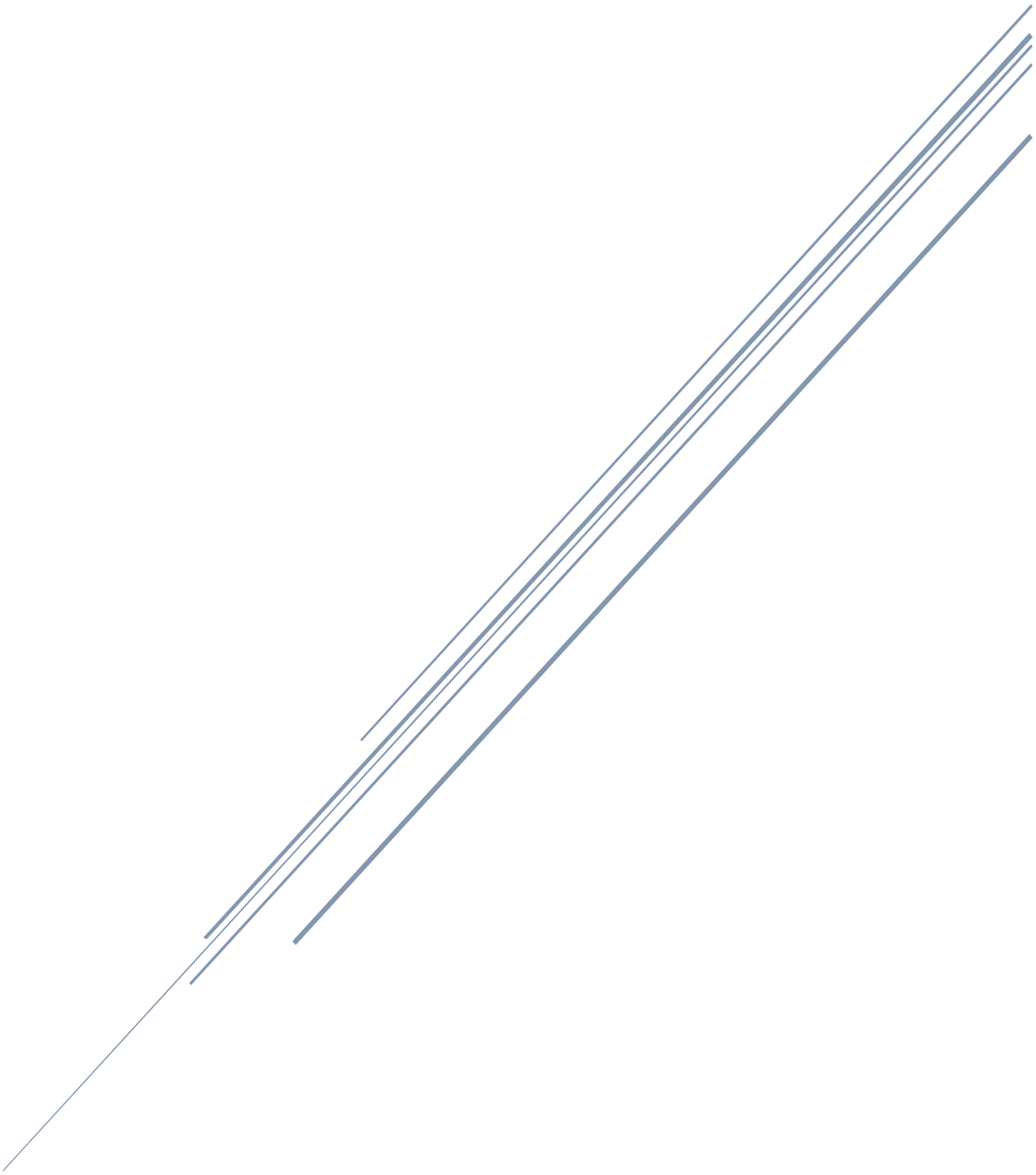
PARADIGMAS DE PROGRAMACIÓN

Declarativo VS Imperativo



2º DAM
Álvaro Bellón Lanz

ÍNDICE

Introducción

La programación, como disciplina en constante evolución, presenta una gran variedad de paradigmas que definen la manera en que los desarrolladores conciben, estructuran y resuelven problemas a través del código. Entre estos paradigmas, dos enfoques fundamentales, el declarativo y el imperativo, son los pilares conceptuales que guían la construcción de software.

¿Qué es un paradigma de programación?

Es un enfoque o modelo conceptual que define la forma en que se estructuran, diseñan y llevan a cabo los programas. Cada paradigma proporciona un conjunto de reglas y principios que guían la escritura de código, la organización del software y la solución de problemas. Los paradigmas de programación influyen en cómo se abordan y resuelven los problemas en el desarrollo de software.

Cuando estás desarrollando software, eliges un paradigma basado en el tipo de problema que estás resolviendo y en tus preferencias personales. Cada paradigma tiene sus propias ventajas y desventajas.

Imagina que estás construyendo una casa. Cada paradigma de programación sería como un conjunto de instrucciones sobre cómo deberías planificar, diseñar y construir esa casa. Algunos paradigmas podrían decirte que uses ciertos materiales o que organices las habitaciones de una manera particular.

Paradigma de programación declarativo

En el mundo de la programación, el paradigma declarativo se presenta como una forma de abordar la creación de software desde una perspectiva más orientada a los resultados deseados y no a los pasos específicos para lograrlos.

Piensa en que le tienes que explicar a un cocinero cómo hacer una tortilla de patata, en vez de explicarle cómo debe cortar la patata o cuánto tiempo debe estar friéndola, nos centraremos en explicarle cómo queremos que quede el plato final. Es decir, el paradigma declarativo se centra en definir los resultados que queremos obtener y no en los pasos que debemos hacer para llegar hasta ellos.

Estas son las características más importantes del paradigma declarativo:

- *La expresión del "Qué":* es algo básico para enfocar este paradigma. Significa que nos enfocaremos en describir el resultado que queremos obtener, sin entrar en detalles de cómo vamos a llegar a ello. Esta característica contribuye a un código más claro, conciso y comprensible.

- *La inmutabilidad:* en este contexto hablaremos de inmutabilidad refiriéndonos a que los datos no cambian una vez hayan sido creados. Es una característica fundamental para evitar errores. Cuando los datos son inmutables, significa que no van a cambiar durante la ejecución del programa. Esta característica contribuye a minimizar la capacidad de

errores producidos por modificaciones inadvertidas y a la creación de programas más seguros.

- *La evaluación perezosa:* es una estrategia en la que los valores se calculan solo cuando son necesarios. En lugar de realizar todos los cálculos de inmediato, el sistema pospone la evaluación hasta el último momento posible. Esto puede mejorar la eficiencia al evitar el cálculo innecesario de valores que podrían no ser utilizados. Esta característica permite un uso más eficiente de los recursos al calcular solo lo que se requiere para obtener el resultado final.

- *La programación funcional:* se basa en el concepto de funciones puras, que son funciones que no tienen efectos secundarios y producen resultados basados únicamente en sus entradas. Es decir, que una función siempre dará el mismo resultado para los mismos datos de entrada, lo que conduce a un código fácil de mantener.

Algunos ejemplos de lenguajes de programación con paradigma declarativo son Haskell o Lisp, aunque también podemos destacar SQL, el cual ya conocemos y que sirve para realizar consultas en bases de datos. Con este lenguaje especificamos qué datos queremos que nos muestre y el sistema de la base de datos se encarga de cómo ejecutar de manera eficiente las operaciones necesarias.

Paradigma de programación imperativo

El paradigma imperativo representa una forma de pensar en la construcción de software que se asemeja a dar instrucciones específicas paso a paso. Este paradigma se centra en la idea de que un programa es esencialmente una serie de instrucciones que se ejecutan secuencialmente.

Al contrario que en el paradigma declarativo, en este caso le daremos al cocinero todos los pasos de manera concisa y ordenada para cocinar la tortilla. En el caso anterior le decíamos qué resultado queríamos, en este nos centraremos en especificarle cómo tiene que cortar las patatas o durante cuánto tiempo debe estar friendo la tortilla.

Estas son las características más importantes del paradigma imperativo:

- *La expresión del "Cómo":* el paradigma imperativo se acentúa en expresar el "cómo" se deben llevar a cabo las operaciones. En lugar de simplemente indicar el resultado deseado, nos centramos en los detalles prácticos de cada paso necesario para lograr ese resultado. Ayuda a que el código tenga un mejor control de errores y mantenimiento.

- *Las variables y estados mutables:* una característica clave del paradigma imperativo es el uso de variables y la capacidad de cambiar el estado del programa durante la ejecución. Pensamos en un variable como un cajón en el que almacenamos información, y durante la ejecución del programa podemos abrir el cajón, manipular la información que hay dentro y cerrar el cajón. Permite al programa adaptarse de forma dinámica durante la ejecución.

- *Las instrucciones explícitas:* las operaciones se expresan a través de instrucciones explícitas, las cuales indican a la máquina qué hacer, cómo hacerlo y en qué secuencia. Esto podría incluir acciones como asignar valores a variables, realizar cálculos específicos, o utilizar estructuras de control de flujo como bucles y condicionales para dirigir el flujo de ejecución. Las instrucciones explícitas hacen que el código sea mucho más claro y se tenga un control más detallado de él.

- *Programación Procedural y Orientada a objetos:* dentro del paradigma imperativo, se destacan dos enfoques principales: la programación procedural y la programación orientada a objetos.

  I. *Procedural:* organizamos tareas en procedimientos o funciones, dividiendo el programa en bloques de código con propósitos específicos.
  II. *Orientado a objetos:* en la programación orientada a objetos, nos centramos en la manipulación de objetos que encapsulan datos y comportamientos relacionados, facilitando la organización y reutilización del código.

Algún ejemplo de ejemplos de lenguajes de programación son Java o Python, entre otros. Estos lenguajes nos permiten estructurar nuestro código como una serie de instrucciones detalladas que la máquina ejecuta secuencialmente. Además, en la programación orientada a objetos, se manipulan objetos que contienen datos y comportamientos, lo que añade una capa adicional de organización y abstracción al código imperativo.

VENTAJAS Y DESVENTAJAS

Paradigma declarativo

El paradigma declarativo presenta ventajas y desventajas, y su idoneidad depende en gran medida del contexto y de los objetivos específicos del desarrollo de software. Estas son algunas ventajas:

- *Abstracción:* la programación declarativa permite una mayor abstracción, donde el programador se enfoca en describir qué resultado se desea en lugar de cómo lograrlo.

- *Facilita el Paralelismo:* dado que el énfasis está en expresar relaciones y resultados, en lugar de pasos detallados, los programas declarativos pueden ser más propensos al paralelismo. Esto facilita la ejecución de operaciones simultáneas, mejorando el rendimiento en sistemas multicore/multinúcleo.

- *Mejora la Mantenibilidad:* la abstracción y la claridad en la expresión del código hacen que sea más fácil mantener y actualizar programas declarativos.

Estas son algunas desventajas:

- *Menor Control Detallado:* la falta de detalles sobre cómo se deben llevar a cabo las operaciones puede resultar en un menor control detallado sobre el flujo de ejecución y

la gestión de recursos. Esto puede ser un inconveniente en situaciones que requieren una optimización precisa.

- *Posible Pérdida de Eficiencia:* la abstracción adicional puede llevar a una pérdida de eficiencia en comparación con soluciones imperativas altamente optimizadas.

- *Aprendizaje Inicial Más Pronunciado:* puede haber una curva de aprendizaje inicial más pronunciada para programadores acostumbrados al paradigma imperativo. La transición de pensar en términos de "cómo" a "qué" puede requerir un cambio de mentalidad.

Paradigma imperativo

El paradigma imperativo también tiene sus ventajas y desventajas, y su elección depende de diversos factores, como la naturaleza del problema, la eficiencia requerida y la preferencia del programador. Estas son algunas ventajas:

- *Control Detallado:* el paradigma imperativo proporciona un control detallado sobre el flujo de ejecución y el estado del programa. Los programadores pueden especificar de manera precisa cómo se deben realizar las operaciones y cómo se deben manipular las variables.

- *Adaptabilidad Dinámica:* la capacidad de cambiar el estado de las variables durante la ejecución permite una adaptabilidad dinámica del programa. Puedes ajustar el comportamiento del programa en respuesta a condiciones cambiantes.

- *Aprendizaje Más Sencillo:* es un modelo fácilmente comprensible para los principiantes.

Estas son algunas desventajas:

- *Mayor Propensión a Errores:* la manipulación directa de variables y el control detallado pueden aumentar la propensión a errores, especialmente en programas grandes y complejos.

- *Dificultad en Paralelismo:* la estructura imperativa puede dificultar la ejecución eficiente en paralelo, ya que el control detallado puede generar dependencias entre las operaciones.

- *Menor Legibilidad en Algunos Casos:* en comparación con el paradigma declarativo, el código imperativo puede ser menos legible para aquellos que no están familiarizados con los detalles de implementación.

CONCLUSIÓN

En conclusión, los paradigmas de programación, declarativo e imperativo, se distinguen principalmente por su enfoque hacia el "qué" y el "cómo" en la resolución de problemas. El

paradigma declarativo se centra en expresar qué resultado se busca, fomentando la abstracción y la claridad al permitir que el sistema determine cómo alcanzar ese resultado. Por otro lado, el paradigma imperativo se inclina hacia la especificación detallada de cómo se deben llevar a cabo las operaciones, otorgando al programador un control preciso sobre el flujo de ejecución y el estado del programa. La elección entre ambos paradigmas depende de las necesidades específicas del proyecto, con el paradigma declarativo destacando en situaciones donde la claridad y la abstracción son cruciales, mientras que el paradigma imperativo sobresale en contextos que requieren un control detallado y una optimización específica.