an integer or a string).

- `get\_size(self)`: Returns the size of the hash table.

- `get\_hash\_value(self, key)`: Returns the hash value for a given key.

- `add(self, key)`: Adds a key to the hash table and returns its position.

- `contains(self, key)`: Checks if the hash table contains a given key.

- `get\_position(self, key)`: Gets the position of a key in the hash table.

- `to\_string(self)`: Returns a string representation of the hash table.