Lab 3 – Documentation

https://github.com/cs-ubbcluj-ro/lab-work-computer-science-2024-georgianapetricele/tree/main/1-Mini-Language-And-Scanner/Lab3_scanner

Lexical Scanner

The scanner operates by reading the source code file and examining its contents line by line. It uses regular expressions to match patterns in the text, allowing it to recognize different tokens based on the mini-language's syntax. These tokens are then classified into two main categories:

● Reserved Tokens: These include keywords, separators, and operators predefined in the mini-language. They are recognized directly by the scanner and are not stored in the Symbol Table. The scanner records these tokens in the Program Internal Form (PIF) with a reference index of -1, indicating that they do not have an associated entry in the Symbol Table.

● Identifiers and Constants: When the scanner detects an identifier (a variable name, for example) or a constant (like a number or a string), it adds an entry to the Symbol Table (ST) if it's not already present. Each entry in the ST is given a unique index, which the scanner then uses to record the token in the PIF. This linkage allows the scanner to efficiently reference identifiers and constants without duplicating information.

Components

Symbol Table

- The Symbol Table is a data structure that maintains a list of all unique identifiers and constants found in the source code. It assigns a unique incremental index to each entry, which is used as a reference in the PIF.

Program Internal Form (PIF)

- The PIF is a list that tracks all tokens with their Symbol Table indices. For reserved tokens, it uses an index of -1, while for identifiers and constants, it uses their respective Symbol Table indices.

Methods:

1. private void readTokens(String tokenFile)
- Reads tokens from a file and stores them in tokens and reservedWords.

2. public void scan(String programFileName) throws IOException
- Scans a program file, generates PIF and symbol table.

3. private void processElement(int index) throws Exception
- Processes a token starting at the given index.

4. private void skipSpaces()
- Skips whitespace and end line characters and updates the current index.

5. private boolean processIdentifier()
- Identifies and processes an identifier.

6. private boolean processIntConstant()
- Identifies and processes an integer constant.

7. private boolean processStringConstant() throws Exception
- Identifies and processes a string constant.
- Throws an exception if the string is not delimitated by both ” ”

8. private boolean processCharConstant() throws Exception
- Identifies and processes a char constant.

9. private boolean processFromTokenList()
- Identifies a token from the token list.