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:Ahmed hazem mohamed

Student ID:200044998

**Under Supervision**

Name of Doctor

Name of T. A.

1. **Introduction**
   1. **Phases of Compiler**
2. **Lexical Analyzer**
3. **Software Tools**
   1. **Computer Program**
   2. **Programming Language**
4. **Implementation of a Lexical Analyzer**
5. **References**

**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.

Lexical Analyzer Project Report

1. Introduction

This project demonstrates a simple implementation of a lexical analyzer using C++.

The analyzer reads an arithmetic expression, divides it into lexemes, and assigns a corresponding token type for each lexeme.

1.1 Phases of Compiler

1. Lexical Analysis

2. Syntax Analysis

3. Semantic Analysis

4. Intermediate Code Generation

5. Code Optimization

6. Code Generation

2. Lexical Analyzer

A lexical analyzer is the first phase of the compiler.

It scans the source code character by character, groups characters into lexemes, and produces tokens that represent variable names, numbers, operators, and punctuation.

3. Software Tools

Operating System: Windows 10

IDE: Code::Blocks

Programming Language: C++

Compiler: GNU GCC

3.1 Computer Program

#include <iostream>

#include <sstream>

#include <cctype>

using namespace std;

int main() {

string line = "x = y + 5 ;";

stringstream ss(line);

string token;

while (ss >> token) {

if (token == "+")

cout << "Next token is: ADD\_OP\t\tNext lexeme is " << token << endl;

else if (token == "-")

cout << "Next token is: SUB\_OP\t\tNext lexeme is " << token << endl;

else if (token == "\*")

cout << "Next token is: MULT\_OP\t\tNext lexeme is " << token << endl;

else if (token == "/")

cout << "Next token is: DIV\_OP\t\tNext lexeme is " << token << endl;

else if (token == "=")

cout << "Next token is: ASSIGN\_OP\tNext lexeme is " << token << endl;

else if (token == ";")

cout << "Next token is: SEMICOLON\tNext lexeme is " << token << endl;

else if (isdigit(token[0]))

cout << "Next token is: INT\_LIT\t\tNext lexeme is " << token << endl;

else

cout << "Next token is: IDENT\t\tNext lexeme is " << token << endl;

}

return 0;

}

3.2 Programming Language

The lexical analyzer was implemented in C++ using standard libraries.

4. Implementation of a Lexical Analyzer

Test Input:

x = y + 5 ;

Expected Output:

Next token is: IDENT Next lexeme is x

Next token is: ASSIGN\_OP Next lexeme is =

Next token is: IDENT Next lexeme is y

Next token is: ADD\_OP Next lexeme is +

Next token is: INT\_LIT Next lexeme is 5

Next token is: SEMICOLON Next lexeme is ;

5. References

Sebesta, R.W. Concepts of Programming Languages, 12th Edition.

CS211 Compiler Design, Summer 2025 Slides.