**Unit-1**

**Introduction to programming concept**

The actual text used to write the instructions for a computer **program**. ... This text is then translated into something meaningful the computer can understand. and Compiler which is a software tool that translates source code into duter**g** is writing computer code to create a **program**, in order to solve a problem. Programs consist of a series of instructions to tell a computer exactly what to do and how to do it.

A **program** is a set of instructions given to a **computer** to perform a specific operation. or **computer** is a computational device which is used to process the data under the control of a **computer program**. ... These **computer** programs are written in a **programming** language which are high level languages.

Programming Methodologies

The approach to analyzing such complex problems, planning for software development and controlling the development process is called **programming methodology**.

### Procedural Programming

Problem is broken down into procedures, or blocks of code that perform one task each. All procedures taken together form the whole program. It is suitable only for small programs that have low level of complexity.

**Example** − For a calculator program that does addition, subtraction, multiplication, division, square root and comparison, each of these operations can be developed as separate procedures. In the main program each procedure would be invoked on the basis of user’s choice.

### Object-oriented Programming

Here the solution revolves around entities or objects that are part of problem. The solution deals with how to store data related to the entities, how the entities behave and how they interact with each other to give a cohesive solution.

**Example** − If we have to develop a payroll management system, we will have entities like employees, salary structure, leave rules, etc. around which the solution must be built.

### Functional Programming

Here the problem, or the desired solution, is broken down into functional units. Each unit performs its own task and is self-sufficient. These units are then stitched together to form the complete solution.

**Example** − A payroll processing can have functional units like employee data maintenance, basic salary calculation, gross salary calculation, leave processing, loan repayment processing, etc.( Lisp, Python, Erlang, Haskell, Clojure, etc)

### Logical Programming

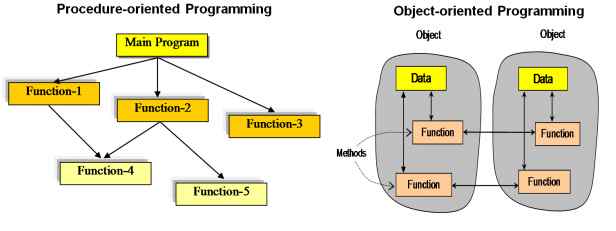
Here the problem is broken down into logical units rather than functional units. **Example:** In a school management system, users have very defined roles like class teacher, subject teacher, lab assistant, coordinator, academic in-charge, etc. So the software can be divided into units depending on user roles. Each user can have different interface, permissions, etc.( Prolog, answer set **programming** (ASP) and Datalog.)

**What is procedural programming with example?**

**Procedural Programming** is the use of code in a step-wise **procedure** to develop applications. For **example**, to develop a simple Bank Account App procedurally: Creating an account for an individual ( account ) Getting an account to deposit or withdraw funds ( getAccount , deposit , withdraw )

**What is OOPs with example?**

**Object Oriented programming** (**OOP**) is a programming paradigm that relies on the concept of classes and objects. ... These functions are defined within the class and perform some action helpful to that specific type of object. For **example**, our Car class may have a method repaint that changes the color attribute of our car.

**Difference between Procedural Programming and Object Oriented Programming:**

| Procedural Oriented Programming | Object Oriented Programming |
| --- | --- |
| In procedural programming, program is divided into small parts called ***functions***. | In object oriented programming, program is divided into small parts called ***objects***. |
| Procedural programming follows ***top down approach***. | Object oriented programming follows ***bottom up approach***. |
| There is no access specifier in procedural programming. | Object oriented programming have access specifiers like private, public, protected etc. |
| Adding new data and function is not easy. | Adding new data and function is easy. |
| Procedural programming does not have any proper way for hiding data so it is ***less secure***. | Object oriented programming provides data hiding so it is ***more secure***. |
| In procedural programming, overloading is not possible. | Overloading is possible in object oriented programming. |
| In procedural programming, function is more important than data. | In object oriented programming, data is more important than function. |
| Procedural programming is based on ***unreal world***. | Object oriented programming is based on ***real world***. |
| Examples: C, FORTRAN, Pascal, Basic etc. | Examples: C++, Java, Python, C# etc. |

**Characteristics of an Object Oriented Programming language**

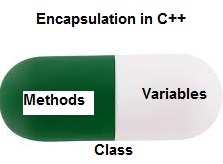
Class: The building block of C++ that leads to Object-Oriented programming is a Class. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

For Example: Consider the Class of Cars. There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc. So here, Car is the class and wheels, speed limits, mileage are their properties.

Object: An Object is an identifiable entity with some characteristics and behaviour. An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

Encapsulation: In normal terms, Encapsulation is defined as wrapping up of 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.

Consider a real-life example of encapsulation, in a company, there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial transactions and keeps records of all the data related to finance. Similarly, the sales section handles all the sales-related activities and keeps records of all the sales. Now there may arise a situation when for some reason an official from the finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of the sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data. This is what encapsulation is. Here the data of the sales section and the employees that can manipulate them are wrapped under a single name “sales section”.



Encapsulation also leads to *data abstraction or hiding*. As using encapsulation also hides the data. In the above example, the data of any of the section like sales, finance or accounts are hidden from any other section.

Abstraction: Data abstraction is one of the most essential and important features of object-oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

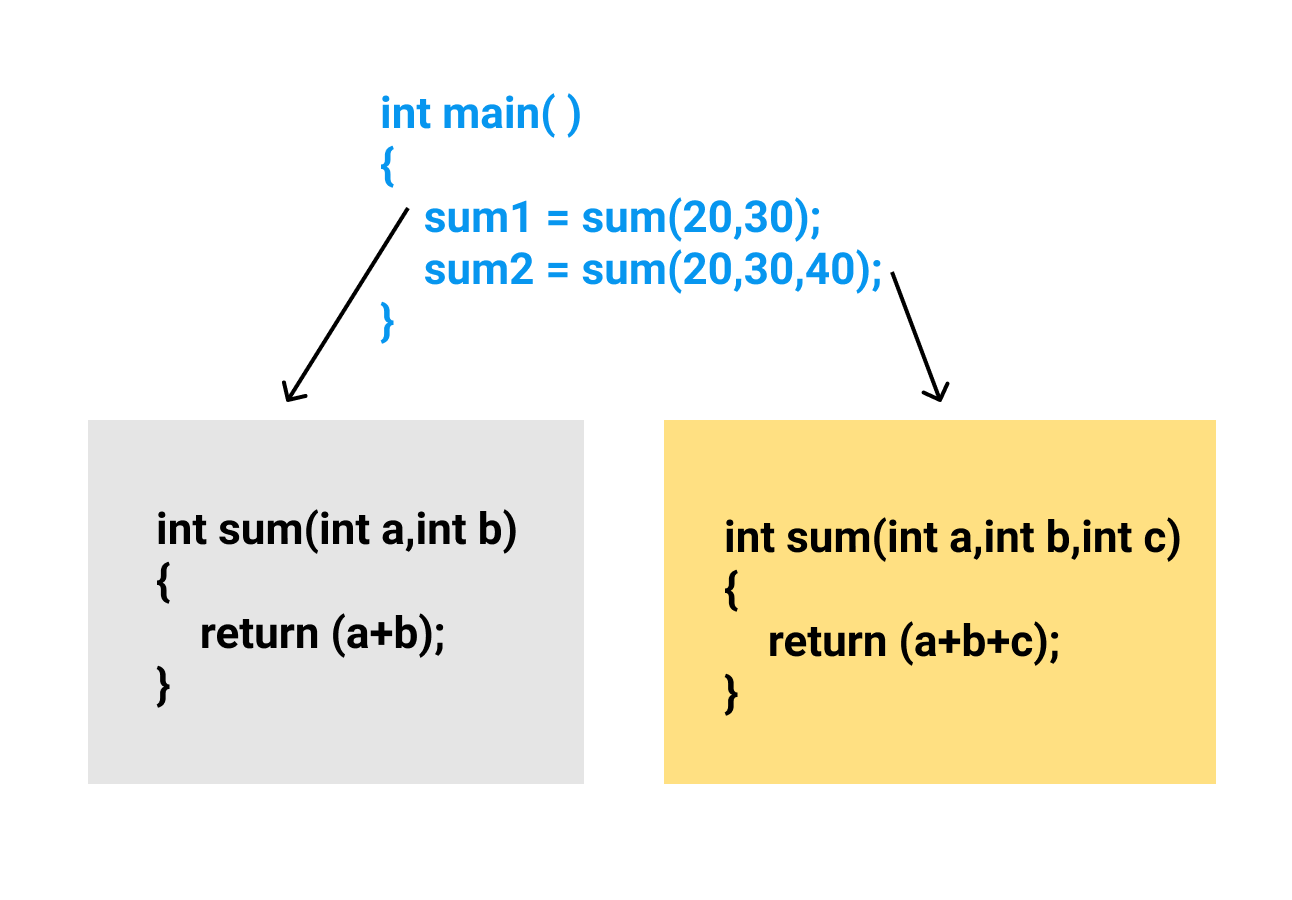
Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car.

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

A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So the same person posses different behaviour in different situations. This is called polymorphism.

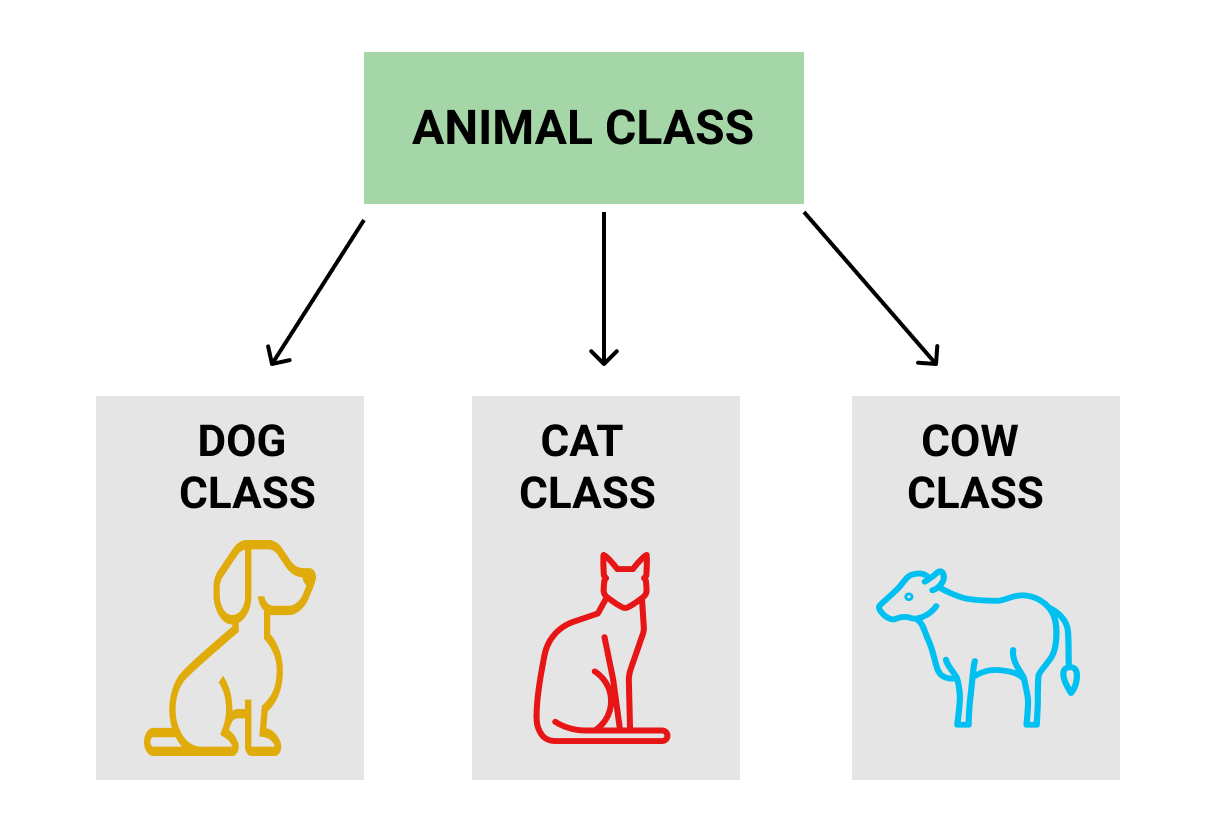
.

.



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 sub class is called Base Class or Super class.
* **Reusability**: 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.  


**Dynamic Binding:** In dynamic binding, the code to be executed in response to function call is decided at runtime. C++ has virtual functions to support this.

**Message Passing:** Objects communicate with one another by sending and receiving information to each other. A message for an object is a request for execution of a procedure and therefore will invoke a function in the receiving object that generates the desired results. Message passing involves specifying the name of the object, the name of the function and the information to be sent.

**OOP advantages**

1- **Improved software development productivity:** OO programming is modular, as it provides separation of duties in object-based program development. It is also extensible, as objects can be extended to include new attributes and behaviors. Objects can also be reused within an across applications. Because of these three factors; extensibility, modularity, and reusability.

2- **Faster development:** Reuse enables faster development. Object-oriented programming languages come with rich libraries of objects, and code developed during projects is also reusable in future projects.

3- **Lower cost of development:** The reuse of software also lowers the cost of development. Typically, more effort is put into the OO analysis and design, which lowers the overall cost of development.

4- **Higher-quality software:** Faster development of software and lower cost of development allows more time and resources to be used in the verification of the software. Although quality is dependent upon the experience of the teams, OO programming tends to result in higher-quality software

**OOP disadvantages** are:

1- **Steep learning curve:** The thought process involved in OO programming may not be natural for some people, and it will take the time to get used to it.

2- **The complexity of creating programs:**it is very complex to create programs based on the interaction of objects. Some of the key programming techniques, such as inheritance and polymorphism, can be a big challenging to comprehend initially.