915-Nistor-Anca/Former-languages-and-compiler-design (github.com)

Documentation – Scanner

The program implements a scanner. It has the lists of operators, separators and reserved words and it also receives the name of the text file the program is in. After scanning, it will print the Program Internal Form and the Symbol Table in two files and also a message if the program is lexically correct or not.

HashTable Class:

__init__(self, size): It is the constructor. It initializes a HashTable object with the given size.

__len__(self): It returns the size of the hashTable.

hash(self, token): The function calculates the hash for a given token. If the token is an integer, it computes the modulus of the token by the table size. If it is a string, it computes the sum of ASCII values of its characters and only after that it calculates the sum modulo the table size.

getPosition(self, token): It computes the hash for the given token. If there is something at that position, that means the token might already be added in the table, but it needs to search for the token at that position. If it really finds it, it returns a pair of (row, column); otherwise, it returns (-1, -1).

add(self, token): Adds the token in the hash table, if the token has not already been added. If it adds it, the function returns 0; if not, it returns 1.

SymbolTable Class:

__init__(self, size): It is the constructor. It initializes a SymbolTable object with a hash table of the given size.

add(self, token): It calls the add function of the hash table with the given token.

getSize(self): It returns the size of the symbol table.

getHashTable(self): It returns the hash table.

__str__(self): It returns a string with the hash table, saving the elements row by row.

getPosition(self, token): It returns the position of the token in the hash table.

ProgramInternalForm Class:

__init__(self): It initializes a ProgramInternalForm class with two lists, one for the tokens and another for the positions of the tokens in the SymbolTable.

getTokens(self): It returns the list of tokens.

getSTPos (self): It returns the list of positions in the SymbolTable.

addTokenSTPos(self, token, pos): It adds a new token in the list of tokens and its position in the SymbolTable in the list of positions.

__str__(self): It returns a string with the Program Internal Form; on each line, there is a token and its corresponding position.

Scanner Class:

__init__(self, operators, separators, reserved_words, file): It initializes a Scanner class which has three lists (for operators, separators and reserved words) and also the name of the input text file which the program is in. It also creates a SymbolTable class entity and a ProgramInternalForm one.

isOperator(self, token): It checks if the given token is in the list of operators.

isSeparator(self, token): It checks if the given token is in the list of separators.

isReservedWord(self, token): It checks if the given token is in the list of reserved words.

isIdentifier(self, token): It checks if the given token is an identifier. An identifier should start with a letter and can be followed by letters, digits or underscores.

isConstant(self, token): It checks if the given token is a constant value. The token can be a nonzero digit, a number (including integers), a boolean value, a character or a string.

detectTokens(self): Its main purpose of this function is to identify the tokens the scanner is working with. At first, it reads the program from a text file and splits everything by space. This is not enough, because there might be tokens which are concatenated (without a space between them).

After we do this, we also have in the main list of tokens the endline character, but we don't consider it as a token, so we have to eliminate each endline from the list and also the tokens which end with an endline.

After that, we eliminate the empty tokens (which were created because we eliminated endlines).

The next step is to split by separators. It takes each token and splits it character by character. If the character it's at is a separator, it adds it, and also adds any substring that was before the separator. If the character is not a separator, it means the program needs to continue adding characters to the substring.

The next step is to split by operators. It is exactly the same logic as splitting by separators.

After that, the program has to make sure that it didn't accidentally split an operator which has two characters, like ,"<=". It goes through each token and concatenates the operators which are next to each other (in this case, "<" and "=").

We do the same thing, but for negative numbers (if i have "-" and "51" after it, for example, it should concatenate them).

The last step is to concatenate back any string which was of type "any string here", because that means it is a constant. It iterates through the tokens and find any tokens which starts with ", because that means that should start a string ("any string here"). As long as we didn't find the token ending with ", we continue to go. After that, we concatenate everything between the first and the last token, and also delete everything in that interval, except the first token, which will have the final result ("any string here").

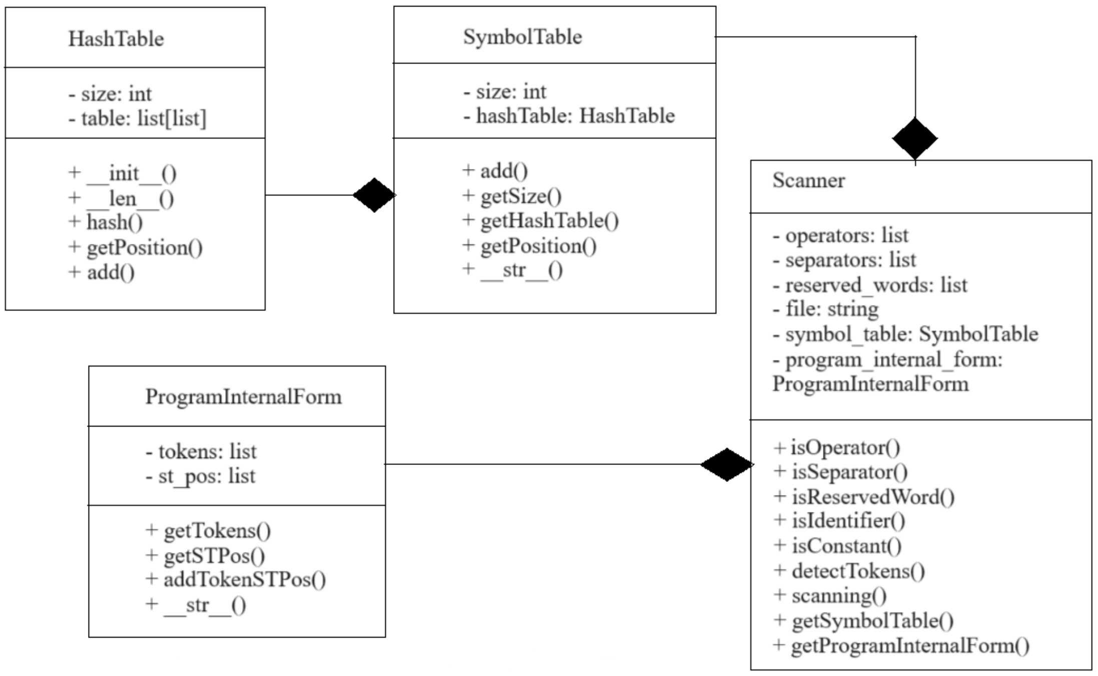
At the end, the function returns the list of tokens.

scanning(self): It takes token by token and tries to clasify it. It firstly checks if the token is a reserved word, operator or separator of it is the comment structure. It any of this is true, it adds the token to PIF, with the position (-1, -1). Otherwise, it checks if the token is an identifier or a constant. If it’s true, it first adds it to the symbol table and gets the position from it. If it is an identifier, it adds to PIF ”id” and the postion from the ST, else it adds ”const” and the position.

If it can clasify the token, it doesn’t add it nowhere. The algorithm also prints corresponding messages.

getSymbolTable(self): It returns the symbol table of the scanner.

getProgramInternalForm(self): It returns the program internal form of the scanner.


HashTable
- size: int
- table: list[list]
+ __init__()
+ __len__()
+ hash()
+ getPosition()
+ add()
SymbolTable
- size: int
- hashTable: HashTable
+ add()
+ getSize()
+ getHashTable()
+ getPosition()
+ __str__()
Scanner
- operators: list
- separators: list
- reserved_words: list
- file: string
- symbol_table: SymbolTable
- program_internal_form: ProgramInternalForm
+ isOperator()
+ isSeparator()
+ isReservedWord()
+ isIdentifier()
+ isConstant()
+ detectTokens()
+ scanning()
+ getSymbolTable()
+ getProgramInternalForm()
ProgramInternalForm
- tokens: list
- st_pos: list
+ getTokens()
+ getSTPos()
+ addTokenSTPos()
+ __str__()