https://github.com/CatalinaArba/LFTC/tree/main/Lab3

Documentation

Project Overview:

The "Lexical Analysis for a Simple Programming Language" project is a Java-based application designed to perform lexical analysis on source code written in a simplified programming language. The project involves tokenizing the input code, identifying lexemes, and categorizing them into various language constructs like operators, separators, reserved words, identifiers, and constants. The application provides a Program Internal Form (PIF) and a Symbol Table (ST) as its primary output.

Components of the Project:

Lexical Analysis Class (LexicalAnalysis):

The LexicalAnalysis class is the core of the project, responsible for processing source code files and generating the PIF and ST.

It utilizes sets for operators, separators, and reserved words and performs tokenization of the source code to identify language constructs.

PifPair Class (PifPair):

The PifPair class represents a token-identifier pair used in the PIF.

It stores a token (lexeme) and an identifier (position in ST) to create an association between tokens and their meaning in the context of the language.

HashTable Class (HashTable):

The HashTable class provides a data structure for storing and managing identifiers in the symbol table.

It allows the insertion, retrieval, and management of symbols and their corresponding identifiers in a hash table.

Main Class (Main):

The Main class serves as the entry point for the project.

It initializes the LexicalAnalysis class, processes a source code file, and prints the PIF and ST as the final outputs.

Key Operations:

Tokenization: The application tokenizes the input source code into lexemes based on operators, separators, spaces, and other language constructs.

Categorization: It categorizes lexemes into operators, separators, reserved words, identifiers, and constants.

PIF Generation: The project generates a Program Internal Form (PIF) by associating tokens with their meaning, facilitating further processing of the code.

Symbol Table (ST): It maintains a Symbol Table (ST) using a hash table data structure to manage identifiers and their corresponding positions in the program.

Error Handling: The application detects and reports lexical errors in the source code.