Hashtable, Symboltable and Scanner Documentation

hashtable.py: This module contains the implementation of a hash table.

- HashTable class:
    - __init__(self, size=100, load_factor=0.7): Initializes a hash table with the specified size and load factor.
    - resize(self, new_size): Resizes the hash table to a new size.
    - hash(self, key): Calculates the hash value for a given key.
    - 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, value=None): Adds a key-value pair to the hash table, handling collisions using linear probing.
    - contains(self, key): Checks if a key is in the hash table.
    - get_variable_count(self, key): Returns the value associated with a key in the hash table.
    - get_position(self, key): Returns the position of a key in the hash table.
    - __str__(self): Returns a string representation of the hash table.

symboltable.py: This module contains the implementation of a symbol table using the HashTable class from hashtable.py.

- SymbolTable class:
    - __init__(self, size=100): Initializes a symbol table with the specified size.
    - add(self, key, value=None): Adds a key-value pair to the symbol table, incrementing the variable count for new variables.
    - has(self, key): Checks if a key exists in the symbol table.
    - get_position(self, key): Returns the position of a key in the symbol table.
    - __str__(self): Returns a string representation of the symbol table.
    - print_sym_table(self): Returns a formatted string representation of the symbol table.

scanner.py: This module contains a scanner for lexical analysis of source code.

- Scanner class:
    - __init__(self, symbol_table): Initializes the scanner with regular expressions for identifying reserved words, operators, separators, identifiers, digits, characters, and strings.
    - add_element(self, token): Adds a token to the program internal form (PIF) table, considering the symbol table.
    - buffering_from_file(self, input_file): Reads source code from a file and calls the buffering method.
    - buffering(self, source_code): Processes the source code and extracts tokens into the PIF table.
    - get_pif_table(self): Returns the PIF table.
    - print_tables(self): Prints the PIF table and the symbol table.
    - write_tables_to_files(self, output_pif_file, output_symbol_table_file): Writes the PIF table and symbol table to output files.

main.py: This is the main program that uses the SymbolTable and Scanner classes.

- Imports SymbolTable and Scanner from the respective modules.
- Initializes a symbol table and sets output file names.
- Creates a scanner and processes the source code from a file.
- Prints the PIF table and the symbol table.
- Writes the PIF table and symbol table to output files.