A black background with grey leaves

AI-generated content may be incorrect.

Lexical Analyzer

Build Scanner

A grey logo on a black background

AI-generated content may be incorrect.

**Prepared By**

Student Name :

**Mustafa hany gomaa**

Student ID:

**200045179**

**Under Supervision**

Name of Doctor:

**Nehal Abdelsalam**

Name of T. A.:

**Fares**

**Introduction**

**The Lexical Analyzer is the first step in a compiler. It takes the source code and splits it into small units called tokens, which are then used by later compiler phases. In this project, we built a simple program to analyze arithmetic expressions.**

* 1. **Phases of Compiler**

**A compiler has multiple phases:**

1. **Lexical Analysis: Converts code into tokens (our focus).**
2. **Syntax Analysis: Checks if the tokens follow language rules.**
3. **Semantic Analysis: Ensures the code makes sense.**
4. **Code Generation: Produces machine code. This project focuses on the Lexical Analysis phase.**
5. **Lexical Analyzer**

**Our lexical analyzer reads and processes arithmetic expressions. For example, with the input Mustafa = ( 2005 + 20 ) \* 10, it identifies identifiers (like Mustafa), numbers (like 2005), operators (like +, \*), and parentheses, printing each token with its code and lexeme.**

**Lexical Analyzer Code Explanation (C Language)  
  
Constants:  
- LETTER: Represents a letter.  
- DIGIT: Represents a digit.  
- UNKNOWN: Represents an unknown character (e.g., operator or punctuation).  
- IDENT: Identifier token.  
- INT\_LIT: Integer literal token.  
- ASSIGN\_OP, ADD\_OP, SUB\_OP, MULT\_OP, DIV\_OP: Operator tokens.  
- LEFT\_PAREN, RIGHT\_PAREN: Parenthesis tokens.  
  
Global Variables:  
- lexeme[]: Array to store the current token characters.  
- nextChar: The current character being analyzed.  
- charClass: Classification of the current character (letter, digit, unknown).  
- input[]: The user or default input string.  
- nextToken: The current token's type.  
  
Functions:  
1. getChar():  
 Reads the next character and determines its class (letter, digit, unknown).  
  
2. addChar():  
 Adds the current character to the lexeme array.  
  
3. getNonBlank():  
 Skips whitespace characters in the input.  
  
4. lookup(char ch):  
 Checks if the character is a recognized operator or parenthesis and assigns the appropriate token.  
  
5. lex():  
 Main function to analyze input and identify tokens. It constructs tokens by combining characters and classifying them.  
  
6. printTokenType(tokenCode):  
 Displays the token type and its corresponding lexeme.  
  
Main Function:  
- Initially processes a default string: "Mustafa = ( 2005 + 20 ) \* 10"  
- Then enters a loop where the user can input expressions repeatedly.  
- Each expression is tokenized and output to the console.**

1. **Software Tools**

**We used simple tools to develop and test this program.**

* 1. **Computer Program**

**We used GCC to compile the C code into an executable (front.exe) and visual studio 2022 to run and test the program.**

* 1. **Programming Language**

**We chose C because it’s fast, simple, and provides good control over file reading and character processing.**

1. **Implementation of a Lexical Analyzer**

**For the input Mustafa = ( 2005 + 20 ) \* 10, it produces the output shown in Table 1.**

**Simple Explanation of the Code**

**#define \_CRT\_SECURE\_NO\_WARNINGS: Prevents warnings about some functions.**

**#include <stdio.h>: Includes input/output functions.**

**#include <ctype.h>: Includes functions to check character types.**

**#include <stdlib.h>: Includes exit() to close the program.**

**int charClass;: Stores the character type (letter, digit, or other).**

**char lexeme[100];: Holds the current token string.**

**char nextChar;: The current character being processed.**

**int lexLen;: Length of the token string.**

**int token;: Current token code.**

**int nextToken;: Next token code.**

**int inputIndex;: Tracks position in the input string.**

**char input[100];: Stores the input expression.**

**#define LETTER 0, etc.: Defines character types.**

**#define IDENT 1, etc.: Defines token types (1 = Identifier, 2 = Number, etc.).**

**int main(): Main function.**

**sprintf(input, "Mustafa = ( 2005 + 20 ) \* 10");: Sets default input.**

**inputIndex = 0;: Starts at the beginning of the string.**

**printf("Default input: %s\n", input);: Prints the default input.**

**getChar(); do { lex(); } while (nextToken != EOF);: Processes the default input.**

**printf("\nEnter an expression: "); fgets(input, 100, stdin);: Gets one user input.**

**printf("Processing input: %s", input);: Prints the user input.**

**getChar(); do { lex(); } while (nextToken != EOF);: Processes the user input.**

**printf("Exiting program.\n"); exit(0);: Prints exit message and closes the window.**

**int lookup(char ch): Assigns token codes to operators and parentheses.**

**void addChar(): Adds a character to the token string.**

**void getChar(): Reads the next character from the input.**

**void getNonBlank(): Skips spaces.**

**int lex(): Analyzes tokens and calls printTokenType.**

**void printTokenType(int tokenCode): Prints the token type, code, and lexeme.**

***TABLE. 1***

**Lexeme --------------------------- token**

**Mustafa Identifier**

**=** **Assignment.** **Operator**

**( Left Parenthesis**

**2005 Integer Literal**

**+ plus. Operator**

**20 Integer Literal**

**) Right Parenthesis**

**\* mult.** **Operator**

**10 Integer Literal**

1. **References**

* **"Compilers: Principles, Techniques, and Tools" by Aho, Sethi, and Ullman.**
* **Online C tutorials from W3Schools.**
* **CONCEPTS OF PROGRAMMING LANGUAGES TWELFTH EDITION ROBERT W. SEBESTA**

**Important Note: -**

Technical reports include a mixture of text, tables, and figures. Consider how you can present the information best for your reader. Would a table or figure help to convey your ideas more effectively than a paragraph describing the same data?

Figures and tables should: -

* Be numbered
* Be referred to in-text, e.g. *In Table 1*…, and
* Include a simple descriptive label - above a table and below a figure.