Object Oriented Programming in C++

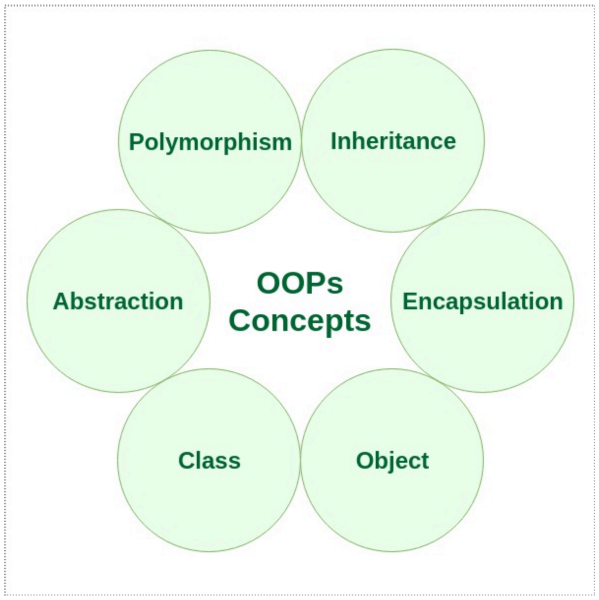
Last Updated : 15 May, 2025

Object Oriented Programming - As the name suggests uses objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc. in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

Characteristics of an Object-Oriented Programming Language

There are some basic concepts that act as the building blocks of OOPs i.e.

Table of Content



Class

The building block of Object-Oriented programming in C++ is a Class. It is a **user-defined data type** that act as a blueprint representing a group of objects which share some common properties and behaviours. These properties are stored as data members, and the behaviour is represented by member functions.

For example, consider the class of **Animals**. An **Animal** class could have common properties like **name**, **age**, and **species** as data members, and behaviors like **eat**, **sleep**, and **makeSound** as member functions. Each individual animal object (such as a dog, cat, or elephant) can be created from this blueprint, and each will have its own unique values for the properties (like the name and age), but all will share the same behaviours defined by the class (like eating, sleeping, and making sounds).

A class is defined using the keyword **class** as shown:

```
class Animal {
public:
    string species;
    int age;
    int name;
```

```
        // sleep for few hrs
    }
    void makeSound () {
        // make sound;
    }
};
```

Here, public is the access specifier that specify what functions are available to others.

Related searches

Java Object Oriented Programming Interview Questions and Answers Pdf

Object

An **Object** is an identifiable actual entity with some characteristics and behaviour. In C++, it is an instance of a class. **For Example**, the Animal class is just a concept or category, not an actual entity. But a black cat named VoidShadowDarkFangReaper is actual animal that exists. Similarly, classes are just the concepts and objects are the actual entity that belongs to that concept.

When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. In the above example, even though we have defined the Animal class, it cannot be used if there are no objects of the class. An object of a class is created as shown:

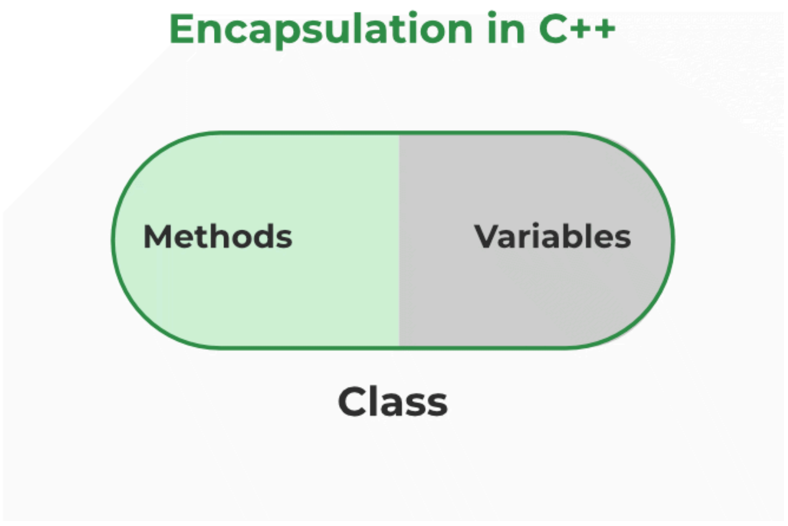
```
#include <iostream>
using namespace std;
```

```
    string species;
    int age;
    int name;

    // Member functions
    void eat() {
        // eat something
    }
    void sleep() {
        // sleep for few hrs
    }
    void makeSound () {
        // make sound;
    }
};

int main() {
    Animal VoidShadowDarkFangReaper;
}
```

Objects take up space in memory and have an associated address like a record in pascal or structure or union. When a program is executed, the objects interact by sending messages to one another. Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and the type of response returned by the objects.

Encapsulation

In simple terms, encapsulation is defined as wrapping up data and information under a single unit. In Object-Oriented Programming, encapsulation is defined as binding together the data and the functions that manipulate them together in a class. Consider an example of the Animal class, the data members species, age and name are encapsulated with the member functions like eat(), sleep, etc. They can be protected by



Abstraction

Abstraction is one of the most essential and important features of object-oriented programming in C++. Abstraction means displaying only essential information and ignoring the other details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. In our case, when we call the **`makeSound()`** method, we don't need to know how the sound is produced internally, only that the method makes the animal sound.

Polymorphism

The word polymorphism means having **many forms**. In simple words, we can define polymorphism as the ability of an entity to behave different in different scenarios. person at the same time can have different characteristics.

For example, the same **makeSound()** method, the output will vary depending on the type of animal. So, this is an example of polymorphism where the makeSound() method behaves differently depending on the Animal type (Dog or Cat).

In C++, polymorphism can be of three types:

- **Function Overloading**: Function overloading is using a single function name to perform different types of tasks.
- **Function Overriding:** Function overriding is changing the behaviour of a function that is derived from the base class using inheritance.

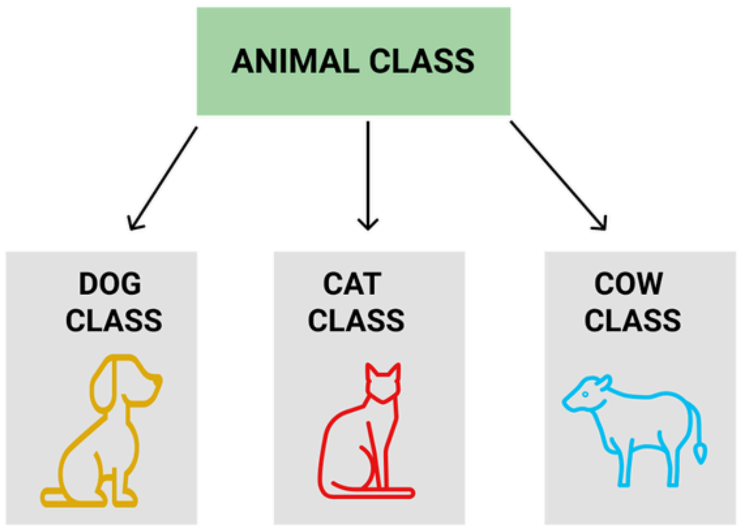
Inheritance

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.

- **Sub Class**: The class that inherits properties from another class is called Sub class or Derived Class.
- **Super Class**: The class whose properties are inherited by a sub-class is called Base Class or Superclass.

Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Example: Dog, Cat, Cow can be Derived Class of Animal Base Class.



Inheritance in C++

Here are key advantages of Object-Oriented Programming (OOP) over Procedure-Oriented Programming (POP):

- **Modularity and Reusability**: OOP promotes modularity through classes and objects, allowing for code reusability.
- **Data Encapsulation**: OOP encapsulates data within objects, enhancing data security and integrity.
- **Inheritance**: OOP supports inheritance, reducing redundancy by reusing existing code.
- **Polymorphism**: OOP allows polymorphism, enabling flexible and dynamic code through method overriding.
- **Abstraction**: OOP enables abstraction, hiding complex details and exposing only essential features

Object Oriented Programming | Video

Visit Course

Comment More info

Campus Training Program

Next Article

C++ Classes and Objects

2. Introduction to C++ Programming Language
3. C++ Programming Language
4. C++ Programming Examples
5. C++ Tutorial | Learn C++ Programming
6. C++ Programming Basics
7. Reference to Dynamic Objects in C++
8. Naming Convention in C++
9. Printing the Address of an Object of Class in C++
10. Output of C++ Program | Set 19

Campus Training Program

Tutorials

Python
Java
C++
PHP
GoLang
SQL
R Language
Android

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
DSA Interview Questions
Competitive Programming

Data Science & ML

Data Science With Python
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
NodeJs
Bootstrap
Tailwind CSS

Python Tutorial

Python Examples
Django Tutorial
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question

Computer Science

GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Databases