# Paradigma de programación

El paradigma de programación es el estilo de programación empleado. Algunos lenguajes soportan varios paradigmas, y otros sólo uno.

- Lenguajes de programación **estructurado**: Divide el problema en partes más pequeñas, que serán realizadas por subprogramas (subrutinas, funciones, procedimientos), que se llaman unas a otras para ser ejecutadas. Ejemplos: C, Pascal.
- Lenguajes de programación **orientada a objetos**: Crean un sistema de clases y objetos siguiendo el ejemplo del mundo real, en el que unos objetos realizan acciones y se comunican con otros objetos. Ejemplos: C++, `Java`.
- Lenguajes de programación **funcional**: La tarea se realiza evaluando funciones, (como en Matemáticas), de manera recursiva. Ejemplo: Lisp, Scala.
- Lenguajes de programación **lógica**: La tarea a realizar se expresa empleando lógica formal matemática. Expresa qué computar. Ejemplo: Prolog.

## Lenguajes que soportan un paradigma

Algunos lenguajes han sido diseñados para soportar un único paradigma de programación

Smalltalk que soporta únicamente la programación orientada a objetos

`Haskell` que solo soporta la programación funcional

Algunos paradigmas prohíben el uso de ciertos mecanismos o técnicas.

En la programación funcional se elimina el uso del efecto secundario en las funciones

En la programación estructurada se desaprueba o incluso elimina el uso de la sentencia goto

## Lenguajes que soporten múltiples paradigmas de programación

Estos lenguajes son aquellos que soportan al menos dos paradigmas.

Dentro de esta categoría podemos encontrar nuevas caras y viejos conocidos:

- Scala: Imperativo, orientado a objetos, funcional, genérico y concurrente
- Erlang: Funcional, concurrente y distribuido
- Perl: Imperativo, orientado a objetos y funcional
- `PHP`: Imperativo, orientado a objetos, funcional y reflexivo
- `Java`Script: Imperativo, orientado a objetos (prototipos) y funcional
- `Java`: Imperativo, orientado a objetos, reflexivo y genérico
- Python y Ruby: Imperativo, orientado a objetos, reflexivo y funcional
- C++: Imperativo, orientado a objetos, funcional y genérico
- C#: Imperativo, orientado a objetos, funcional (lambda), reflexivo y genérico

## Lenguajes que soportan muchos paradigmas de programación

Existen lenguajes como `Oz` que soporta nueve paradigmas de programación

Para algunos es un acierto y para otros un error

## Programación Imperativa

En la programación **imperativa** se describen sentencias que modifican el estado de un programa.

En muchos sentidos la programación imperativa es la programación natural para las CPUs que se basan en ese paradigma al nivel más básico.

En este paradigma se expresa como debe solucionarse un problema especificando una secuencia de acciones a realizar a través de uno o más procedimientos denominados subrutinas o funciones.

Dentro de esta categoría se engloban:

- La programación estructurada que restringe el uso de la instrucción goto
- La programación modular
- La programación orientada a solución

## Programación declarativa

La solución es alcanzada a través de mecanismos internos de control.

No se especifica exactamente como llegar a ella.

Las variables son utilizadas con transparencia referencial, es decir una expresión puede ser sustituida por el resultado de ser evaluada en el programa sin alterarlo semánticamente.

Dentro de esta categoría se engloban

- La programación `funcional` cuyo lenguaje más expresivo y culmen sea seguramente el lenguaje `Haskell`.
- La programación `lógica` donde sin duda el campeón es Prolog (ampliamente usado en ambientes académicos)
- La programación `restringida` o con restricciones entre otras.

En los lenguajes funcionales puros como `Haskell`, todas las funciones son puras, es decir, no tienen efectos secundarios, y los cambios de estado están solo representados como funciones que transforman el estado. Aunque no son imperativos, por norma general proporcionan algún mecanismo por el que describir el efecto de una función como una serie de pasos.

## Diferencias principales

La principal diferencia entre ambos paradigmas es que en la programación imperativa se describe paso a paso un conjunto de instrucciones que han de ejecutarse con la finalidad de variar el estado del programa y resolver un problema para hallar una solución. Es decir, se describe un algoritmo en el que se detallan los pasos secuenciales necesarios a seguir para la resolución de un problema.

Y en la programación declarativa solo se describe el problema pero no los pasos necesarios para llegar a su solución, la cual es hallada mediante mecanismos internos de inferencia de información a partir de la descripción del problema en si. La programación imperativa se basa en la máquina de Turinq mientras que la programación declarativa se basa en el cálculo lambda.

¿Cómo diferenciar una de la otra?

Podemos diferenciarlas sin temor a equivocamos en los siguientes supuestos:

- Un programa que describe que problemas deben resolverse pero no como, está programado con programación declarativa.
- Cualquier programa que evita los efectos secundarios o es referencialmente transparente, está programado con programación declarativa
- Todo lo que no cuadre con las dos afirmaciones anteriores usa programación imperativa o una mezcla de ambas