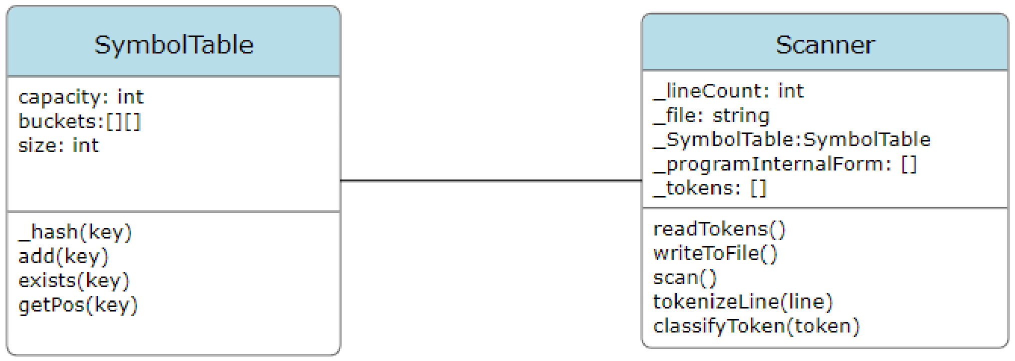
https://github.com/daniel357/flcd/blob/main/scanner.py ->scanner.py



Scanner.py Documentation

Input:

1. **file_info**: A string containing the code to be scanned.
2. The file **token.in**: Contains a list of predefined tokens, one token per line. These tokens are used for classification during scanning.

Output:

1. **PIF.out**: A file containing the Program Internal Form (PIF) generated during scanning. The PIF is a list of token-classification pairs.
2. **ST.out**: A file containing the Symbol Table (ST) which records the identifiers and constants encountered during scanning. This table also stores their positions in the code.
3. Print statements for feedback, indicating whether the code is lexically correct or not.

Functions and Their Functionality:

find_unbalanced_brace(braces_list)

- Description: This function checks if the curly braces **{}** in the code are balanced or unbalanced.
- Input:
  - **braces_list**: A list of curly braces in the code.
- Output:

    - Returns **None** if all braces are balanced, or the first unbalanced brace encountered.
- Usage: Used to check for unbalanced curly braces in the code.

Scanner.__init__(file_info)

- Description: The constructor of the **Scanner** class initializes the scanner and performs the scanning of the input code.
- Input:
    - **file_info**: A string containing the code to be scanned.
- Output: Initializes the **Scanner** object and generates the PIF and ST. It may raise a **ValueError** in case of lexical errors.

Scanner.readTokens()

- Description: Reads the predefined tokens from the **token.in** file and stores them in the **_tokens** list.
- Input: None.
- Output: Populates the **_tokens** list with predefined tokens.
- Usage: Used for reading the predefined tokens from a file.

Scanner.writeToFile()

- Description: Writes the Program Internal Form (PIF) and Symbol Table (ST) to output files.
- Input: None.
- Output: Creates **PIF.out** and **ST.out** files with the respective contents.
- Usage: Used to save the scanning results to files for further analysis.

Scanner.scan()

- Description: Performs lexical scanning of the input code, classifies tokens, updates the PIF and ST, and checks for balanced curly braces.
- Input: None (uses the code in **file_info**).
- Output: Generates the PIF, ST, and prints a message indicating if the code is lexically correct or not.

Scanner.tokenizeLine(line_string)

("[^"]+"|[a-zA-Z0-9]+|[^a-zA-Z0-9"\s]+) -> regex for splitting lines

    "[^"]+" -> match a string between double quotes

    [a-zA-Z0-9]+ ->match alphanumeric char

[^a-zA-Z0-9"\s]+ -> looks for one or more chars that are not alphanumeric or quotes

- Description: Tokenizes a line of code and handles special cases like making sure to get the right tokens, “==” or “<>”
- Input:
  - **line_string**: A string representing a line of code.
- Output: Returns a list of tokenized elements.
- Usage: Used to split a line into tokens for further processing.

Scanner.classifyToken(token)

^[a-zA-Z]+[a-zA-Z0-9]*$ ->match a variable name that should start with a letter and can be followed by one or more alphanumeric values

'^"[a-zA-Z0-9\s]+"$' -> match a string constant

^\'[a-zA-Z0-9\'$]' ->match a single char

'^0$|^(\+|-)?[1-9][0-9]*$' -> match a digit, either 0 or a +/- non zero digit

- Description: Classifies a token as either an identifier, string constant, character constant, or integer constant based on regular expressions.
- Input:
  - **token**: A string representing a token.
- Output: Returns an integer code for the token type (1 for identifier, 2 for string constant, 3 for character constant, 4 for integer constant, 0 for unclassified).
- Usage: Used to classify tokens for adding to the Symbol Table and PIF.