History of programming languages :-

① 1883 : Lady Ada Lovelace

⇒ Charles Babbage is known as the father of computer because he created the first working machine.

⇒ But the first computer program was written by Lady Ada Lovelace. [a program to compute bernoulli numbers]

② 1949 : Kathaleen Booth

⇒ The first assembly language was written by Kathaleen Booth

③ 1957 :

⇒ The first high level programming language called FORTRAN was designed by John Backus for scientific & engineering calculations.

④ 1958 :

⇒ some more high level languages were developed like ALGOL (Algorithmic language) & LISP for symbolic calculation. ALGOL laid the foundation of structured language block format

⑤ 1959 :

⇒ COBOL (Common Business Oriented Language) was formed for business data processing in finance.

⑥ 1964 :

⇒ BASIC (Beginner's All purpose Symbolic Instruction Code) was designed to be beginner friendly and teaching programming to others.

⑦ 1972 :

⇒ One of the most influential language called the mother of

all programming language, C was developed by Dennis Ritchie in USA.

→ It laid many important concepts which are used by all the programming language today.

(8) subsequently various other programming languages were developed for different sectors & applications

Pascal - (1970)
C++ - (1983)
Pearl - (1984)
Python - (1991)
Java - (1995), Javascript, Ruby
C# - (2000)

Characteristics & features of a programming language

1. Readability
2. Writeability
3. Reliability
4. Portability → Platform / machine independence
5. Library & community support
6. Performance
7. Scalability
8. Security
9. Documentation
10. Flexibility → paradigms.

1. Readability → The lang. should have easy to read, clear and concise syntax to make the code visually appealing.

2. Writability → It should allow developers to express their ideas in efficient code.

3. Reliability → It should offer proper error handling mechanisms and well-defined behaviour.

4. Portability → The lang. should be compatible across diff. platforms and machines i.e. same code should be able to run on various platforms.

5. Library & community support → Active community of developers can significantly enhance a programming language's capability and problem solving. This is because of wide range of libraries, tutorials, forums & resources.

6. Performance → A programming language should be efficient in code execution, memory management & optimisation.

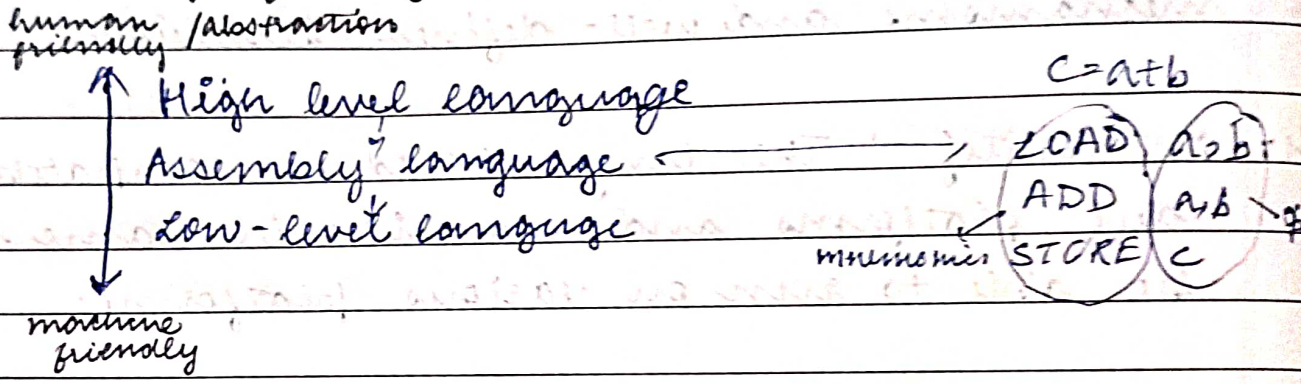
7. Scalability → The lang. should provide tools and features for developing both small scale & large scale applications acc" to the requirements.

8. Documentation → The lang. should have a well structured & comprehensive documentation in order to guide new developers or to share knowledge.

9. Security → The lang. should be able to handle common threats & security vulnerabilities.

10. Flexibility → Language should allow different programming paradigms to cater preferences and requirements of diff. developers.

Hierarchy of programming language :-



High Level language :- They are designed to be more human friendly and understandable. They are highly abstract from the details of computation.
Ex. C, C++, JAVA, Python

Assembly language :- They use mnemonics & symbols to represent machine instructions making them slightly understandable. They have lower abstraction & are machine dependent.
Ex. X86 Assembly, ARM Assembly

Machine language :- They consists of binary code and instruction & work directly with the computer hardware. They have almost no abstraction & are difficult to understand by humans.

	HLL	AL	ML
Abstraction	highest	medium	lowest
Understability	highest easily readable	medium understable if meanings of Mnemonics are known	lowest difficult to understand
Portability	highly portable	less	not
Efficiency	least efficient	moderate	highly efficient
Development time	least time	moderate time	highly / most time
Debugging	〃	〃	〃
Application areas	general purpose programming	OS, hardware interfacing & optimisation	used by compilers & assemblers
Control on hardware	least	moderate	highest control

Ex.

HLL → c = a+b;

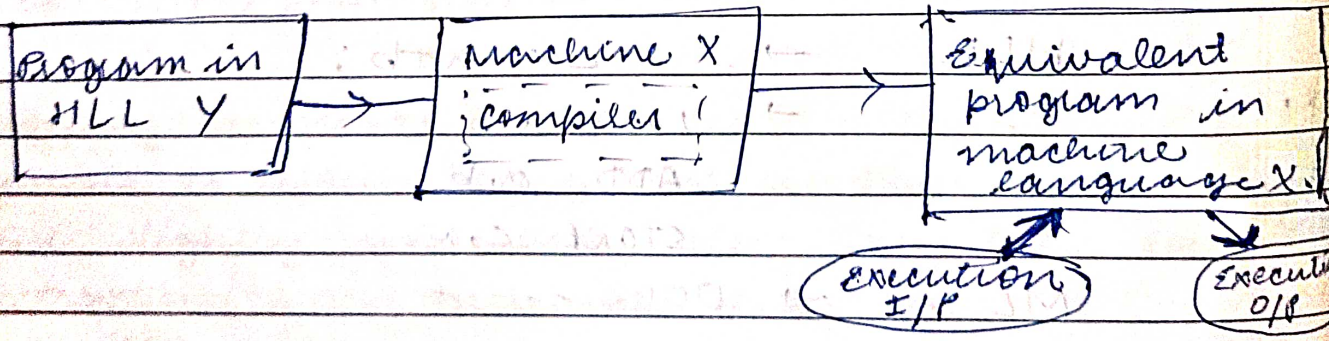
AL → LOAD a,b
ADD a,b
STORE c,a

ML → 001.---.---

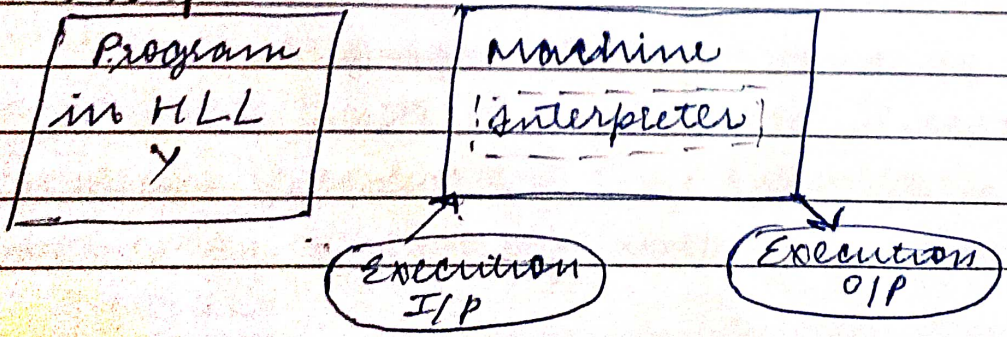
Language Translators :-

	Compiler	Assembler	Interpreter
I/P	entire source code in HLL	Assembly instructions & entire code	line by line source code in
O/P	produces an executable file of machine code	machine code/ object code	outputs the code directly line by line
Efficiency (Execution)	highly efficient because entire program is executed at once.	highly efficient	less efficient beca it will always conv the program line by l
Errors	All errors are reported at once	limited error checking	Errors are reported line by line as they occur.
Debugging	difficult bcoz all errors have to be solved before execution	NIL	easy because th are solved at the time of their occur
Portability	not portable bcoz program needs to be recompiled for diff platform.	not portable	portable (independ for each platform)
Memory	requires less overhead memory.	least memory usage.	more runtime & memory usage bec interpreter is loaded into memory each time.

1) Compiler →



2) Interpreter →



Stages of program execution :-

1. Writing / editing → (Text editor)
2. Compilation / Interpretation →
3. Linking

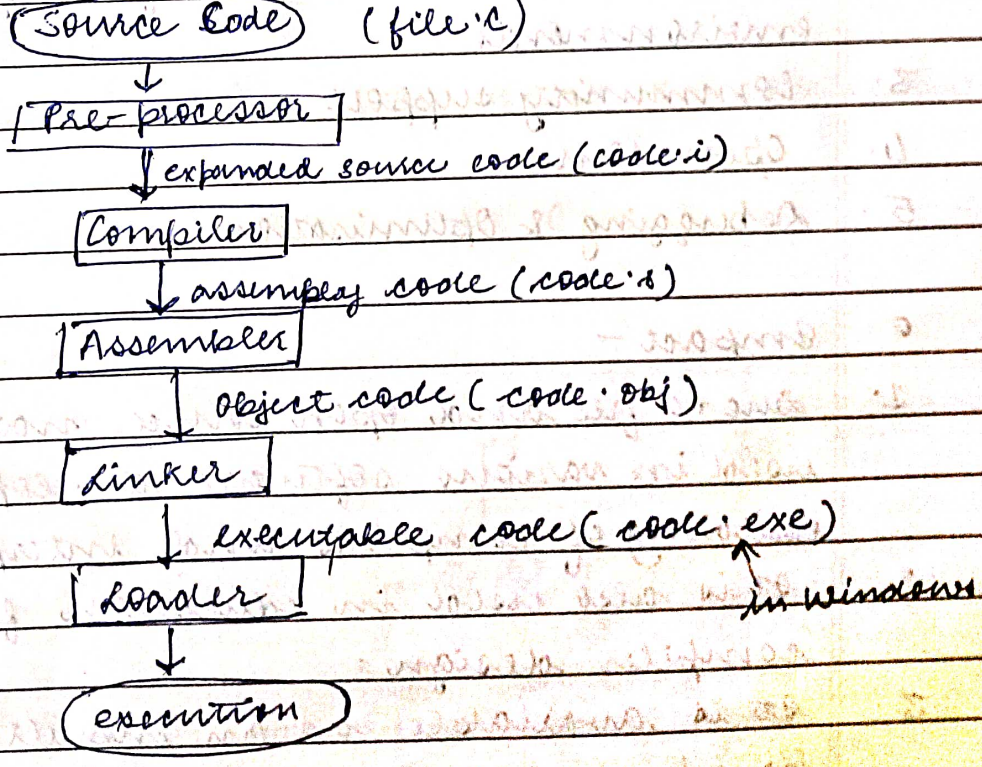
Linking in compiled languages, the object code might need to be linked with external libraries and other files to create a complete executable code. This process of linking is done by compiler itself.

4. Loading → (OS)

After the object file is generated, the task of compiler is finished. Now, the OS loads the executable file into memory (RAM). It includes allocation of all the program elements into seperate memory locations along with their addressing.

5. Execution :-

CASE STUDY OF C :-



Case study on GCC:-

- History: GCC stands for (GNU compiler collection) It is widely used open-source compiler system developed by FSF (Free software Foundation) in early 1980s.
- Its first official version was released in 1987 (GCC 1.0) with the support for compiling C language.

- Features :-
1. Language support → It supports C, C++, Ada, FORTRAN etc.
   Formula Translator
2. Portability (cross-platform) → GCC can target multiple platforms due to its cross compilation property. Developers can write code for one system and generate executable files for different machine environments.
3. Community support
4. Open source
5. Debugging & Optimisation

- Impact -
1. Due to free and open source nature, GCC is widely used in various software development projects, building of compilers and interpreters.
2. It is also used in education field for teaching compiler design.
3. It is available as an inbuilt compiler on all Linux based operating system.

#

Syntax :→ It is defined as the arrangement of words / elements in a statement of a programming language. It is basically a set of rules concerned with the arrangement of elements and not the meaning of statements.

Characteristics :→

A syntax should be easily :

1. readable
2. writable
3. Verifiable
4. Translatable
5. Unambiguous

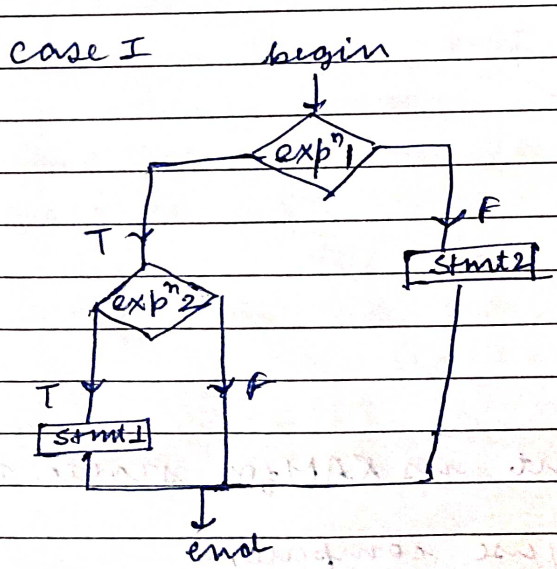
Ex. of unambiguous :-

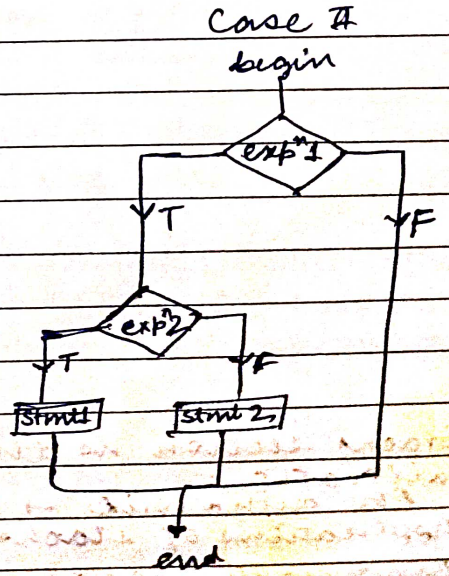
Pascal or ALGOL

(I) if expression ~~then~~ statement 1 else statement 2

(II) if expression then ~~statement 1~~ stmt 1

⇒ if expression then if exp^n 2 then stmt 1 else stmt 2





≡ Syntactic elements of a programming langu

1. Character set (A-Z, a-z, 0-9)

2. Identifier rules

3. Operators → <= -(a (shellscript)

4. Keywords - if [ ] while [ ]
then do
else done

5. Comments → // # (shellscript)

6. Delimiters → { }, ( ), [ ], , ; ; CSV

7. Blank spaces → Ex. Python (INDENTATION SENSITIVE) | C, C++ (NOT INDENT)

8. Free-fields or fixed-field formats
C, C++, Java Assembly lang.

9. Operations or expressions

Semantics :- It refers to the meaning associated with the statements of a language.

→ Semantic errors are usually encountered at runtime.

Ex. name collisions,
division by 0,
type mismatch

Data objects :-

It represents a location or container for storing data values in memory. Whenever a data object is created certain properties or characteristics get associated with it. These properties are called binding.

Ex. name
value
type
location
size
lifetime

variable, constants, literal

variable → value can change

constants → value doesn't change in the whole program.

Ex. # define MAX 30 ← value of constant
MAX → constant

Data Type - A class of data objects together with a set of operations for creating and manipulating them.

Also, → Built-in, → Derived, → User-defined

Data can be classified as :

- Elementary (containing a single data object)
- Structured (aggregate of various data objects)
  - Linear
  - Non-linear

Specification of elementary data types :-

1. Attributes → Features that distinguish the data object from others like name, type, etc. location
all the attributes in a data object are stored in a descriptor (Dope (VECTOR))
array

2. Values → An ordered set of possible values (range) that data type can contain

3. Operations → Operations are mathematical functions with a well defined and range of values.
→ Each operation has a signature to specify its rules and descriptions.

Syntax : Operation name : arg type x arg-type ..... arg-ty
↓
result typ

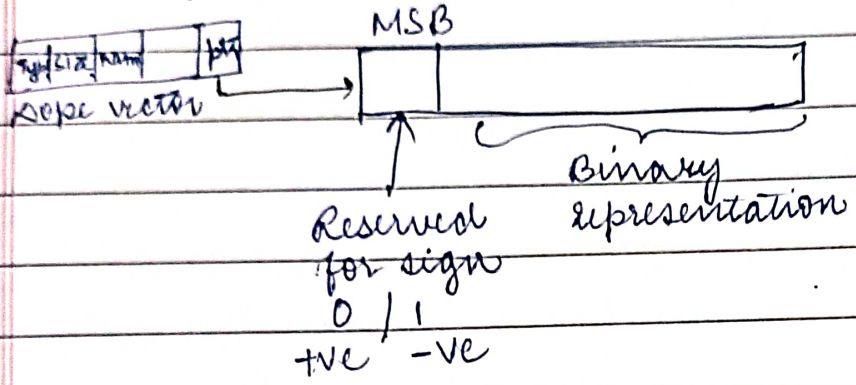
example: + : int x int → int
== : int x int → Boolean
SQRT : real → real

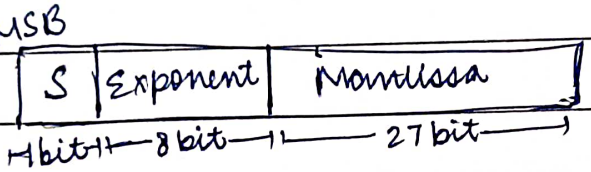
4. sub-type : If a data type is a part of a larger class then it is declared as its sub type.
Ex. in C; long, short, char all are sub types of integ
In Pascal; the data type small Integer is a sub type of integer with range 1 to 20.

Implementation of elementary data types :-

1. Storage representation → All the values contained data objects along with the decoded instructions are represented in strings of binary no.s. The representation is strongly influenced by the underline hardware of the machine.

Storage of integers in memory :-





2. Operation implementation →

i) Directly as a hardware operation.
Ex. basic logic gates like AND, OR or NOT are used to perform logical or relational operations, circuits such as half adder, full adder, parallel adder are used to perform basic arithmetic operations.

ii) Implemented as a func" or sub-program.
Ex. complex operations such as square root, multiplication can be implemented by an algo. to be executed on hardware.