UNIT-1

① Language Evaluation Criteria -

	CRITERIA		
Characteristic	Readability	Writability	Reliability
Simplicity	✓	✓	✓
Orthogonality	✓	✓	✓
Data Types	✓	✓	✓
Syntax Design	✓	✓	✓
Support for Abstraction		✓	✓
Expressivity		✓	✓
Type Checking			✓
Exception Handling			✓
Restricted Aliasing			✓

The fourth primary criterion is 'cost', which is not included in the table because it is only slightly related to the other criteria and the characteristics that influence them.

→ Readability - Ease with which programs can be read and understood
Simplicity - language should be as small as possible avoid redundant feature
Orthogonality - A relatively small set of primitive constructs can be combined in a relatively small number of ways to build the control and data structure of the language.
Data Types - It should be defined clearly with meaning.
Syntax Design - Special words and 'form and meaning' affects readability.

→ Writeability - How easy to create or write a program for given problem.
Support for abstraction - Need to able to define and utilize abstraction.
Expressivity - provide convient ways for specifying computations.

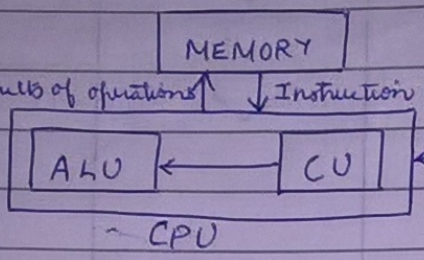
→ Reliability - Program performs to its specification under all conditions.
Typechecking - Testing for type errors in a given program.
Exception handling - Ability to intercept run-time errors.
Restricted aliasing - Two or more distinct names that can be used to access the same memory cell.

② Influences on language design -

Factors that influences the base design of programming languages.

(1) language evaluation criteria
(2) Computer architecture
(3) Programming design methodologies

Computer architecture (Von Neumann Architecture) -



CU → CONTROL UNIT
ALU - Arithmetic logic Unit
CPU - Central Processing Unit

Programming Design methodologies -

1960 → structured programming (ALGOL, ADA)
1970 → procedure-oriented programming (C, GO, FORTRAN, PASCAL)
late 1970s → data-oriented programming (SIMULA 67)
early 1980s → Object oriented programming (C++, JAVA, smalltalk)

③ language Categories -

Imperative, functional, logic and object oriented.

Imperative - Uses variables, assignments statements & iteration
Functional - Uses functions, parameters & recursion
logic - Uses facts and rules
Object oriented - Uses data abstraction, inheritance & dynamic method binding

④ Programming Paradigms -

It provides the view that the programmer has of the execution of the program. There are mainly four computational models that describe most programming today -

(1) Imperative or procedural programming / languages
(2) Functional or Applicative programming / languages
(3) logical or Rule-based programming / languages
(4) Object oriented programming / languages.

Imperative languages -

They are command driven or statement oriented languages. A program consists of a sequence of statements, and the execution of each statement causes the computer to change the value of one or more location in its memory.

Syntax - Statement 1;
Statement 2;
.....
Statement n;

Eg - C, C++, FORTRAN, PASCAL, ADA

Functional languages -

Here we look at the function that must be applied to initial machine state, by accessing the initial set of variables and combine them in a way that leads to the problem solution.

Syntax - $function_n(\ldots function_2(function_1(data)\ldots))$

Eg - LISP, ML

Logical languages -

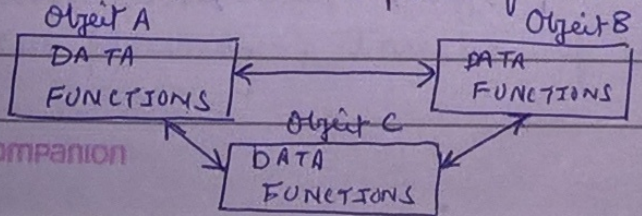
It checks for the presence of a certain enabling condition and when present, executes an appropriate action.

Syntax - $enabling\ condition_1 \rightarrow action_1$
$enabling\ condition_2 \rightarrow action_2$
.....
$enabling\ condition_3 \rightarrow action_3$

Eg - Prolog

Object-oriented programming (OOP) -

It creates partitioned memory area for both data and functions that can be used as templates for creating copies of such modules on demand.



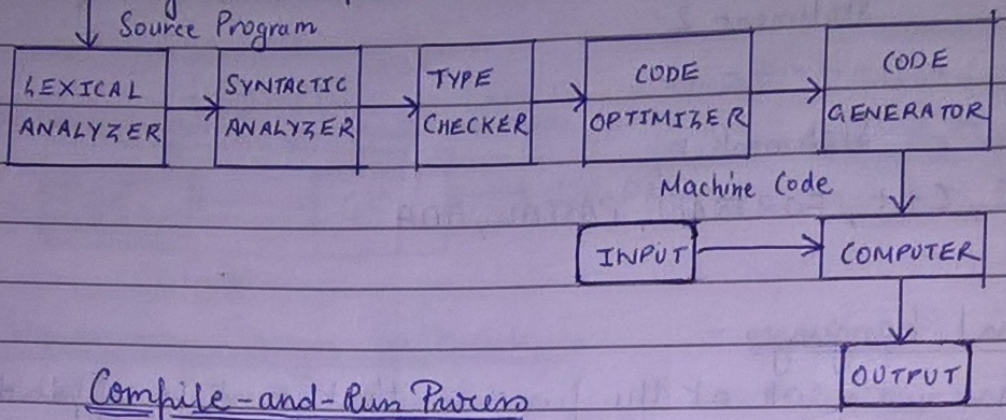
Eg:- Small Talk, C++, JAVA

⑤ Programming language Implementation –

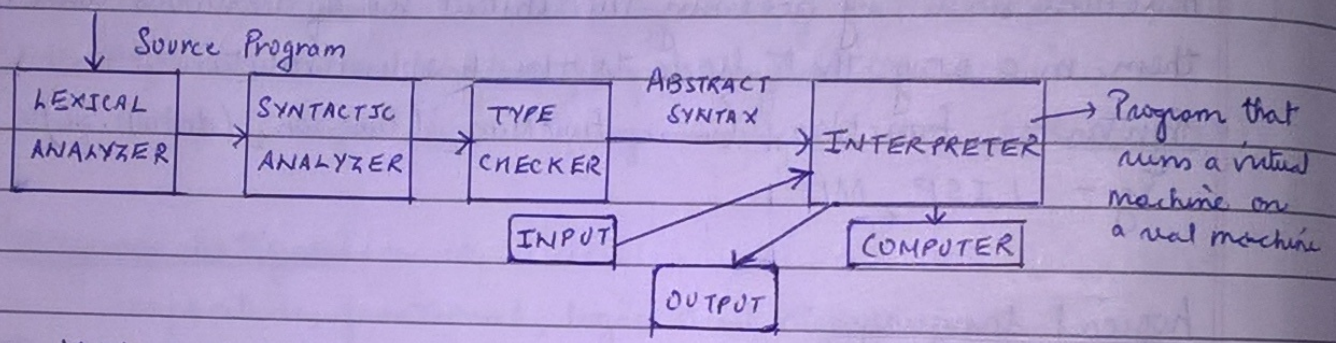
An implementation of a programming language requires that programs in the language be analyzed, and then translated into a form that can be either:-

(1) Run by a computer (i.e. "Real Machine") called compiling

(2) Run by an interpreter (i.e. "Virtual Machine") called interpreting.



Compile-and-Run Process



Virtual Machines and Interpreter

⑥ Programming Environments –

A programming environment is the collection of tools used in the development of software. This collection may consist of only a file system, a text editor, a linker, and a compiler, or it may include a large collection of integrated tools, each accessed through a uniform user interface.

Eg: UNIX (1970s), Borland JBuilder, Microsoft Visual Studio .NET (C#, Visual Basic .NET, JScript, F#, C++/CLI), Netbeans (Java, JavaScript, Ruby, PHP)

⑦ Issues in language Translation -

→ Syntax -

It describes the structure of programs without any consideration of their meaning

Key criteria concerning syntax -

Readability, Writeability, Verifiability, Translatability and lack of ambiguity

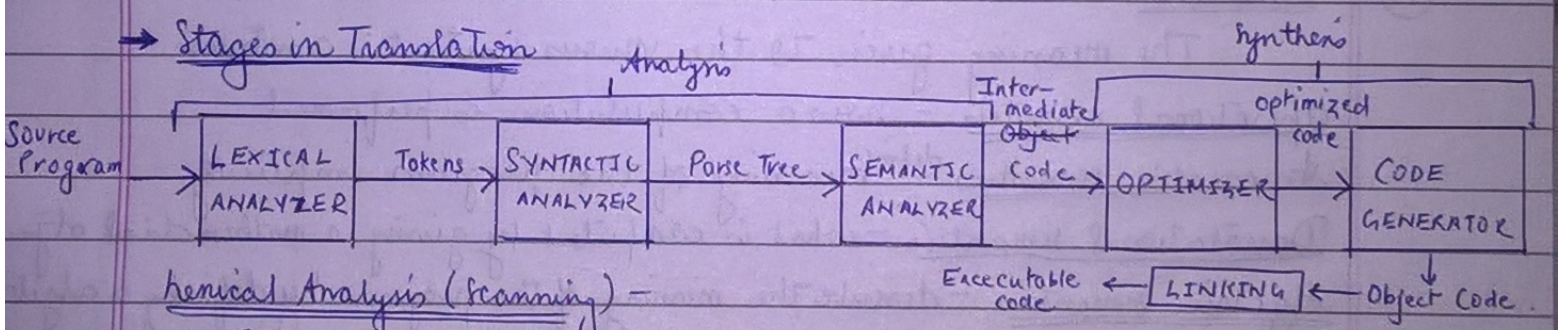
Basic Syntactic concepts -

Character Set (Alphabets), Identifiers (String of letter of digits usually begins with a letter), Operator Symbols (-, +, *, /), Keywords, Noise Words (optional words to improve readability), Comments, Blanks, Delimiters (used to denote the beginning and the end of syntactic constructs), Expressions, Statements

Overall program - subprogram structure -

(1) Separate subprogram definitions → separate compilation, linked at last time
(2) Separate data definitions → General approach in OOP
(3) Nested subprogram definitions
(4) Separate interface definitions → C/C++ Header files.
(5) Data description separated from executable statements
(6) Unsuppressed subprogram definitions

→ Stages in Translation



Lexical Analysis (Scanning) -

Identifying the tokens :- keywords, identifiers, constants.

Syntactic Analysis (Parsing) - determining the structure of the program

Semantic Analysis - Assigning meaning to the syntactic structures.

Basic Semantic tasks - Symbol-table maintenance, Error detection, Macro processing, Insertion of implicit information.

Optimization → Removing redundant statements
Code Generation - obtaining the object code of the program
Linking and loading - obtaining the executable code of the program

(8) Context Free Grammer (CFG) -
A context free grammer or CFG is represented by 4-tuple (V, T, P, S) where
V → set of variables or non-terminals (phrase structures)
T → set of terminals (words or token symbols)
P → set of productions (produce the things of language)
S → Starting variable.

(9) Backus-Naur Form (BNF) - (equivalent to CFG)
If CFG productions has used only the metasymbols "→" and "|". (Sometimes parenthesis are allowed to group things together) It is used to describe another language.

(10) Parse Tree - A hierarchical representation of a derivation.

(11) Semantics -
The meaning given to the various syntactic constructs.
Operational semantics - how a computation is performed
Axiomatic semantics - define meaning of the program implicitly
Denotational semantics - what is computed by giving a mathematical object
Algebric semantics - describe the meaning of the program by defining a algebra
Translation semantics - how to translate a program into an other language.