¿Qué es un IDE? Cite ejemplos

Un entorno de desarrollo integradoo entorno de desarrollo interactivo, en inglés *Integrated Development Environment* (IDE), es una aplicación informática que proporciona servicios integrales para facilitarle al desarrollador o programador el desarrollo de software. Los IDE están diseñados para maximizar la productividad del programador proporcionando componentes muy unidos con interfaces de usuario similares. Los IDE presentan un único programa en el que se lleva a cabo todo el desarrollo. Generalmente, este programa suele ofrecer muchas características para la creación, modificación, compilación, implementación y depuración de software. Uno de los propósitos de los IDE es reducir la configuración necesaria para reconstruir múltiples utilidades de desarrollo, en vez de proveer el mismo set de servicios como una unidad cohesiva. Reduciendo ese tiempo de ajustes, se puede incrementar la productividad de desarrollo, en casos donde aprender a usar un IDE es más rápido que integrar manualmente todas las herramientas por separado.

¿Qué es programación Estructurada?

La programación estructurada es un [paradigma de programación](https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n) orientado a mejorar la claridad, calidad y tiempo de desarrollo de un [programa de computadora](https://es.wikipedia.org/wiki/Programa_de_computadora) recurriendo únicamente a [subrutinas](https://es.wikipedia.org/wiki/Subrutina) y tres estructuras básicas: [secuencia](https://es.wikipedia.org/wiki/Bloque_de_c%C3%B3digo), selección (*[if](https://es.wikipedia.org/wiki/If" \o "If)* y *[switch](https://es.wikipedia.org/wiki/Switch" \o "Switch)*) e [iteración](https://es.wikipedia.org/wiki/Iteraci%C3%B3n) ([bucles](https://es.wikipedia.org/wiki/Bucle_(programaci%C3%B3n)) *[for](https://es.wikipedia.org/wiki/Bucle_for" \o "Bucle for)* y *[while](https://es.wikipedia.org/wiki/Bucle_while" \o "Bucle while)*); asimismo, se considera innecesario y contraproducente el uso de la instrucción de [transferencia incondicional](https://es.wikipedia.org/wiki/Transferencia_incondicional) ([GOTO](https://es.wikipedia.org/wiki/GOTO)), que podría conducir a [*código espagueti*](https://es.wikipedia.org/wiki/C%C3%B3digo_espagueti), mucho más difícil de seguir y de mantener, y fuente de numerosos [errores de programación](https://es.wikipedia.org/wiki/Error_de_programaci%C3%B3n).

Surgió en la década de 1960, particularmente del trabajo de [Böhm](https://es.wikipedia.org/wiki/Corrado_B%C3%B6hm" \o "Corrado Böhm) y Jacopini,[1](https://es.wikipedia.org/wiki/Programaci%C3%B3n_estructurada#cite_note-1)​ y un famoso escrito de 1968: «La sentencia goto, considerada perjudicial», de [Edsger Dijkstra](https://es.wikipedia.org/wiki/Edsger_Dijkstra" \o "Edsger Dijkstra).[2](https://es.wikipedia.org/wiki/Programaci%C3%B3n_estructurada#cite_note-dijkstra1968-2)​ Sus postulados se verían reforzados, a nivel teórico, por el [teorema del programa estructurado](https://es.wikipedia.org/wiki/Teorema_del_programa_estructurado) y, a nivel práctico, por la aparición de lenguajes como [ALGOL](https://es.wikipedia.org/wiki/ALGOL), dotado de [estructuras de control](https://es.wikipedia.org/wiki/Estructuras_de_control) consistentes y bien formadas.

A finales de los años 1970 surgió una nueva forma de programar que no solamente permitía desarrollar programas fiables y eficientes, sino que además estos estaban escritos de manera que se facilitaba su comprensión en fases de mejora posteriores.

El [teorema del programa estructurado](https://es.wikipedia.org/wiki/Teorema_del_programa_estructurado), propuesto por Böhm-Jacopini, demuestra que todo programa puede escribirse utilizando únicamente las tres instrucciones de control siguientes:

* Secuencia.
* Instrucción condicional.
* Iteración (bucle de instrucciones) con condición inicial.

¿Qué es programación Orientada a Objetos?

La programación orientada a objetos (POO, u OOP según sus siglas en inglés) es un [paradigma de programación](https://es.wikipedia.org/wiki/Paradigma_de_programaci%C3%B3n) que viene a innovar la forma de obtener resultados. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial.

Muchos de los objetos prediseñados de los [lenguajes de programación](https://es.wikipedia.org/wiki/Lenguajes_de_programaci%C3%B3n) actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas.

Está basada en varias técnicas, incluyendo [herencia](https://es.wikipedia.org/wiki/Herencia_(inform%C3%A1tica)), [cohesión](https://es.wikipedia.org/wiki/Cohesi%C3%B3n_(inform%C3%A1tica)), [abstracción](https://es.wikipedia.org/wiki/Abstracci%C3%B3n_(inform%C3%A1tica)), [polimorfismo](https://es.wikipedia.org/wiki/Polimorfismo_(inform%C3%A1tica)), [acoplamiento](https://es.wikipedia.org/wiki/Acoplamiento_(inform%C3%A1tica)) y [encapsulamiento](https://es.wikipedia.org/wiki/Encapsulamiento_(inform%C3%A1tica)).

Su uso se popularizó a principios de la [década de 1990](https://es.wikipedia.org/wiki/D%C3%A9cada_de_1990). En la actualidad, existe una gran variedad de [lenguajes de programación](https://es.wikipedia.org/wiki/Lenguajes_de_programaci%C3%B3n) que soportan la orientación a objetos.

Los objetos son entidades que tienen un determinado "estado", "comportamiento (método)" e "identidad":

* La identidad es una propiedad de un objeto que lo diferencia del resto; dicho con otras palabras, es su identificador (concepto análogo al de identificador de una [variable](https://es.wikipedia.org/wiki/Variable_(programaci%C3%B3n)) o una [constante](https://es.wikipedia.org/wiki/Constante_(programaci%C3%B3n))).

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados [métodos](https://es.wikipedia.org/wiki/M%C3%A9todo_(programaci%C3%B3n_orientada_a_objetos)), que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

¿Qué es un lenguaje de bajo nivel?

Un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) de características bajo nivel es aquel en el que sus instrucciones ejercen un control directo sobre el hardware y están condicionados por la estructura física de las computadoras que lo soportan. El uso de la palabra *bajo* en su denominación no implica que el lenguaje sea menos potente que un [lenguaje de alto nivel](https://es.wikipedia.org/wiki/Lenguaje_de_alto_nivel), sino que se refiere a la reducida [abstracción](https://es.wikipedia.org/wiki/Abstracci%C3%B3n_(inform%C3%A1tica)) entre el lenguaje y el hardware. Por ejemplo, se utiliza este tipo de lenguajes para programar tareas críticas de los [sistemas operativos](https://es.wikipedia.org/wiki/Sistemas_operativos), de aplicaciones en tiempo real o [controladores de dispositivos](https://es.wikipedia.org/wiki/Controlador_de_dispositivo).

Los lenguajes de bajo nivel como ser el lenguaje ensamblador puede ser más difícil que los lenguajes de alto nivel debido a que está más íntimamente relacionado con las características técnicas del hardware. Además estos suelen carecer de abstracciones de mayor nivel. No es posible una abstracción fuera de lo estipulado para el conjunto del [microcódigos](https://es.wikipedia.org/wiki/Microc%C3%B3digo" \o "Microcódigo) del [microprocesador](https://es.wikipedia.org/wiki/Microprocesador) de un ordenador. Por otro lado, es más fácilmente traducible a [lenguaje de máquina](https://es.wikipedia.org/wiki/Lenguaje_de_m%C3%A1quina), tarea que realiza esencialmente un [compilador](https://es.wikipedia.org/wiki/Compilador).

¿Qué es un lenguaje de alto nivel?

Un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) de alto nivel se caracteriza por expresar los [algoritmos](https://es.wikipedia.org/wiki/Algoritmo) de una manera adecuada a la capacidad [cognitiva](https://es.wikipedia.org/wiki/Cognici%C3%B3n) humana, en lugar de la capacidad con que los ejecutan las máquinas.

En los primeros lenguajes, la limitación era que se orientaban a un área específica y sus instrucciones requerían de una sintaxis predefinida. Se clasifican como lenguajes procedimentales o [lenguajes de bajo nivel](https://es.wikipedia.org/wiki/Lenguaje_de_bajo_nivel). Otra limitación de estos es que se requiere de ciertos conocimientos de programación para realizar las secuencias de instrucciones lógicas. Los lenguajes de alto nivel se crearon para que el usuario común pudiese solucionar un problema de procesamiento de datos de una manera más fácil y rápida.

Por esta razón, a finales de los [años 1950](https://es.wikipedia.org/wiki/A%C3%B1os_1950) fue fundada en 1950 y surgió un nuevo tipo de lenguajes de programación que evitaba estos inconvenientes, a costa de ceder un poco en las ventajas. Estos lenguajes se llaman *de tercera generación* o *de nivel alto*, en contraposición a los *de bajo nivel* o *de nivel próximo a la máquina*.

¿Qué es un intérprete?

En [ciencias de la computación](https://es.wikipedia.org/wiki/Ciencias_de_la_computaci%C3%B3n), intérprete o interpretador es un [programa informático](https://es.wikipedia.org/wiki/Programa_(computaci%C3%B3n)) capaz de analizar y ejecutar otros programas. Los intérpretes se diferencian de los [compiladores](https://es.wikipedia.org/wiki/Compilador) o de los [ensambladores](https://es.wikipedia.org/wiki/Ensamblador) en que mientras estos traducen un programa desde su descripción en un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_de_cuarta_generaci%C3%B3n) al [código de máquina](https://es.wikipedia.org/wiki/C%C3%B3digo_de_m%C3%A1quina) del sistema, los intérpretes sólo realizan la [traducción](https://es.wikipedia.org/wiki/Proceso_de_traducci%C3%B3n_de_programas) a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.

Usando un intérprete, un solo archivo fuente puede producir resultados iguales incluso en sistemas sumamente diferentes (ejemplo. una [PC](https://es.wikipedia.org/wiki/Computador_personal) y una PlayStation 4). Usando un compilador, un solo archivo fuente puede producir resultados iguales solo si es compilado a distintos ejecutables específicos a cada sistema.

Los programas interpretados suelen ser más lentos que los compilados debido a la necesidad de traducir el programa mientras se ejecuta, pero a cambio son más flexibles como entornos de [programación](https://es.wikipedia.org/wiki/Programaci%C3%B3n) y [depuración](https://es.wikipedia.org/wiki/Depuraci%C3%B3n_de_programas) (lo que se traduce, por ejemplo, en una mayor facilidad para reemplazar partes enteras del programa o añadir módulos completamente nuevos), y permiten ofrecer al programa interpretado un entorno no dependiente de la máquina donde se ejecuta el intérprete, sino del propio intérprete (lo que se conoce comúnmente como [máquina virtual](https://es.wikipedia.org/wiki/M%C3%A1quina_virtual)).

Para mejorar el desempeño, algunas implementaciones de programación de lenguajes de programación pueden interpretar o compilar el código fuente original en una forma intermedia más compacta, y después traducir eso al código de máquina (ej. [Perl](https://es.wikipedia.org/wiki/Perl), [Python](https://es.wikipedia.org/wiki/Python" \o "Python), [MATLAB](https://es.wikipedia.org/wiki/MATLAB), y [Ruby](https://es.wikipedia.org/wiki/Ruby)). Algunos aceptan los archivos fuente guardados en esta representación intermedia (ej. Python, [UCSD Pascal](https://es.wikipedia.org/wiki/UCSD_Pascal) y [Java](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_Java)).

¿Qué es un compilador?

Un compilador es un [programa informático](https://es.wikipedia.org/wiki/Programa_inform%C3%A1tico) que [traduce un programa](https://es.wikipedia.org/wiki/Proceso_de_traducci%C3%B3n_de_programas) que ha sido escrito en un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) a un lenguaje común, [1](https://es.wikipedia.org/wiki/Compilador#cite_note-1)​ usualmente [lenguaje de máquina](https://es.wikipedia.org/wiki/Lenguaje_de_m%C3%A1quina), aunque también puede ser traducido a un código intermedio (*[bytecode](https://es.wikipedia.org/wiki/Bytecode" \o "Bytecode)*) o a texto y que reúne diversos elementos o fragmentos en una misma unidad. Este proceso de traducción se conoce como *compilación*, compilar consiste en traducir un programa escrito en un cierto lenguaje a otro.[2](https://es.wikipedia.org/wiki/Compilador#cite_note-2)​

La construcción de un compilador involucra la división del proceso en una serie de fases que variará con su complejidad. Generalmente estas fases se agrupan en dos tareas: el análisis del programa fuente y la síntesis del programa objeto.

* [Análisis](https://es.wikipedia.org/wiki/An%C3%A1lisis_de_algoritmos): se trata de la comprobación de la corrección del programa fuente, e incluye las fases correspondientes al [análisis léxico](https://es.wikipedia.org/wiki/Analizador_l%C3%A9xico) (que consiste en la descomposición del programa fuente en componentes léxicos), [análisis sintáctico](https://es.wikipedia.org/wiki/Analizador_sint%C3%A1ctico) (agrupación de los componentes léxicos en frases gramaticales ) y análisis semántico (comprobación de la validez semántica de las sentencias aceptadas en la fase de análisis sintáctico).
* [Síntesis](https://es.wikipedia.org/wiki/S%C3%ADntesis): su objetivo es la generación de la salida expresada en el lenguaje objeto y suele estar formado por una o varias combinaciones de fases de generación de código (normalmente se trata de código intermedio o de código objeto) y de optimización de código (en las que se busca obtener un código lo más eficiente posible).

Alternativamente, las fases descritas para las tareas de análisis y síntesis se pueden agrupar en:

* Analizador o *front-end*: es la parte que analiza el [código fuente](https://es.wikipedia.org/wiki/C%C3%B3digo_fuente), comprueba su validez, genera el árbol de derivación y rellena los valores de la [tabla de símbolos](https://es.wikipedia.org/wiki/Tabla_de_s%C3%ADmbolos_(compilador)). Esta parte suele ser independiente de la plataforma o sistema para el cual se vaya a compilar, y está compuesta por las fases comprendidas entre el análisis léxico y la generación de código intermedio.
* Generador o *back-end*: es la parte que genera el código máquina, específico de una plataforma, a partir de los resultados de la fase de análisis, realizada por este generador.

Esta división permite que el mismo generador se utilice para crear el [código máquina](https://es.wikipedia.org/wiki/C%C3%B3digo_m%C3%A1quina) de varios [lenguajes de programación](https://es.wikipedia.org/wiki/Lenguajes_de_programaci%C3%B3n) distintos y que el mismo analizador que sirve para examinar el código fuente de un [lenguaje de programación](https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n) concreto sirva para producir [código máquina](https://es.wikipedia.org/wiki/C%C3%B3digo_m%C3%A1quina) en varias plataformas. Suele incluir la generación y optimización del código dependiente de la máquina.