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**

Jolie Nixon Ezzat

200039028

**Under Supervision**

Dr / Nehal

T. A./ Eng. Eslam

1. **Introduction**
   1. **Phases of Compiler**

I used Visual Studio Compiler . **Compiler** operates in various phases each phase transforms the source program from one representation to another. Every phase takes inputs from its previous stage and feeds its output to the next phase of the compiler.  
There are 6 phases in a compiler. Each of these phases helps in converting the high-level language to the machine code. The phases of a compiler are :

1 - Lexical analysis

2 - Syntax analysis

3 - Semantic analysis

4 - Intermediate code generator

5 - Code optimizer

6 - Code generator

1. **Lexical Analyzer**

Lexical Analysis is the first phase when compiler scans the source code. This process can be left to right, character by character, and group these characters into tokens.  
Here, the character stream from the source program is grouped in meaningful sequences by identifying the tokens. It makes the entry of the corresponding tickets into the symbol table and passes that token to next phase.

The primary functions of this phase are :

1 - Identifying the lexical units in a source code

2 - Classifying lexical units into classes like constants , reserved words , and enter them in different tables . It will ignore comments in the source program

3 - Identifying token which is not a part of the language

**For example :** x = y \* 2

Lexeme Token

X identifier

= assign\_op

Y identifier

\* mult\_op

2 int\_literal

1. **Software Tools**
   1. **Computer Program**

Visual Studio is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment" \o "Integrated development environment) (IDE) developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft" \o "Microsoft). It is used to develop [computer programs](https://en.wikipedia.org/wiki/Computer_program" \o "Computer program) including [websites](https://en.wikipedia.org/wiki/Web_site" \o "Web site), [web apps](https://en.wikipedia.org/wiki/Web_app" \o "Web app), [web services](https://en.wikipedia.org/wiki/Web_service" \o "Web service) and [mobile apps](https://en.wikipedia.org/wiki/Mobile_app" \o "Mobile app). Visual Studio uses Microsoft software development platforms including [Windows API](https://en.wikipedia.org/wiki/Windows_API" \o "Windows API), [Windows Forms](https://en.wikipedia.org/wiki/Windows_Forms" \o "Windows Forms), [Windows Presentation Foundation](https://en.wikipedia.org/wiki/Windows_Presentation_Foundation" \o "Windows Presentation Foundation) (WPF), [Microsoft Store](https://en.wikipedia.org/wiki/Microsoft_Store" \o "Microsoft Store) and [Microsoft Silverlight](https://en.wikipedia.org/wiki/Microsoft_Silverlight" \o "Microsoft Silverlight). It can produce both [native code](https://en.wikipedia.org/wiki/Machine_code" \o "Machine code) and [managed code](https://en.wikipedia.org/wiki/Managed_code" \o "Managed code).

Visual Studio provides developers a feature rich development environment to develop high-quality code efficiently and collaboratively.

Visual Studio IDE provides many [features](https://learn.microsoft.com/en-us/visualstudio/ide/writing-code-in-the-code-and-text-editor?view=vs-2022) that make it easier for you to write and manage your code with confidence. For example, code quickly and accurately with [AI-assisted development](https://learn.microsoft.com/en-us/visualstudio/ide/ai-assisted-development-visual-studio?view=vs-2022) tools. These tools include [GitHub Copilot](https://learn.microsoft.com/en-us/visualstudio/ide/visual-studio-github-copilot-extension?view=vs-2022) and [IntelliCode](https://learn.microsoft.com/en-us/visualstudio/ide/intellicode-visual-studio?view=vs-2022). Make quick improvements to your code using light bulbs that suggest actions, or expand/collapse blocks of code using outlining. Organize and explore your code with the **Solution Explorer** that shows your code organized by files or the **Class View** that shows your code organized by classes.

[GitHub Copilot](https://learn.microsoft.com/en-us/visualstudio/ide/visual-studio-github-copilot-extension?view=vs-2022), [GitHub Copilot Chat](https://learn.microsoft.com/en-us/visualstudio/ide/visual-studio-github-copilot-chat?view=vs-2022), and [IntelliCode](https://learn.microsoft.com/en-us/visualstudio/ide/intellicode-visual-studio?view=vs-2022) assist developers in writing code faster and with greater accuracy, help develop a deeper understanding of the codebase, and help with other development tasks such as writing unit tests, [debugging](https://learn.microsoft.com/en-us/visualstudio/debugger/debug-with-copilot?view=vs-2022), and profiling.

You can compile and build your applications to create builds immediately and test them in a debugger. You can run multi-processor builds for C++ and C# projects. Visual Studio also provides several options that you can configure when you build applications. You can create a custom build configuration in addition to the built-in configurations, hide certain warning messages, or increase build output information.

Integrated debugging in Visual Studio enables you to debug, profile, and diagnose with ease. You step through your code and look at the values stored in variables, set watches on variables to see when values changes, examine the execution path of your code. Visual Studio offers other ways to debug your code while it runs.

You can write high-quality code with comprehensive testing tools in Visual Studio. Unit tests give developers and testers a quick way to find logic errors in code. You can analyze how much code you're testing and see instant results in a test suite. Know the impact of every change you make with advanced features that test code while you type.

Visual Studio Live Share enables real-time collaborative development. With Live Share you can share your project with your peers, no matter the language or platform. Get to the bottom of an issue fast by allowing your team to connect, navigate, set break points, and type in your editor session.

* 1. **Programming Language**

I wrote the code in c++ language . C++ is a [high-level](https://en.wikipedia.org/wiki/High-level_programming_language" \o "High-level programming language), [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language" \o "General-purpose programming language) created by Danish computer scientist [Bjarne Stroustrup](https://en.wikipedia.org/wiki/Bjarne_Stroustrup" \o "Bjarne Stroustrup). First released in 1985 as an extension of the [C programming language](https://en.wikipedia.org/wiki/C_(programming_language)" \o "C (programming language)), it has since expanded significantly over time; as of 1997, C++ has [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming" \o "Object-oriented programming), [generic](https://en.wikipedia.org/wiki/Generic_programming" \o "Generic programming), and [functional](https://en.wikipedia.org/wiki/Functional_programming" \o "Functional programming) features, in addition to facilities for [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language" \o "Low-level programming language) [memory](https://en.wikipedia.org/wiki/Memory_(computing)" \o "Memory (computing)) manipulation for systems like [microcomputers](https://en.wikipedia.org/wiki/Microcomputer" \o "Microcomputer) or to make operating systems like [Linux](https://en.wikipedia.org/wiki/Linux" \o "Linux) or [Windows](https://en.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows). It is usually implemented as a [compiled language](https://en.wikipedia.org/wiki/Compiled_language" \o "Compiled language), and many vendors provide [C++ compilers](https://en.wikipedia.org/wiki/List_of_compilers" \l "C.2B.2B_compilers" \o "List of compilers), including the [Free Software Foundation](https://en.wikipedia.org/wiki/Free_Software_Foundation" \o "Free Software Foundation), [LLVM](https://en.wikipedia.org/wiki/LLVM" \o "LLVM), [Microsoft](https://en.wikipedia.org/wiki/Microsoft" \o "Microsoft), [Intel](https://en.wikipedia.org/wiki/Intel" \o "Intel), [Embarcadero](https://en.wikipedia.org/wiki/Embarcadero_Technologies" \o "Embarcadero Technologies), [Oracle](https://en.wikipedia.org/wiki/Oracle_Developer_Studio" \o "Oracle Developer Studio), and [IBM](https://en.wikipedia.org/wiki/IBM" \o "IBM).

C++ was designed with [systems programming](https://en.wikipedia.org/wiki/Systems_programming" \o "Systems programming) and [embedded](https://en.wikipedia.org/wiki/Embedded_software" \o "Embedded software), resource-constrained software and large systems in mind, with [performance](https://en.wikipedia.org/wiki/Performance_(software)" \o "Performance (software)), efficiency, and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications,  including [desktop applications](https://en.wikipedia.org/wiki/Application_software" \o "Application software), [video games](https://en.wikipedia.org/wiki/Video_game_development" \o "Video game development), [servers](https://en.wikipedia.org/wiki/Server_(computing)" \o "Server (computing)) (e.g., [e-commerce](https://en.wikipedia.org/wiki/E-commerce" \o "E-commerce), [web search](https://en.wikipedia.org/wiki/Web_search_engine" \o "Web search engine), or [databases](https://en.wikipedia.org/wiki/Database" \o "Database)), and performance-critical applications (e.g., [telephone switches](https://en.wikipedia.org/wiki/Telephone_switches" \o "Telephone switches) or [space probes](https://en.wikipedia.org/wiki/Space_probes" \o "Space probes)).

C++ is one of the world's most popular programming languages.

C++ can be found in today's operating systems, Graphical User Interfaces, and embedded systems.

C++ is an object-oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.

C++ is portable and can be used to develop applications that can be adapted to multiple platforms.

1. **Implementation of a Lexical Analyzer**

A **lexical analyzer**, also known as a lexer or tokenizer, is an integral part of the compiler whose main function is to divide the input source code into logical units called tokens. These tokens are then transferred to the next phase of the compilation process.

 Lexical analyzers work by scanning the input source code character by character and grouping them into tokens based on predefined patterns or rules. The process mainly consists of **5 major steps**:

1. Reading input characters from the source code
2. Identifying and categorizing lexemes
3. Generating tokens with attributes such as type and value
4. Handling whitespace and comments present in the source code
5. Detecting and handling various errors in the source code

**Code :**

#include <iostream> // a library for input and output

#include <cctype> // a library for character classification

#include <string> // a library for using std::string

using namespace std;

// Character classes

#define LETTER 0 // Alphabets

#define DIGIT 1 // Digits

#define UNKNOWN 99 // Anything else as operators

#define EOF\_TOKEN -1 // End of file

// Token codes

#define INT\_LIT 10 // Integer literal

#define IDENT 11 // Identifier ( x , y ... )

#define ASSIGN\_OP 20 // equal sign ( = )

#define ADD\_OP 21 // Addition operator ( + )

#define SUB\_OP 22 // Subtraction operator ( - )

#define MULT\_OP 23 // Multiplication operator ( \* )

#define DIV\_OP 24 // Division operator ( / )

#define LEFT\_PAREN 25 // Left parenthesis " ( "

#define RIGHT\_PAREN 26 // Right parenthesis " ) "

// Global variables

int charClass; // Current character class

char nextChar; // Actual character

string lexeme; // Current lexeme

int nextToken; // the token after the current lexeme

// Function declarations

void addChar(); // Adding next character to the lexeme

void getChar(); // Getting the value of the next character

void getNonBlank(); // Skips over whitespace

int lex(); // Lexical analyzer

int lookup(char ch); // Adding operators to tokens

// Main function

int main() {

cout << "Enter the input text (press Enter and then Ctrl+D to stop input):" << endl;

// Start reading the input directly from user input (cin)

getChar(); // Getting the value of character

do { // Starting a loop

lex(); // A function to get the lexical analyzer

} while (nextToken != EOF\_TOKEN); // if the next token doesn't equal the end of file token

return 0; // Return value 0 when finished

} // A curly brace to end the main function

void addChar() { // Adds nextChar to lexeme

lexeme += nextChar; // Lexeme will be added to the next character

} // A curly brace to end the addChar function

void getChar() { // Reads next character and sets charClass

if (cin.get(nextChar)) { // If the next character is successfully got

if (isalpha(nextChar)) // Checking if the next character is a letter

charClass = LETTER; // The current character class is a letter

else if (isdigit(nextChar)) // Another condition to check if the next character is a digit

charClass = DIGIT; // The current character is a digit

else // Another condition to check if the next character is something else

charClass = UNKNOWN; // The current character is something else ( operator )

}

else { // If input ends

charClass = EOF\_TOKEN; // End of file

} // A curly brace to end the else condition

} // A curly brace to end the getChar function

void getNonBlank() { // Skips whitespace

while (isspace(nextChar)) { // Using a while loop to read the next characters till we meet a non space character

getChar(); // A function that calls the non space character

} // A curly brace to end the while loop

} // A curly brace to end the getNonBlank function

int lookup(char ch) { // A function that adds operators to tokens

switch (ch) { // Using switch case to add characters to parenthesis and operators

case '(':

addChar();

return LEFT\_PAREN; // (

case ')':

addChar();

return RIGHT\_PAREN; // )

case '+':

addChar();

return ADD\_OP; // +

case '-':

addChar();

return SUB\_OP; // -

case '\*':

addChar();

return MULT\_OP; // \*

case '/':

addChar();

return DIV\_OP; // ' / '

case '=':

addChar();

return ASSIGN\_OP; // =

default: // If none of the above cases is true

addChar(); // Unknown character

return EOF\_TOKEN; // End of file

} // A curly brace to end the switch case

} // A curly brace to end the lookup function

int lex() { // Lexical analyzer

lexeme.clear(); // Starting with empty lexeme

getNonBlank(); // Skips whitespace

switch (charClass) { // Using switch case

case LETTER: // In case the current character is a letter

addChar(); // Add character to lexeme

getChar(); // Reading characters

while (charClass == LETTER || charClass == DIGIT) {

addChar(); // Add letter or digit to lexeme

getChar(); // Read characters

} // A curly brace to end the while loop

nextToken = IDENT; // The next token is an identifier

break; // Stop loop

case DIGIT: // In case the current character is a digit

addChar(); // Add character to lexeme

getChar(); // Read characters

while (charClass == DIGIT) {

addChar(); // Add digit to lexeme

getChar(); // Read characters

} // A curly brace to end the while loop

nextToken = INT\_LIT; // The next token is an integer

break; // Stop loop

case UNKNOWN:

nextToken = lookup(nextChar); /\* In case the current character is an operator

look up the next character \*/

getChar(); // Read characters

break; // Finish the case

case EOF\_TOKEN:

lexeme = "EOF"; // In case of end of file the lexeme equals EOF

nextToken = EOF\_TOKEN;

break; // Finish the case

} // A curly brace to end the switch cases

cout << "Next token is: " << nextToken << ", Next lexeme is: " << lexeme << endl; // Output the token and lexeme

return nextToken;

} // A curly brace to end the lex function

1. **References**

Guru99

Wikipedia

Learn Microsoft

W3Schools

GeeksforGeeks

**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.