## Simple Lang Compiler

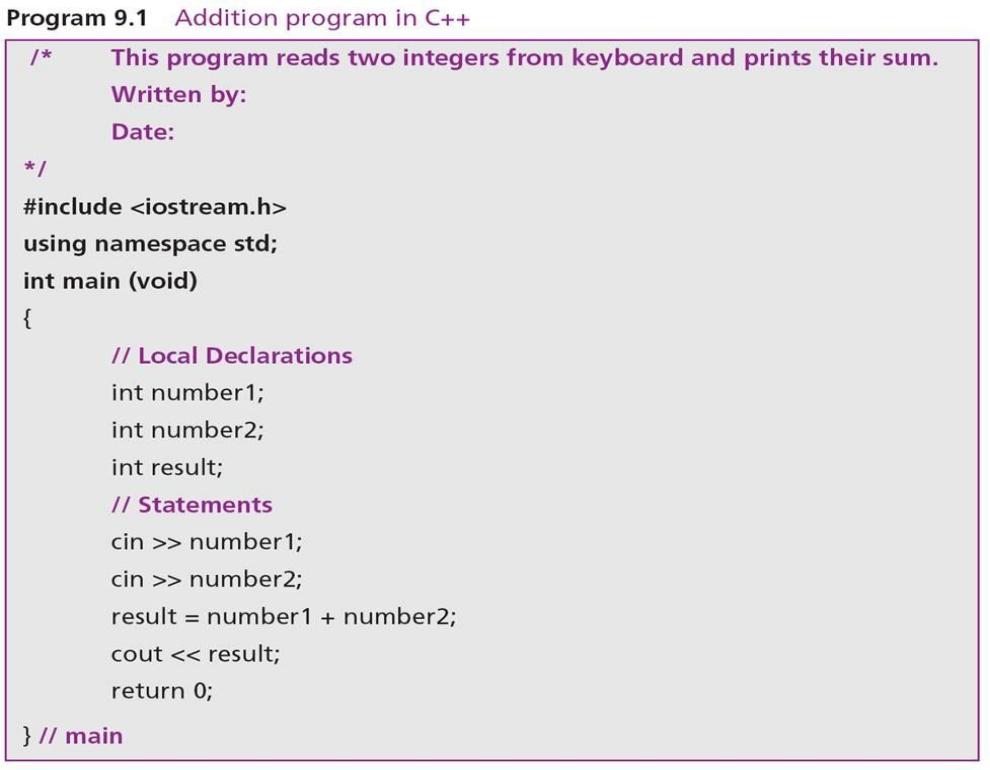
Programmed to work with 8 Bit CPU

**CONENTS:-**

* **Introduction to Simple Lang**
* **Features of Simple Lang**
* **Memory Management**
* **Instruction Set Overview**
* **Control Structures**
* **Data Types and Variables**
* **Error Handling**
* **Paeser in Literature**
* **Lexer**
* **Lexer vs. Paeser**

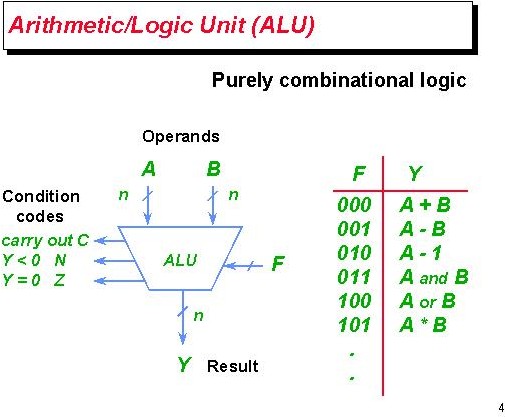
## INTRODUCTION

* SimpleLang is a minimalistic high-level language designed to run on an 8-bit CPU. It includes basic constructs such as variable declarations, assignments, arithmetic operations, and conditional statements, but it does not include loops. This language aims to be easy to understand and implement for educational purposes.
* This typically allows for the total of 256 different values to be represented from 0 to 255.
* A compiler is a computer program that helps in translating the computer code from one programming language into another language. Basically, it translates the program written in the source language to the machine language.

Simple Lang is a high-level programming language designed for 8-bit CPUs.

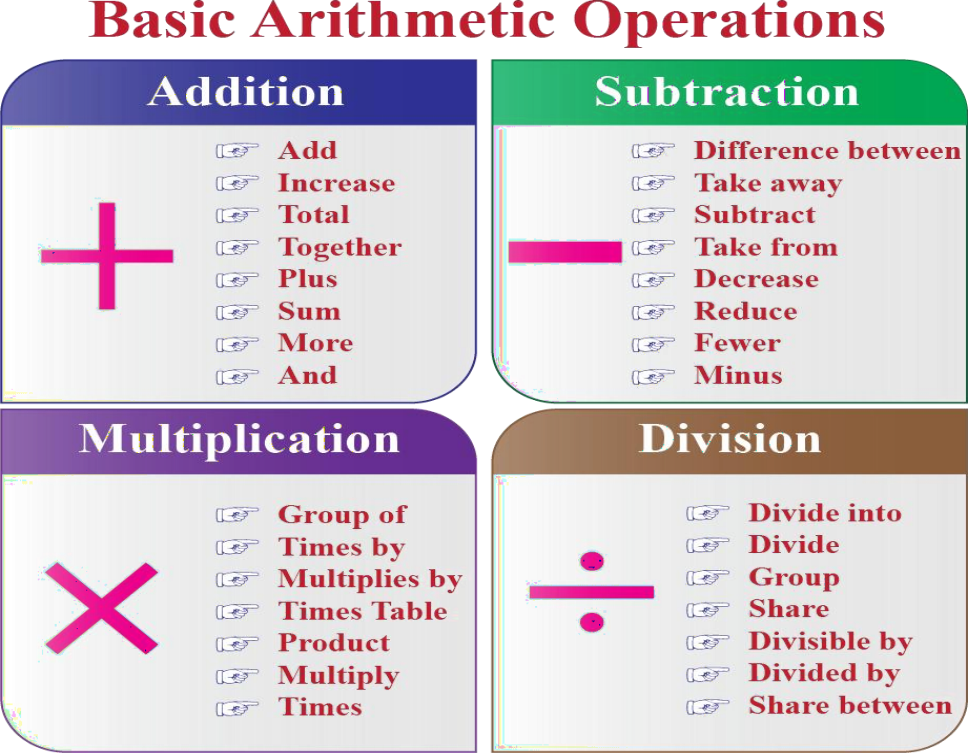
It aims to simplify development by abstracting hardware details.

The language is particularly suited for educational purposes and small-scale applications.

Simple Lang supports basic arithmetic and logical operations.

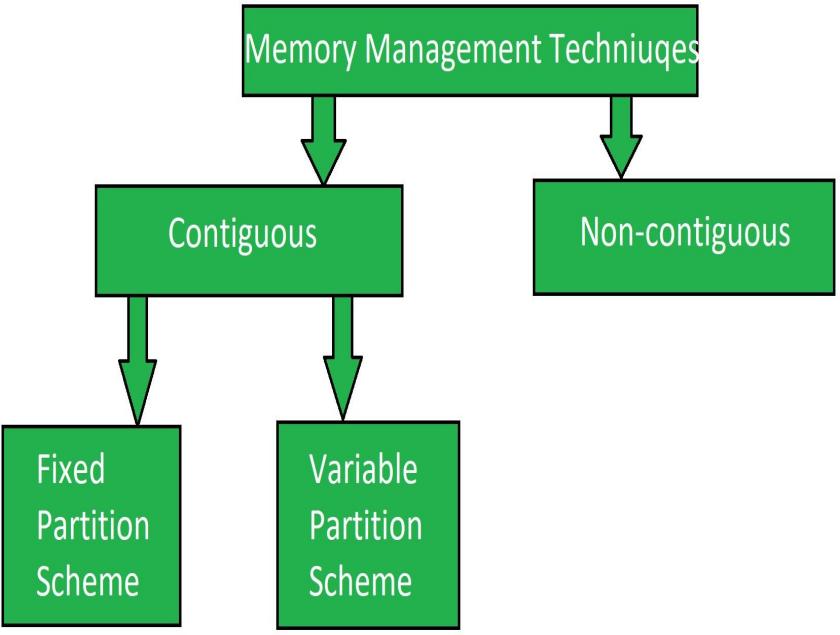
It includes control structures like loops and conditional statements.

The language is designed to be intuitive, making it easy for beginners to learn.

Simple Lang supports basic arithmetic operations such as addition, subtraction, multiplication, and division.

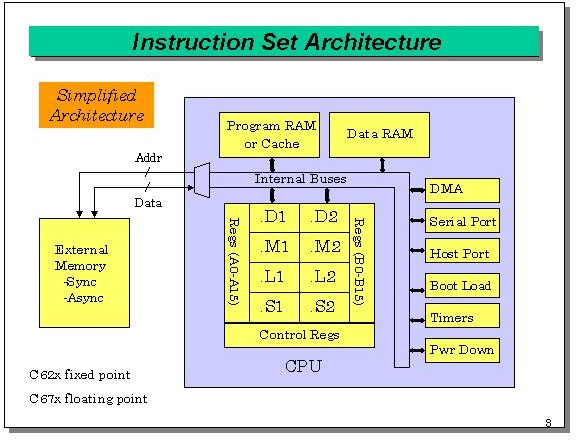
It includes control structures like loops and conditionals to facilitate decision-making.

The language also allows for variable declaration and basic input/output operations.

Simple Lang uses a straightforward memory model that maps directly to 8-bit architecture.

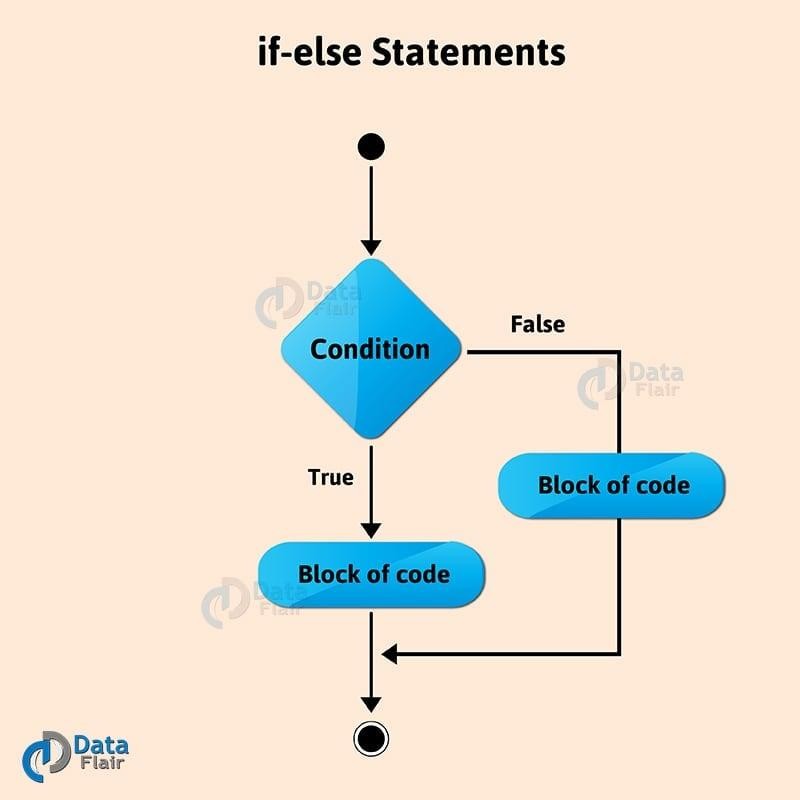
Variables are stored in a fixed number of registers and memory locations.

The language allows for easy access and manipulation of memory, crucial for performance.

The Simple Lang instruction set includes basic commands for data manipulation.

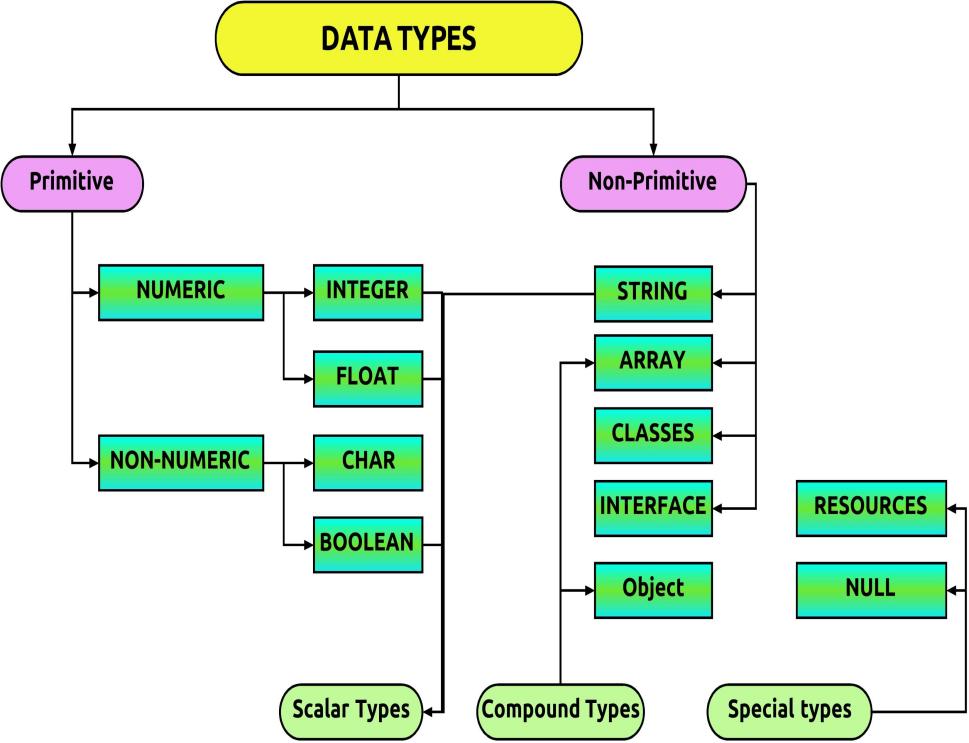
Instructions are represented in a compact binary format suitable for 8- bit CPUs.

Common operations include load, store, add, subtract, and conditional jumps.

Simple Lang provides if-else statements for decision-making processes.

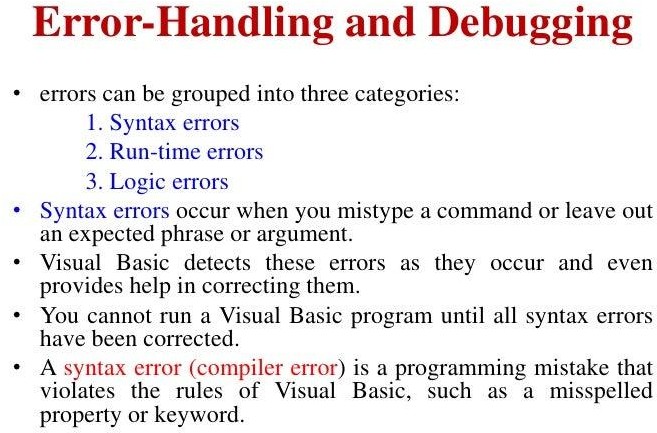
Looping constructs such as for and while allow repeated execution of code blocks.

These control structures enhance the ability to write complex algorithms simply.

Simple Lang primarily supports integer data types due to the 8-bit architecture.

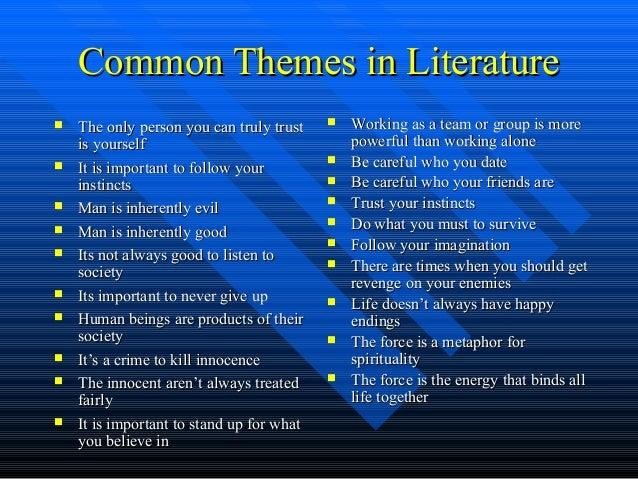
Variables are dynamically allocated, allowing flexibility in coding.

The simplicity of data types helps prevent common programming errors.

Error handling in Simple Lang is straightforward, focusing on common issues.

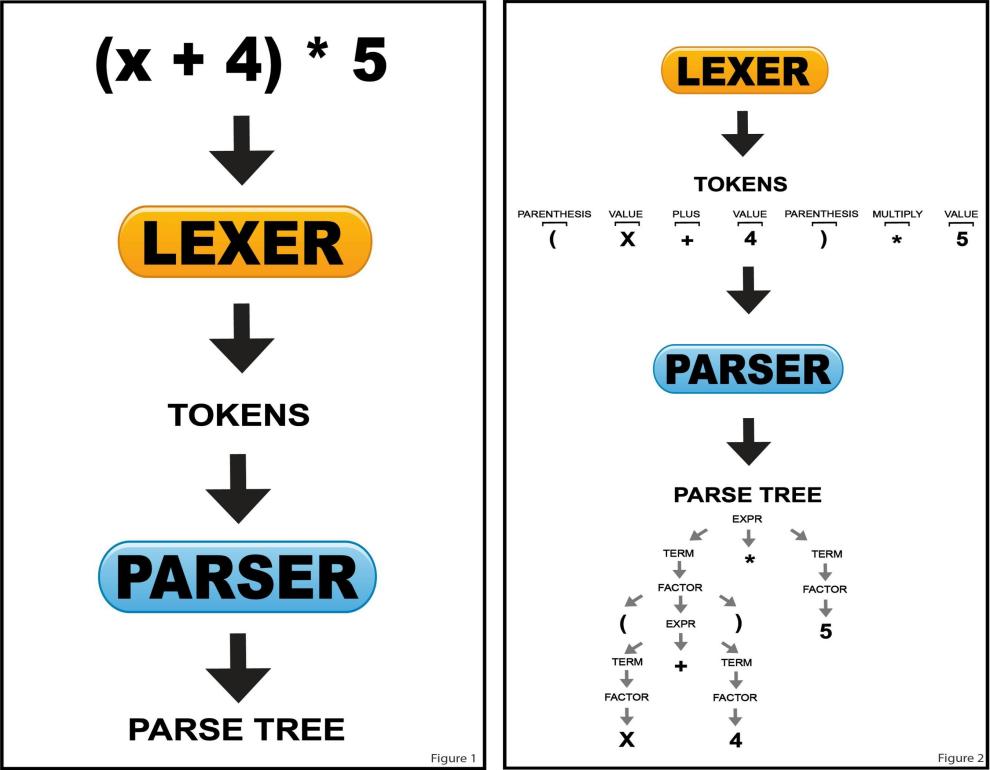
The language provides basic debugging tools to track runtime errors.

Clear error messages help developers quickly identify and fix problems.

Literature often reflects themes related to paeser, shaping narratives and character development.

Authors may use paeser to convey complex ideas and emotions in their works.

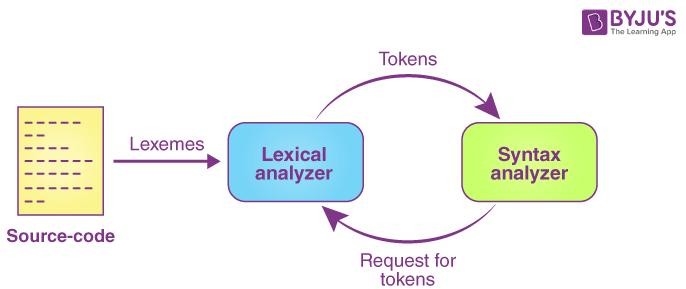
Understanding this relationship enriches the reading experience and interpretation of texts.

The lexer simplifies the parsing process by breaking down the input into manageable pieces.

It identifies keywords, operators, identifiers, and other significant symbols.

By classifying the input, lexers help ensure that the syntax of the language is correctly interpreted.

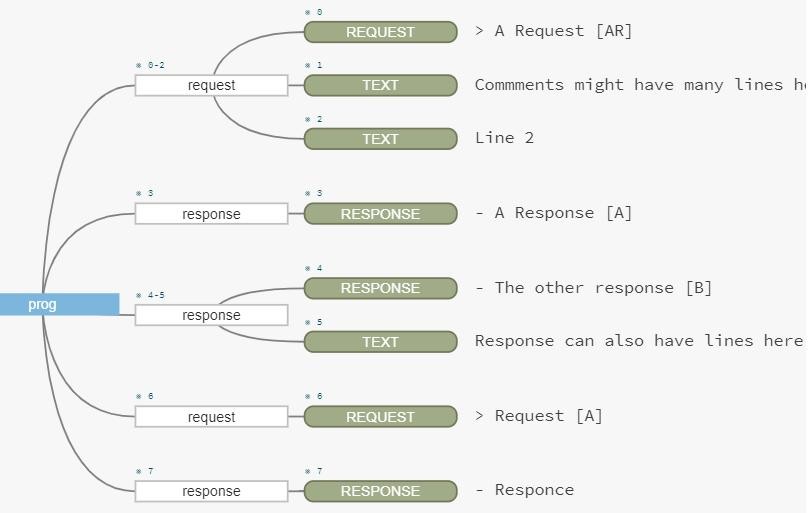
# Lexers

A lexer, or lexical analyzer, is a crucial component of a compiler or interpreter.

Its primary function is to convert a sequence of characters into a sequence of tokens.

Tokens are meaningful symbols that represent basic elements of the programming language.

# Lexer vs. Parser

While the lexer focuses on token generation, the parser deals with the structure of those tokens.

The parser uses the tokens to build parse trees or abstract syntax trees for further processing.

Both components work together to ensure that source code is correctly understood by the compiler.

## Thank-you

The link to the project, https://github.com/Sireeshanarayanasetti/recruit---vicharak.git,

As instructed the repository is private and "recruit---vicharak" has been invited, Hopefully everything is to your liking,

Please do contact me for any further discussion, Have a good day!

Best Regards,

Manepalli Navya Gnana Sindhu

navyamanepalli44mail.com