①

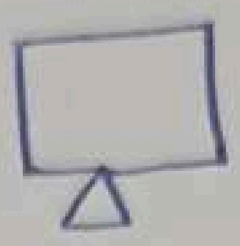
03/04/25

→ L2 - Basics of Programming(II)

→ Programming Language?

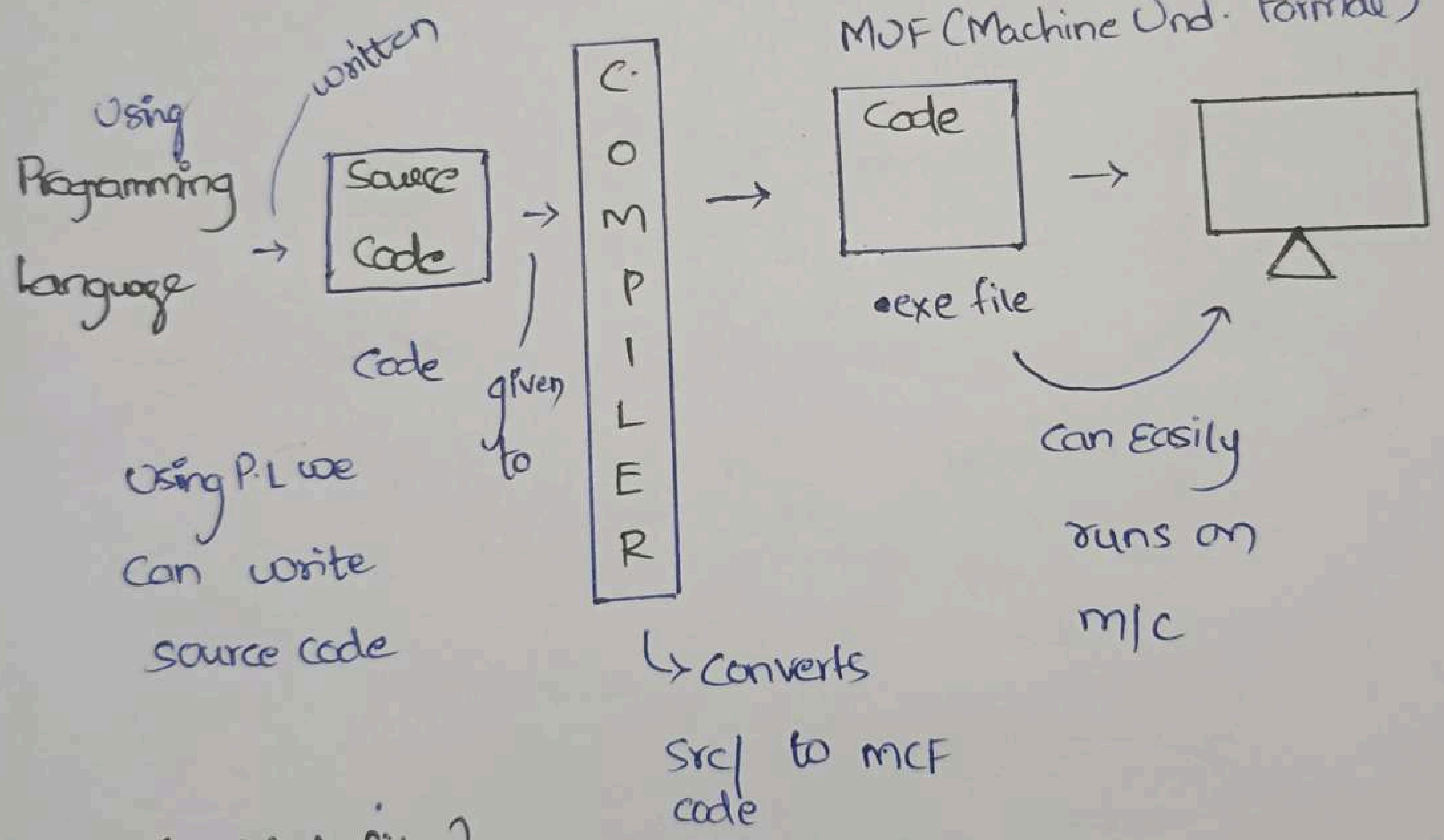
what? * Set of Instructions given to machine to perform some tasks.

why?



* Set of Instructions → C++, Java, python, . . .

→ Compilation Process:



Using Programming Language
written
Source Code
Code
given to
Using P.L we can write source code
COMPILER
↳ converts src/code to MCF
MOF (Machine Ond. Format)
Code
•exe file
Can Easily runs on m/c

What is object file?
what is exe file?
} H/w (open & Explore)

→ where to code?

Vs-code (windows) (or) Codehelp.in/quick-compiler

→ First Code:

⊙ C++ Code
  ↳ Starting point
    ↳ int main / void main
      ↳ (discussed further)

returntype ↑

⊙ int main() → function name ↳ function

```
{
    ≡
    return 0; → termination
}
```

↳ Scope of function.

$\text{i/p} \rightarrow \boxed{\text{function}} \rightarrow \text{o/p compulsory}$

(may or may not take)

⊙ C++ prog will be runned by the Operating System

$$\begin{tikzcd}
\text{Operating System} \arrow[r, "\text{calls}"] & \text{main() func} \arrow[r] & \text{Executes code Inside main} \arrow[r] & \text{return0;} \arrow[lll, bend left, "\text{tells Successful Execution to OS}"]
\end{tikzcd}$$

(that code runned Succ without any Error)

Pre-Processing-Directive.
↑

```
⊙ #include <iostream>   → It is a file contains Input/output func code.
using namespace std;    ↳ folder
int main()
{
  cout << "Asif Nihal";  → cout is used to print o/p
}
```

• Print (keyword/obj/method)
  ↳ cout << "Asif Nihal";

- The Defination of cout is available in <iostream> file contains Input/output code. ff It is available in Standard namespace (std) folder.

- why Header file ~~only~~ ~~have~~ ~~#~~ ~~why~~ ~~#~~ is not ending with ;

Ans: Headerfile is not ending with ; because It is following the syntax ff rules of prog.

→ <u>Input/output In C++:</u>

$$\begin{tikzcd}
 & \text{Keywords} \arrow[dl] \arrow[dr] & \\
\text{Input/Read} \arrow[d] & & \text{output/print} \arrow[d] \\
\text{Cin} & & \text{cout}
\end{tikzcd}$$

<u>Syntax:</u>
cin>>

Syntax: cout<<

```
Ex: Cout << 2;
    Cout << "mohit";
    Cout << 5 << endl;
    cout << endl << endl << 2 << endl;
```

For nextline we use: endl;

Alternate ~~[illegible]~~ of endl: "\n" (newline character)

Input: cin >> syntax

The value of Input will be storing in the memory

Q) The Smallest memory block containing address refers to

bit or byte.

$\downarrow$ 0,1

$\downarrow$ 8bits (00001010)

$\hookrightarrow$ This contains address that can be refereable.

204
$\boxed{8}$
Ramesh $\hookrightarrow$ It is a Var. name

→ Variable:

- Variables are the Containers of storing data values

Data $\rightarrow$ Stores in Mem. Block $\rightarrow$ uska ek address rehta hai (add) $\hookrightarrow$ Usko harbar yaad nai rakh sakte.

Address ko yaad rakhne ke liye Variable use karte hai $\hookrightarrow$ ab hum variable ko uske nam se bula sakte hai instead of (rem add & calling)

→ **Datatypes:**

- The datatype Specifies the size and type of Information in which the Valiable will Store

Datatype
- ↳ int -ve, +ve, 0
- ↳ Chae 'a', 'b', . . 'z', 'A', 'B', . . . 'Z',
- ↳ boolean true/false
- ↳ void · Discussed fuether
- ↳ float, double 1.23, 3.14823567

Ex: int ageOfAariz = 12; $\boxed{12}$ AgeOfAariz 4bytes   bool asif = 1 or true

float pi = 3.14f;

chae grade = 'A';   double b = 4.13;

Vaiable ⟶ ① Declaration (No initialization)

int age; , float price;

② Defination

int age = 12; float price = 3.14;

③ Initialization ↷ refers to defⁿ

→ Q) what is smallest addressable Space byte or bit?

Ans: In boolean We are using 1byte for representing true or false

Size of Datatypes

Datatypes	32-bit CPU	64-bit CPU
Chae	1	1
Short	2	2
int	4	4
long	4	8
longlong	8	8
float	4	4
double	8	8

→ **Naming Conventions of variables**

Rules:

✓ It should begin with an alphabet

Ex: float 1a ✗ float a ✓

✓ There may be morethan one alphabet, but without any spaces between them

Ex: float age of aari ✗ ✗ float ageofAari ✓

✓ Digits may be used but only after alphabet Ex: int 10age ✗

int age10 ✓

✓ No special symbols can be used except the (_) symbol. when multiple words are needed, an underscore should seperate them

Ex: int age.+of+babbar ; int a+b+c ;

Snakecase·
int age_babbar ✓

int ~~ar-bear~~ age_of_asif ✓

Camel case: ageOfAariz

age → Lower
Of → upp
Aariz → Upp

w1 w2 w3 w4
w1 ↓ L·C
w2 ↓ 1stchar U·C remaining l·c
w3, w4 ↑ 1stchar U·C & rem l·c

✓ No keywords or command can be used as a variable name
Ex: int main = 5 ; int cout = 5 ;

✓ All statements in C++ language are case-sensitive. Thus a variable (in uppercase) is considered different from a variable declared (in lowercase)

Ex:
```
int love=5;  }
int Love=5;  } both are different
```

H/w: Binary N·S } in learn C++
Decimal N·S }

Q) Difference in storage of +ve and -ve Integers?

Ans:

int a=5

$\hookrightarrow$ 101

$\nearrow$ data storage in memory

0 0000000 00000000 00000000 00000101

+ve

Sign bit

$\hookrightarrow$ leftmost

$\hookrightarrow$ 0 $\to$ +ve

$\to$ 1 $\to$ -ve numb

int age =10

0 0000000 00000000 00000000 00001010

+ve

1's Compliment, 2's Compliment, Conversion.

Signed $\to$ -ve, +ve, 0

Unsigned $\to$ +ve

ASCII values

a - 65

b - 66

char $\to$ -128 to +127

Ex: char ch= 'a'; $\to$ -128 to 127

A $\to$ 97

$\hookrightarrow$ stores signed char

Unsign char ch = 'a' 0 $\to$ 255

Datatype $\to$ Range Signed $\boxed{-2^{n-1} - 2^{n-1}-1}$

Unsigned $0 - 2^n-1$

er: char = 1byte = 8bits $-2^{8-1} - 2^{8-1}-1$ (-128 - 127)

→ H/w:

(1) HLL, LLL (High-level lang, low-level lang)

Sol^n HLL → written by programmers
LLL → Understandable by machine

→ Operators:

• Operators are used to perform operations on variables and values.

→ Arithmetic [+, -, *, /, %, ++, --]

→ Relational [==, != , >, <, >=, <=] → comparision based operator
↓ Result will be true or false

→ ~~Arithmetic~~ [=] Assignment

→ Logical [&&, ||, !]

→ Bitwise [&, |, ~, ^, >>, <<]

→ Arithmetic Operators:

+, -, *, /, %, ++ (Incre), -- (Decre)

+	-	*	/	%
add^n	subtr^n	mult	Divsin	modulo

Ex:
```
int a=5;
int b=10;

int answer = a + b;
                 -
                 *
                 /
                 %
Cout << "answer" << answer << endl;
```

→ Assignment Operator : assigning a value to a Variable.

Ex: int $\underset{\text{assigned / given}}{a=5}$

→ Relational operator: Comparision (Result will be true/false)

[==, !=, >, <, >=, <=]

```
int a=10;
int b=10;

cout<< (a==b);
cout << (a!=b);
Cout << (a>b);
Cout<< (a<b);
cout << (a>=b);
Cout << (a<=b);
```

→ Logical Operator: Condⁿ → (T/F) as o/p

&& - Logical AND (both true)

a	b	o/p
T	T	T
T	F	F
F	T	F
F	F	F

|| - Logical OR (atleast one true)

a	b	o/p
T	T	T
T	F	T
F	T	T
F	F	F

!= Logical NOT

a	!a
T	F
F	T