

CS424-Compiler Construction

Assignment#1

Assignment Report

Muhammad Omer Bin Atique (2020335)

List of Contents

[1 INTRODUCTION 1](#_Toc159319772)

[2 IMPLEMENTATION 2](#_Toc159319773)

[2.1 Scanner’s Design 2](#_Toc159319774)

[2.2 Implementation Details 2](#_Toc159319775)

[2.3 Test Cases 4](#_Toc159319776)

# INTRODUCTION

Imagine a small, new programming language called "MiniLang." MiniLang is designed to be simple yet powerful enough to demonstrate key programming concepts. It supports basic arithmetic operations, variable assignments, if-else conditions, and print statements. My task was to design and implement a scanner for MiniLang using C++ or python.

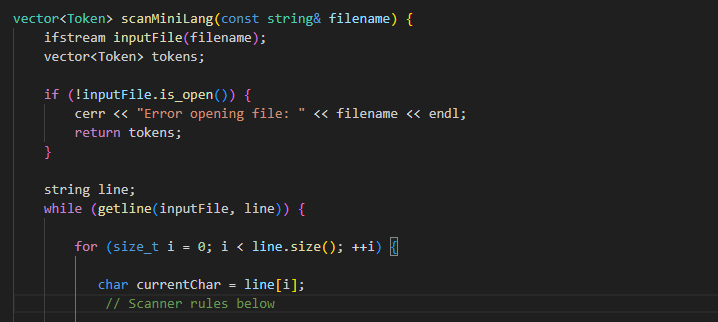
This assignment helped me understand the complexities of lexical analysis and prepared me for the subsequent stages of compiler design, moreover, it encouraged me to be creative with the test cases and explore the process of tokenizing different elements of the MiniLang language.

# IMPLEMENTATION

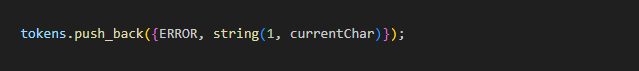
## Scanner’s Design

The scanner is designed to tokenize MiniLang source code based on the language's specifications. It aims to recognize various token types such as integers, identifiers, keywords, boolean literals, operators, comments, and errors. The design revolves around iterating through each character in the input file, identifying patterns, and categorizing them into appropriate token types.

## Implementation Details

* **Tokenization Logic:** The scanMiniLang function implements the core logic for scanning MiniLang source code. It reads the content of the input file line by line and processes each character to identify tokens. The function employs a series of checks and conditions to recognize different token types according to the language specifications.
* A black rectangle with white text

  Description automatically generated**Token Types:** The TokenType enum defines different categories of tokens that the scanner can recognize. These include integers, booleans, identifiers, operators, keywords, literals, comments, errors, and datatypes.
* **Character Processing:** The function processes each character in the input file to identify tokens. It checks for whitespace characters, integers, identifiers (including keywords, boolean literals, and datatypes), comments, and operators.
* **Error Handling:** If the scanner encounters an unrecognized character, it categorizes it as an error token. This helps in identifying and reporting lexical errors in the MiniLang source code.



* A black background with white text

  Description automatically generated**Output:** The tokens recognized by the scanner are stored in a vector of Token structures. Each token consists of a type (from TokenType) and the lexeme associated with it.
* **Printing Tokens:** The printTokens function is responsible for printing the tokens recognized by the scanner. It iterates through the vector of tokens and displays the token type along with its lexeme.

A computer screen with text

Description automatically generated

* **Main Function:** The main function serves as the entry point of the program. It specifies the input filename (example.minilang in this case), scans the MiniLang source code using the scanMiniLang function, retrieves the tokens, and prints them using the printTokens function.

A black screen with colorful text

Description automatically generated

This implementation provides a basic yet effective scanner for MiniLang, capable of tokenizing source code and identifying various language constructs. Further enhancements and refinements can be made to handle additional language features and improve error detection and reporting.

## Test Cases

* **Simple Case:**

**A screenshot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated**

* **A screen shot of a computer

  Description automatically generatedA screenshot of a computer program

  Description automatically generatedComlex Case:**