

C++ OOPs Concepts

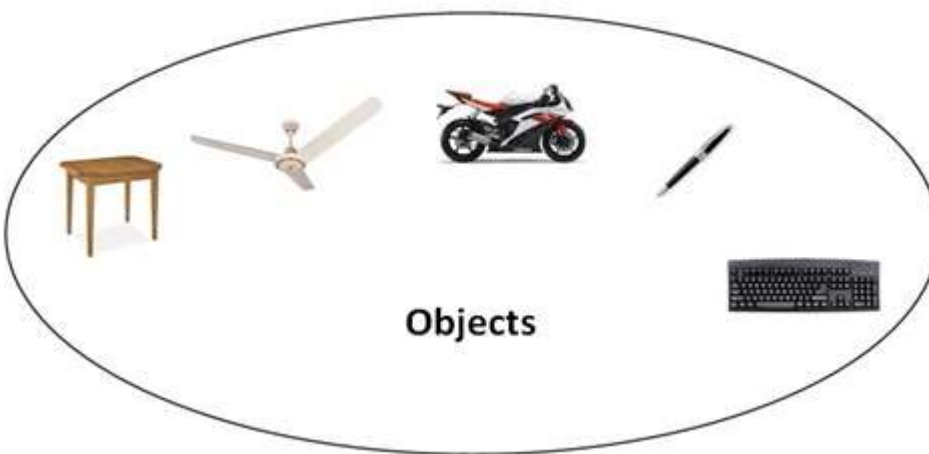
The major purpose of C++ programming is to introduce the concept of object orientation to the C programming language.

Object Oriented Programming is a paradigm that provides many concepts such as inheritance, data binding, polymorphism etc.

The programming paradigm where everything is represented as an object is known as truly object-oriented programming language. Smalltalk is considered as the first truly object-oriented programming language.

OOPs (Object Oriented Programming System)

Object means a real word entity such as pen, chair, table etc. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing some concepts:



- **Object**
- **Class**
- **Inheritance**
- **Polymorphism**

- **Abstraction**
- **Encapsulation**

Object

Any entity that has state and behavior is known as an object. For example: chair, pen, table, keyboard, bike etc. It can be physical and logical.

Class

Collection of objects is called class. It is a logical entity.

A Class in C++ is the foundational element that leads to Object-Oriented programming. A class instance must be created in order to access and use the user-defined data type's data members and member functions. An object's class acts as its blueprint. Take the class of cars as an example. Even if different names and brands may be used for different cars, all of them will have some characteristics in common, such as four wheels, a speed limit, a range of miles, etc. In this case, the class of car is represented by the wheels, the speed limitations, and the mileage.

Inheritance

When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

- 1. Sub class - Subclass or Derived Class refers to a class that receives properties from another class.**
- 2. Super class - The term "Base Class" or "Super Class" refers to the class from which a subclass inherits its properties.**
- 3. Reusability - As a result, when we wish to create a new class, but an existing class already contains some of the code we need, we can generate our new class from the old class thanks to**

inheritance. This allows us to utilize the fields and methods of the pre-existing class.

Polymorphism

When one task is performed by different ways i.e. known as polymorphism. For example: to convince the customer differently, to draw something e.g. shape or rectangle etc.

Different situations may cause an operation to behave differently. The type of data utilized in the operation determines the behavior.

Abstraction

Hiding internal details and showing functionality is known as abstraction. Data abstraction is the process of exposing to the outside world only the information that is absolutely necessary while concealing implementation or background information. For example: phone call, we don't know the internal processing.

In C++, we use abstract class and interface to achieve abstraction.

Encapsulation

Binding (or wrapping) code and data together into a single unit is known as encapsulation. For example: capsule, it is wrapped with different medicines.

Encapsulation is typically understood as the grouping of related pieces of information and data into a single entity. Encapsulation is the process of tying together data and the functions that work with it in object-oriented programming. Take a look at a practical illustration of encapsulation: at a company, there are various divisions, including the sales division, the finance division, and the accounts division. All financial transactions are handled by the finance sector, which also maintains records of all financial data. In a similar vein, the sales section is in charge of all tasks relating to sales and maintains a record of each sale. Now, a scenario could occur when, for some reason, a

financial official requires all the information on sales for a specific month. Under the umbrella term "sales section," all of the employees who can influence the sales section's data are grouped together. Data abstraction or concealing is another side effect of encapsulation. In the same way that encapsulation hides the data. In the aforementioned example, any other area cannot access any of the data from any of the sections, such as sales, finance, or accounts.

Dynamic Binding - In dynamic binding, a decision is made at runtime regarding the code that will be run in response to a function call. For this, C++ supports virtual functions.

Advantage of OOPs over Procedure-oriented programming language

- 1. OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.**
- 2. OOPs provide data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.**
- 3. OOPs provide ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.**

Why do we need oops in C++?

There were various drawbacks to the early methods of programming, as well as poor performance. The approach couldn't effectively address real-world issues since, similar to procedural-oriented programming, you couldn't reuse the code within the program again, there was a difficulty with global data access, and so on.

With the use of classes and objects, object-oriented programming makes code maintenance simple. Because inheritance allows for code reuse, the program is simpler because you don't have to write the same

code repeatedly. Data hiding is also provided by ideas like encapsulation and abstraction.

Why is C++ a partial oop?

The object-oriented features of the C language were the primary motivation behind the construction of the C++ language.

The C++ programming language is categorized as a partial object-oriented programming language despite the fact that it supports OOP concepts, including classes, objects, inheritance, encapsulation, abstraction, and polymorphism.

1) The main function must always be outside the class in C++ and is required. This means that we may do without classes and objects and have a single main function in the application.

It is expressed as an object in this case, which is the first time Pure OOP has been violated.

2) Global variables are a feature of the C++ programming language that can be accessed by any other object within the program and are defined outside of it. Encapsulation is broken here. Even though C++ encourages encapsulation for classes and objects, it ignores it for global variables.

Overloading

Polymorphism also has a subset known as overloading. An existing operator or function is said to be overloaded when it is forced to operate on a new data type.

Conclusion

You will have gained an understanding of the need for object-oriented programming, what C++ OOPs are, and the fundamentals of OOPs, such as polymorphism, inheritance, encapsulation, etc., after reading this course on OOPS Concepts in C++. Along with instances of polymorphism and inheritance, you also learned about the benefits of C++ OOPs.

