**Open Bootcamp**

**B**

[Conceptos de la Programación 1](#_Toc122538093)

[Que es un lenguaje de programación 1](#_Toc122538094)

[Concepto 1](#_Toc122538095)

[Tipos de Paradigmas de Programación 1](#_Toc122538096)

[Imperativos vs Declarativos 1](#_Toc122538097)

[Ventajas del paradigma imperativo 2](#_Toc122538098)

[Desventajas de paradigma imperativo 2](#_Toc122538099)

[Ventajas del paradigma declarativo 3](#_Toc122538100)

[Desventajas del paradigma declarativo 3](#_Toc122538101)

[Funcionales vs Procedimentales 4](#_Toc122538102)

[División por niveles 4](#_Toc122538103)

[Lenguaje Maquina 4](#_Toc122538104)

[Lenguaje Ensamblador 5](#_Toc122538105)

[Bajo Nivel 6](#_Toc122538106)

[Clasificación por nivel 6](#_Toc122538107)

[Proceso de Conversión 6](#_Toc122538108)

[Compilador 7](#_Toc122538109)

[Interprete 7](#_Toc122538110)

[Diferencia entre un lenguaje interpretado y un lenguaje compilado 7](#_Toc122538111)

[Ventajas y desventajas de un lenguaje interpretado 8](#_Toc122538112)

[Ventajas y desventajas de un lenguaje compilado 9](#_Toc122538113)

[Desarrollo de código (programación) 9](#_Toc122538114)

[Entorno de desarrollo integrado IDE 10](#_Toc122538115)

[Control de versiones 10](#_Toc122538116)

# Conceptos de la Programación

## Que es un lenguaje de programación

### Concepto

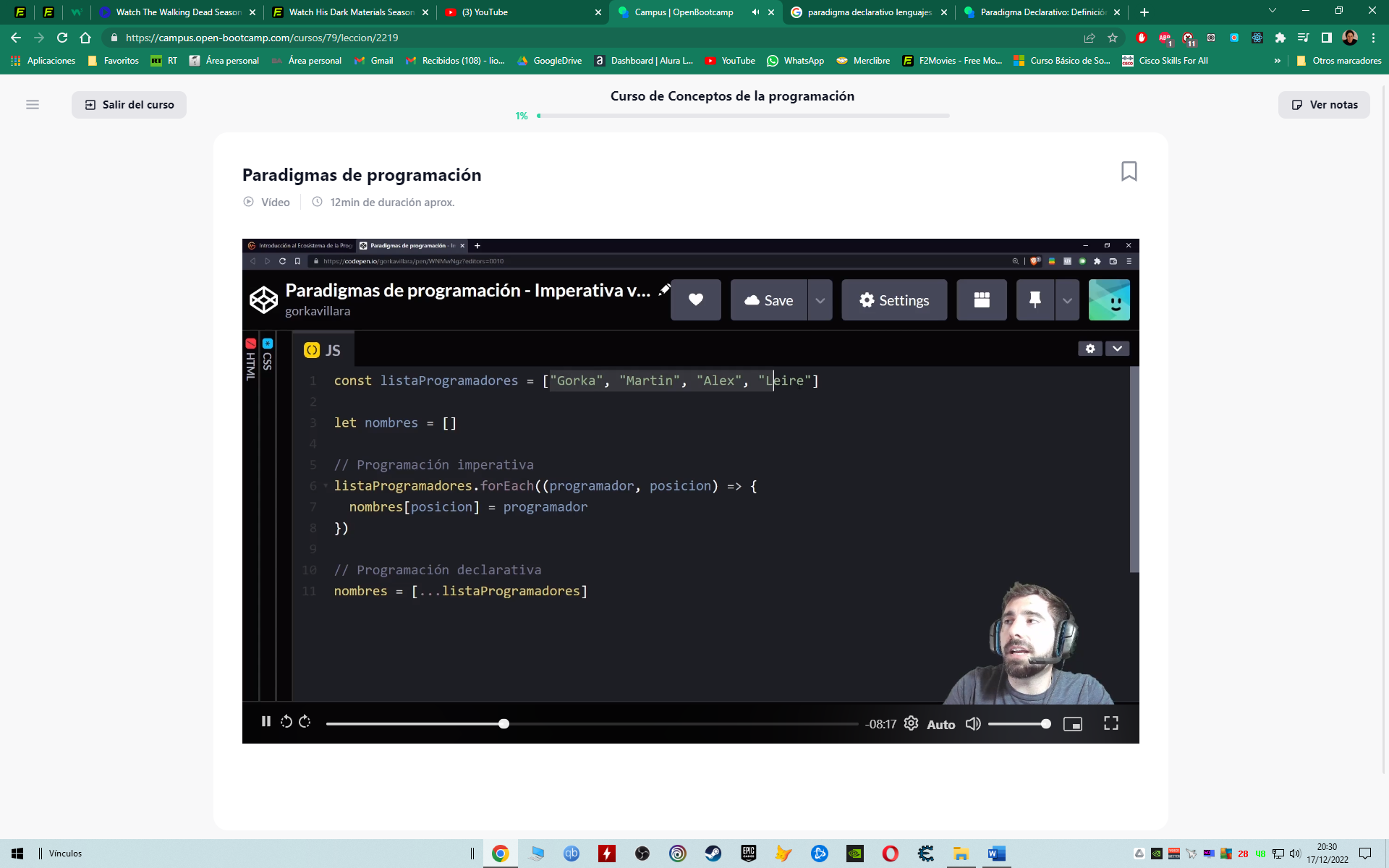
Un lenguaje de programación es un lenguaje formal, es decir un lenguaje con reglas gramaticales bien definidas, que le proporciona al programador, la capacidad de escribir (o programar) una serie de instrucciones o secuencias de órdenes en forma de algoritmos con el fin de controlar el comportamiento físico o lógico de un sistema informático, de manera que se puedan obtener diversas clases de datos o ejecutar determinadas tareas.

## Tipos de Paradigmas de Programación

### Imperativos vs Declarativos

Los **paradigmas imperativos** son el paradigma más antiguo. Los programas consisten en una sucesión de instrucciones o conjunto de sentencias, como si el programador diera órdenes concretas. El desarrollador describe en el código paso por paso todo lo que hará su programa.

El **paradigma declarativo** es una forma de desarrollar software que se centra en el resultado final, es decir, el programador delimita lo que espera obtener con ese segmento de código y el sistema genera el resultado necesario.



#### Ventajas del paradigma imperativo

Ahora, veamos algunas de las ventajas que ofrece el paradigma imperativo a los programadores que lo usan:

* Aún hoy día hay muchos lenguajes de programación que están basados en el paradigma imperativo.
* Es el enfoque original de la programación, es decir, en los años 50, los lenguajes fueron pensados e implementados usando este paradigma.
* El código desarrollado con el paradigma imperativo es muy fácil de leer, es como leer las instrucciones de una receta o manual, esto es útil cuando estás comenzando en el mundo de la programación.
* Esa misma facilidad de lectura, es importante a la hora de corregir errores en el código, pero solo cuando se trata de programas sencillos, que no tienen mucha extensión.

#### Desventajas de paradigma imperativo

Entre las desventajas del paradigma imperativo es importante considerar:

* Cuando la complejidad del problema exige que el código sea más extenso, es complicado leerlo e identificar los posibles errores que se produzcan al momento de ejecutarlo.
* El programador debe ser muy cuidadoso al modificar cualquier línea de código porque puede alterar el funcionamiento de lo que se ha desarrollado y producir errores.

#### Ventajas del paradigma declarativo

Entre las ventajas que ofrece desarrollar software siguiendo el paradigma declarativo destacan:

* Reduce la mutabilidad. Cuando dentro del código hay cambios frecuentes en las estructuras de datos, es posible que se produzcan errores. Dentro del paradigma declarativo se trabaja con estructuras de datos inmutables que evitan esto.
* Facilidad para reparar errores. Si una de las sentencias utilizadas está fallando, es fácil detectar dónde está ocurriendo la incidencia y corregirla sin afectar el resto del programa.
* El código es más corto. El paradigma declarativo te permite emplear múltiples veces todas las estructuras o funciones que ya has desarrollado, sin necesidad de copiar el código completo nuevamente.

Ahora bien, es importante que sepas que estas ventajas únicamente puedes aprovecharlas si el estado final, lo que quieres lograr con el código, está perfectamente establecido.

#### Desventajas del paradigma declarativo

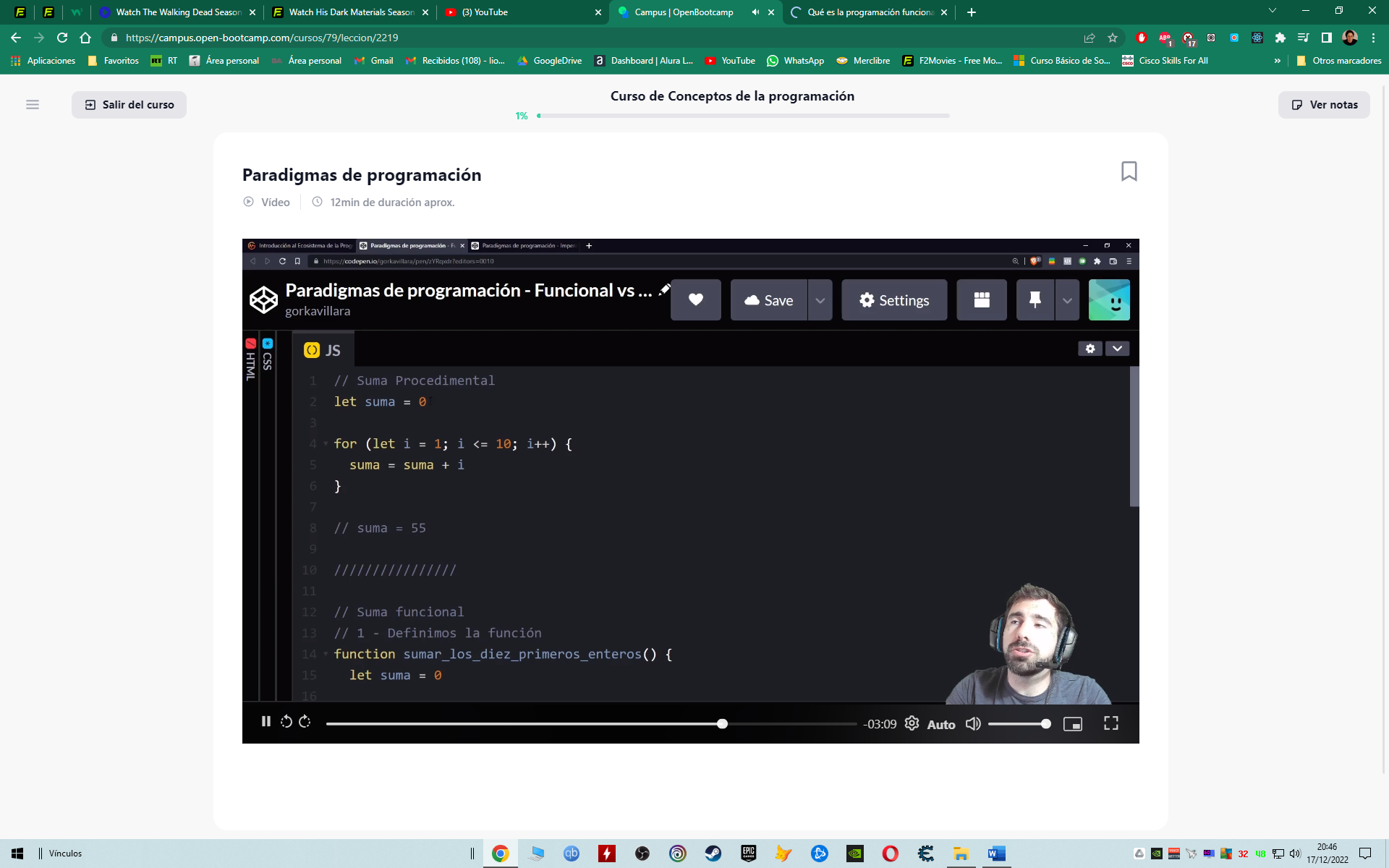
Si revisamos las desventajas del paradigma declarativo encontramos que:

* Para la mente humana es difícil orientar la programación sin verla como pasos a seguir para la solución de un problema. Este paradigma enfoca su atención en el resultado que se espera, sin enunciar los pasos a seguir para alcanzar este. Así que puede resultar complicado entenderlo, sobre todo al inicio.
* El código desarrollado siguiendo el paradigma declarativo suele ser altamente abstracto, eso hace que a medida que crece en complejidad es más difícil que sea entendido por otras personas.

### Funcionales vs Procedimentales

El **paradigma funcional** está basado en el modelo matemático de composición funcional. En este modelo, el resultado de un cálculo es la entrada del siguiente, y así sucesivamente hasta que una composición produce el valor deseado.

La **programación procedimental o programación por procedimientos** es un paradigma de la programación. Muchas veces es aplicable tanto en lenguajes de programación de bajo nivel como en lenguajes de alto nivel. Este paradigma consiste en basarse de un número muy bajo de expresiones repetidas, englobarlas todas en un procedimiento o función y llamarlo cada vez que tenga que ejecutarse.



### División por niveles

#### Lenguaje Maquina

El lenguaje de maquina es aquel cuyas instrucciones son directamente entendibles por la computadora y no necesitan traducción posterior para que la UCP pueda comprender y ejecutar el programa.

Las instrucciones en lenguaje máquina se expresan en términos de la unidad de memoria más pequeña (bit) = digito binario 0 o 1, en esencia una secuencia de bits que especifican la operación y las celdas de memoria implicadas en una operación.

Como se observa estas instrucciones son fáciles de leer por una computadora y difíciles para un programador y viceversa. Por esta razón se hace difícil escribir programas en código o lenguaje de máquina. Y se requiere otro lenguaje para comunicarse con la computadora pero que se hace más fácil de escribir y de leer por el programador. Para evitar la tediosa tarea de escribir programas en este lenguaje se han diseñado otros programas de programación que facilitan la escritura y posterior ejecución de los programas.

#### Lenguaje Ensamblador

Son una representación más entendible para el humano de los códigos del lenguaje máquina. Cada instrucción en lenguaje ensamblador representa una instrucción en el lenguaje máquina. El lenguaje ensamblador aparece casi a la par con el lenguaje máquina, esto debido a que los fabricantes de hardware diseñan sus Chips pensando ya en las instrucciones de un lenguaje de ensamblado.

A pesar de que el lenguaje ensamblador es más fácil de entender por las personas sigue teniendo las desventajas del lenguaje de máquina. Realmente lo que hace es ayudar un poco a que el código sea más legible. El lenguaje ensamblador adicionalmente necesita un traductor (ensamblador) capaz de convertir dichos códigos en lenguaje máquina.

Las computadoras tienen su propio lenguaje, compuesto de un código binario representado por ceros y unos en secuencias únicas y difícilmente comprensibles para el ojo humano.

Los lenguajes de programación empleados en el trabajo diario de los desarrolladores, están conformados por instrucciones puntuales, con una sintaxis estricta, pero que facilitan el proceso de hacer programas o desarrollar aplicaciones.

El lenguaje ensamblador está en un nivel intermedio entre estos dos tipos de lenguaje. Usa código mnemotécnico para formular instrucciones básicas que son interpretadas por los computadores, procesadores, controladores y cualquier circuito integrado que sea programable.

En cuanto a los tiempos de ejecución, el lenguaje ensamblador es un poco más lento que el lenguaje máquina, ya que cada instrucción necesita ser traducida para que el procesador la entienda y lleve a cabo lo que se le indica.

Un ejemplo de una instrucción en lenguaje máquina y en lenguaje ensamblador:

Ejemplo de Lenguaje Máquina vs. Lenguaje Ensamblador

|  |  |
| --- | --- |
| **Lenguaje Máquina** | **Lenguaje Ensamblador** |
| 0010111000000001  0000000000001010 | mov ax, 10 |

#### Bajo Nivel

Un lenguaje de programación de bajo nivel, es aquel en el que sus instrucciones ejercen un control directo sobre el hardware y por lo tanto están condicionados por la estructura física de las computadoras que lo soportan.

#### Clasificación por nivel

##### Lenguajes de alto nivel

Son muy parecidos al lenguaje humano. Por lo tanto, es relativamente sencillo comprender las instrucciones del código. Los desarrolladores usan estos lenguajes para crear programas o aplicaciones que pueden ser usados en diversos dispositivos.

Java, Python, C++, Perl, JavaScript, Fortran, SQL y PHP.

##### Lenguaje de bajo nivel

Es el lenguaje de la máquina y es prácticamente imposible para el ojo humano comprender las instrucciones porque están compuestas por ceros, unos (lenguaje binario) y combinaciones de letras y números (lenguaje ensamblador).

##### Lenguaje de nivel medio

Este concepto no es aceptado por todo el gremio de la programación. Se refiere a lenguajes en los que se puede escribir código, pero que también pueden trabajar directamente con los registros del sistema. C, Fortran y Basic son incluidos en este apartado por quienes establecen esta parte de la clasificación.

## Proceso de Conversión

### Compilador

Un compilador es un programa informático que traduce todo el código fuente de un proyecto de software a código máquina antes de ejecutarlo. Solo entonces el procesador ejecuta el software, obteniendo todas las instrucciones en código máquina antes de comenzar. De esta manera, el procesador cuenta con todos los componentes necesarios para ejecutar el software, procesar las entradas y generar los resultados.

El primer compilador fue desarrollado por Grace Hooper, una científica y militar de nacionalidad estadounidense que se empeñó en lograr que los seres humanos interactuaran de una manera sencilla con las computadoras.

El compilador toma la totalidad del código escrito y lo traduce, generando un archivo o fichero de código ejecutable que la máquina podrá interpretar directamente.

Algunos ejemplos de lenguaje compilado son: Haskell, C, C++, Pascal, Cobol, Basic, Algol, Basic, etc.

### Interprete

Un intérprete es un programa informático que procesa el código fuente de un proyecto de software durante su tiempo de ejecución, es decir, mientras el software se está ejecutando, y actúa como una interfaz entre ese proyecto y el procesador. Un intérprete siempre procesa el código línea por línea, de modo que lee, analiza y prepara cada secuencia de forma consecutiva para el procesador. Este principio también se aplica a las secuencias recurrentes, que se ejecutan de nuevo cada vez que vuelven a aparecer en el código. Para procesar el código fuente del software, el intérprete recurre a sus propias bibliotecas internas: en cuanto una línea de código fuente se ha traducido a los correspondientes comandos legibles por máquina, esta se envía directamente al procesador.

El proceso de conversión no finaliza hasta que se ha interpretado todo el código. Solo se interrumpe prematuramente si se produce un fallo durante el procesamiento, lo que simplifica mucho la resolución de los errores, ya que la línea de código problemática se detecta inmediatamente después de ocurrir el fallo.

Algunos ejemplos de lenguaje compilado son: Perl, Python, Ruby, PHP ,etc.

#### Diferencia entre un lenguaje interpretado y un lenguaje compilado

Quizá estás pensando que este concepto y el de lenguaje compilado son lo mismo, sin embargo, hay una diferencia entre ambos que afecta la forma en que se hace la traducción del código fuente.

En el caso del lenguaje compilado, un compilador convierte la totalidad del código que ha hecho el programador a un archivo ejecutable que podrá ser entendido por la máquina.

Ahora bien, en el caso del lenguaje interpretado, no es necesario hacer una compilación del código completo. La máquina, mediante un intérprete, va ejecutando una a una las líneas del programa en tiempo real.

Cuando se trabaja con un lenguaje interpretado, si el intérprete encuentra algún error, la ejecución se detiene del todo hasta que este sea corregido y, nuevamente, pueda ser puesto en marcha cuando se resuelve.

#### Ventajas y desventajas de un lenguaje interpretado

##### Entre las ventajas de un lenguaje interpretado están:

El programa podrá ejecutarse en cualquier máquina que tenga el intérprete instalado. De cumplirse esa condición, cualquier código escrito en ese lenguaje podrá ser ejecutado sin dificultad, sin tomar en cuenta el sistema operativo que tenga instalado el computador.

A la hora de corregir errores o agregar funciones nuevas al código, el lenguaje interpretado nos brinda la posibilidad de hacerlo sin necesidad de tener que esperar para hacer una nueva compilación.

El tiempo necesario para desarrollar aplicaciones en los lenguajes interpretados es menor, puesto que para ir haciendo pruebas no es necesario esperar a compilar la totalidad del código.

Por lo general, son más sencillos a la hora de aprender que los lenguajes compilados.

##### Al hablar de las desventajas de un lenguaje interpretado considera:

Puesto que el código se va ejecutando al mismo tiempo que la máquina lo está leyendo, la velocidad de procesamiento es más lenta que en el caso de los programas compilados.

El código fuente puede ser modificado por cualquier persona que tenga acceso a él. En el caso de los derechos de autor, esto puede significar una dificultad considerable.

Para ejecutar el código deberás tener acceso a este y al intérprete, no solo al ejecutable.

##### Ejemplos de lenguaje interpretado

* Perl
* Bash
* Python
* Ruby
* PHP
* Basic
* ASP
* TCL

#### Ventajas y desventajas de un lenguaje compilado

##### Entre las ventajas de los lenguajes compilados están:

Eficiencia en la corrección de errores, cuando se hace el proceso de compilación, el compilador emite una lista de todos los errores que necesitan corrección.

Su ejecución es más rápida que en el caso de los lenguajes interpretados, puesto que se ejecuta cuando se ha compilado todo el bloque de código y no instrucción a instrucción.

##### Como desventajas, los lenguajes compilados tienen:

Los errores no se pueden corregir hasta que se complete la compilación.

Una vez que se revisa un error es necesario volver a hacer el proceso de compilación desde cero.

Algo importante que debes tomar en cuenta cuando uses lenguajes compilados es que el hecho de que el proceso de compilación sea exitoso no quiere decir que el programa esté funcionando.

Para verificar esto será necesario que realices pruebas que comprueben que el código está realizando las tareas específicas para las que lo construiste, de manera efectiva.

## Desarrollo de código (programación)

### Entorno de desarrollo integrado IDE

### Control de versiones