https://github.com/biancaszekely32/FLCD

Documentation – Lab3

- Program Internal Form

The ProgramInternalForm class represents the Program Internal Form (PIF) used in lexical analysis. It defines the rules and properties of the programming language being analyzed. It includes information about operators, separators, reserved words, and provides methods to determine the type of a token. It manages a list of pairs, where each pair contains a token and a pair of codes.

Fields:

- *pif:* A list containing pairs of tokens and their associated codes.
- *operators*: A list containing operators like addition, subtraction, multiplication, division, modulus, comparison operators, logical operators, etc.
- *separators*: A list of characters used to separate tokens, such as brackets, parentheses, braces, colons, semicolons, commas, quotes, spaces, newlines, and tabs.
- *reservedWords*: A list of predefined words in the language, such as control flow statements (if, else, for, while), I/O operations (read, write), data types (integer, string), etc.
- *tokens*: A hash map that associates tokens (e.g., identifiers, constants) with their respective codes. It facilitates easy retrieval of the code for a given token.
-

Methods:

- *ProgramInternalForm():* initializes operators,separators, reservedWords, tokens and it loads the list of Tokens by calling the function
- *loadListOfTokens()*: Initializes the tokens hash map by associating each token type with a unique code. It also assigns codes to reserved words, operators, and separators.
- *getCode(String token)*: Returns the code associated with a given token.
- *isOperator(String token)*: Checks if a token is an operator.
- *isPartOfOperator(char op):* Determines if a character is part of an operator, considering cases like ">=" , "<=", "==", "!=".
- *isSeparator(String token):* Checks if a token is a separator.
- *isReservedWord(String token):* Determines if a token is a reserved word.
- *isIdentifier(String token):* Checks if a token is a valid identifier.

-identifierPattern: Matches valid identifiers, which start with a letter and can be followed by letters, digits, or underscores.
-characterPattern: Matches valid character constants enclosed in single quotes.
-stringPattern: Matches valid string constants enclosed in double quotes, which can contain various characters including spaces and special symbols.

- *isIntegerConstant(String token):* Checks if a token is a valid integer constant- can be either 0, or can be a number formed with digits, or by positive or negative numbers formed with digits
- *public void addToPif(String token, Pair<Integer, Integer> codes):* Adds a new pair to the Program Internal Form.
- *@Override public String toString():* Generates a string representation of the Program Internal Form.

➢ Analyzer

The Analyzer serves as a lexical analyzer for processing source code files. It reads a source code file, tokenizes it, and builds symbol tables for identifiers and constants. Additionally, it generates a Program Internal Form (PIF) which is a data structure used in further stages of a compiler.

Fields:

- *filename*: Stores the name of the source code file.
- *symbolTableString*: An instance for storing string symbols.
- *symbolTableInteger*: An instance for storing integer symbols.
- *pif*: An instance for storing the Program Internal Form.

Methods:

- *public Analyzer(String filename):* Initializes filename, symbolTableString, symbolTableInteger, and pif.
- *public void analyze():* Reads the source code file line by line.
- *private ArrayList<String> tokenize(String line):* Breaks a line of code into individual tokens based on language-specific rules.
- *public String identifyString(String line, int index):* Identifies a string constant starting from the specified index in the given line.
- *public String identifyOperator(String line, int index):* Identifies an operator starting from the specified index in the given line.
- *public String identifyToken(String line, int index):* Identifies a token starting from the specified index in the given line.
- *private void buildTables(List<Pair<String, Integer>> tokensWithLine):* Builds symbol tables and PIF based on the provided list of tokens. Identifies and classifies tokens into reserved words, identifiers, constants, or invalid tokens.

- *private void writeResults():* Writes the results (symbol tables and PIF) to two separate text files: SymbolTable.txt and ProgramInternalForm.txt.
- *private void deleteIfExists(String filePath):* Checks if a file exists at the specified path and deletes it if it does.

➢ Class Diagram