BCE ~ Dr. pavan thakur 45k gandhi

Page No
Date

Definition A computer is an electronic device that can accept, store and process information to produce a required result.
[info → meaningful data]

Functions- A comp can be defined in terms of its functions:
<1> Accept data <2> Store data <3> Process data <4> Retrieve data as & when required
<5> Print data in desired format.

Features-
- Speed
- Accuracy
- Storage
- Diligence (लगातार)
- Versatility
- Power of remembering
- Parallelism (multitasking)
- Improve / upgrade

Limitation-
- Does not have own Iq
- Does not have feelings
- No self debugging
- Cannot work without intervention
- Garbage in, Garbage out (faulty result if faulty program input)

Generations → I^{st} gen → 1942-1955
Main innovation → Vaccuum tube → made possible formation of future comp.
Main memory → Punched cards
Input / output devices → Punched cards & papers
Language → Low level machine language
OS → No O.S., the human operators were set to toggle switches.
Size → Main frame (wall size)
Cons → costly, gets heated very much

→ IInd Gen → 1955-64
Main Inno - Transistors
memory → RAM & ROM, external storage, magnetic tapes, disks.
Lang → Assembly language (low-level), BASIC etc (High level)

OS → Human-handled punch cards
Size → Smaller than Ist - table size.

→ IIIrd Gen → 1964-75
Main Inno → (IC) - integrated circuits, large scale integration (LSI)
memory → PROM (Programmable) & DRAM (Dynamic), floppy disk
I/o devices → keyboards & monitors
OS → Complete OS was introduced (MS)
size → Mini (godium)

→ IVth Gen → 1975-1990
Main Inno → (VLSI) very large scale integration.
Memory → EPROM (erassable programmable) & SRAM (static), floppy disk, hard disk
I/o devices → keyboard & monitor
Language → High level languages eg → application softwares (MS office)
size → Micro computer
OS → MS-DOS

→ Vth Gen → 1990 - today
Main Inno → (ULSI) ultra large scale integration.
Memory → EEPROM (electrically erassable programmable), SIM, DIMM, Modified magnetic optical disk & pendrive/flash drive
I/o devices → scanners, printers etc

Language → AI expert system
OS → GUI based os like windows XP
size → laptops, notebooks, (Tablet) palmtops, pocket pc (mobiles)

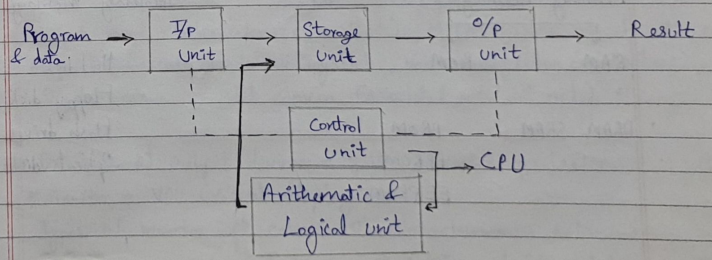
Classification of Computers:

⇒ Based on Application:
i) Analog
ii) Digital
iii) Hybrid eg → Modem

⇒ Based on Purpose:
i) General purpose ⟨ii⟩ Special purpose

⇒ Based on Size and Capability:
i) Micro ⟨ii⟩ Mini ⟨iii⟩ Mainframe ⟨iv⟩ Super comp.
eg → PARAM-10000

Organization of Computers:



Registers

Memory Address Register	MAR	→ Pointer → has address
Memory Buffer Register	MBR	temp storage
Program Counter	PC	record of executed instructions
Instruction Register	IR	

Buses

Data bus
Address bus
Control bus

Instruction set

A combination of operational codes and a way of specifying an operant by its location or address in memory

Memory

A computer memory is used to store data and instructions

Memory types

- Primary Memory
  - RAM
    - DRAM
    - SRAM
  - ROM
    - PROM
    - EPROM
    - EEPROM
- Secondary memory
  - Hard disk
  - Floppy disk
  - Flash drives
  - Optical disks

Primary memory - It is a part of computer where a data or program resides for execution.

* ROM - ROM is a primary memory that stores necessary info. to operate system.
  eg → Programmes to boot the computer.

Secondary memory - eg → Hard disk, Floppy disk, flash drives, Optical disks etc

Input / output devices → * Input devices bring info into the computer
* Output devices bring info. out of the computer.

Functions of input devices :-

- To accept the data from outside world
- To convert it into ASCII code or Binary code.
- Send the code to CPU

* Examples of input devices - keyboard, mouse (GUI based device), light pen, OCR (Optical character reader → scanner), OMR (Optical mark reader → exam sheets), MICR (Magnetic ink character reader cheques), Barcode reader, etc. Joystick, Voice recognition (microphone), Digital camera, Web camera (associated with internet) etc.

* Examples of output devices - Speaker, printer, projector,
* Monitor → VDU (Visual display Unit)
  Types of monitor → colour → Monochrome, Coloured
  Technology → CRT, LCD, LED

features of monitor - Resolution, Response time, contrast, refresh rate, Aspect ratio, Dot pitch (gap between pixels)

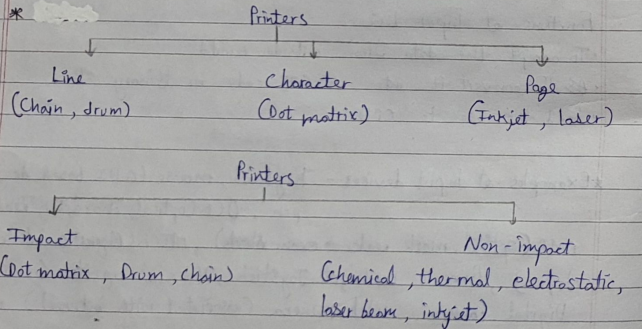
Monitor Display modes :- Resolution & colours

i) CGA (Colour graphics Adapter) in 1981
   * 4 colours * 320 x 200 (pixels)

ii) EGA (Enhanced graphic Adapter) in 1984
   * 16 colours * 640 x 350

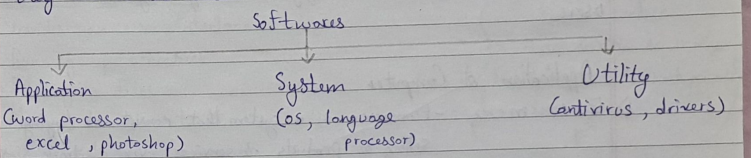
iii) VGA (Video graphic array) in 1987
   * 16 colours * 640x480 or *256 colours * 320x200

iv) SVGA (Super video graphic array) in 1990
   * True colours (16,536) * 1024x768 in million colours.



Softwares

It refers to a set of computer programs that describe the program how they are to be used.



System software → It is a collection of programs which are designed to operate control and extend the processing capability of computer.

* Operating system - It is a system software which is used for operating each and every task of a computer.
  eg like memory management, user instructions, hardware components, file and application softwares.
  eg → Unix, linux, windows, mac etc.

* Language processor - It is used to translate the instruction of any programming language to machine language.
  Three types:
  → Compiler - Reads whole code at a time, occupies less memory, is fast, used nowadays.
  → Interpreter - Processes one statement at a time, requires more space in memory.
  → Assembler - Converts high level language to low level language

Computer ethics

These are defined as application of classical ethical principles to the use of computer technology. These give guidelines to computer users.

Application of Computer

1) E-commerce - Ecommerce is a system that consists of buying and selling of products or services over electronic systems such as the internet and other computer networks.

2) Bioinformatics - The science of bioinformatics actually develops algorithm & biological software to analyze and record the data related to biology such as genes, drug ingredients etc.

3) Remote sensing - It is a computer software application that processes remote sensing data

4) Health care - Comp. in healthcare include computerized physician order entry system for tasks such as ordering tests and medication, advanced monitoring devices, robotic surgery system

5) GIIS - As a comp hardware & software system, it is designed to

6) Gaming <7> Multimedia <8> Animation

Operating System

User Interface Application

↑ ↓

[OS] { O.S. Interface; Scheduler & resource manager; File systems & Network protocols; Device drivers }

↑ ↓

Hardware

Functions of an O.S :

i) Provides an user-interface
ii) Running programmes.
iii) Managing files
iv) Memory management
v) I/o system management
vi) Scheduling jobs (set priority of jobs)
vii) Task management
viii) Controlling network
ix) Security

Types of O.S :

i) Batch processing - O.S that operates whole batch of programs at a time. (OS)
ii) Multi-user & single user O.S (CMS Dos & mobiles → Single user)
iii) Multitasking & single tasking O.S. (Old cell phones) (single tasking)
iv) Real time O.S.
v) Distributed O.S. (Super computers)
vi) Embedded O.S. (P.D.A → personal digital assistant)

File management

The task of managing and organising files is called file management.

File Attributes:

i) Read only
ii) Hidden
iii) System
iv) Volume label
v) Directory
vi) Archive

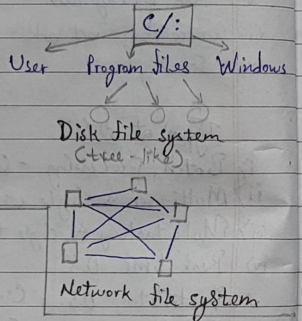
File operations

i> Copy
ii> move
iii> Delete
iv> Rename
v> Merge
vi> Set attributes
vii> Truncate

File System

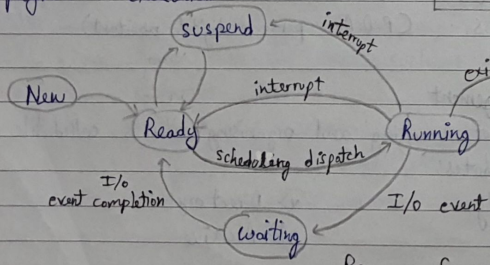
It is a structure in comp. which is used to store and organize data.

Types of file system:
i> Disk files system
ii> Network file system
iii> Special purpose file system
(Database file system, transactional FS.)



Process

A process can be defined as a program in execution



PROCESS STATE

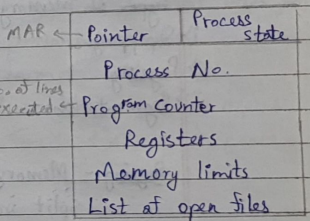
Process states:

i> New
ii> Ready
iii> Running
iv> Blocked
v> Suspended
vi> waiting
vii> Terminated

Process control blocks

The O.S. groups all information that it needs about a particular process into a data structure called a process control block.

i> Process state
ii> CPU registers
iii> Memory management info.
iv> Program counter
v> Accounting info.
vi> I/O status info.
vii> CPU scheduling info.



Process Scheduling

i> Long-term scheduler → Job-queue
ii> Medium-term scheduler → ready-queue
iii> short-term scheduler → Device-queue

time-sharing

Scheduling Algorithm

i> (FCFS) - First come first serve - inefficient
ii> Shortest Job first scheduling Algorithm (SJFS) - less inefficient
iii> Round Robin Scheduling (RRS) - time slices allotted to jobs in job queue

stack

most efficient

Page No.
Date

Memory Management

It is concerned with allocation of physical memory of finite capacity to requesting process.

Requirements of memory management:

* Process may require more memory than available.
* Load processes to different locations each time they execute
* Need to swap all or parts of a process to secondary memory during execution.
* Memory protection between the process.

Types/Ways of memory management schemes:

i) Swapping → main mem to sec mem & back
ii) Paging
iii) Segmentation

Paging - Memory management technique in which a process is split into a no. of fixed equal sized partitions and allocated non contiguous memory spaces.

↳ Ans. → continuous with a common diff.

Segmentation - Segmentation is a logical unit such as the main program procedure, functions, methods, objects, local variable, global variables, symbol table, arrays.

NLP → Natural language processing
↳ Auto completion/correction

Microsoft Office

M.S. Word

It is a word processing software program which is used to type letters, reports and other documents.

Word processing refers to typing, editing, formatting, storage and printing of any kind of document.

<write features in exam>

M.S. Excel

It is one of the most popular spreadsheet applications that helps you to manage data, create charts and thought-provoking graphs.

Excel is supported by both mac and pc platform.

It can also be used to balance a cheque book, create an expense report build formulas and edit them. <write features in exam>

Introduction to Algorithm And Programming Languages

Algorithm - An algorithm is a well defined list of steps for solving a particular problem.

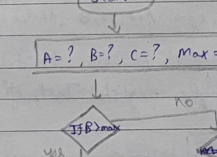
eg → step 1> start
step 2> store 5 into A & 10 into B
step 3> calculate Sum = A+B
step 4> Print value of Sum
step 5> Stop

time complexity (O^n)
Space Complexity

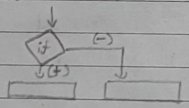
Flowchart - Flowchart is the pictorial representation of an Algorithm.

- start / stop
- Step
- Decision / Conditional
- Action
- Flow line

(For the above eg)



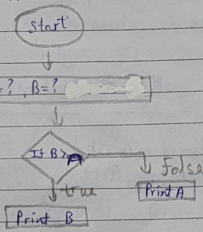
For conditional



Symbols -

- Connector
- or symbol
- multiple document
- Database

Q> ⇒ Draw the flowchart to print greater number among the two input number.



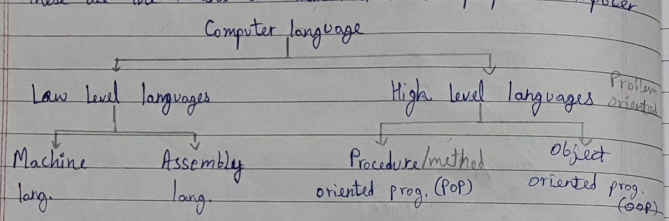
⇒ Print greater of three number

Advantages of Flowchart

1) It is easier for Programmer to explain the logic.

Computer / Programming languages

These are the means of communication b/w people & computer



Low level language → Machine oriented & require extensive knowledge of computer hardware & its configuration.
eg → machine lang. (01100) 1st gen, assembly lang. 2nd gen (Code words, eg → Add, 12, B ! device specific)

High level languages → Instructions are written in form of simple english. Instructions are converted machine language by compiler.
These are problem oriented.
eg → (COBOL) Common Business Oriented Language,
(FORTRN) Formula Translation.
(BASIC) Beginners All Purpose Symbolic Instruction Code.
Adv → machine independent, easy to run & learn, errors can be easily removed, easy to maintain
Dis Adv → Inefficient code, slower

Prolog → FPGA
Prolog → CPP designing
Field programmable gate

Types of high level language:

1) Algebraic formula type processing (Maths)
eg → BASIC, FORTRAN, Programming Lang Version-1 (PL-1), Algorithic Lang. (ALGOL), A Prog Lang. (APL), Matrix laboratory (MATLAB)

2) Business Data Processing (Business)
eg → COBOL, Report Program Generator (RPG) (Power BI)

3) String and List Processing
eg → List Processing (LISP), Program in Logic (ProLog)

4) Object oriented Programming language
eg → C++, java, .Net, Python

5) Procedure oriented programming.
eg → C, Pascal.

6) Visual Programming Language.
eg → Visual Basic, Visual Java, Visual C++

Generation of Programming Language

1) 1GL → I^st Generation Lang Programming Language.
eg → machine lang (01101010)

2) 2GL → II^nd Gen Lang.
eg → Assemble Lang (Add 12,5)
Code words are defined)

3> 3GL - 3rd Gen Lang.

eg → C, C++, Python, Java - - -

```
Ceg→ # include <stdio.h>
     int main () {
         puts ("Hello World");
         return 0; }
```

4> 4GL - 4th Gen lang. - Designed to be closer to natural lang.

eg → Extract all customers where "previous purchases" total more than 1000.

5> 5GL - 5th Gen lang. - It allowed you to easily envision OOP, class hierarchy and drag icons to assemble program components.

Procedure Oriented Programming (POP)

Vs Object Oriented Programming (OOP)

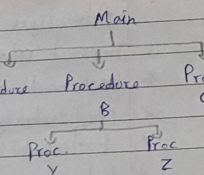
POP → POP consists of writing a list of instructions for the computer to follow & organize these instructions into groups known as functions

Features:

* Emphasis is on algorithm.
* Large programs are divided into smaller programmes called functions.
* Most of the functions share global data.
* Data moves around anywhere from function to functions.
* Function transforms data from one form to another.
* It employs top-down approach for program designing

Advantages:

* Easy to use and understand
* Easy to learn
* Error can be easy to remove
* Simple programming



POP

Disadvantages:

* It is time consuming
* No security for data.
* No structure or code reusability.
* Slow performance
* Difficult maintainance
* No proper method for exception handling

OOPs → In OOPs languages, all applications are built around class and object.

Features:

* Emphasis is on data but not procedure.
* Programs are divided into objects.
* Data structures are designed such they characterize the object.
* Functions that operate on the data of an object are tied together in the data structure
* Data is hidden and cannot be accessed by external function
* New data & functions can be easily added whenever necessary.
* It follows bottom-up approach in program designing.

Advantages:
* Improved reliability & flexibility
* Real world modelling
* Modifiability
* Extensiability
* Reduced maintainance
* High code reuseability

Disadvantages:
* Object oriented development is not a solution.
* It is not a technology
* It is not yet completely accepted by major vendors.

OBJECT Oriented Programming

① Objects - Objects are the basic run-time entities in object oriented system with some characteristics and behavior.
<with examples>

Class - A class is a collection of objects of similar type that share common properties and relationship.

② Data abstraction - Abstraction refers to the act of representing essential feature without including the background details and explanations.

③ Encapsulation - Wrapping up of data and methods in a class is known as encapsulation.

④ Inheritance - The mechanism of deriving a new class from a base class is called inheritance

⑤ Polymorphism - It is the concept that supports the capability of an object of a class to behave differently in response to a message action.

C++ Programming Language

C++ is an object oriented programming language used in the development enterprise and commercial application.

→ Features of C++:

Features:
* Object oriented
* Classes and object
* Data abstraction
* Inheritance
* Polymorphism
* Portability
* Versitile cross platform
* Data encapsulation

character set:
* Alphabets (A,a,B,b,---)
* Digits (1,2,3,---)
* Special Symbols ($,#,&,---)
* White Space Characters (Space, tab, \n ---)

token - The smallest individual unit of C++ program is known as a token or a lexical unit.
* Keywords
* Identifiers
*

Keywords - These are the reserved words that have special meaning to the language compiler and they cannot be used as identifiers in a programme.

C → 32
C++ → 52

eg → break, const, while, do, for, new, try, switch, int, char, float, if, else, double, string etc

Identifiers - These are the user defined words.

eg → first_var , Var_1 , _var

Constant - These are data items that never change during the execution of program.

eg → const float pi = 3.14

Seperator - Symbols that seperate each component of a program.

eg → : , ( ) , { } , [ ] , , , ; , . , = etc

Postfix and Prefix Operator

x = 1

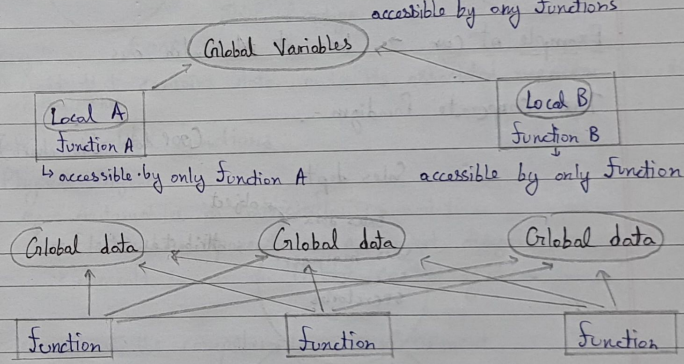
y = x++ (y = 1) (x = 2)

z = ++x (z = 3) (x = 3)

[Post left to right → first assign
x++ ⇒ x = x+1
x-- ⇒ x = x-1

[Pre right to left → first increment
++x , --x

Operator types	Operator	Association
→ Multiplicative	*, /, %	Left to Right
→ Additive	+, -	Left to right
→ Biterwise		
AND	&	Left to right
XOR	^	
OR	|	
→ Conditional		
AND	&&	Left to right
OR	||	Left to right
Conditional	?:	Right to Left
→ Assignment	=, +=, -=, /=, *=, %=, <<=, >>=, &=, ^=, |=	Right to Left



P.T.O

The Procedural Paradigm

→ Real world modelling :

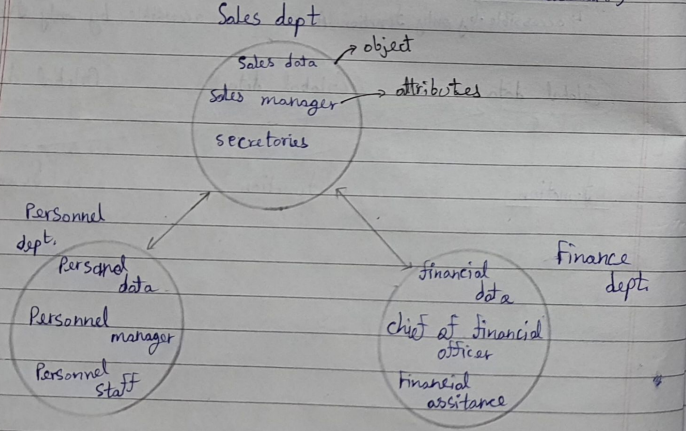
Attributes { colour, dimension, engine life

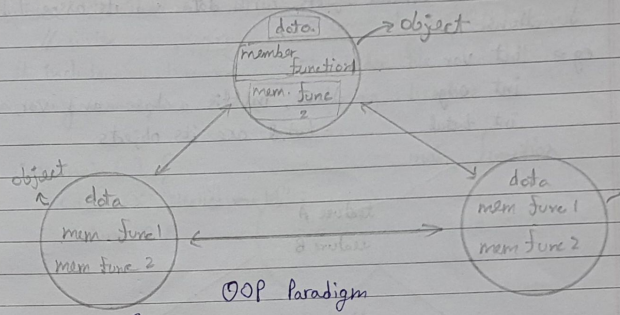
Behaviour { speed, mileage, avg life

Example of car → Class of Mahindra cars

The Corporate Paradigm -

(OOP data flowchart)





OOP Paradigm

Characteristics of OOPs

Object → A unit that contains data and its associated member functions.
or runtime entities

Some real-life functions :

(1) • Physical Objects
eg → Automobiles in traffic flow simulation
→ Electrical components in a circuit design
→ Aircraft in an air traffic control system

(2) Elements of the computer user environment
eg → Graphic objects like line, rectangles, circles.
→ Mouse, keyboard, disk drive etc.

(3) Human entities
eg → employees, students, customers etc.

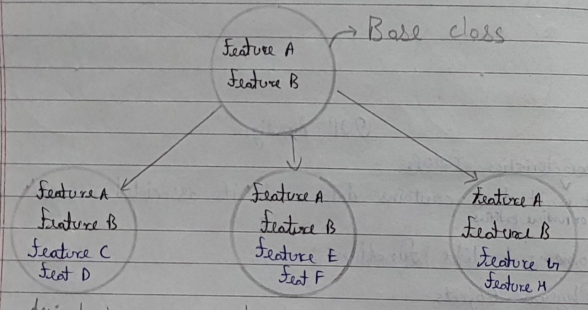
(4) User defined datatypes.
eg → angles, complex numbers, points on plane etc.

Classes - A collection of similar objects with data and its associated functions.

eg → int var
int avg
int total

Here, 'int' is a class and var, avg, total are its objects

Inheritance -



(benefit → code reusability)

eg → Vehicles
cars
mopads / 2-wheelers
3 wheelers
trucks

Deriving new class from base class.

whitespace → gap b/w two words
space, tab, newline

Program Structure :

```
// write a program          → comment optional
# include <iostream>        → header file mandatory
using namespace std;        → using directive

int main() {                → main function
    cout << "Hello world";
    return 0;
}
```

① Comments → i) Single line //
ii) Multiline line /* ... */
Not a statement
Replaced by a single whitespace

⇒ Compiler takes the comment as a single whitespace

② Header Files → # include → preprocessor directive
(/libraries)
eg → iostream → Input Output Stream → basically a class having (cin) and (cout) as objects, creating input & output flows.

③ Using directive

④ Main function → * function prototype
syntax - return type function_name (arguments)
eg → int main (void)

* function body is defined within curly braces { } which contains programming statements within.
* use of semicolon (;) → to terminate a program statement.

I bala gose Swami

⑤ Cin & Cout → Cout - Output stream

syntax - cout << "- - - -"; (insertion operator)

cin - Input stream

syntax - cin >>;

extraction operator

⑥ String → * Any inside double quotes (" ") :

* \ → escape sequence . eg → \n → newline (written inside)

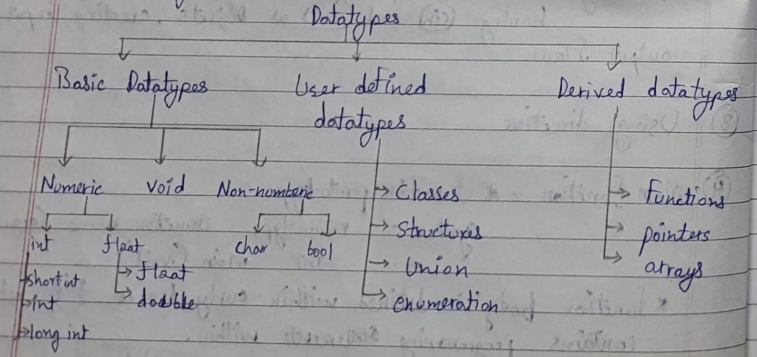
* endl manipulator → newline

⑦ return → * If void is return type then only 'return;'

* In turbo C++ we can end function with getch(), in place of return, which is used to hold the o/p screen.

Datatypes in C++

Datatypes define the size & type of the variable to be used in the program.



int → * short ; short varname ; [16 bits]
(-32768 to 32767 → 2 bytes, in all OS)

* int : int varname ;
(-2,147,483,648 to 2,147,483,647 → 4 bytes) (2bytes in DOS, 4 bytes in windows ≥ 32 bits, same as short)

* long int : long varname ;
(0 to 4,294,967,295) → 4 bytes → unsigned

Q) Write a program to ask the user to input a number b/w the short integer no. and display the output on screen.

```
# include <iostream>
using namespace std;

int main ()
{ short number;
  cout << "Enter a number between -32768 to 32767 :";
  cin >> number;
  cout << "\n The number is : " << number;
  return 0;
}
```

Float = 4 bytes
Double = 8 bytes
in memory

Q> Write a program to ask a user to input their weight.

Ans
```
#include <iostream>
using namespace std;

int main()
{ float weight;
  cout << "Enter your weight : ";
  cin >> weight;
  cout << "\n Your weight is : " << weight;
  return 0;
}
```

float → * float : takes 4 bytes in memory
* double : takes 8 bytes in memory.

char → takes 1 byte
stores character in ASCII code

Q> WAP to ask the user to input the size of their T-shirt, 'S' for small, 'M' for medium, 'L' for Large.

Ans
```
#include <iostream>
using namespace std;
int main() { char size;
cout << "Enter the size of your shirt : \'S\' for small, \'M\' for medium, \'L\' for large";
cin >> size;
cout << "\n Your size is : " << size;
return 0; }
```

Struct → various datatypes at once
array → single datatype at a time

bool → * two values → true → 1
→ false → 0

User defined datatypes

Classes - They are the user defined datatypes that contain some data variables and their associated functions
They are a group of similar objects.

Structure - Structure is a datatype which can contain a variety of datatypes in it.

Union - Union is a memory location that is shared by two or more different variables at different times.

Enumeration - It is an alternative method for naming integer constants
~~eg - pi, phi~~

Derived datatypes

array - An array is a container object that holds a fixed number of values of a single type.

function → It is a derived datatype which is built for special purpose and with some standard definition.

pointer → It is a special type of variable which holds address of another variable

Global var → var which are declared out side every function

5+14×22/3%4 → equal priority operations then priority given to operator that comes first from left to right

Syntax of pointer → int *ip;

ip = &x

↳ for assigning address

↳ for declaring pointer datatype to point

Operators

Operators are the special symbols that perform mathematical or logical operator on one, two or three operands and then return a result.

They are of various types:

1) Arithmetic operators → +, -, *, /, % → Higher priority → binary operators
2) Assignment operators → = (right to left) (left to right)
3) Relational operators → ==, !=, >, >=, <, <= → comparision
4) Increament / decrement operators ++a, a++, --a, a-- → condition
5) Logical operators (!, &&, || operate on boolean) → boolean
6) Conditional operators (? :)
7) Compound operator or assignment (x=, +=, -=, /=, %=, >>=, <<=, &=, ^=, |=)
8) Bitwise operators (&, |, ^, ~, <<, >>) → shift right, shift left
9) The size of () operator → size in terms of bytes.

→ Expression is smt that can be solved to (yield) a value

→ A grp of programming statement enclosed in curly brackets {} , making a block, is called compound statement

→ Types of control statement
- * Sequence structure → Straight line
- * selection structure → branching
- * Looping structure → iteration / repetation

Prefix > postfix > x++ = 1 (fast/speed)

A=10011100

B=01010101

00010100 = 20

right shift → no. decreases

left shift → no. increases

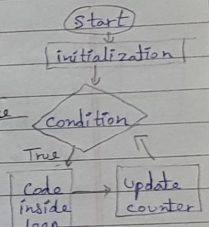
For Loop

for loop executes a series of statement till condition is True

Syntax → for (initialization ; condition ; update) {
}

eg → for (int i=1 ; i<=5; ++i) {
cout << i << " " ; }

output → 1 2 3 4 5



→ Multiple initialization

eg → for (int i=0, j=0 ; i<3 ; j++, i++) {
cout << endl ;
cout << "i= " << i << "j= " << j << endl ; }

output → i=0 j=0
i=1 j=1
i=2 j=2

→ Optional expression

eg → int counter = 0;
for (; counter < 5 ;) {
counter++ ;
cout << "Looping : " ; }
cout << "\n Counter: " << counter << ".\n";

output → Looping : Looping : Looping : Looping : Looping :
Counter : 5.

→ Empty for loop

eg →
```
int counter = 0;
int max = 3;
for ( ; ; ){
if (counter < max)
  { cout << "Hello! \n";
    counter++; }
else break;
}
```

Output →
```
Hello!
Hello!
Hello!
```

→ Null for loop

eg →
```
for (int i=0, i<5; cout<<"i: "<<i++<<endl);
```

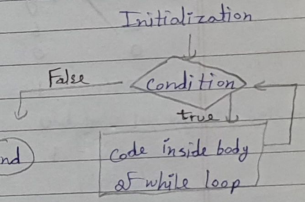
→ Nested for loop eg →
```
int rows = 3, columns = 4;
for (int i=0; i<rows; i++)
  { for (int j=0; j<columns; j++)
    { cout << "X"; }
    cout << "\n";
  }
```

Output →
```
XXXX
XXXX
XXXX
```

While loop

While statement continues to execute a block of statements while a particular condition is true.

Syntax →
```
while (condition)
  { // code
    // update }
```



eg →
```
int i = 0
while (i<=5){
  cout << i << "\n";
    i++; }
cout << "Complete. Counter: " << i << ".\n";
```

output →
```
0
1
2
3
4
5
Complete. Counter: 6
```

→ do while loop → The difference b/w do-while and while is that it evaluates its conditional statement (expression) at the bottom of the loop instead of the beginning of loop. Therefore the statements within the do-while block are always executed atleast once.

eg →
```
int counter = 2;
do { cout << Hello \n";
     counter--; } while (counter > 0);
cout << "Counter is : " counter << endl;
```

Output →
```
Hello
Hello
Counter is : 0
```

Bitwise operators

1) **AND** operator (&)
returns true only if both bits are 1.

a	b	a&b
0	0	0
0	1	0
1	0	0
1	1	1

eg → 12 = 0000 1100
&25 = 0001 1001
0000 1000 = 8 (in decimal)

→ int a=12, b=25;
cout << "a&b = " << (a&b) << endl;
output → a&b = 8

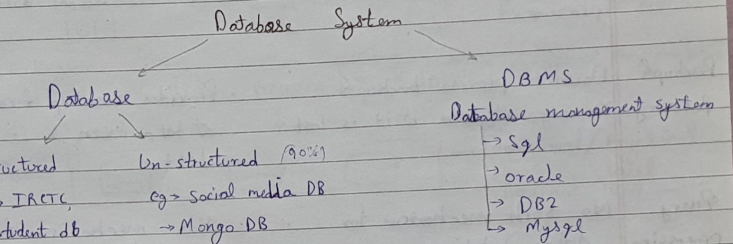
2) **OR** operator (|)
returns 1 if atleast one of the operands is 1

a	b	a|b
0	0	0
0	1	1
1	0	1
1	1	1

eg → int a=12, b=25;
cout << "a|b = " << (a|b) << endl;
output → a|b = 29

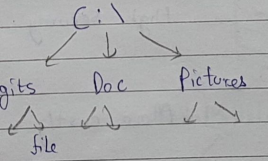
Unit-5

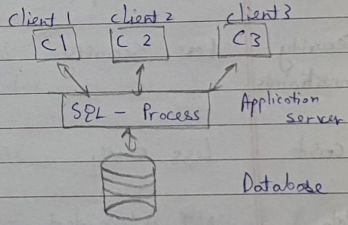
Data- A collection of facts.
Information- Useful data



File system Vs DBMS

NTFS - tree structure
(New technology file system)

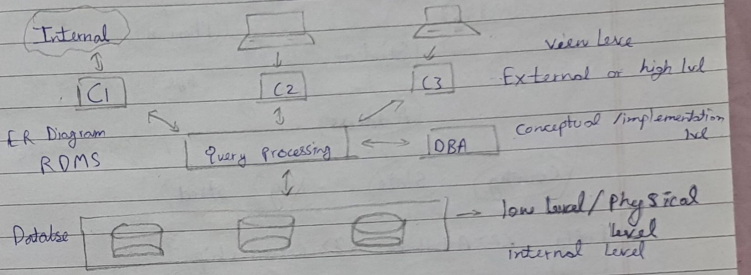




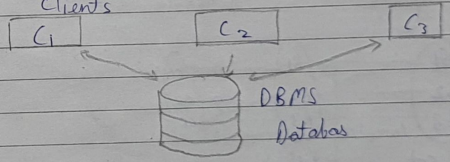
① Structure
② NTFS / EXT Extended file system - These are s/w that manage & organize file within a computer.

Aspect	Filesystem	DBMS
Structure		S/w for managing database
Redundancy	Redudant data can be present	No redundant data
Backup & Recovery	Doesn't provide backup & recovery of data if it is lost	Provides backup & recovery of data even if it is lost.
Query processing	No such mechanism for data fetching	Has mechanism for data fetching
Consistency	There is less data consistency	More data consistency
Complexity	Less complex	Highly complex
Security constraints	Less security	High security
Cost	Less costly	More costly
Data independence	Not independent	is independent
User Access	Only one user can access data at a time	Multiple users can access data at a time

DBMS 3-tier Architecture



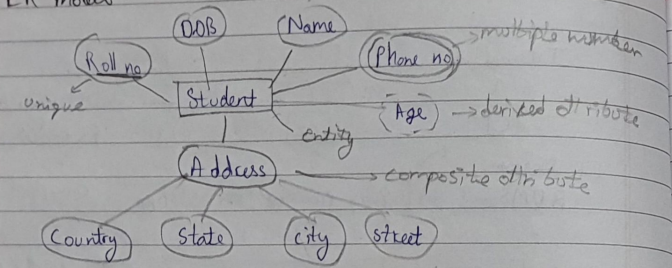
DBMS 2-tier Architecture
Client-server architecture



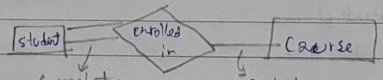
Data Models in DBMS

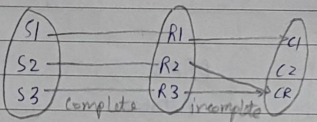
- Conceptual or object-based data model
  - ER Model
  - Object oriented model
- Representational or record based data model
  - Hierarchial model
  - Network Model
  - relational model
- Physical Data Model

ER model



1) Entity <2) Attribute <3) Relationship

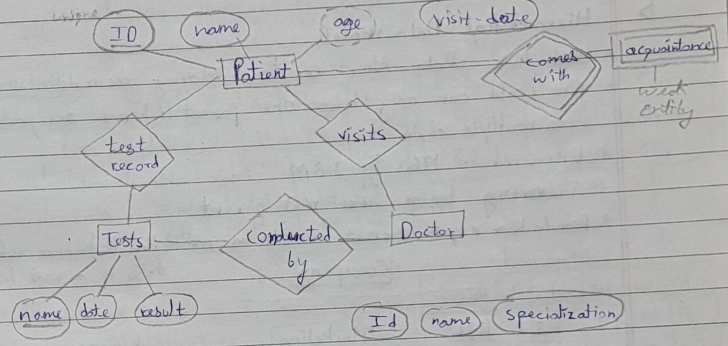




Entity set Relationship set Course set

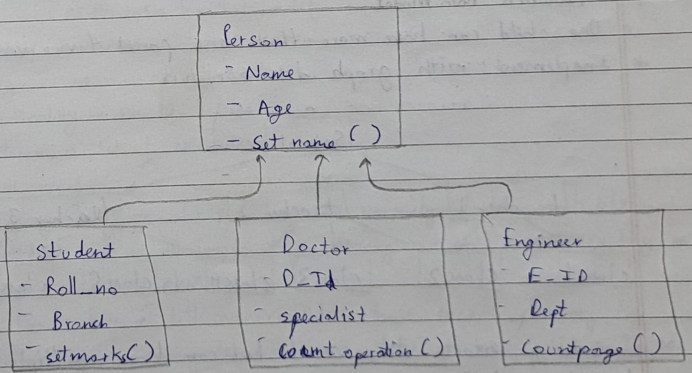
Object-based Data model

ER Data model



entity, entity set, relationship

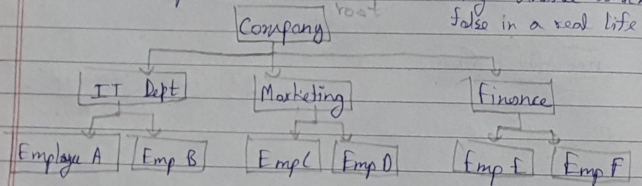
Object oriented Data Model



Record-based data model

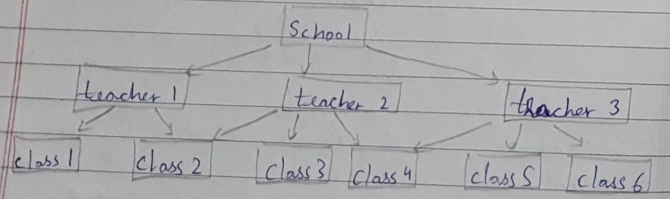
1) Hierarchical Data model

* uses tree Data structure for storing records
* One child can only have one parent whereas one parent can have multiple child
* Discovered in 1960 by IBM
* Traversing begins from root parent
* Drawback → emp B cannot work in marketing & finance which is false in a real life scenario



2) Network Data model

* One child can have more than one parent & vice versa.
* Implement with graph dat structure.



* Traversing starts from root but can have multiple paths to get to destination

3) Relational Data model

* uses tables as data structure to store data

student

Roll no	Name	Add	Ph.	Age	branch	Sem
1	Ram	Delhi	12	18	CS	I
2	Shyam	JBP	23	17	IT	II
3	Sita	Mumbai	34	19	EC	I
4	Rohit	Delhi	45	18	CS	III

Branch_code

branch code	branch name
CS	Computer Sciencen
IT	Info Tech
EC	Electronics

Branch_sub

Branch code	Sem	C1	C2	C3
CS	I	Bmf	BCE	Phy
EC	II	-	-	-
IT	III	-	-	-

composite key

* Attribute → Column names
* Relation Schema → Structure

schema → eg → student (Roll no, name, add, Ph, age, branch, Sem)

* tuple → a row which includes the value of attributes for that entity
* Relation instance →
* Degree → no. of attributes
* Cardinality → no. of tuples
* Column → set of attributes
* Null values
* Primary Key → unique key to uniquely identify a tuple.

  Candidate Key → all unique keys among which a primary key is chosen.

  Alternate Key → left over candidate keys except primary key

  Foreign key → Primary key in one table, non-primary key in another table.

Cloud computing

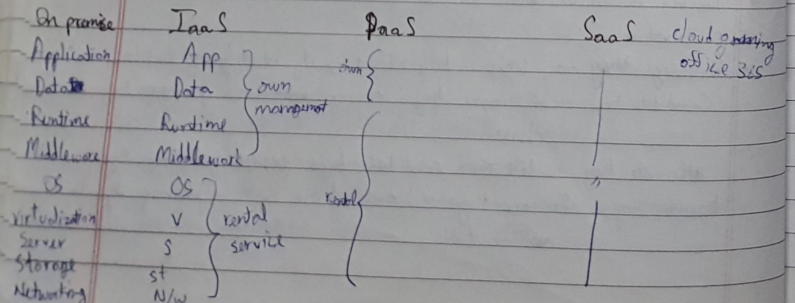
Cloud computing refers to manipulating configuring & accessing the applications online. It occurs online data storage infrastructure and application.

⇒ Essential characteristics of cloud computing

1) On-demand self-service
2) Broad Network Access (It provides you broad network access)
3) Resource pooling (large quantity of sources)
   eg → Storage, processing, memory, network, bandwidth
4) Rapid elasticity
5) Measured service - metered service

⇒ Service models of cloud computing

IaaS - Infrastructure as a service
PaaS - Platform as a service
SaaS - Software as a service



Types of cloud (Deploy)

i) Private cloud
ii) Community cloud
iii) Public cloud
iv) Hybrid cloud

Malware (Malicious S/w)

It is specifically designed to gain access or damage the Comp

eg → 1) Spyware 2) Keyloggers 3) Viruses 4) Worms
5) Trojans 6) Adware 7) Ransom ware 8) Rogue ware S/w

Money Laundering

1) Placement 2) Layering 3) Integration

Info. theft

1) Credit card no. theft 3) Pin Capturing 5) Electronic cash
2) ATM spoofing 4) Database theft

Email spoofing

Pretending to be an authentic email.

Denial of Serive (DOS)

blocking website server by purposely overloading it with requests,

→ Cyber stalking → Hacking
→ logic bombs → Pharming Attack
→ Computer Ethics DIY

Class & object in C++

class → most imp feature → leads to OOP

User defined data type → holds own data members & member funcs which can be accessed & used by creating instance of class

→ variables inside class defn → data members and the func are called member func.

→ eg → class of birds, all birds can fly and birds can fly (they all have wings) & beaks → flying → behaviour and wings & beaks → part of their characterstics

→ many diff birds in class with diff names but they all possess same behaviour & characteristics.

→ class → just a blueprint, which declares & defines characteristics & behaviour, namely data members & mem func resp.

Obj

→ Class → blueprint or template → no storage assigned to when class defined

→ obj are instances of class. which holds the data variables declared in class & mem func work on these class obj

→ each obj have different data variables.

Obj are initialized using special class functions called constructors

eg →
```
Class Student {
    public :
        string int roll_no;
        void print_roll () {
            cout << "Roll no. is " << roll_no;
        } };
```

```
int main () { Student Rohan;
        Rohan . roll_no = 123 ;
        Rohan . print_roll () ;
        return 0;
    }
```

Output → Roll no. is 123

Friend Functions & protected

→ OOP data hiding → restricts access of private members outside the class.

→ However, a feature in C++ called FF breaks this rule & allows us to access mem func from outside.

→ basically a func that is used to access all pri & prot mem of class

→ considered non mem func → declared by using prefix friend keyword.

eg →
```
class A { int a;
    public :
        A () { a=0; }
        friend void showA (A&); };
void showA (A& x) { cout << "A :: a=" << x.a;
int main () { A a;
        showA (a) ;
        return 0; }
```

FF	Mem F
→ declared in any no. of cla using Friend keyword.	→ only dec in private, pub or prot scope of particular cla
→ can be called without any obj	→ has to create obj of some class to call mem func
→ Friend keyword	→ no keywork
→ used to modify or change pri & prot	→ code reusability, make code maintainable
→ not a part of class	→ part of call defn & invoked by par obj

Virtual func

→ func in base class which is overridden in derived class & tells compiler to perform late binding on this func.

→ Virtual keyboard is used to make a mem func of base class virtual

* ensures that correct func is called for an obj regardless of types of reference (or pointer) used for function call.
* mainly used to achieve runtime polymorphism.
* Func are declared with virtual keyword in base class.
* Resolving of func call is done at runtime.

eg→
```
class base {
    public:
        virtual void print () { cout << "print base class \n"; }
        void show () { cout << "show base class \n"; } };
class derived : public base {
public:
    void print () { cout << "print derived class \n" ; }
    void show () { cout << "show derived class \n"; } };
int main () {
```

Constructor

→ special mem func of a class that is executed whenever we create new obj of that class.
→ exact same name as class & it doesn't have any return type not even void
→ very useful for setting initial values for certain mem vars
→ special class func perform initialization of every obj.
→ compiler call const to init vals to obj mems after storage alloc to obj

Types of consts:

① Default const
→ does n't take params.

② Parameterized const → takes params
→ used to provide diff vals to data membs of diff objs by pass appropriate values as arguments

③ Copy const → special type of const which takes obj as args & used to copy vals from data mems of one obj to another.

eg→
```
class construct {
    public:
        float area;
        construct () { area = 0; }
        construct (int a, int b) { area = a * b; }
        void disp () { cout << area << endl; } };
int main () { construct o;
              construct o2 (10, 20);
              o.disp ();
              o2.disp ();
              return 0; }
```

output → 0
200

	Const	Destructor
① →	init obj of class	→ destroy instances
② →	Classname (args if any) { const's body }	→ ~Classname (no args) { }
③ →	either accept args or not	→ can't have args
④ →	called when instance/obj of class is created	→ called when obj of class is freed or deleted

③ → dloc mem to instance	⑥ deallloc mem of inst/obj.
④ → can be overloaded	⑥ It can't be overloaded.
⑤ → same name as class	⑦ same name but preceded by (~) symbol
⑧ multiple const.	⑧ single destr
⑨ concept of copy const → init an obj from another obj	⑨ no copy destructor concept

Overloading

→ concept used to avoid redundant code where same method name is used multiple times but with a diff set of parameters.

→ actual method that gets called during runtime is resolved at compile time, avoiding runtime errors.

→ provides code clarity, eliminates complexity & enhances runtime performance.

Function overloading in C++

→ feature of oop where two or more func can have same name but diff params.

→ func name overloaded with diff jobs = FO.

→ FO is considered as eg of polymorphism feature in C++.

```
eg→ void print (int i) { cout << " Here is int " << i << endl; }
    ——x— (double f) { ——x—— float —x f ——x—— }
    ——x— (char const *c){ ——x—— char* —x— c ——x— }
int main ( ) { print (10);
               print (10.10);
               print ("ten");
               return 0; }
```

output: Here is int 10
——x— float 10.1
——x— char* ten

Scope resolution operator

→ ~~In C++,~~ scope resolution operator is used to reference global var or mem func that is out of the scope.

→ ∴ scope resolution operator to access the hidden variable or function of a program. ~~The operator~~.

→ Operator is represented as double colon (::) symbol.

eg → when global & local variable or func has same name in a program, and when we call the variable, by default → local variable is ∴ it hides glo var or func

To overcome this situation, we use scope res oper to fetch a program's hidded variable or function.

```
int num 50;
class oper { public:
                Void fun ( ); };
Void oper :: fun ( ) { cout << " It is mem func of class "; }
int main ( ) {
 int num = 100;
 cout << " local = " << num;
 cout << " global = " << :: num;
 ~~oper~~ oper op;
  op.fun ( );
 return 0; }
```

output → local = 100;
global = 50
It is mem func of class.

Inheritence

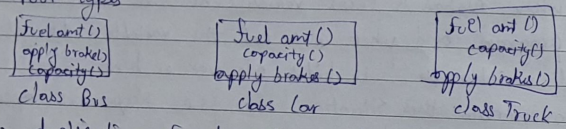
→ one of most imp concepts in OOP.

→ derive a new class from another ~~class~~ class

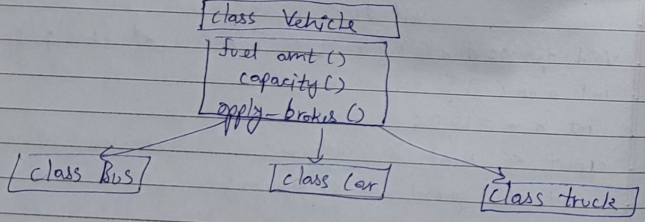
→ existing class = base class, new class = derived class

→ when creating a class, instead of writing completely new data members and member func it provides a facility of inheriting members of an existing class.

four types



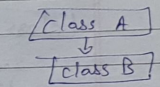
→ duplication of code 3 times → ↑es chances of error & data redundancy. → To avoid this type of situation inheritance is used.



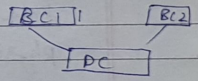
Eg → class Parent { public :
int id_p; };
class Child : public Parent { public:
int id_c; };
int main() { Child obj1;
obj1.id_c = 5;
obj1.id_p = 10;
cout << " Child = " << obj1;
cout << " Parent = " << obj1;
return 0; }

Diff types :

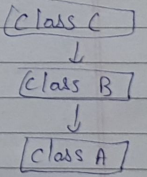
(1) Single inheritance → a class → only allowed to inherit from one class.
→ i.e one subclass derived from one base class
class subclass : access mode base class { }



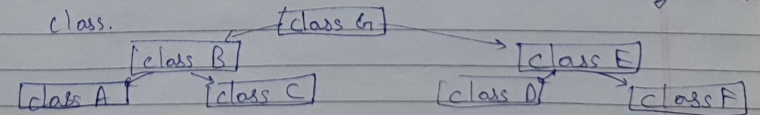
(2) Multiple Inheritance → C++ → class can inherit from more than one classes
→ one subclass → more than one base class
class subclass : access-mode base class1, a_m b_c2, ---
{ }



(3) Multilevel Inheritance → a derived class is derived from another derived class



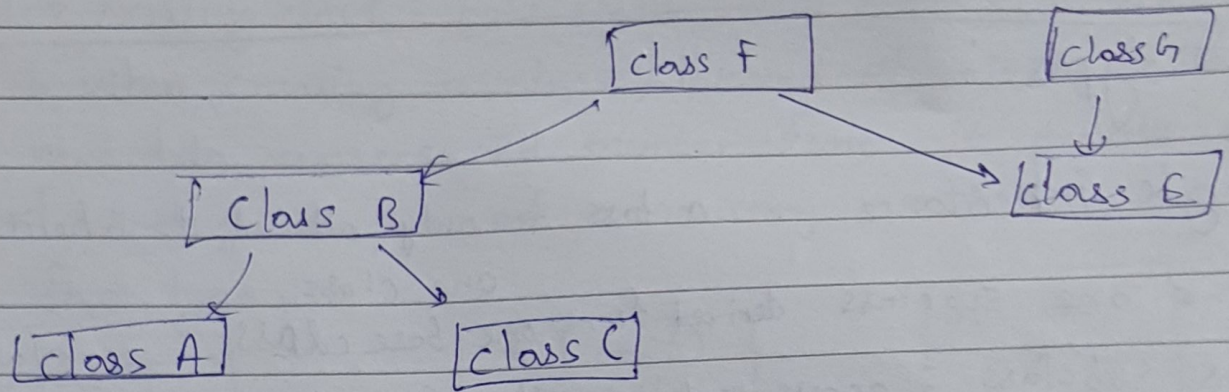
(4) Hierarchical Inheritance → more than one subclass is inherited from an single base class.
i.e more than one derived class is created from a single base class.



③ Hybrid Inheritance

→ combining more than one type of inheritance.

eg → combining Hierarchical inheritance & Multiple Inheritance



Sender

application →

presentation → check data → compress → easier

session → maintain connection b/w sender & Receiver

Transport → check for compressibility, virus

Network

data link → gives data link to access data with link

physical

5-bit → 6 bit code system
Combination
32 instructions

Sitanshu Jain
CS assitant

1950 - Machine
1970 - Assembly - mnemonic codes, 90% of machine, operand, symbol table

⇒ Performance parameter: Complexity (time, space)
Redundancy
Size of program (line of codes)

⇒ ~~Object~~ Procedure oriented Programming (POP)
or Function " "

1) Divide programme into no. of functions. (funcs can be called multiple times)
2) Focus on func rather than data

⇒ ANSI → ASCII
American national standard institute
American standard code for info interchange

⇒ Assembler, compiler, interpreter
interpreter → generates token for each line & continues execution from previous error line after debugging.

EBCDIC system 8 bit

⇒ Java → UNICODE → 16 bit combination
Universal code

⇒ ISO CODE → 32 bit
International standard organisation code

A programming language is defined by a set of certain rigid rules, much more & inflexible than any natural language.

Lexicon - The rules that determine which symbols (letters, digits, punctuation marks etc) could be used in language

Syntax - Set of rules that define how these symbols are to be used.

Semantics - Every statement is expressed in a syntax manner and to recognize its meaning is called semantics
Should be →
→ Lexically correct
→ Syntactically & semantically correct

* Intruction list → predefined keyword

* "High level languages make a bridge b/w the human lang & the machine lang"

→ Source code

Structure of C Programming

.h ① Header Files → stores specific predined definitions eg → stdio.h, conio.h (console) ↓ clrscr(), getch()
② Structure
③ Declaration of func
④ Main Func

① ⇒ header files → stdio.h (Standard input output) eg → printf, scanf
→ conio.h (console input output) eg → ~~console~~ clrscr(), getch() (clear screen, get character)
→ string.h eg → strlen(), strCmp(), strupr(), strlwr()
→ math.h eg → cos, sin, sqrt
→ dos.h eg → delay
→ process.h eg → exit(0)

declaring header file → # include <stdio.h> → searches entire system
→ # include "stdio.h" → searches only current directory
Program starts processing from # ↵ preprocessor directive ↵ directories store files Keyword

② ⇒ Main → Starting point of compilation & execution
→ User defined func, stored in library and called by operating sys.

⇒ Tokens → The smallest individual units of a programme are tokens.
i) Variables ii) Constants iii) Key words iv) Tokens
v) Literals vi) Identifiers vii) Operators

Lexicons → symbols

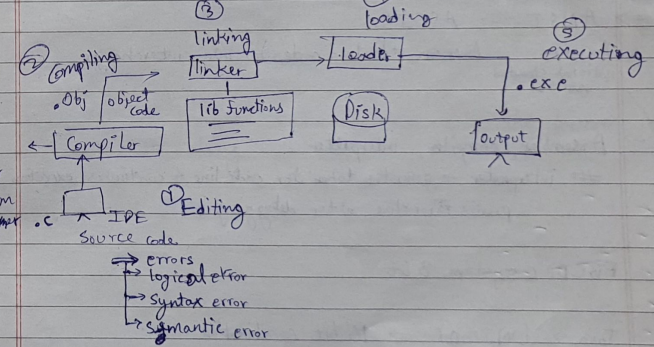
Top-down → execution of current func depends on exec of previous func.

Bottom-up → Obj is linked with func. Even if func doesn't execute, obj still exists

Compiler
Source code → ~~Source co~~ code having probability of errors.
↓ compiler
Object code → Code without errors.
↓
Machine code

.c
.obj
.exe

Phases of Program / Program lifecycle



Checks for error & shows em to programmer

compiler → memory space resides inside memory & acts as simulated comp.

I/p & O/p statements

stdio.h → printf
scanf

① printf

Syntax → printf (" ");

eg → ① IDE to User screen

```
printf (" My name is ");
O/p → My name is

printf (" My first program");
O/p → My first program
```

display

puts → put string (only str)
printf → every datatype. (every datatype)

accept value from user

gets → string
scanf

★ Jis bhi entity are scope hota hai usko terminate (;) Be Nahi karte eg → int main (void) ; ✗
→ int main (void) { }

Format Specifier

Datatypes	Format conversion Specifier
int	%d
float	%f
char	%c
string	%s
double	%lf
pointer / address	%u

int = 2
float = 4
double = 8
long = 4
long double = 10

Tokens

1) Variables → assigning a name to memory space.
2) Slots / [containers / boxes] in memory ram where data is stored & data can be changed during the execution of program.

Varible declaration →

datatypes → inbuilt types / predefined → int, float, double, char, long, (2 byte) (4 byte) (8 byte)
→ user derived → derived from predefined → variable, strings, pointers, array, enum, func
→ user defined → ~~→ main~~
↳ both predefined & user derived.
structure, class, main

identifiers → logically declared entities

eg → i) void main ()
{ float x = 3.125 ;
printf (x); }
O/p → nothing

ii) int main ()
{ int x = 3;
return x; }
O/p → 3

iii) int main ()
{ int x = 3;
printf (x);
return 0; }

iv) int main ()
{ float x = 3.125;
printf ("%f", x);
return x; }
→ 3 cuz
O/p

; → Semicolon → End of line

~ tilde
^ Caret
& ampersand

C 11

$0 \times 8^2 + 0 \times 8^1 + 1 \times 8^0$
$8 + 1 = 9$

* eg → (1) int a;
int b;
char c;
int d;

(2) int a, b, d;
char c;

(3) int a = 10;
int b = 20;
int d = 30;
char c = 'A';

int x = 10;
i) name → x
type → int
value → 10

address → 0xBFCC02
size → 2 bytes
range → -32768 to 32767

Rules to declare identifiers

* Cannot start with a digit but we can use digits after alphabets
eg → 10x ✗ x10 ✓

* Cannot use any special character except underscore
eg → x$ ✗ $y ✗ -y ✗ %x ✗
int _x ✓, x_ ✓, x_x ✓

* Cannot use keywords (predefined standards) as identifiers.
eg int int ✗, int float x — —
but int Int, int flOAt ✓ — — — case sensitive

* Cannot use space b/w two characters
eg → a b ✗ a_b ✓

wild characters Only print & not scan
"\n" → new line
"\t" → tab
"\b" → backspace
"\r" → return carraige → returns last value
"\v" → vertical representation
"\\" → forward slash

(ii) From memory to user screen
Syntax →
printf ("format specifier", variable name)

eg → int x = 10;
printf ("%d", x);

float y = 3.125;
printf ("%f", y);

→ char c = 'A';
printf ("%c", c);

Char → single quotes

→ int a, b, sum;
a=10; b=20; sum=30;
printf ("Sum %d + %d = %d", a, b, Sum);

Variable declarations : (datatype)
(1) Return type var_name;
(2) Return type var1, var2, var3;
(3) Return type var1 = value, var2 = value;

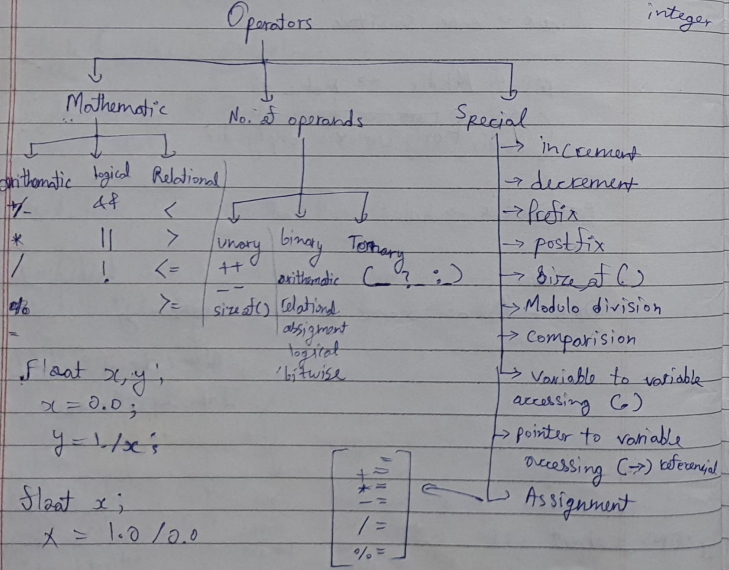
Literals : value held by a variable

125,0, -125 ← int → char → 'A', 'a', 'B'
12.5, 2e5 ← float → string → "Abc"

format conversion specifiers → %d, %f, %s

Type casting → Up casting → low size to high size eg int → float
→ low casting → high " to low size → float → int

$6.62607 \times 10^{-34} = 6.62607E-34$ or $6.62607e-34$
(value after e should be integer)



float x, y;
x = 0.0;
y = 1./x;

float x;
x = 1.0 / 0.0

Before word formatting we set page size & the margins & do some preliminary settings
→ Ruler → Page setup (Custom A4 size) ① → Page size
② → margin
③ preliminary settings

=rand(1, 3) → no. of lines in para.
↳ no. of paragraph
→ to generate random text

Format painter – It is used to copy the formatting from one text and apply it to another.

Right click on heading 1 and select update heading 1 to match selection

+ – – – + – – + – – + → table

shift + alt + arrow ↓ → move row
shift + alt + enter → split table
shift + alt + up ↑ → merge

Kali linux → hacking → Socket level Security

move source_dir destination_folder

Linux Command

→ Linux is a command line user interface operating system.
→ It is worked on directories

(a) Types of OS:

(1) Batch O.S → batch of instructions as (.bat) file get executed automatically.

(2) Single & multi user OS → Linux

(3) Single tasking & Multitasking os
bDOS → windows

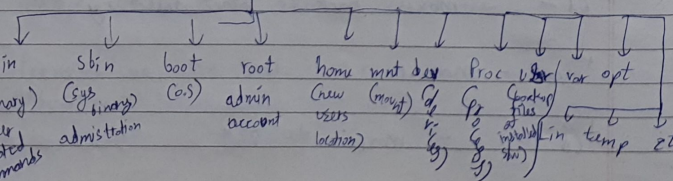
(4) Real time OS → weather forecast.

(5) Embedded O.S → Mobiles, modem, routers

(6) Distributed O.S. → ticket counter (no cpu only keyboard, mouse, monitor)

(14) directories in Linux

1) Root directory (Slash) /



Modulus → find digits in number
(%) find out leap year, prime number, even/odd

fast
prefix > postfix > i = I + 1
reduces instruction sets

C → a → command prompt

DOS → case insensitive
Linux → case sensitive

MD & Mkdir → make dir
RD → remove dir
(Rmdir only removes empty dir

Attributes → hide, write, read

DNS → Domain name system → resolving
Name to IP
IP to Name

→ DNS, telnet, FTP

POST → Power on self test (checks)
BIOS
MBR → Masterboard record → priority of boot

* 'touch' - makes empty file (Files) eg → touch <filename>
* 'Cat' - Non empty file (Concatenate) → 3 duties → create the file
  → to see content inside file
  → to append some more content on existing file.

only works with redirectional operator →(>) Input Redirectional op.
→(<) O/p " "
→(>>) append " "

* 'File *' - to check if file is empty

eg → cat > file_name
gives prompt to enter some value
ctrl + D to save
→ cat filename
output
→ cat < filename
output

→ cat >> file_name
append some value

* 'mkdir' → to make directory / directories

Permissions → Read → r
(Attributes in DOS) → Write → w
→ Execute → x

[root @ localhost ~] #
PC name
~ = cwd
# = user → Admin user (#) (root)
→ Normal user ($)
@ = link sign

(List)
* 'ls' - This command is used to check the content inside the directory.

Dark blue → Directory, black → file, Dark green → executable file

'ls -l' - notations → d → directory

'pwd' - present working directory

'clear' - clears console

'cd' - change directory eg → cd Desktop/

'cd ..' - moves back to prev. directory

'cd ../..' - moves back two prev. dirs

'cd -' - moves back to start directory

'man' - manual

'cd ~' - moves to root

Permissions ~~assignment~~ in Linux

- user (Creator)
- group (co workers)
- other (spectator)

`drwxr-xr-x 1 root root 57 Dec 21 14:14 ai`

- user → user
- group → group
- other → other
- 1 → ~~user~~ links
- root → user name
- root → group name
- 57 → size in bytes
- Dec 21 14:14 → date/time
- ai → file name

Assigning permission → 'chmode'

→ symbolic → + (assign), - (-) revoke
→ numeric

Eg → # chmode u+x ai → file name
- u → user
- + → assign permission
- x → execute permission

chmode u-x ai
- u → user
- - → revoke permission
- x → execute permission

→ # chmode u+x, g+w, o+w ai
- u → user
- g → group
- o → other

or

chmode go+w ai
- go → both ~~together~~ together

'init 0' → shutdowns linux PC

CLI → Command Line Interface
GUI → Graphical User Interface