Paradigmas de Programación

Paradigma

Tomando las definiciones clásicas del concepto paradigma, encontramos lo siguiente. Según la RAE (Real Academia Española) un paradigma es *"Ejemplo o ejemplar"*; y tiene una segunda definición: *"Teoría o conjunto de teorías cuyo núcleo central se acepta sin cuestionar y que suministra la base y modelo para resolver problemas y avanzar en el conocimiento"*.

Según el diccionario de Oxford es: *"A pattern or model, an exemplar; (also) a typical instance of something, an example"*, traducido al español es: *"Un modelo o patrón, un ejemplo; (también) una representación típica de algo, un ejemplo"*.

En la aplicación científica, el concepto de paradigma fue enunciado por Thomas Khun en el libro "La estructura de las revoluciones científicas", publicado en el año 1962, refiriéndose a la visión del mundo que poseen los científicos en un determinado momento histórico.

Haciendo una síntesis de las definiciones dadas, podemos decir que un paradigma determina una manera especial de entender el mundo, explicarlo y manipularlo.

Paradigma de programación

Un Paradigma de Programación es un enfoque particular o filosofía para la construcción del software. Se puede entender como una cultura sobre cómo programar.

Un paradigma tiene una filosofía y un conjunto de fundamentos teóricos que lo definen. Los lenguajes de programación generalmente se clasifican dentro de algún paradigma fundamental aunque también pueden tomar aspectos de otros paradigmas. Esta cultura de programación agrupa estilos, técnicas, sintaxis comunes, pero, sobre todo, generan una filosofía de pensamiento sobre cómo entender las situaciones del mundo real y, por ende, sobre cómo definir el modelo para luego describirlo mediante un lenguaje de programación que sustente dicha filosofía y que pueda construir el modelo planteado en un soporte computacional.

Existen varios paradigmas de programación y cada uno tiene sus características que los distinguen. En este curso de paradigmas de programación veremos el paradigma orientado a objetos, el paradigma funcional y el paradigma lógico.

Programación

La programación es el acto que realiza una persona para programar una computadora. Este acto de programar incluye la tarea de comprender algún fenómeno o situación del mundo real, armar un modelo del mismo (por medio de un proceso de abstracción) y, luego, construir y recrear dicho modelo en una computadora. Dicha construcción se realiza mediante un lenguaje de programación.

La actividad de programar es una actividad beneficiosa para el ser humano y para la comunidad. Varios autores señalan la importancia de aprender lenguajes de programación, ya que traen tres consecuencias beneficiosas para la mente humana:

1. La programación activa centros de aprendizaje del cerebro.
2. Cambia la forma de pensar.
3. Da impulso a la memoria y a las capacidades cognitivas de la persona.

Podemos tomar las palabras de Steve Jobs, creador de Apple, quien dijo: *"Programming teaches you how to think"* , en español *"la programación te enseña cómo pensar"*. En la misma línea se inscribe la opinión de Alan Perlis, creador del lenguaje Algol, quien dijo: *"learning a programming language can change the way a person thinks"*, en español *"aprender un lenguaje de programación puede cambiar la forma en que una persona piensa"* . Y fue por más diciendo: *"a language that doesn't affect the way you think about programming, is not worth knowing"* , traducido, *"No vale la pena conocer un lenguaje que no afecta la forma en que piensas sobre la Programación"*.

Si bien son opiniones personales, el espíritu de las mismas se basa en que el acto de programar obliga a una persona a pensar cómo resolver un problema de una manera que pueda ser explicado a una computadora y, para ello, intervienen elementos tales como: descomponer el problema en subproblemas, realizar abstracciones, priorizaciones, se debe ser preciso, se debe tener un conocimiento completo de todos los elementos que forman parte de la solución, entre otros elementos.

Lenguajes de programación

Los lenguajes de programación son básicamente un medio de comunicación entre las personas y las computadoras.

Un lenguaje de programación es un conjunto de reglas y símbolos que permiten escribir instrucciones, pasos, declaraciones, que una computadora puede entender y ejecutar. Estos lenguajes proporcionan una forma de comunicarse con la máquina y expresar algoritmos y procesos de manera particular y comprensible. Los lenguajes de programación varían en su sintaxis, semántica y enfoque (paradigma).

Lenguajes Imperativos

Los lenguajes de programación imperativos son un tipo de lenguaje de programación que se centra en describir cómo lograr un resultado específico mediante la especificación de una secuencia de comandos o instrucciones. Estos lenguajes se

basan en el paradigma de programación imperativa, que se centra en cambiar el estado de la computadora mediante la ejecución de instrucciones.

Los programas imperativos consisten en una secuencia de instrucciones que se ejecutan en un orden determinado. Cada instrucción especifica una acción que debe realizar la computadora. Hacen uso de variables para almacenar y manipular datos. Utilizan estructuras de control como bucles (for, while) y condicionales (if, else) para controlar el flujo de ejecución del programa. Organiza el código modularmente con el uso de procedimientos y funciones. Al ejecutarse un programa y modificarse el estado de las variables se va modificando el estado global del programa, estos cambios de estado son una característica distintiva de los lenguajes imperativos.

Una mención especial merecen los lenguajes orientados a objetos. Los lenguajes orientados a objetos son, en esencia, imperativos, pero con una perspectiva diferente en la forma en que organizan y estructuran el código. La orientación a objetos es un paradigma de programación que se basa en el concepto de "objetos", que son instancias de clases que encapsulan datos y operaciones que se pueden realizar sobre esos datos.

Aunque la programación orientada a objetos introduce conceptos como encapsulamiento, herencia y polimorfismo, estos se basan en el principio subyacente de la ejecución de instrucciones para lograr un resultado específico, que es la esencia del paradigma imperativo. En este sentido, los lenguajes orientados a objetos agrupan instrucciones para manipular datos y realizar operaciones dentro de objetos, y las interacciones entre objetos se realizan a través de mensajes o métodos, lo que implica, en última instancia, la ejecución de una secuencia de instrucciones para lograr un resultado.

Lenguajes declarativos

Los lenguajes declarativos son un tipo de lenguaje de programación que se centra en describir el qué se quiere lograr en lugar de cómo lograrlo. En lugar de proporcionar una secuencia de comandos e instrucciones detalladas para alcanzar un resultado específico (imperativo), los lenguajes declarativos permiten a los programadores expresar sus intenciones y dejar que el sistema determine la mejor manera de cumplirlas. Este enfoque contrasta con los lenguajes imperativos, donde se especifica paso a paso cómo se deben realizar las operaciones.

Está basado en el desarrollo de programas especificando o "declarando" un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla (tan sólo se le indica a la computadora que es lo que se desea obtener o que es lo que se está buscando).

El énfasis está en definir la lógica del problema y centrar los programas en describir relaciones y restricciones.

Los paradigmas de programación Funcional y Lógico entran en esta categoría.

Paradigmas fundamentales

Los paradigmas más importantes son:

- Paradigma Imperativo
- Paradigma de Objetos
- Paradigma Lógico
- Paradigma Funcional

A continuación daremos una muy breve descripción de cada uno de los paradigmas.

Paradigma Imperativo

En el paradigma imperativo, el énfasis recae en describir paso a paso cómo se deben realizar las operaciones. Los programas escritos en este paradigma se componen de una secuencia de instrucciones que indican al computador qué hacer y en qué orden. Se utilizan variables para almacenar y modificar datos, y las estructuras de control, como bucles y condicionales, dirigen el flujo de ejecución del programa. Este enfoque se centra en cambiar el estado de las variables a lo largo del tiempo.

Algunos ejemplos de lenguajes imperativos son: C, Pascal, Cobol.

Este paradigma es ampliamente utilizado en diversas áreas, desde el desarrollo de sistemas operativos y software de bajo nivel hasta aplicaciones de negocios y de alto rendimiento. Aunque es versátil, su énfasis en el control de flujo y el cambio de estado puede hacer que ciertas tareas sean más complejas en comparación con otros paradigmas.

Paradigma Orientado a Objetos

El paradigma orientado a objetos organiza el código alrededor de entidades llamadas "objetos", que encapsulan datos y las operaciones que se pueden realizar sobre esos datos. Conceptos como encapsulamiento, herencia y polimorfismo permiten modelar el mundo real de manera efectiva. Los programas orientados a objetos están estructurados en clases que definen los objetos, y las interacciones entre ellos se realizan mediante mensajes o métodos.
Algunos ejemplos de lenguajes son: Smalltalk, Java, C#, Python, Ruby.

Es ampliamente utilizado en diversas áreas de desarrollo de software debido a su capacidad para modelar sistemas de manera modular y extensible. Se adapta bien a la modelización de sistemas complejos y la resolución de problemas en el ámbito del software de negocios, pero también es utilizado en sistemas embebidos, sistemas distribuidos, desarrollo de interfaces gráficas de usuario, sistemas de control, entre otros.

Paradigma Lógico

En el paradigma lógico, los programas se construyen en base a reglas lógicas que describen relaciones y restricciones. Este enfoque utiliza la inferencia lógica para derivar conclusiones a partir de hechos y reglas establecidas. La programación lógica se centra en expresar intenciones sin especificar explícitamente el flujo de control del programa.

Un ejemplo de lenguaje de programación lógica es Prolog.

Se aplica principalmente en inteligencia artificial, sistemas expertos, procesamiento de lenguaje natural y problemas que pueden expresarse en términos de reglas y relaciones lógicas. Puede ser menos adecuado para problemas que requieren una manipulación eficiente de datos o cálculos numéricos intensivos.

Paradigma Funcional

En el paradigma funcional, se trata la computación como la evaluación de funciones matemáticas. Se enfoca en la inmutabilidad de datos, lo que significa que los datos no cambian una vez que se han creado. Las funciones son tratadas como entidades de primera clase, lo que permite pasarlas como argumentos y devolverlas como resultados. El paradigma funcional evita el cambio de estado mutable y promueve la escritura de funciones puras que producen resultados basados únicamente en sus entradas, sin efectos secundarios.

Algunos ejemplos de lenguajes de programación funcional: Haskell, Lisp, Scala, Erlang.

Es eficiente para programación concurrente, procesamiento de datos y desarrollo de algoritmos matemáticos. Se destaca en la creación de programas robustos y concisos. Aunque no es tan común en todas las áreas como otros paradigmas, su popularidad ha ido en aumento, especialmente en el procesamiento de datos y la programación distribuida.

Si bien algunos paradigmas de programación son más adecuados para ciertas áreas de aplicación, no hay restricciones estrictas que impidan aplicar cualquier paradigma a cualquier dominio. La elección del paradigma depende de varios factores, como la naturaleza del problema, los requisitos del proyecto, las preferencias del programador y la eficiencia deseada.

Cabe destacar que muchos lenguajes de programación admiten múltiples paradigmas, permitiendo a los programadores elegir la mejor aproximación para un problema específico. Por ejemplo, lenguajes como Python y JavaScript son conocidos por ser multiparadigma y permitir la programación imperativa, orientada a objetos y funcional. La elección del lenguaje y del paradigma depende del contexto del proyecto y de los requisitos particulares de la aplicación que se esté desarrollando.