1. Introduction to computer

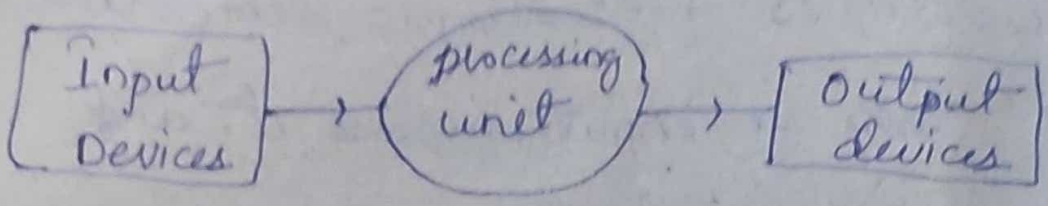
Computer: It is a electronic device which take a input, process it and gives output

→ A computer is a electronic machine which reduces the human effort

→ Computer is consisting of 2 major components i.e. Hardware, Software

* Hardware: The components which exists in real world in physical nature.
  → The components which we can touch & feel.

* Software: It is virtual in nature and doesnot have structure.
  → It will be stored in memory.



Ex: Keyboard, mouse, scanner, camera, Joysticks, Microphone, Touch screen etc.

i) ALU
ii) CU
iii) MU

Ex: Monitor, Speaker, Printer, Projector etc.

1) ALU (Arithmetic and Logical Unit):

i) Arithmetic and logical unit (+, -, *, /, %)

ii) logical (>, <, >=, <=, ==, !=)

2) CU (control unit): It is used to control all the hardware devices connected to the computer.

3) MU (memory unit): i) primary memory (main memory) (volatile)

ii) Secondary memory (Auxiliary) (Non-volatile)

i) Volatile : It is tempotary memory. when computer is shutt down then the data is loss.

eg: RAM, ROM (Non-volatile)
BIOS (Basic Input output system).

ii) Non-volatile : It stores the data permenantly irrespective of power.

eg: HDD, SSD, CD, DVD, memory card, pendrive etc.

⇒ Ups: Uninterrupted power Supply.

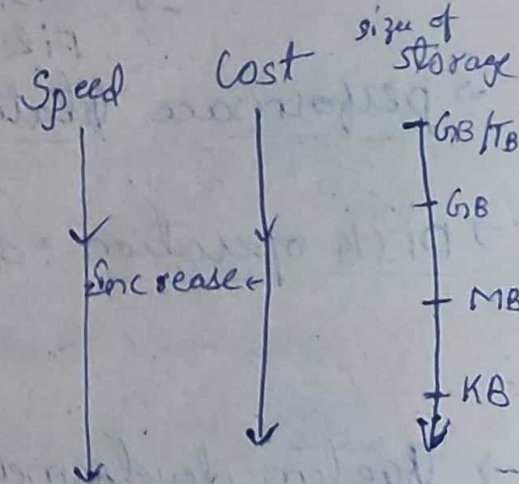
31/12/21

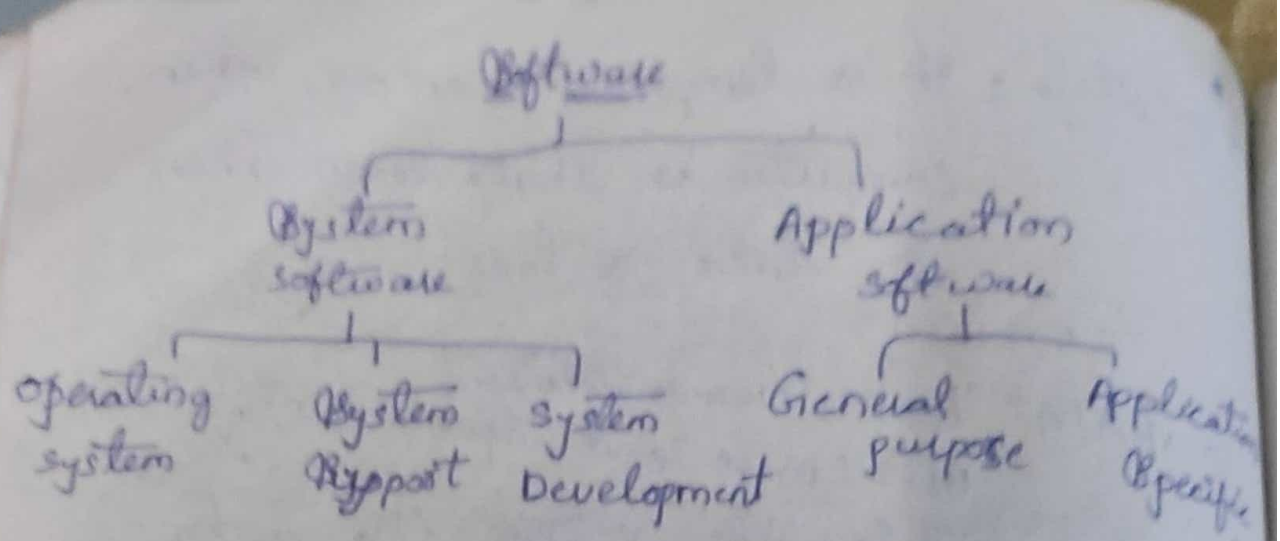
HDD - Secondary memory

RAM - primary memory

Cache - It is b/w RAM & processor

CPU Registers - Fastest memory type





→ Operating System: It manages all the hardware attached to it.

→ System Support: It gives the performance statistics of our computer.

Eg: Task manager, file wall, Disk operation, File system management.

→ Performace Statistic: memory, Cpu, Network

⇒ Disk operation: defragmentation, format, partition

→ System development: creating new softwares using existing softwares.

* Translator programs (compiler, Assembler)
* Debugging tools, testing tools.
* IDE (Integrated Development Environment)

* Debugging : It helps us to rectify errors in our program.

* Testing tool : Test the quality of Software.

* IDE : It reduces our load of debugging, testing etc; It performs all the functions performed by the debugging tools, testing tool.

Application software :

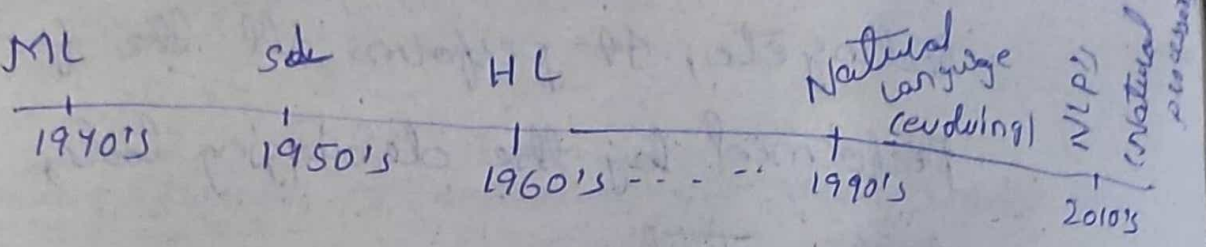
→ General purpose : There is no barrier of domain. This is used by everyone.

→ Application specific : It has some specific functions.

ex: Billing Scanner, Ledger System etc.

3/12/21 Computer Languages

1) Machine Language (1940's)
2) Symbolic Language (1950's)
3) High level Language (1960's)



2) Symbolic language (1950's) [Assembly language]:

→ Discovered by Grace Hopper and introduced 'Translator program' (or) (Assembler) which translate Symbolic language into "Machine language

⇒ Symbols (mnemonics):

+ - Add	Example:
- - SUB	READ, 2, R_1
* - MUL	READ 3, R_2
/ - DIV	ADD R_1, R_2
% - MOD	OUT ACC
READ $R_1, R_2 - - - R_6$	PCB - (programme control block) which programme is executed in processor.
ACC, PCB.	

→ Symbolic language fails because it includes hardware. Hardware is different from one compiler to other.

→ It depends on compiler.

→ They are not supporting portability (machine dependent)

---

3) High-Level Language:

```
#include <stdio.h>
int main()
{
    int a = 10;
    int b = 20;
    printf("sum= %d", a+b);
}
```

⇒ compiler: It converts the high-level Programming languages into machine Language. ex: C, C++, Java.

→ It scan whole programme in one scan.

⇒ Interpreter: It scan the programme line by line.

ex: python.

⇒ Natural Language processors: They convert our natural language into M.C.

example: Alexa, siri, ok google, cortana etc.

4/12/21

CREATING AND WRITING PROGRAMS:

1) Writing and editing the program.
2) Compiling the program.
3) Linking the program.
4) Executing the program.

⇒ The error b/w writing & compiling is called "Syntax errors".

Eg: ";"

⇒ The error which we have identify by ourselves is "logical errors".

Eg: '2+3' instead of '2*3'.

⇒ program Development Steps:

1) Designing Structure charts
2) Algorithms
   → pseudo code
   → flow chart
3) programs [coding]
4) Testing
5) Deployment
6) Maintainance.

⇒ Algorithms: It is an Step by Step Solution to Solve given computational problem.

Example 1: Write an algorithm for making tea;

Step: 1) Start, on the Stove.

Step 2: Boil water in a utensil on stove

Step 3: Add tea powder and sugar.

Step 4: Add milk.

Step 5: After 5 minutes, ~~Serve the tea~~ turn off the stove and serve the tea.

Step 6: End stop.

Pseudo Code

→ It is a detailed description about the algorithm.

→ An informal description about the algorithm.

① Making tea pseudo code.

⇒ Onn the Stove, keep the vessel on the Stove pour some water in it. Let it boil for a minute, then add tea powder, Sugar and milk then again let it boil for 5 mins, then off the Stove.

Flow Chart: It is diagramatic or pictorial representation of algorithm.

→ In order to draw the flowchart we have some standardise symbols.

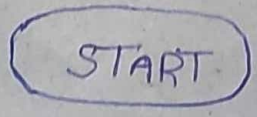
→ flowchart symbols are divided into two.

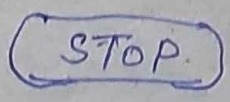
1) Auxiliary Symbols
2) primary Symbols.

1) Auxiliary Symbols: It is used for decoration purpose.

a) Terminals: We use oval shape.

→ It is used to indicate the starting point and stoppingpoint



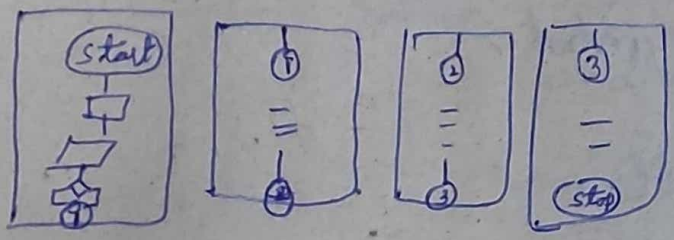


b) Flowlines: It is used to connect the symbols.

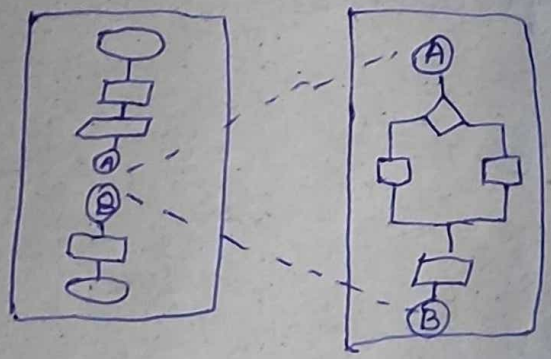
⇄ ↓↑

→ it shows the sequence of symbol.

c) Connectors: It is used to connect the different parts of flowchart.



→ When it gets difficult to fit some part of flowchart we use connectors



② primary Symbols: a) sequence Symbol.
b) Selection/condition/decisionmaking Symbol
c) Repitetive/Looping/Iterate Symbol.

a) Sequence Symbols: No skipping, No repitation

i) Null Symbol: No Symbol is used, only flowl are used.

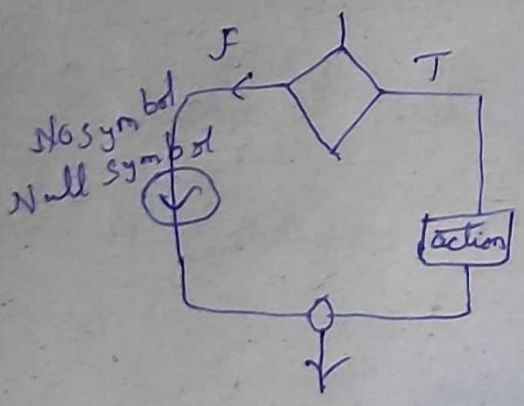
ii) Assignment Symbol:

iii) Input/output symbol

(5)
iv) Compound symbol.
(4)

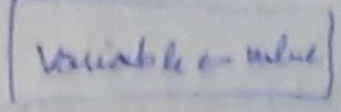
v) Module call Symbol.

i) Null Symbol: No symbol
only flowlines:



ii) Assignment Symbol: It is used to assign the value to the variable.

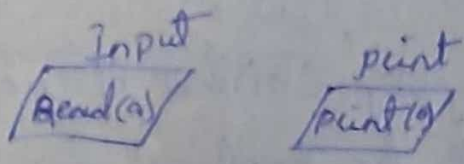
→ rectangular box is used.



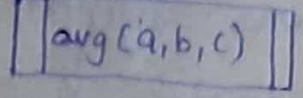
→ Arrow is directed towards variable from value.

iii) Input/output Symbol: It is used for reading and writing

→ parallelogram is used.



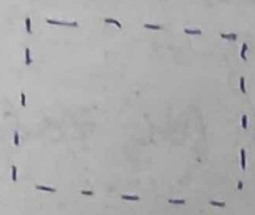
iv) module call Symbol: To show the function call (or) To sub-divide the given problem into ~~two~~ number of modules.



8/12

→ All sequence statements in the program will be considered by the program control exactly once. No skipping, repetation of sequence statement will be there. If we want the skipping of st^n or repeatation of st, we should use "control structures"

⇒ Compound Symbol: In order to merge or in order to group multiple statements as a single statement compound symbol is used.



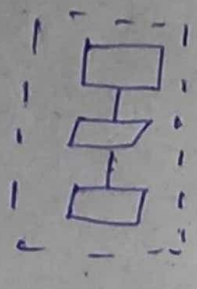
→ Dotted ▭ is used.

Ex: control structure:

⇒

→ If we apply control structure it will be applied to only first statement.

⇒ control structure with compound symbol.



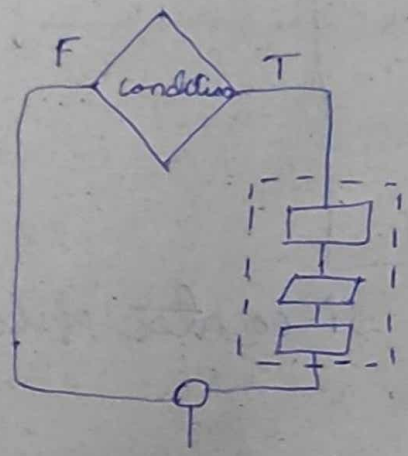
→ If we apply control structure with compound symbol. Now, it will be considered as single statement & control structure is applied to all st.

b) Conditional / Decision making / Selection Symbol:

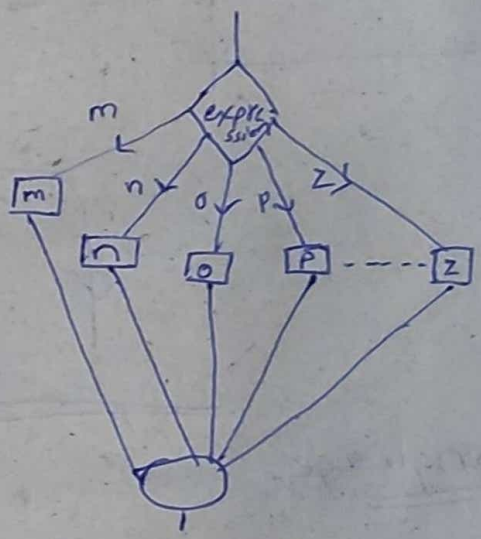
→ It allows "Skipping."

i) Two way Selection.
ii) Multi way Selection.

i)

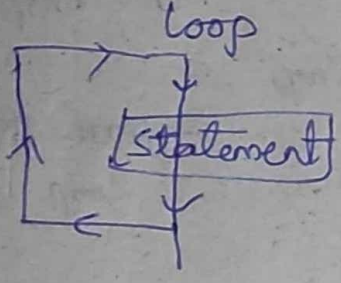


ii) Expression (a+b).



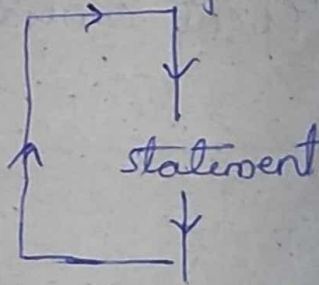
c) Repitative / Looping / Iterate Symbol:

Initialisation i=1, i<10



Infinite loop

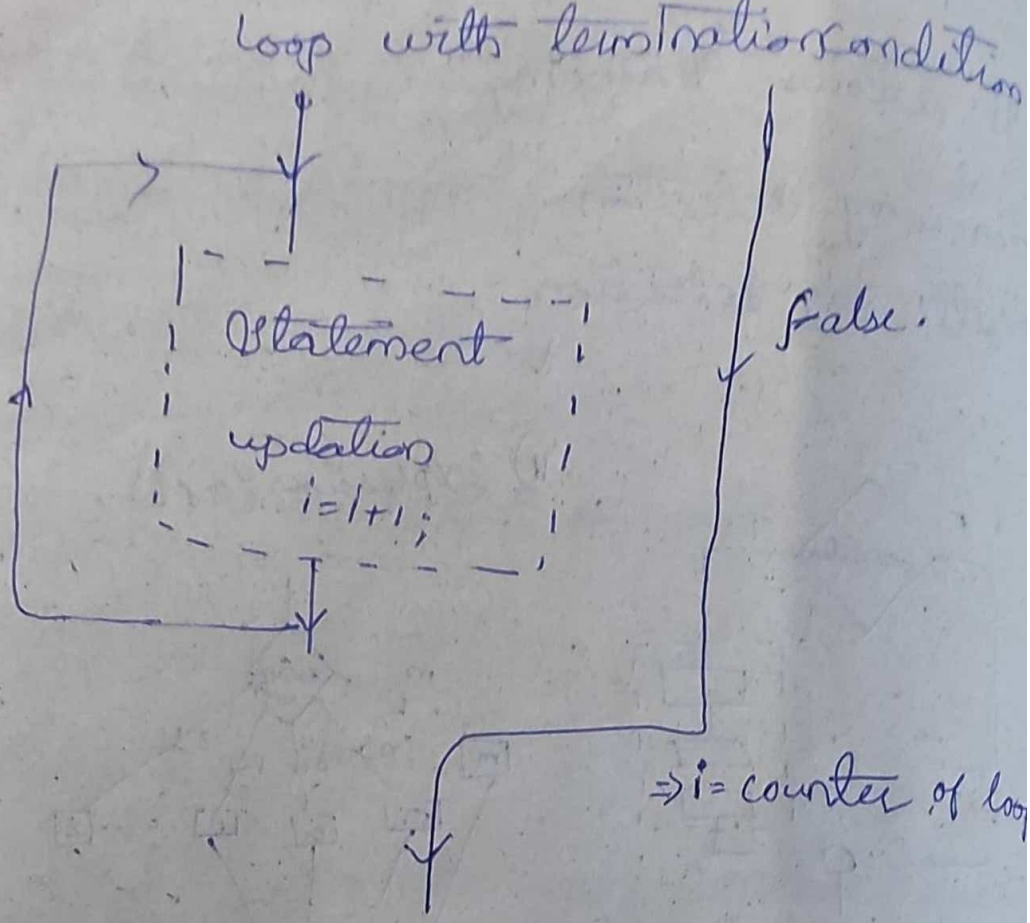
Loop with termination con



Infinite loop

n/12/21

i=1 i<10



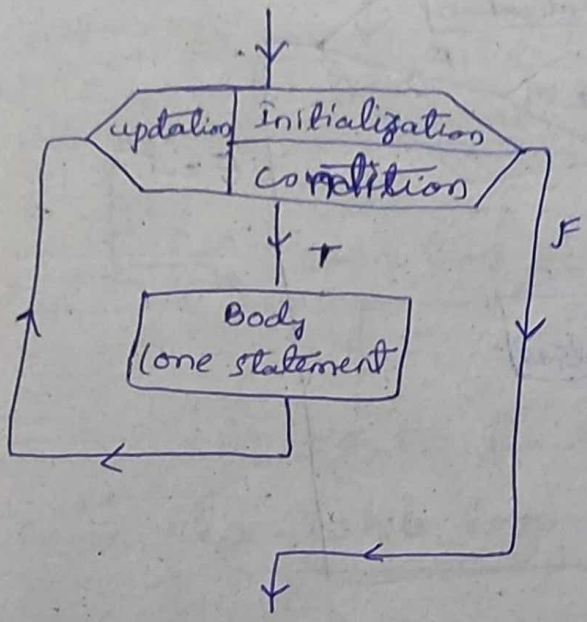
C-language:

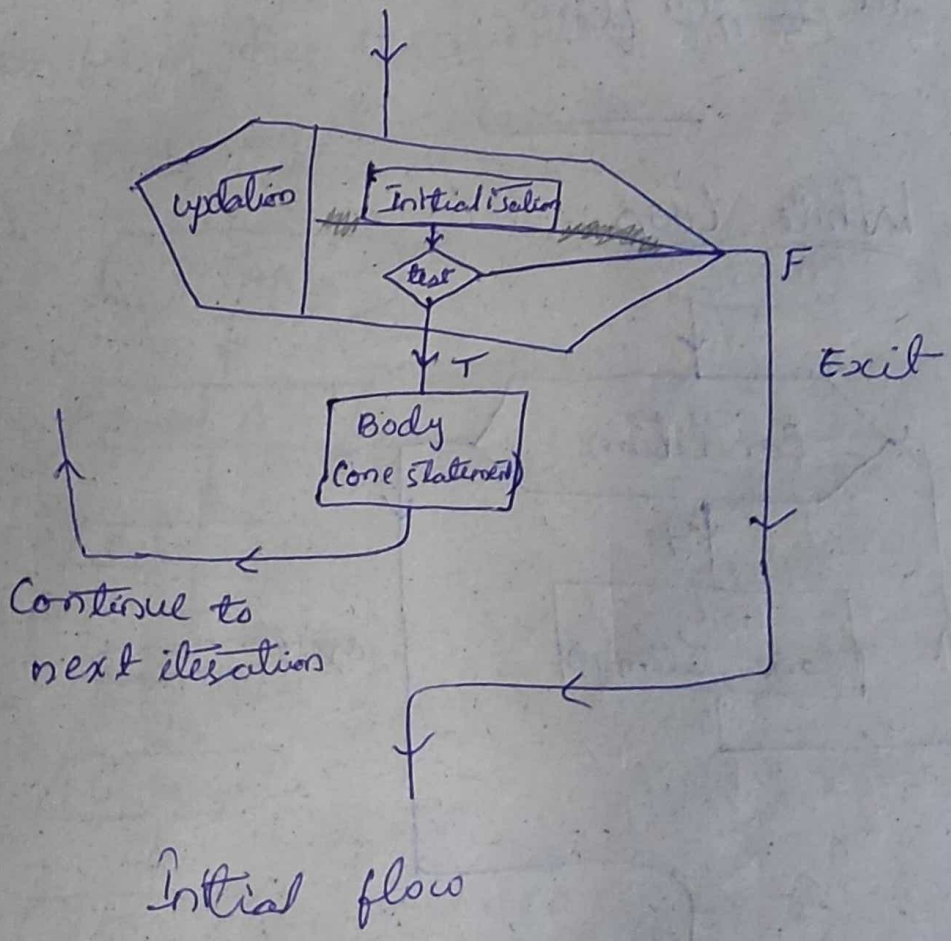
i) for loop - counter controlled loop } pre test loop

ii) while loop - } event controlled loop

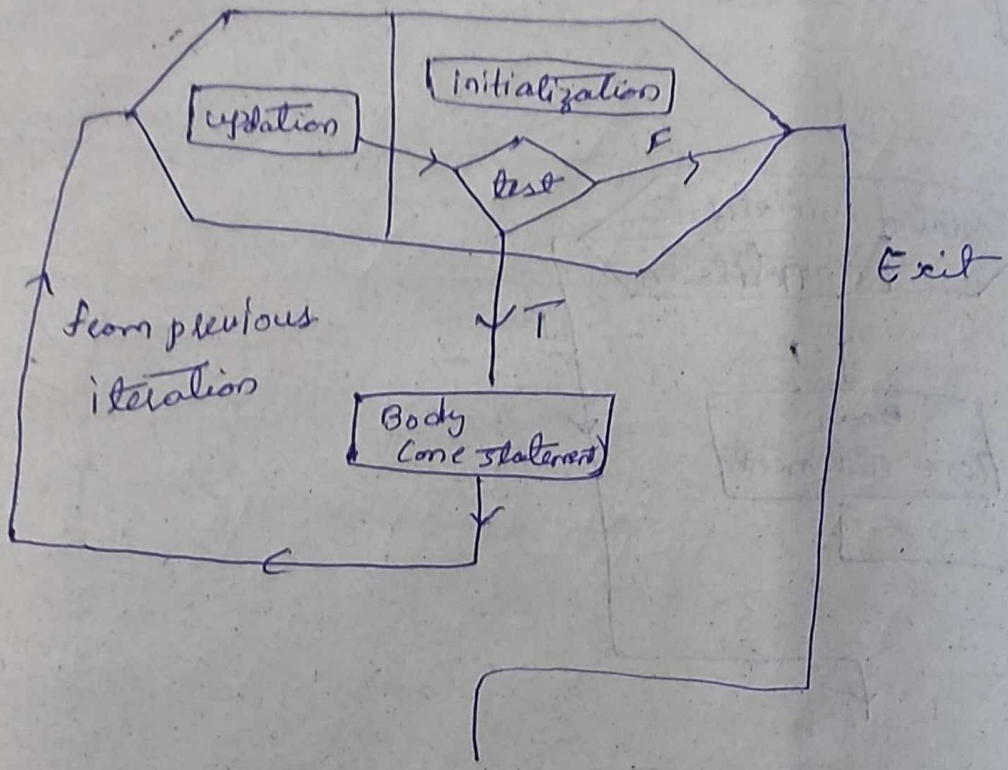
iii) do while loop - } event controlled loop → post test loop

→ For loop flowchart.



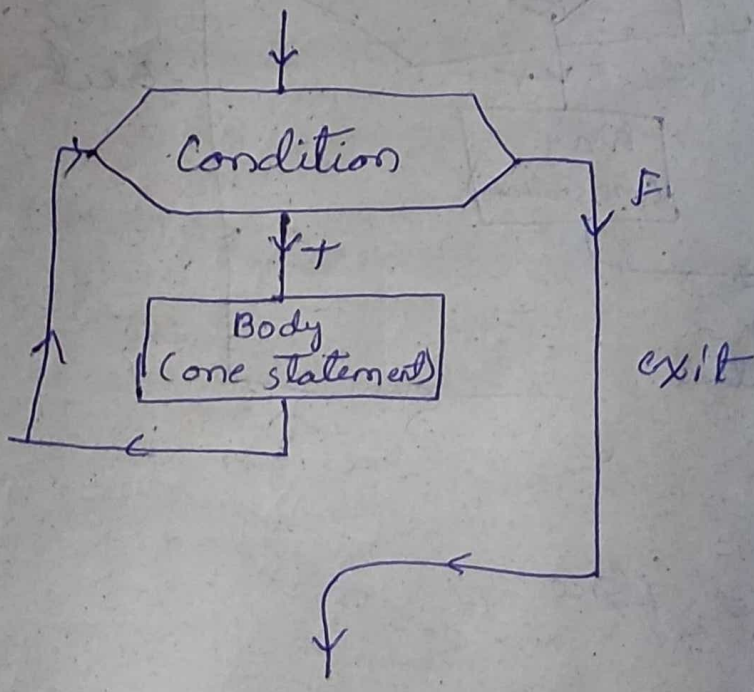


Initial flow

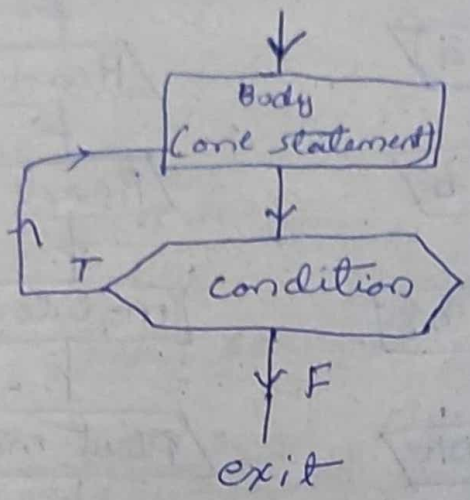


2nd ~~to~~ nth flow.

While Loop



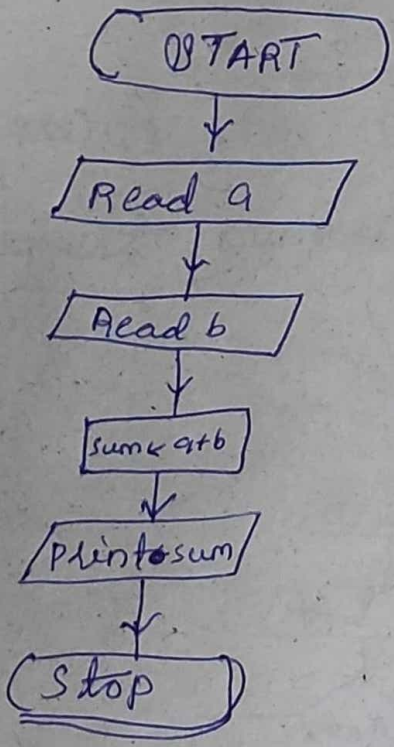
Do while

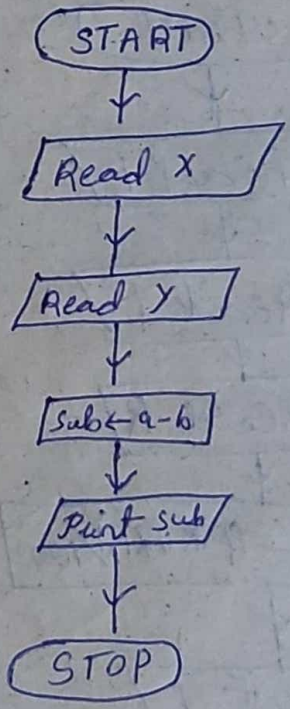


do - while loop

Ex:1: Draw a flowchart to add, sub, mut, div and to find the remainder of two numbers & display the result.

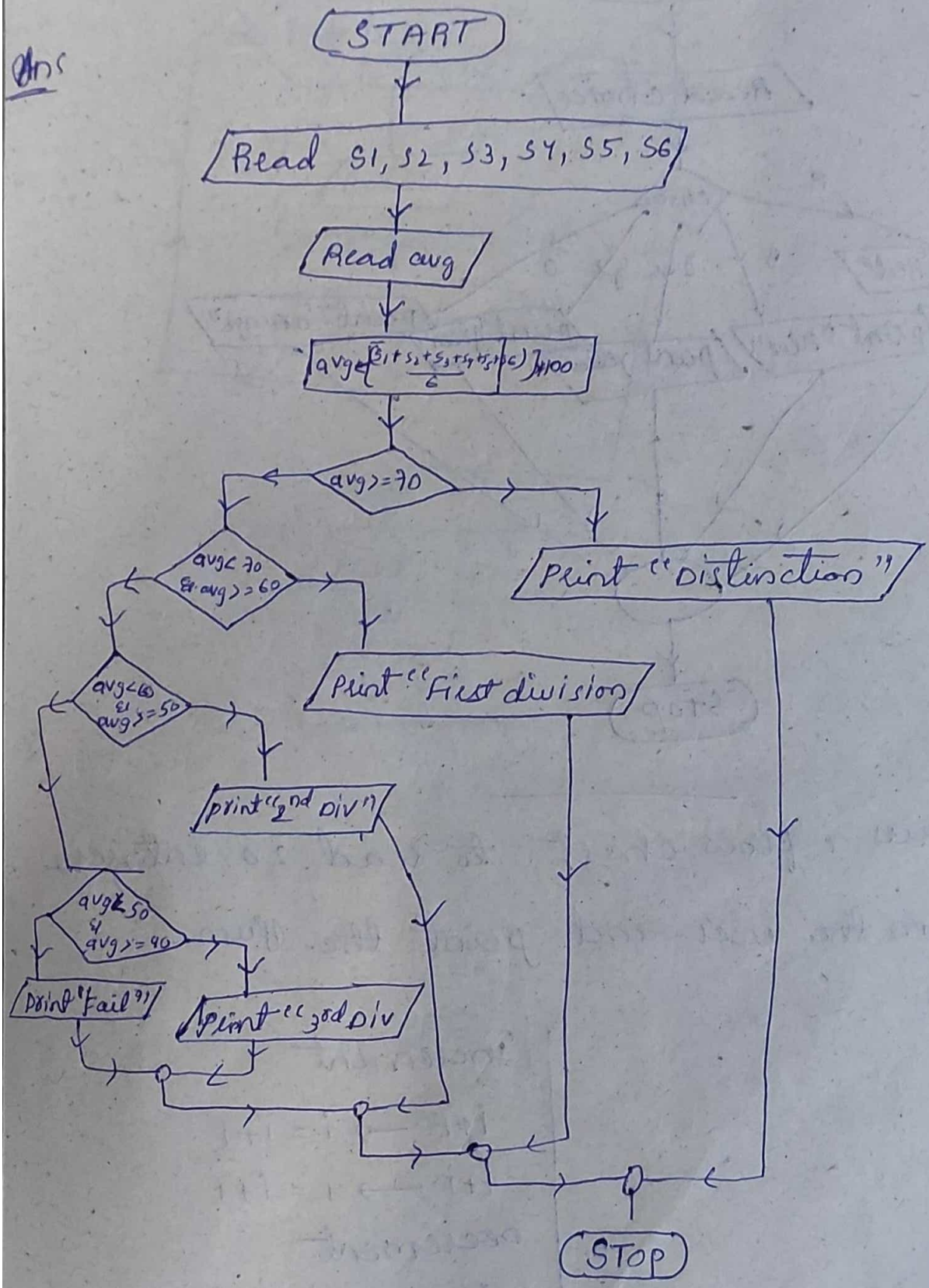
Ans





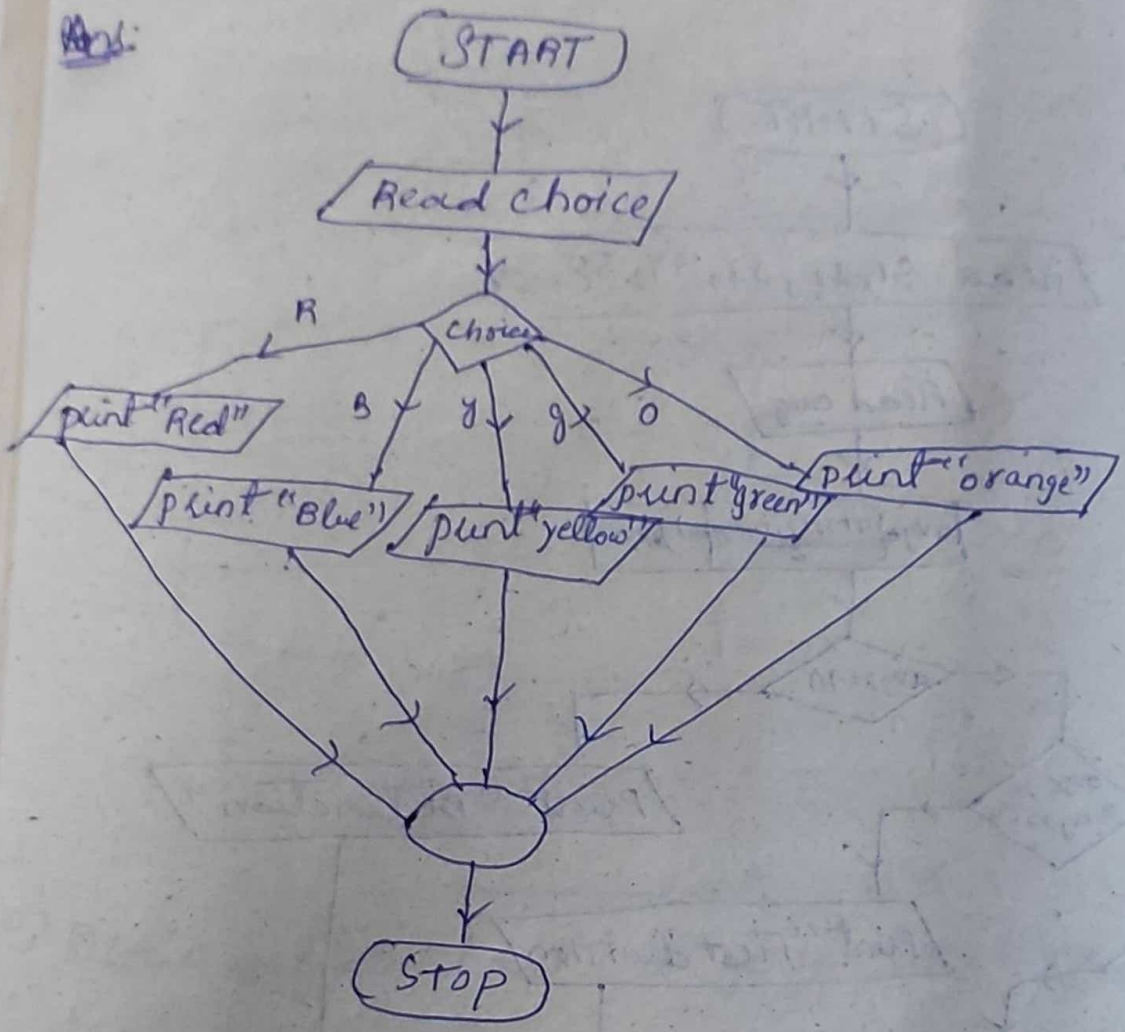
6) Draw a flowchart to print ~~the~~ a student performance by considering 6 subject marks.

Ans



Q) Draw a flowchart to print rainbow colors as per user input.

Ans:



Q) Draw a flow chart to read 20 integers from the user and print the sum.

Increment

i++ → i = i+1

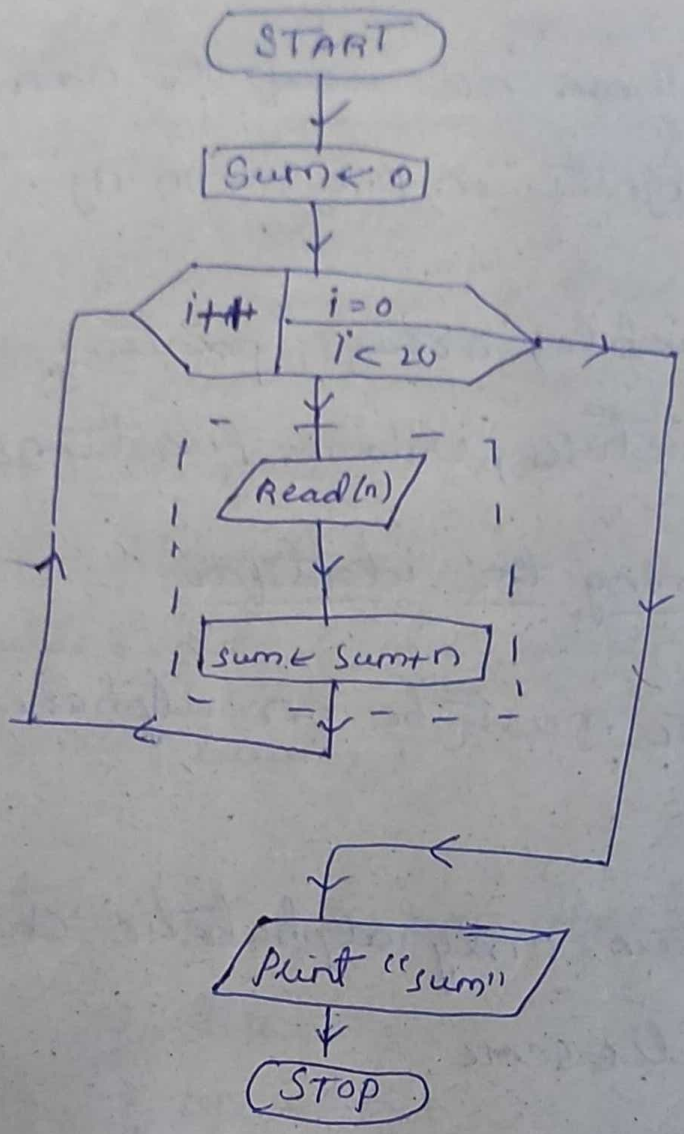
++i → i = i+1

Decrement

i-- → i = i-1

--i → i = i-1

Ans



13/12/21

→ Identifiers: These are used to name the programming objects in programming.

* They are: variable, arrays, pointers, structures, unions, functions.

→ Rules for framing the identifier.

1) First character must be an alphabet (or) underscore.
2) It must contain only alphabetic characters digits or underscore.
3) First 63 characters are significant.
4) It cannot be a keyword.

Ex: room temperature ×
9abc ×
abc@123 ×

_abc ✓
room_temperature ✓

⇒ Key words: Keywords are reserved words in our programming they will have Special meaning when used.

→ 37 - keywords are there.

Ex: int = integer

char = character

float = decimal

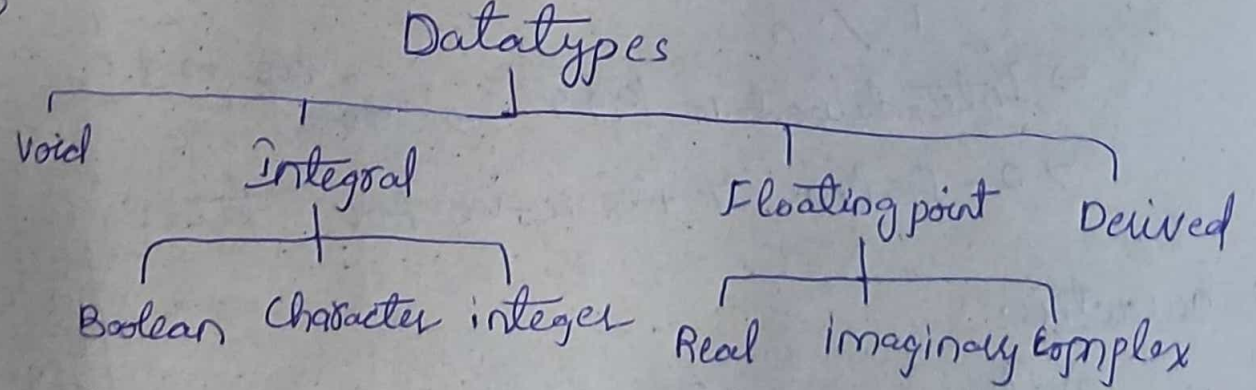
double, void, if, else, while, bool, exit, continue, break, return etc.

⇒ Datatypes: It Specifies the type of value (variable)

→ Size

→ operations.

→



⇒ primitive / Basic type: Independent (void, Integral, Floating points)

⇒ Derived: Dependent on Primitive datatypes.

Ex: Arrays, pointers, structures, Unions, file etc.

→ void - No values & No operations.

Ex. → int main(int)
← return value
↳ parameter (input)

→ Void main(void) - No input, no output

→ No ammount of memory allocated to void data types.

→ Integral:

→ Boolean: To declare the variable boolean, bool keyword is used.

Ex: True / false
1 / 0

→ Introduced in 1999 → C99. (→ version of C)

→ 0 - false & +ve & -ve for - true.

→ Character:

a) char → 1 byte → 2^8 bits → 256 bits

Ex: A-Z, a-z, 0-9, white space, tab, @, &, # etc.

b) wchar_t → 2 byte → 2^{16} bits → 65,536 bits

white character set

⇒ Size of ( ) : It is a special type which is useal to return the allocated memory size in no. of bytes.

→ Integers : Number without any fraction part.

→ 4 different types of integers.

ex: short, int, long, long long

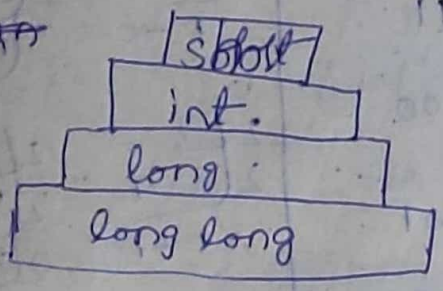
↓ ↓ ↓ ↓

short int int long int long long int

⇒ Size of (short) < Size of (int) < size of (long) < size of (long long)

2 bytes 4 bytes 4 bytes 8 bytes

→ ~~bits~~



⇒ To calculate range for short

-ve ---- 0 ---- +ve

n bits

2^{n-1} ---- 0 ---- $2^{n-1}-1$

-ve +ve

2^{16-1} ---- 0 ---- $2^{16-1}-1$

-32,768 ---- 0 ---- 32,767

int

2^{32-1} ---------- 0 ---------- $2^{32-1}-1$

2^{31} ---------- 0 ---------- $2^{31}-1$

-2147483648 ---- 0 ---- 2147483648

Decimal	Binary
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

2|0
0

2|1 (0.5)
0

2|2
1 -0

2|3
1 -1

2|4
2|2 -0
1 -0

2|5
2|2 -1
1 -0

2|6
2|3 -0
1 -1

2|7
2|3 -1
1 -1

2|8
2|4 -0
2|2 -0
1 -0

2|9
2|4 -1
2|2 -0
1 -0

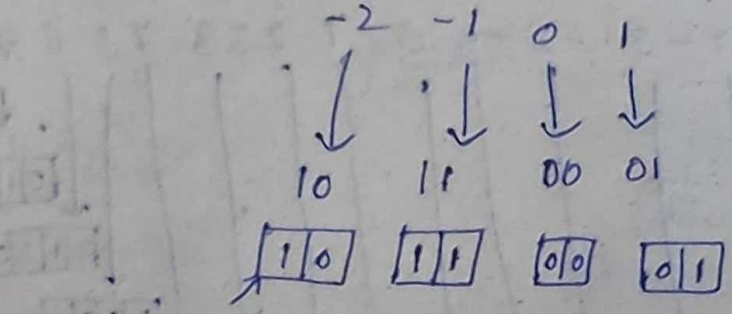
2|10
2|5 -0
2|2 -1
1 -0

2|11
2|5 -1
2|2 -1
1 -0

2|12
2|6 -0
2|3 -0
1 -1

n = 2 bits

$2^{n-1} \; -- \; 0 \; -- \; 2^{n-1}-1 \quad (n>1)$

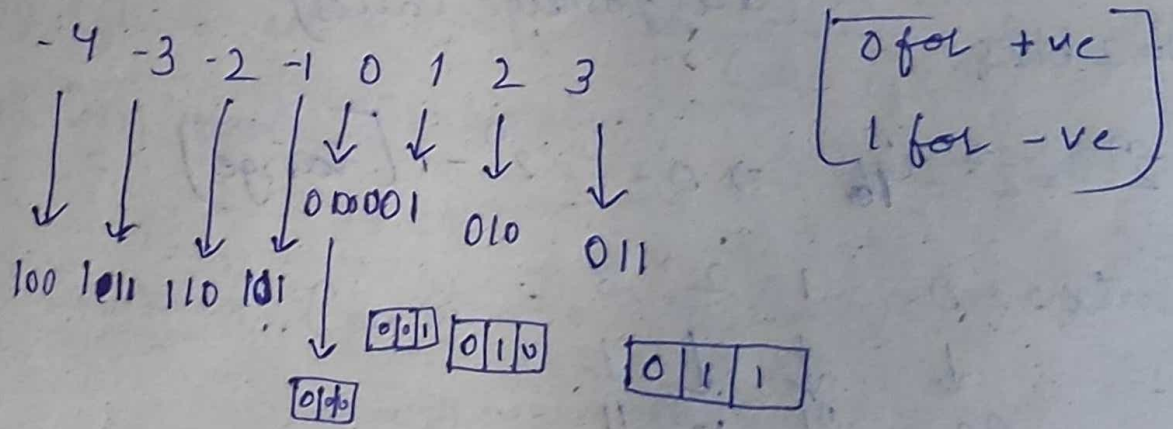


lms - left most significant cell used to indi

the sig

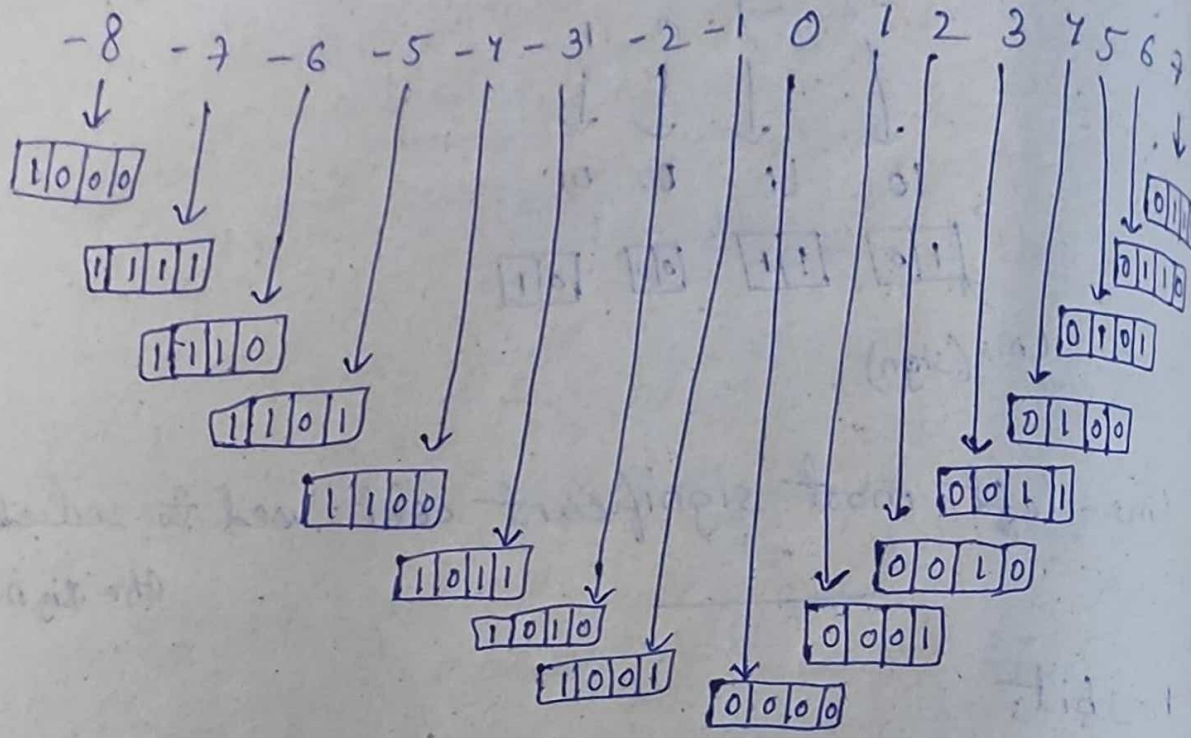
n = 3 bits

$2^{3-1} \; -- \; 0 \; -- \; 2^{3-1}-1 \quad (n>1$



n = 4 bits

2^{4-1} - - - - 0 - - - - $2^{4-1} - 1$



Unsigned int : [no negative values]
[all positive values]

n = 2 bits

$2^n = 2^2 = 4$ ⇒ 0 - - - $2^n - 1$ [range]

Ex: 0, 1 2 3

00 01 10 11

---

n = 3 bits

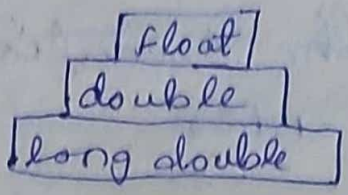
0 1 2 3 4 5 6 7

000 001 010 011 100 101 110 111

⇒ Floating-point types:

⇒ Real number: It is number with fraction part.

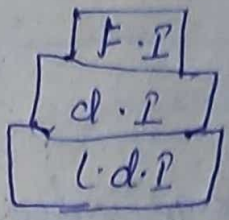
i) Float
ii) double
iii) long double



size of (float) < size of (double) < size of (long double)

⇒ imaginary: $\sqrt{-1} = i$ ⇒ The number which have $i^2 = -1$ -ve with square root.

i) float imaginary
ii) double imaginary
iii) long double imaginary

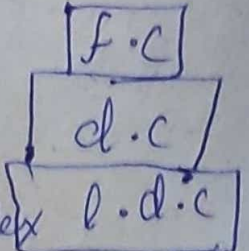


size of (f·i) < size of (d·i) < size of (l...

⇒ Complex: Combination of real number and imaginary number.

ex: a+bi

i) float complex
ii) double complex
iii) long double complex



size of (f·c) < size of (d·c) < size of (l·d·c)

⇒ Variable: A variable is named memory location.

→ A variable is a storage area which holds the value during the execution of a program.

Ex: a = 1, b = 2

start - alphabets
cannot - digits
start - underscore

→ i) Give a name (identifiers)
- variable declaration

ii) Specifying type (data types)
- variable definition

iii) Assigning value (constants)
- variable initialisation

int temp
↳ Garbage value.

⇒ Garbage value: Left over value from the previous use program.

→ It is not useful.

⇒ Constants: It is a fixed value and we cannot change its value.

→ Constants are data values that cannot be changed during programme execution.

⇒ Boolean, character, integer, real, complex & string

⇒ These are called literal

⇒ Boolean constants:

T/F

1 0

→ To use boolean constants, a header file <stdbool.h> must be used.

⇒ character constants: All alphabets, digits & specials characters (All the printable symbols on the keyboard).

→ To use a we should write 'a'.

Eg: 'a', 'b', 'c' --- 'z'

'A', 'B', 'C' --- 'Z'

'[', ']', '@' ----

'0', '9', '2' ..

⇒ non printable characters : (escape characters)

→ '\n' - new line → '\\' - to print
→ '\t' - Horizontal tab → '\'' - To print single
→ '\v' - vertical tab → '\"' - To print double

18/10/21

⇒ integer constants :

@host - Supplementary/primitive.
int (default) - 15 U → unsigned.
long (L) - 15 L U
long long (LL) - 15 LL U

Signed (default)
unsigned (U)

⇒ real number constants :

float (F) - 3.14 → F
double (default) - 3.147
long double (L) - 3147 (L)

⇒ imaginary & complex constants:

float complex (.F) - 4.2345+3.147fi

double complex (default) - 1.234+3.417i

long double complex (L) - 1.234L+3.417i

⇒ String:

"Hyderabad", "Irfan", "_"

"Hello world \n"

"Hellow \t world"

Syntax Error

;

{ (f)

C)

(") ,,

Compiler detect these errors.

- compile time errors

Logical Error / runtime

programmer should correct these errors.

Ex. int a, b;

printf("add = %d", a

=> In this we have written '*' in place