Lab Scanner documentation

https://github.com/tdrts/Labs_FLC/tree/main/Lab%204

Class Scanner

- **classify_tokens –** reads from a file all the tokens in my programming language and classifies them in 3 lists: separators, operators and reserved words
- **print_tokens –** displays on the screen the 3 lists obtained in classify_tokens
- **scan_file(filename) –** filename = file that contains the source code that has to be scanned

  - opening the file, reading line by line, using the function tokenize to get all the tokens in the current line. Then we classify each token. Separators, operators and reserved words are added in pif. If we identify a constant (numerical or string) we add it to the constant symbol table and then in pif with the corresponding position from the symbol table. If we identify an identifier we add it to the identifiers symbol table and then in pif with the corresponding position from the symbol table. If all the above fail, it means we have a token that cannot be classified which means we have a lexical error.

  - after scanning the entire file, a success message is printed on the screen or if there are errors, a message with the line number and wrong token are printed

  - pif, constants symbol table and identifiers symbol table are written in 3 separate files

  - Regex for numerical: **^0|[+-]?[1-9][0-9]*\.?[0-9]*$**

  - Regex for identifier: **^[a-zA-Z]([a-zA-Z]|[0-9])*$**
- **tokenize(line) –** line = string corresponding to a line in the program

  - splitting the line into corresponding separators, operators, reserved words, constants or identifiers by going char by char

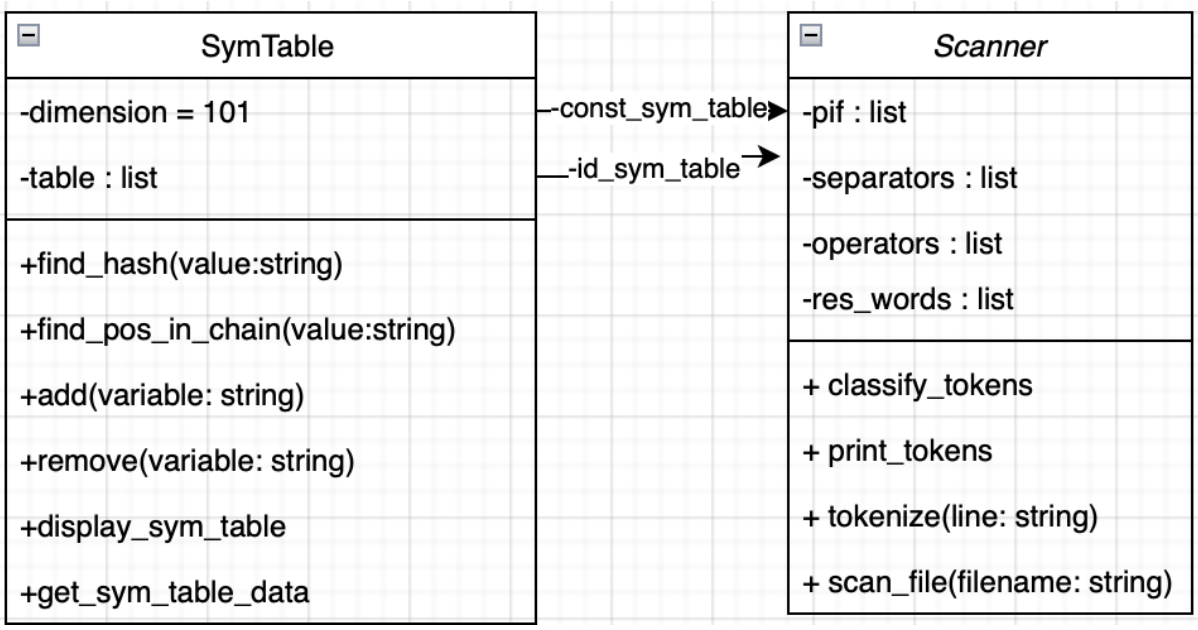
Symbol Table

- Hash Table with a hashing function that computes the sum of ascii codes of the value modulo 101 (size of the array).
- Collisions are solved by simple chaining.
- There is are 2 symbol tables: one for identifiers and one for constants.

Class Symbol Table

- **find_hash(value)** - returns the hash value corresponding for the value
- **add(variable)** – compute the hash value for the variable and add it to the table if it's not already added
- **remove(variable)** – computes the hash value for the variable and removes it from the symbol table if it exists
- **display_sym_table** – that prints on the screen the values in the hash table
- **find_pos_in_chain(value)** – returns the position that value has in the chain
- **get_sym_table_data** – returns a list will all the elements in the table

UML Diagram


SymTable
-dimension = 101
-table : list
+find_hash(value:string)
+find_pos_in_chain(value:string)
+add(variable: string)
+remove(variable: string)
+display_sym_table
+get_sym_table_data
-const_sym_table
-id_sym_table
Scanner
-pif : list
-separators : list
-operators : list
-res_words : list
+ classify_tokens
+ print_tokens
+ tokenize(line: string)
+ scan_file(filename: string)