MISR UNIVERSITY
FOR SCIENCE & TECHNOLOGY
College of Information Technology

جـامعـة مـصـر
للعلـوم والتكنـولـوجيـا
كليـة تكنولوجيـا المعلومـات

LEXICAL ANALYZER

Build Scanner

Prepared By

Student Name:

Mohamed Talal

Student ID:

200049026

Under Supervision

Name of Doctor

Name of T. A.

- ## Introduction

This document provides an overview of the implementation of a Lexical which is a ,Analyzer
the ,It covers the phases of a compiler .fundamental phase in compiler design role of a lexical
analyzer, software tools used, and the implementation details

- ### Phases of Compiler

including: ,A compiler consists of several phases
Tokenizing the input code. :Lexical Analysis .1
Checking grammatical structure. :Syntax Analysis .2
Ensuring meaningful statements. :Semantic Analysis .3
Creating an intermediate representation. :Intermediate Code Generation .4
Improving performance and efficiency. :Optimization .5
.Producing machine code :Code Generation .6

- ## Lexical Analyzer

A Lexical Analyzer is responsible for scanning the source code and converting it into tokens.
It identifies keywords, operators, identifiers, and other elements

- ## Software Tools

Various software tools are used in compiler construction.

- ### Computer Program

It .A compiler is a special type of program that translates source code into machine code ensures the correctness of syntax and semanti

- **Programming Language**

Lexical analyzers are often implemented using programming languages like Python, C, or Java. The implementation in this document is in Python.

- **Implementation of a Lexical Analyzer**i

charClass = None lexeme = [] nextChar = '' lexLen = 0 token = None nextToken = None

def addChar(): global lexeme, lexLen, nextChar if lexLen <= 98: lexeme.append(nextChar) lexLen += 1 else: print("Error - lexeme is too long")

def getChar(): global nextChar, charClass nextChar = in_fp.read(1) if nextChar == '': charClass = 'EOF' elif nextChar.isalpha(): charClass = 'LETTER' elif nextChar.isdigit(): charClass = 'DIGIT' else: charClass = 'UNKNOWN'

def getNonBlank(): global nextChar while nextChar.isspace(): getChar()

def lookup(ch): global nextToken if ch == '(': addChar() nextToken = 'LEFT_PAREN' elif ch == ')': addChar() nextToken = 'RIGHT_PAREN' elif ch == '+': addChar() nextToken = 'ADD_OP' elif ch == '-': addChar() nextToken = 'SUB_OP' elif ch == '*': addChar() nextToken = 'MULT_OP' elif ch == '/': addChar() nextToken = 'DIV_OP' else: addChar() nextToken = 'EOF' return nextToken

def lex(): global lexLen, charClass, nextToken, lexeme lexLen = 0 getNonBlank()

```
                                    if charClass == 'LETTER':
                                                    addChar()
                                                    getChar()
                    while charClass == 'LETTER' or charClass == 'DIGIT':
                                                    addChar()
                                                    getChar()
                                            nextToken = 'IDENT'
                                    elif charClass == 'DIGIT':
                                                    addChar()
                                                    getChar()
                                    while charClass == 'DIGIT':
                                                    addChar()
                                                    getChar()
                                            nextToken = 'INT_LIT'
```

```
        elif charClass == 'UNKNOWN':
            lookup(nextChar)
            getChar()
        elif charClass == 'EOF':
            nextToken = 'EOF'
        lexeme = ['E', 'O', 'F']

    print(f"Next token is: {nextToken}, Next lexeme is
                {''.join(lexeme)}")
        return nextToken
```

if *name* == "*main*": try: in_fp = open("front.in", "r") except FileNotFoundError: print("ERROR - cannot open front.in") else: getChar() while nextToken != 'EOF': lex()

- **References**

Create Your Own Domain-Specific :Language Implementation Patterns **.(2022) .T ,Parr.1**
and
***General Programming Languages with Python*.**
.Introduction to Compiler Design **.(2021) .D ,Parsons.2**

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.