https://github.com/921-Beltechi-Lois/Formal-Languges-and-Compiler-Design

Lab 3 - Scanner

Scanner Class Description:

The Scanner class is responsible for constructing the program's internal structure, building the symbol table, and verifying the lexical correctness of the input program. The internal structure consists of pairs comprising a string token and its position in the symbol table. For regular tokens, the position in the symbol table is considered (-1, -1).

Scanner Operations:

setProgram(program: string): This operation sets the input program for the scanner.

readTokens(): This operation reads tokens from the "token.in" file, categorizes them into reserved words and other tokens, and stores them for future reference.

skipSpaces(): This operation skips spaces within the program, updating the current line and index as it progresses.

skipComments(): This operation handles the skipping of comments within the program, updating the index accordingly.

treatStringConstant(): Boolean: This operation handles the case when a string constant is encountered in the program. It checks the validity of the string, ensuring there are no invalid characters and that quotes are correctly balanced. If the string constant is valid, it adds it to the symbol table (if not already present) and the program's internal form, updates the index, and returns true. If the string constant is invalid, it returns false.

treatIntConstant(): Boolean: This operation handles the case when an integer constant is encountered in the program. It checks the validity of the number (ensuring it contains only digits), adds it to the symbol table (if not already present) and the program's internal form if valid, updates the index, and returns true. If the integer is invalid, it returns false.

checkIfValid(possibleIdentifier: string, programSubstring: string): Boolean: This operation checks whether a possible identifier is valid. It does so by verifying if it's part of a declaration or if it already exists in the symbol table. If the possible identifier doesn't meet these conditions, it's considered an invalid token, and the operation returns false. Otherwise, it returns true.

treatIdentifier(): Boolean: This operation handles the case when an identifier is encountered in the program. It checks if the identifier is valid, following specific rules such as starting with an underscore or a letter, containing only letters, digits, and underscores, being part of a declaration, or already defined in the symbol table. If the identifier is valid, it adds it to the symbol table (if not already present) and the program's internal form, updates the index, and returns true. If the identifier is not valid, it returns false.

treatFromTokenList(): Boolean: This operation checks if the current program element is a reserved word or another token. If it is, it adds it to the program's internal form and returns true. Otherwise, it returns false.

nextToken(): This operation processes the current token, and if no corresponding case is found, it throws an exception indicating a lexical error at the current line and index.

scan(programFileName): This operation reads the program from the specified file, scans for the next token until the end of the program, and writes the program's internal form and symbol table to output files. If no exceptions are encountered during scanning, it displays a message indicating that the program is lexically correct. If an exception is caught, it displays the corresponding exception message.