Source code: https://github.com/TeodoraVlad12/University-Work/tree/main/Third%20Year/Formal%20Languages%20and%20Compiler%20Design/Lab3

Documentation for `HashTable` Class

Attributes

- **Size (int)**: This represents the total number of slots (buckets) in the hash table.
- **HashTable (list)**: This is a list of lists, where each sublist stores keys that hash to the same index.

Methods

1. **`GetSize()`**:
    - **Description**: Returns the size of the hash table.
    - **Returns**: An integer representing the number of slots in the table.
2. **`HashFunction(key)`**:
    - **Description**: Computes a hash value for a given string key.
    - **Parameters**:
        - `key`: A string whose hash value is to be computed.
    - **Returns**: An integer that is the hash value, calculated as the sum of ASCII values of the characters in the string modulo the size of the table.
3. **`GetHashValue(key)`**:
    - **Description**: Retrieves the hash value for a key, if the key is a string.
    - **Parameters**:
        - `key`: A string for which to get the hash value.
    - **Returns**: An integer representing the hash value, or -1 if the key is not a string.
4. **`HasValue(key)`**:
    - **Description**: Checks whether a specified key exists in the hash table.
    - **Parameters**:
        - `key`: A string to search for in the table.
    - **Returns**: `True` if the key exists, otherwise `False`.
5. **`Add(key)`**:
    - **Description**: Adds a new key to the hash table.
    - **Parameters**:
        - `key`: A string that you want to add to the table.
    - **Returns**: A tuple containing the hash value and the index in the sublist where the key was added. If the key already exists, it returns the existing position.
6. **`Delete(key)`**:
    - **Description**: Removes a key from the hash table if it exists.
    - **Parameters**:
        - `key`: A string to be removed from the table.
    - **Returns**: None.

7. `GetValuePosition(key)`:
    - **Description**: Retrieves the position of a key in the hash table.
    - **Parameters**:
        - `key`: A string whose position is to be found.
    - **Returns**: A tuple containing the hash value and the index in the sublist, or (-1, -1) if the key is not found.

Documentation for `SymbolTable` Class

Attributes

- **Size (int)**: This indicates the number of slots allocated for the symbol table.
- **HashTable (HashTable)**: This is an instance of the `HashTable` class that handles the underlying storage and retrieval of items.

Methods

1. `Add(item)`:
    - **Description**: Adds an item (identifier or constant) to the symbol table.
    - **Parameters**:
        - `item`: A string that represents the item to be added.
    - **Returns**: A tuple containing the hash value and the index of the added item, or the existing position if the item already exists.
2. `Delete(item)`:
    - **Description**: Removes an item from the symbol table if it exists.
    - **Parameters**:
        - `item`: A string that specifies the item to be removed.
    - **Returns**: None.
3. `GetValuePosition(item)`:
    - **Description**: Retrieves the position of an item in the symbol table.
    - **Parameters**:
        - `item`: A string whose position is to be found.
    - **Returns**: A tuple containing the hash value and the index of the item, or (-1, -1) if not found.
4. `HasValue(item)`:
    - **Description**: Checks whether a specified item exists in the symbol table.
    - **Parameters**:
        - `item`: A string to search for in the symbol table.
    - **Returns**: `True` if the item exists, otherwise `False`.