PAGE No.
DATE / /

OOPS

OBJECT ORIENTED PROGRAMMING

Things Entry | Based on | Language | code

1. Supports different programming languages

   Example: C, C++, JAVA, PYTHON, etc.

2. It is used for development of software.

3. The main aim of OOP is to bind together the data and functions that operate on them so that no other part of the code can access this data except that function.

4. OOP aims to implement real world entities like inheritance, hiding, polymorphism, etc in programming.

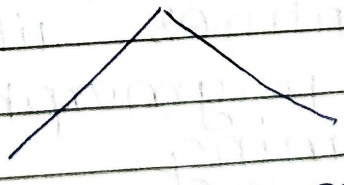
PAGE No.
DATE / /

Key concepts of OOPS:

1] Object

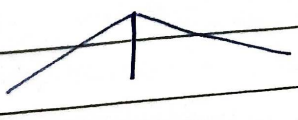
1. Basic unit of OOPs and represents the real-life entities
2. Instance of class
3. When class is defined → no memory allocated but when instantiated (object is created) memory is allocated.
4. An object contains data and code to manipulate the data
5. An object has an identity, state and behaviour.
6. Object

Entities / Things



characteristics property

Example: Book



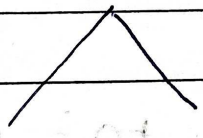
colour Author Price

PAGE No.
DATE / /

2] Classes

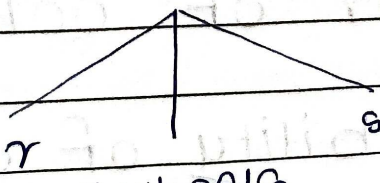
1. user-defined data type
2. Contains objects with singular characteristics
3. Consists of data members and member functions, which can be accessed and used by creating an instance of that class.
4. Represents set of properties or methods that are common to all objects of one type.
5. Class

Blueprint / Templates



object Properties

Example: CAR



3] Encapsulation

1. Defined as → wrapping up of data under a single unit.

2. It is the mechanism that binds together code and the data it manipulates

3. variables / data of a cls → hidden from → other class → & can be accessed only through any member function of their class in which they are declared.

4. Data in cls is hidden from other classes

5. known as data-hiding.

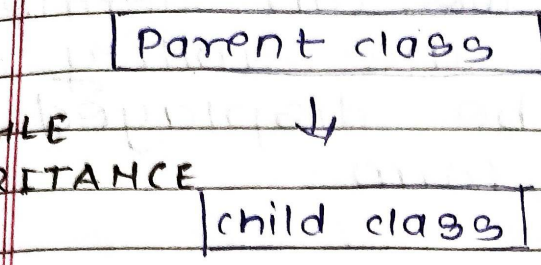
En → Inside
capsule → cover to protect.

4] Inheritance

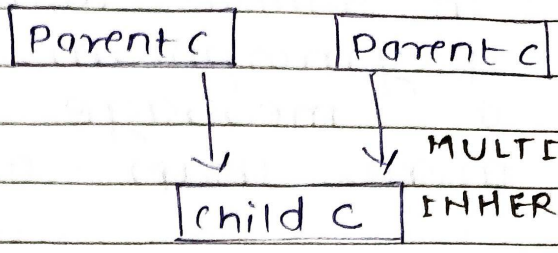
1. Imp pillar of OOP.

2. The capability of a class to derive properties and characteristics from another class is called inheritance.

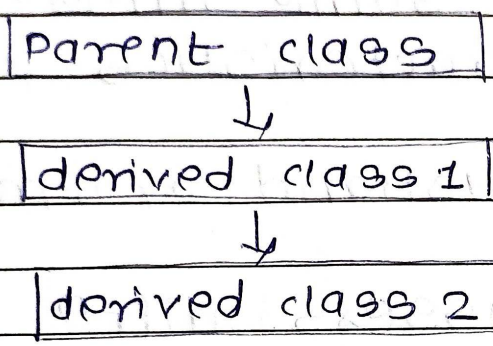
3. Develops new class from the existing classes



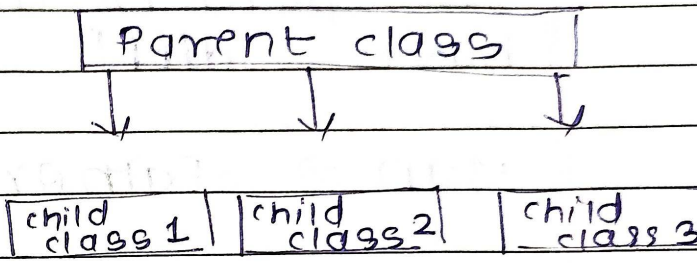
SINGLE INHERITANCE



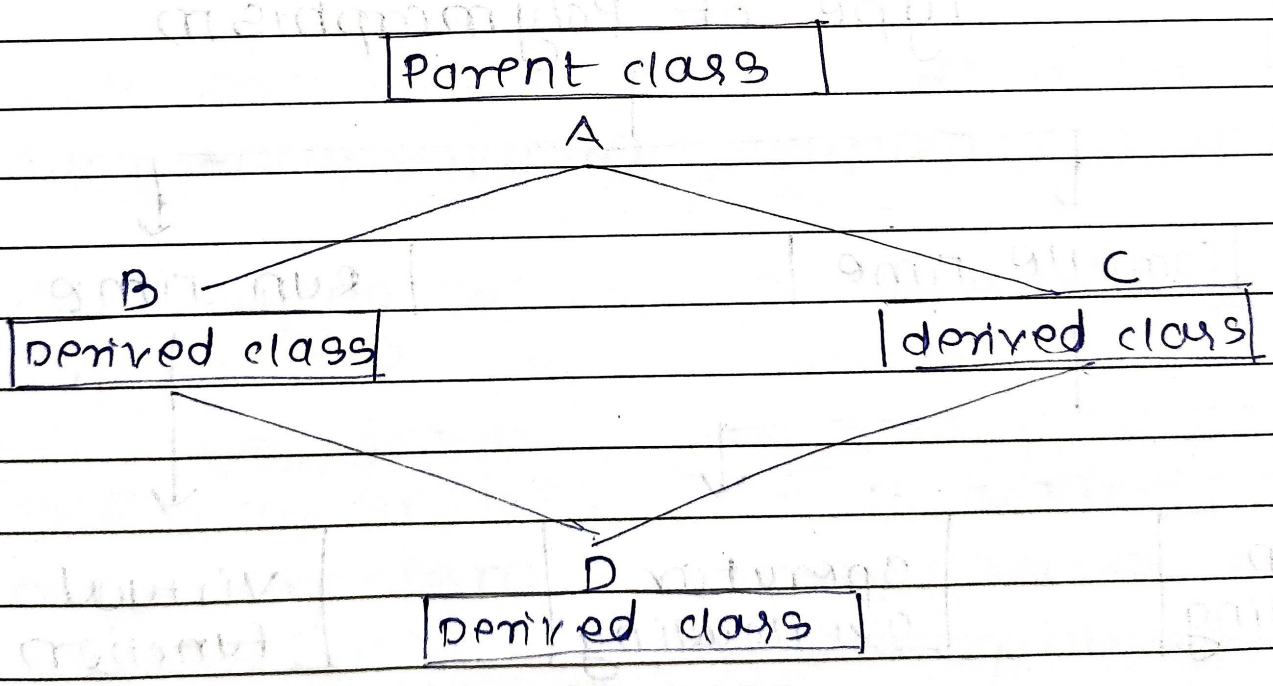
MULTIPLE INHERITANCE



MULTILEVEL INHERITANCE



HIERARCHICAL INHERITANCE



HYBRID CLASS

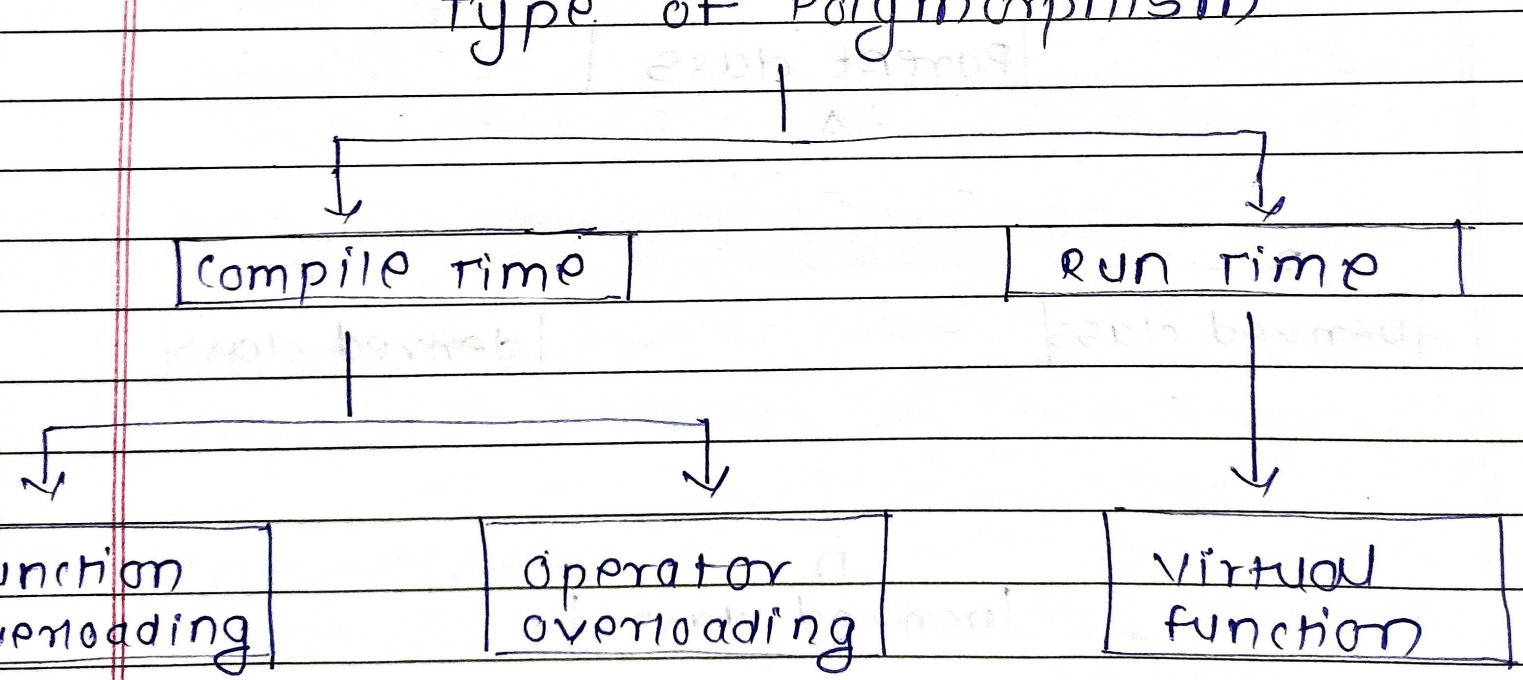
5] Polymorphism

1. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

2. Example:
A person at a same time can have diffⁿ characteristics.

Man → Father, Husband or Employee

Same person possess diffⁿ behaviour in diffⁿ situations.



PAGE No.
DATE / /

6] Data Abstraction.

1. Most essential and imp feature of OOP.

2. DA refers to providing only essential information about the data to the outside word, hiding the bg details or implementation.

Difference btwn OOP's & POP

	OOPS	POP
1.	OOP focuses on objects & classes	1. POP focuses on procedures & functio
2.	OOP encourages data encapsulation	2. POP doesn't support encapsulatio
3.	OOP supports inheritance	3. It typically lacks inheritanc
4.	OOP emphasizes polymorphism	4. It doesn't emphasizes poly-morphism
5.	Code reusability present	5. Code reusability absent.
6.	Ex: C, FORTRAN, Pascal, Basic	6. Ex: C++, JAVA PYTHON, C#*

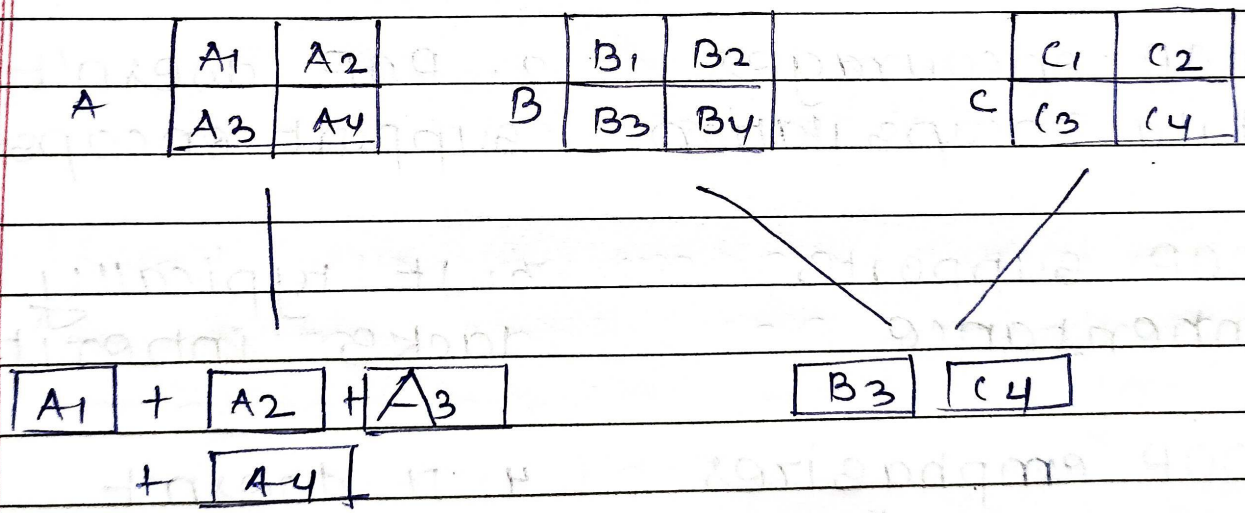
Different Paradigms for solving problems in c++

1] Inheritance Paradigm

Inheritance allows you to create a new class based on an existing class.
This faciliates code reuse and the creation of hierarchial relationships between classes.

2] Composition Paradigm

Mixture / combination of different components



A1 + A2 + A3 + A4 + B3 + C4

3] Encapsulation Paradigm

Encapsulation involves bundling data and the methods that operates

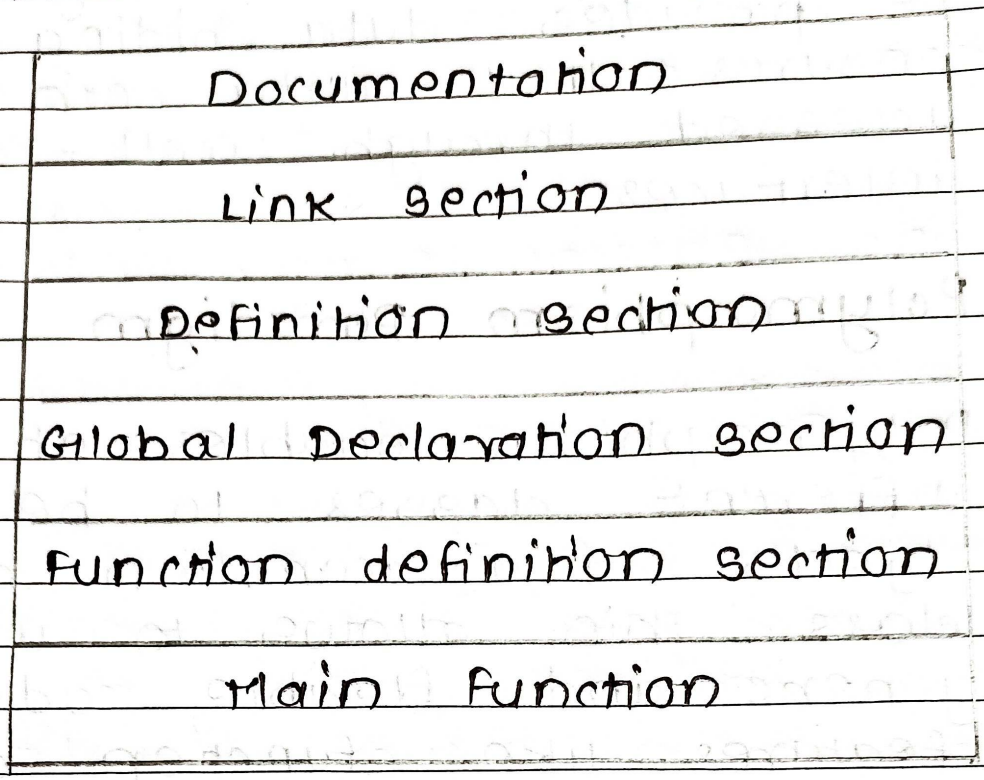
on the data within single unit (class). It provides data hiding and ensures that data can only be accessed through well-defined interfaces.

4] Polymorphism Paradigm

Polymorphism enables objects of different classes to be treated as objects of a common base class. This allows to write more generic and flexible code through features like function overloading and virtual functions.

5] Interface Paradigm

Structure of c++ program



1. Documentation section

1. First section in structure of c++
2. Used to doument logic of the program that the programmer going to code.
3. It can be also used to write the purpose of program.
4. Whatever written in documentation section is the comment and is not compiled by compiler.

DATE / /

2. Linking section

1. Header files

a. Generally a program includes various programming elements like built in functions, classes, keywords, etc. that are already defined in the standard c++ library.

b. In order to use such pre-defined elements in program, an appropriate header must be included in the program.

c. Standard headers are specified in a program through the pre-processor directive #include.

2. Namespaces

a. A Namespace permits grouping of various entities like classes, objects, functions, etc.

b. Any user can create seperate namespaces of its own and can use them in any other program

3. Definition section

1. It is used to declare some constants and assign them some value.

2. In this section, anyone can define your own datatype.

4. Global Declaration section

1. The variables and the class definitions which are going to be used in the program are declared to make them global.

2. The scope of variable declared in this section lasts until the entire program terminates.

5. function Declaration section

1. It contains all the functions which our main functions need.

2. usually, this section contains the user-defined functions.

DATE / / /

6. Main function

1. The main function tells the compiler where to start the execution of the program.

2. All the statements to be executed should be written in main function.

3. The compiler executes all the instructions which are written in the curly braces { } which encloses the body of the main function.

Data Types in C++

1] Primary Data Type
These data types are built-in or predefined data types and can be used by the user directly to declare variables.

Primitive data types in C++ are:
Integer
character
Boolean
floating point
Double floating point
valueless or void
wide character.

2] Derived Data Type
DDT that are derived from the primitive or built in datatypes are reffered to as DDT

four types of DDT :
function
Array
Pointer
Reference

3] Abstract or User-defined data type
A or UDDT are defined by the user itself. Like, defining a class in c++ or a structure.

User-defined data types:
class
Structure
Union
Enumeration
Typedef defined datatype.

PAGE NO.
DATE / /

Pointers

1. Variables that stores another variable's memory address or location.

2. It enables / allows to work direcly with memory address.

3. It requires careful handling so that to avoid errors and security issue.

4. It provides a way to acces or manipulate the data indirectly.

5. It is denoted by or represen by * [aestric]

Strings

1. A string is a data type that is used in programming to represent text rather than numbers.

2. It is a one-dimensional array.

3. A string is a sequence of characters and can contain letters, numbers, symbols & spaces.

4. It is represented by " "
{Double Quotation marks}
↓
Text in string is enclosed by -
"2", "a", "&"

5. String is end using NULL ptr
"\0"
It is important to use end pointer to end string otherwise it will be infinite.

PAGE No.
DATE / /

Structure

↓

storage area / method

1. It is used to store data
2. It is a collection of different elements / variables into one place
3. It consist of different data type
4. Example : Student as structure

↓

Name, Rollno, Phn no, Email

Different data types.