PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos (Object Oriented Programming, OOP) es un modelo de programación informática que organiza el diseño de software en torno a datos u objetos, en lugar de funciones y lógica. Un objeto se puede definir como un campo de datos que tiene atributos y comportamiento únicos.

La programación orientada a objetos se centra en los objetos que los desarrolladores quieren manipular en lugar de enfocarse en la lógica necesaria para manipularlos. Este enfoque de programación es adecuado para programas que son grandes, complejos y se actualizan o mantienen activamente.

La organización de un programa orientado a objetos también hace que el método sea beneficioso para el desarrollo colaborativo, donde los proyectos se dividen en grupos.

Los beneficios adicionales de la programación orientada a objetos incluyen la reutilización, la escalabilidad y la eficiencia del código. Incluso cuando se utilizan microservicios, los desarrolladores deben seguir aplicando los principios de la programación orientada a objetos.

El primer paso en OOP es recopilar todos los objetos que un programador desea manipular e identificar cómo se relacionan entre sí, un ejercicio que a menudo se conoce como modelado de datos.

Los ejemplos de un objeto pueden variar desde entidades físicas, como un ser humano que se describe por propiedades como nombre y dirección, hasta pequeños programas informáticos, como widgets.

Una vez que se conoce un objeto, se etiqueta con una clase de objetos que define el tipo de datos que contiene y cualquier secuencia lógica que pueda manipularlo. Cada secuencia lógica distinta se conoce como método. Los objetos pueden comunicarse con interfaces bien definidas llamadas mensajes.

Principios de OOP

La programación orientada a objetos se basa en los siguientes principios:

Encapsulación

La implementación y el estado de cada objeto se mantienen de forma privada dentro de un límite definido o clase. Otros objetos no tienen acceso a esta clase o la autoridad para realizar cambios, pero pueden llamar a una lista de funciones o métodos públicos. Esta característica de ocultación de datos proporciona una mayor seguridad al programa y evita la corrupción de datos no intencionada.

Abstracción

Los objetos solo revelan mecanismos internos que son relevantes para el uso de otros objetos, ocultando cualquier código de implementación innecesario. Este concepto ayuda a los desarrolladores a realizar cambios y adiciones más fácilmente a lo largo del tiempo.

Herencia

Se pueden asignar relaciones y subclases entre objetos, lo que permite a los desarrolladores reutilizar una lógica común sin dejar de mantener una jerarquía única. Esta propiedad de OOP obliga a un análisis de datos más completo, reduce el tiempo de desarrollo y asegura un mayor nivel de precisión.

Polimorfismo

Los objetos pueden adoptar más de una forma según el contexto. El programa determinará qué significado o uso es necesario para cada ejecución de ese objeto, reduciendo la necesidad de duplicar código.

Lenguajes de programación orientados a objetos

Si bien Simula se acredita como el primer lenguaje de programación orientado a objetos, los lenguajes de programación orientada a objetos más populares son:

Java
JavaScript
Python
C++
Visual Basic .NET
Ruby
Scala
PHP

Críticas a la programación orientada a objetos

El modelo de programación orientada a objetos ha sido criticado por los desarrolladores por múltiples razones. La mayor preocupación es que la programación orientada a objetos hace demasiado hincapié en el componente de datos del desarrollo de software y no se centra lo suficiente en la computación o los algoritmos. Además, el código OOP puede ser más complicado de escribir y tardar más en compilarse.

Los métodos alternativos a la programación orientada a objetos incluyen:

programación funcional
programación estructurada
programación imperativa

Los lenguajes de programación más avanzados brindan a los desarrolladores la opción de combinar estos modelos.

PROGRAMACIÓN ESTRUCTURADA

Dentro de los paradigmas de la programación, la estructurada es una de los más comunes y a menudo el primer paso de los estudiantes que comienzan a tratar con el código. En este artículo queremos explicarte qué es exactamente y qué aportó en el mundo de la programación, hasta qué punto es importante hoy en día y otros detalles interesantes.

Qué es la programación estructurada

Es un paradigma de la programación, es decir, un estilo de codificar los algoritmos que se aplica en los lenguajes de programación.

La programación estructurada se establece como paradigma de programación en torno a los años 70, siendo uno de sus principales padres Dijkstra, con el objetivo de mejorar las prácticas de programación y facilitar la creación de programas más complejos, aumentando también la facilidad de mantenimiento del software.

Consiste en un estilo de programación en el que encontramos las estructuras básicas ya conocidas en los lenguajes de programación anteriores, como ciclos y condicionales, a los que se le añade la posibilidad de ejecutar subrutinas o funciones.

Qué había antes de la programación estructurada

Antes de aparecer la programación estructurada existía ya un estilo de programación imperativo, en el que se definían paso por paso las acciones para resolver un problema, el algoritmo, con las estructuras de control mencionadas anteriormente, como bucles o condicionales. Ese estilo imperativo ha perdurado hasta hoy y sigue siendo usado en la mayoría de los casos, sin embargo, para controlar el flujo de ejecución de las sentencias, antes de la programación estructurada, se usaba una instrucción denominada "GOTO".

El "GOTO", que viene de las palabras "go to" (ir a), permitía mencionar una línea de código a la que el flujo de ejecución debía saltar. Gracias a esta instrucción era posible volver a ejecutar un pedazo de código, saltarse unas cuantas líneas de código y cosas así. Durante mucho tiempo se había programado con esa costumbre, pero daba como resultado programas difíciles de entender, ya que no existía una clara distribución de las responsabilidades y tampoco facilitaba la organización del código.

Probablemente muchos de los lectores no habrán tenido que lidiar con la sentencia "GOTO" y no se hagan una idea de cómo eran los programas en aquella época. Básicamente consistían en bloques de código lineal, carentes de forma y estructura, con muy complicado mantenimiento. Sobre todo, era imposible ver a simple vista cómo iba

a ejecutarse en programa y para hacerse una idea era necesario seguir una traza, que saltaba de un lugar a otro del maremagnum de sentencias, de manera enredada.

De hecho, el término del "código spaghetti" viene justamente de esa falta de forma y la disposición del código sin una estructura definida, todo mezclado.

Qué aportó la programación estructurada

En lugar del "GOTO" la programación estructurada propone una pieza nueva que no existía anteriores paradigmas de la programación, que es la subrutina, o función.

Gracias a las funciones era posible organizar el código de una manera mucho más clara, creando pequeñas piezas de código que eran fácilmente utilizables, que aportaban semántica al código y que conseguían definir una estructura jerárquica.

Con las funciones se pudo comenzar a separar las partes del código en pequeñas piezas fácilmente comprensibles y mantenibles, a delegar la responsabilidad en rutinas que eran capaces de resolver un problema y sobre las cuales el programador podía despreocuparse de su funcionamiento interno, porque sabía que ellas serían capaces de resolver sus problemas.

A la capacidad de ser capaces de obviar los detalles no relevantes de algo es a lo que llamamos "abstracción". En términos de programación vino gracias a las funciones. El programador puede simplemente entender qué hace una función sin preocuparse en cómo está desarrollada por dentro, lo que le permite centrarse en las partes que le interesan en cada momento y deshacer la complejidad del software.

Ventajas de la Programación Estructurada:

- Capacidad de organización jerárquica del código. Unas funciones principales llaman a otras funciones secundarias y éstas llaman a otras, creando una jerarquía de funciones fácilmente entendibles.

- Permiten la abstracción, evitando que el programador necesite entender todo el código como un único bloque.

- Aumenta la capacidad de depuración, ya que cada una de las unidades "función" se pueden probar por separado.

- Se puede aumentar drásticamente el número de líneas de las aplicaciones sin que éstas se vuelvan un caos.

- Aumenta la velocidad de desarrollo y, además, la capacidad de que varias personas puedan participar en el desarrollo de un proyecto.

- Todo ello reduce la complejidad del software y los costes de desarrollo.

Tipos de funciones

A las subrutinas, el concepto que hoy conocemos por simplemente como "funciones", algunos lenguajes de programación lo dividieron en dos tipos de estructuras:

- Las funciones tenían la particularidad de devolver valores.
- Los procedimientos realizaban procesos, pero no devolvían ningún valor.
- Los lenguajes de programación que se usan comúnmente en la actualidad no difieren entre funciones y procedimientos, los tratan de la misma manera, con la misma estructura.

¿Se usa hoy la programación estructurada?

Aunque la programación estructurada es algo que ya viene de lejos hoy en día todavía es altamente usada en muchos proyectos y son la base de la mayoría de los lenguajes de programación actuales. De hecho, muchos lenguajes populares como Javascript o PHP la fomentan de base, igual, aunque también incorporan otros paradigmas distintos.

Cuando una persona comienza con la programación a menudo se empieza conociendo la programación estructurada, porque resulta más fácil y posibilita una curva de aprendizaje más suavizada. Como los lenguajes de programación actuales incorporan todo lo que la programación estructurada ha aportado, es muy sencillo aplicarla en la ruta de aprendizaje con tecnologías comúnmente utilizadas.

Pero, que se use para aprender, no quiere decir que no se use todavía para resolver los problemas actuales. Lenguajes como Javascript, aunque soportan otros paradigmas, la tienen muy arraigada y aplicaciones realmente complejas están resueltas en Javascript con una organización del código basada en funciones. Igual ocurre con otros lenguajes como PHP.

Muchos otros lenguajes de programación como Java usan cosas de programación estructurada, pero requiere el desarrollo de clases, de programación orientada a objetos, por lo que no siempre son usados como primer lenguaje para aprender a programar. también hay que decir que en la actualidad los equipos de desarrollo están más y más organizando el código con clases, aunque los lenguajes puedan ser usados solo con funciones. Quizás la comunidad donde más se usan las funciones para organizar el código sea la de Javascript, aunque en este lenguaje se usan mucho los objetos, incluso objetos creados de manera literal, como con JSON.

Qué vino después de la programación estructurada

Después de la programación estructurada aparecieron otros paradigmas como la programación orientada a objetos, que es la que domina el panorama actual de la programación.

La programación orientada a objetos hereda todo lo que tiene la programación estructurada, trayendo un nuevo concepto que es la "clase", la cual permite agrupar datos heterogéneos y funcionalidades asociadas a esos datos. Es un tema que puedes aprender mejor en el manual de programación orientada a objetos. Java es el lenguaje de programación más popular que nos obliga a programar siempre con clases.

Otros paradigmas como la programación funcional también vinieron después y son capaces de resolver de manera muy elegante ciertos problemas. Actualmente hay muchos lenguajes de programación funcional como Haskell o Scala, aunque a decir verdad no terminan de cuajar al 100%, porque no facilitan el desarrollo de todos los tipos de aplicaciones.