**Paradigmas de Programación**

**1. ¿Qué es un Paradigma de Programación?**

Un **paradigma de programación** es un estilo o enfoque que define cómo se deben estructurar y ejecutar los programas. Determina la forma en que los desarrolladores abordan la resolución de problemas y organizan el código.

Existen distintos paradigmas de programación, cada uno con su propio conjunto de principios, ventajas y casos de uso. Algunos lenguajes de programación combinan múltiples paradigmas para ofrecer mayor flexibilidad.

**2. Principales Paradigmas de Programación y Ejemplos**

**🔹 Paradigma Imperativo**

El paradigma imperativo se basa en la ejecución de instrucciones secuenciales que modifican el estado del programa. Su enfoque está en describir cómo se debe realizar una tarea mediante comandos y estructuras de control como bucles y condicionales.

**Ejemplos de lenguajes imperativos:**

* **C**
* **Pascal**
* **Fortran**
* **BASIC**

**🔹 Paradigma Orientado a Objetos (POO)**

El paradigma de programación orientado a objetos se basa en la creación de **objetos** que encapsulan datos y comportamientos dentro de estructuras llamadas **clases**. Este paradigma favorece la reutilización de código y la organización modular.

**Ejemplos de lenguajes orientados a objetos:**

* **Java**
* **C++**
* **Python**
* **C#**
* **Ruby**
* **Swift**
* **Kotlin**

**🔹 Paradigma Funcional**

El paradigma funcional se basa en funciones puras, sin modificar el estado global del programa ni usar variables mutables. Fomenta el uso de funciones de orden superior y evita efectos secundarios.

**Ejemplos de lenguajes funcionales:**

* **Haskell**
* **Lisp**
* **Erlang**
* **Scala** (tambien admite POO)
* **Elixir**
* **F#**

**🔹 Paradigma Lógico**

El paradigma lógico se basa en el uso de reglas y hechos para resolver problemas mediante inferencias lógicas. En este enfoque, el programador define el problema en términos de relaciones, y el motor de inferencia del lenguaje deduce la solución.

**Ejemplos de lenguajes lógicos:**

* **Prolog**
* **Mercury**
* **Datalog**

**🔹 Paradigma Declarativo**

El paradigma declarativo se centra en especificar **qué** debe hacerse en lugar de **cómo** hacerlo. Se utiliza en bases de datos, consultas y lenguajes de definición de estructuras.

**Ejemplos de lenguajes declarativos:**

* **SQL** (Lenguaje de consulta estructurado para bases de datos)
* **XQuery** (Lenguaje para consultar datos XML)
* **HTML + CSS** (Definen estructuras y estilos en la web, aunque no son lenguajes de programación en sí)

**3. Otros Paradigmas y Enfoques Modernos**

**🔹 Paradigma Reactivo y Concurrente**

Este paradigma se centra en el procesamiento en paralelo y la respuesta a eventos asincrónicos, lo que lo hace ideal para sistemas de alta disponibilidad y aplicaciones modernas.

**Ejemplos de lenguajes con soporte reactivo/concurrente:**

* **JavaScript** (con Node.js para programación asíncrona)
* **RxJS** (Extensión reactiva de JavaScript)
* **Erlang**
* **Elixir**
* **Go** (con concurrencia nativa mediante goroutines)

**4. Conclusión**

✅ No hay un paradigma mejor que otro; cada uno tiene ventajas y desventajas dependiendo del tipo de problema que se quiera resolver.

✅ Algunos lenguajes modernos, como **Python, JavaScript y Scala**, combinan varios paradigmas (imperativo, orientado a objetos y funcional) para ofrecer mayor flexibilidad.

✅ Elegir el paradigma adecuado depende del contexto del desarrollo, la naturaleza del problema y las preferencias del equipo de programación.