Data hiding, Encapsulation, Abstraction, Polymorphism.

Data hiding: It is a concept in OOP which confirms the security of members of a class from unauthorized access.

It is a technique of protecting the data members from being manipulated or hacked from any other source.

It reduces some complexity of the system. Data hiding can be achieved through the encapsulation, as encapsulation is a subprocess of data hiding.

It guarantees restricted data access to class members and maintain object integrity.

Private, public and protected are three types of protection/access specifiers available within the class. Usually, the data within a class is private and the functions are public. The data is hidden, so that it will be safe from accidental manipulation.

Encapsulation: It binds the data and functions together which keeps both safe from outside interference. Data encapsulation leads to data hiding.

The private members of a class are only accessible to the objects of that class only, and the public members are accessible to the objects of that class as well as they are accessible from outside the class. Encapsulation helps the end user of a system to learn what to do with the system instead of how it must do.

Encapsulation makes the system easier to operate by the end user.

Abstraction: It is primarily used to hide the complexity.

It indicates the necessary characteristics of an object that differentiates it from all other types of objects.

An abstraction concentrates on the external aspect of an object. For an object, abstraction provides the separation of the crucial behavior from its implementation.

A proper abstraction emphasizes on the details that are important for the reader or user and suppresses features that are, irrelevant and deviant.

Types of abstraction:

Procedural abstraction – It includes series of instructions having the specified functions. Procedural abstraction provides mechanism to abstracting well defined procedures or operations as entities. The implementation of the procedure requires a number of steps to be performed.

Data abstraction – It is a set of data that specifies and describes a data object. this principle is at the core of object orientation. In this form of abstraction, instead of just focusing on operations, we focus on data first and then the operations that manipulates the data. Happens in classes and objects, functions /procedures specific to certain data.

Basis	**Data hiding**	Encapsulation	Abstraction
Definition	Hides the data from the parts of the program.	Concerns about wrapping data to hide the complexity of the system.	Extracts only relevant information and ignore inessential details.
Purpose	Restricting or permitting the use of data inside the capsule.	Enveloping or wrapping the complex data.	Hide he complexity.
Access Specifier	The data under data hiding is always private and inaccessible.	The data under encapsulation may be private or public.	-

Polymorphism: The word polymorphism means having many forms. We can define polymorphism as the ability of a message to be displayed in more than one form.

In C++ polymorphism is mainly divided into two types:

- Compile time polymorphism.
- Run time/Dynamic polymorphism.

Compile time polymorphism: - This type of polymorphism is achieved by function overloading and operator overloading.

Function overloading: When there are multiple functions with the same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in number of arguments or/and change in type of arguments.

Rules for function overloading:

Function declaration that differs only in the return type are equivalent.

Parameter declarations that differ only in a pointer * versus an array [] are equivalent.

Parameter declarations that differ only in that one is a function type and the other is pointer to the same function type are equivalent.

Parameter declarations that differ only in the presence or absence of const and/or volatile are equivalent.

Two parameter declarations that differ only in their default arguments are equivalent.

Operator Overloading: C++ also provides option to overload operators.

E.g.: We can make the operator ('+') for string class to concatenate two strings. We know that this is the addition operator whose task is to add two operands. So single operator '+' when placed between integer operands adds them and when placed between two string operands, concatenates them.

Run-time polymorphism:

It is also known as dynamic polymorphism or late binding. In run-time polymorphism, the function call is resolved at the run time.

In contrast, to compile time or static polymorphism, the compiler deduces the object at run time and then decides which function call to bind to the object. This type of polymorphism is achieved by function overriding.

Function overriding: Function overriding occurs when a derived class has a definition for one of the member functions of the base class. The base function is said to be overridden.