OOPS AND STRUCTURAL PROGRAMMING LANGUAGES

Procedural Programming

Procedural Programming can be defined as a programming model which is derived from structured programming, based upon the concept of calling procedure. Procedures, also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out. During a program's execution, any given procedure might be called at any point, including by other procedures or itself.

Languages used in Procedural Programming:

FORTRAN, ALGOL, COBOL,

BASIC, Pascal and C.

Object-Oriented Programming

Object-oriented programming can be defined as a programming model which is based upon the concept of objects. Objects contain data in the form of attributes and code in the form of methods. In object-oriented programming, computer programs are designed using the concept of objects that interact with the real world. Object-oriented programming languages are various but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

Languages used in Object-Oriented Programming:

Java, C++, C#, Python,

PHP, JavaScript, Ruby, Perl,

Objective-C, Dart, Swift, Scala.

Procedural Programming vs Object-Oriented Programming

Below are some of the differences between procedural and object-oriented programming:

Procedural Oriented Programming	Object-Oriented Programming
In procedural programming, the program is divided into small parts called **functions**.	In object-oriented programming, the program is divided into small parts called **objects**.
Procedural programming follows a **top-down approach**.	Object-oriented programming follows a **bottom-up approach**.
There is no access specifier in procedural programming.	Object-oriented programming has access specifiers like private, public, protected, etc.
Adding new data and functions is not easy.	Adding new data and function is easy.
Procedural programming does not have any proper way of 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, there is no concept of data hiding and inheritance.	In object-oriented programming, the concept of data hiding and inheritance is used.
In procedural programming, the function is more important than the data.	In object-oriented programming, data is more important than function.
Procedural programming is based on the **unreal world**.	Object-oriented programming is based on the **real world**.

Procedural programming is used for designing medium-sized programs.	Object-oriented programming is used for designing large and complex programs.
Procedural programming uses the concept of procedure abstraction.	Object-oriented programming uses the concept of data abstraction.
Examples: C, FORTRAN, Pascal, Basic, etc.	**Examples:** C++, Java, Python, C#, etc.

What are examples of object-oriented programming languages?

While Simula is credited as being the first object-oriented programming language, many other programming languages are used with OOP today. But some programming languages pair with OOP better than others. For example, programming languages considered pure OOP languages treat everything as objects. Other programming languages are designed primarily for OOP, but with some procedural processes included.

For example, popular pure OOP languages include:

- Ruby
- Scala
- JADE
- Emerald

Programming languages designed primarily for OOP include:

- Java
- Python
- C++

Other programming languages that pair with OOP include:

- Visual Basic .NET
- PHP
- JavaScript

What are the benefits of OOP?

Benefits of OOP include:

- **Modularity.** Encapsulation enables objects to be self-contained, making troubleshooting and collaborative development easier.

- **Reusability.** Code can be reused through inheritance, meaning a team does not have to write the same code multiple times.
- **Productivity.** Programmers can construct new programs quicker through the use of multiple libraries and reusable code.
- **Easily upgradable and scalable.** Programmers can implement system functionalities independently.
- **Interface descriptions.** Descriptions of external systems are simple, due to message passing techniques that are used for objects communication.
- **Security.** Using encapsulation and abstraction, complex code is hidden, software maintenance is easier and internet protocols are protected.
- **Flexibility.** Polymorphism enables a single function to adapt to the class it is placed in. Different objects can also pass through the same interface.

Criticism of OOP

The object-oriented programming model has been criticized by developers for multiple reasons. The largest concern is that OOP overemphasizes the data component of software development and does not focus enough on computation or algorithms. Additionally, OOP code may be more complicated to write and take longer to compile.

Alternative methods to OOP include:

- **Functional programming.** This includes languages such as Erlang and Scala, which are used for telecommunications and fault tolerant systems.
- **Structured or modular programming.** This includes languages such as PHP and C#.
- **Imperative programming.** This alternative to OOP focuses on function rather than models and includes C++ and Java.
- **Declarative programming.** This programming method involves statements on what the task or desired outcome is but not how to achieve it. Languages include Prolog and Lisp.
- **Logical programming.** This method, which is based mostly in formal logic and uses languages such as Prolog, contains a set of sentences that express facts or rules about a problem domain. It focuses on tasks that can benefit from rule-based logical queries.