



# Programación

01 Introducción a la Programación de Software

José Luis González Sánchez



# Contenidos

1. Algoritmos y Programas
2. Paradigmas de Programación
3. Fases de la Programación
4. Tipos de Lenguajes
5. Compiladores e Intérpretes
6. Lenguajes más usados



# Algoritmos y Programas

Antes de programar...

# Algoritmo y Programas

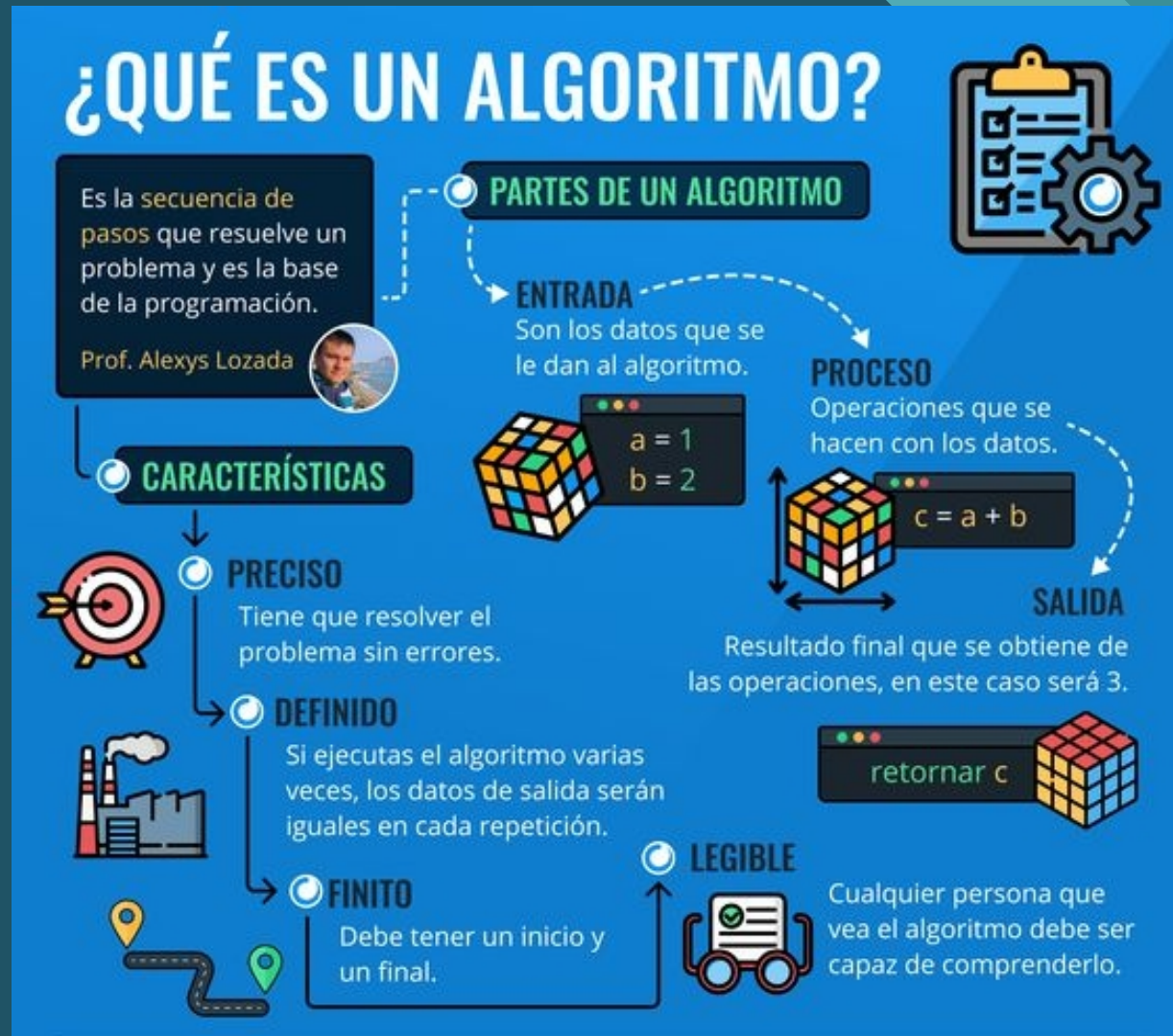
- **Algoritmo:** secuencia ordenada de pasos, descrita sin ambigüedades, que conducen a la solución de un problema dado.
- Los algoritmos son independientes de los lenguajes de programación y de las computadoras donde se ejecutan. Un mismo algoritmo puede ser expresado en diferentes lenguajes de programación y podría ser ejecutado en diferentes dispositivos. Piensa en una receta de cocina, ésta puede ser expresada en castellano, inglés o francés, podría ser cocinada en fogón o vitrocerámica, por un cocinero o más, etc. Pero independientemente de todas estas circunstancias, el plato se preparará siguiendo los mismos pasos.
- La diferencia fundamental entre **algoritmo** y **programa** es que, en el segundo, los pasos que permiten resolver el problema, deben escribirse en un determinado lenguaje de programación para que puedan ser ejecutados en el ordenador y así obtener la solución.

# Algoritmo y Programas

- Los lenguajes de programación son sólo un medio para expresar el algoritmo y el ordenador un procesador para ejecutarlo. Mediante el lenguaje creamos programas que ejecutan uno o más algoritmos en un sistema específico.
- En esencia, todo problema se puede describir por medio de un algoritmo y las características fundamentales que éstos deben cumplir son:
  - Debe ser preciso e indicar el orden de realización paso a paso.
  - Debe estar definido, si se ejecuta dos o más veces, debe obtener el mismo resultado cada vez.
  - Debe ser finito, debe tener un número finito de pasos.



# Algoritmo y Programas



# Paradigmas de Programación

Las Reglas del juego

# Paradigmas de Programación

- **Paradigma de programación:** es un modelo básico para el diseño y la implementación de programas. Este modelo determinará como será el proceso de diseño y la estructura final de un programa.
- Son las reglas del juego. Tienes una pelota o balón y una portería, pero no es lo mismo jugar al fútbol que al balomano. Pero en ambos tu objetivo es introducir el balón en la portería y anotar más que tu rival. Igual pasa con los lenguajes, existen reglas y según las usemos podremos hacer unas cosas u otras. Tipos:
  - Programación Estructurada
  - Programación Modular
  - Programación Declarativa
  - Programación Funcional
  - Programación Lógica
  - Programación Orientada a Objetos
  - Programación Reactiva



# Paradigmas de Programación

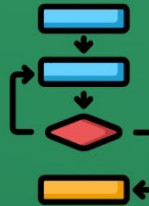
## ¿QUÉ SON LOS PARADIGMAS DE PROGRAMACIÓN?

Los paradigmas son los diferentes estilo de usar la programación para resolver un problema.



### PROGRAMACIÓN ESTRUCTURADA

Programación secuencial con la que todos aprendemos a programar. Usa ciclos y condicionales.



### PROGRAMACIÓN REACTIVA

Observa flujos de datos asíncronos y reacciona frente a sus cambios.



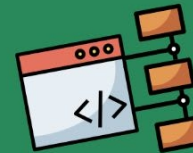
### PROGRAMACIÓN ORIENTADA A OBJETOS

Divide los componentes del programa en objetos que tienen datos y comportamiento y se comunican entre sí.



### PROGRAMACIÓN FUNCIONAL

Divide el programa en tareas pequeñas que son ejecutadas por funciones.



# Fases de la Programación

Secuencias que debemos seguir

# Fases de Programación

- El proceso de creación de software puede dividirse en diferentes fases:
  - Fase de resolución del problema: Análisis y Diseño.
  - Fase de implementación: Codificación y Pruebas.
  - Fase de explotación y mantenimiento.

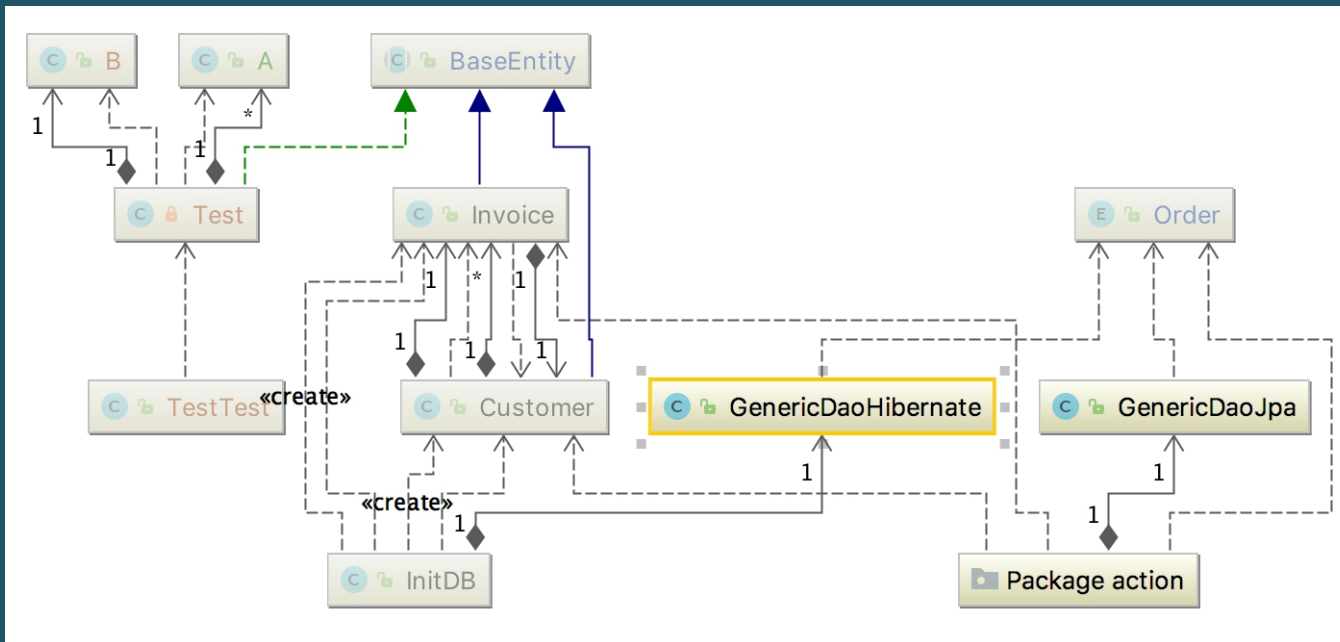
# Análisis

- Por lo general, el análisis indicará la especificación de requisitos que se deben cubrir.
- El análisis inicial ofrecerá una idea general de lo que se solicita, realizando posteriormente sucesivos refinamientos que servirán para dar respuesta a las siguientes cuestiones:
  - ¿Cuál es la información que ofrecerá la resolución del problema?
  - ¿Qué datos son necesarios para resolver el problema?
- La respuesta a la primera pregunta se identifica con los resultados deseados o las salidas del problema.
- La respuesta a la segunda pregunta indicará qué datos se proporcionan o las entradas del problema.
- En esta fase debemos aprender a analizar la documentación de la empresa , investigar, observar todo lo que rodea el problema y recopilar cualquier información útil.



# Diseño

- Consiste en plantear la aplicación como una única operación global, e ir descomponiéndola en operaciones más sencillas, detalladas y específicas. Consiste en crear los “planos” de nuestra aplicación, detallar los algoritmos y los pasos necesarios para que partiendo de las entradas indicadas obtengamos las salidas deseadas procesando la información.



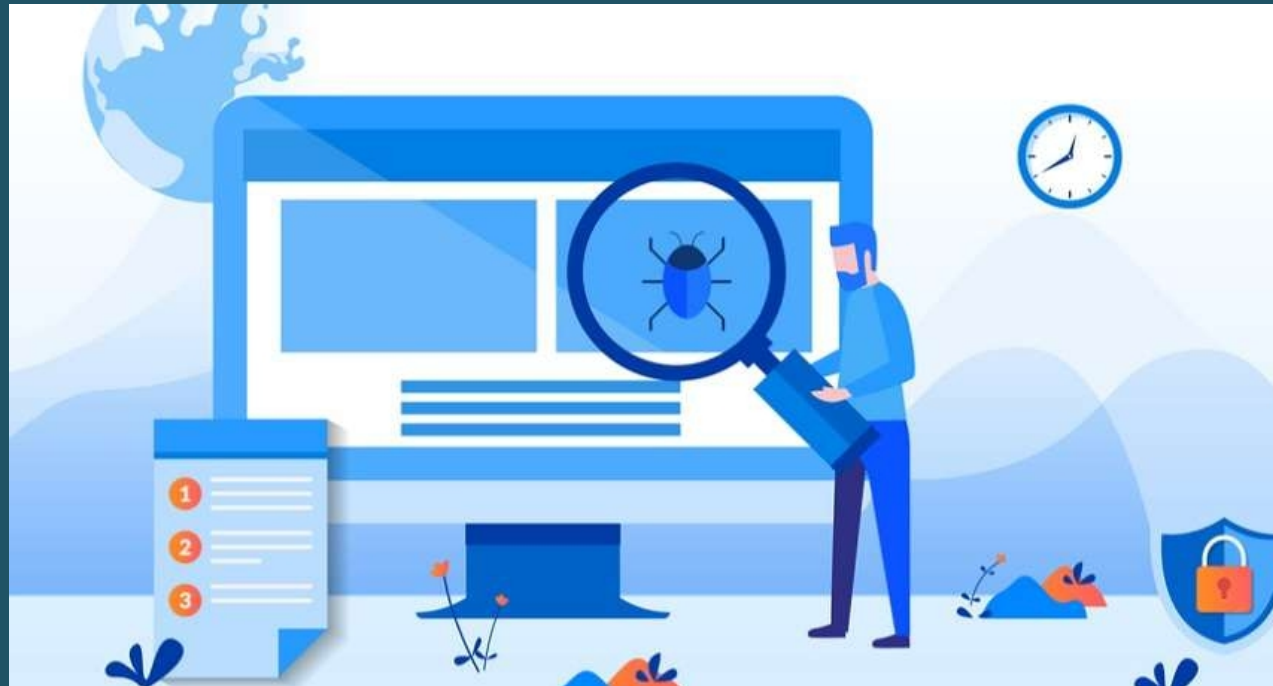
# Codificación

- Esta etapa consiste en transformar o traducir los resultados obtenidos a un determinado lenguaje de programación.

```
ws.on("message", m => {  
  let a = m.split(" ")  
  switch(a[0]){  
    case "connect":  
      if(a[1]){  
        if(clients.has(a[1])){  
          ws.send("connected");  
          ws.id = a[1];  
        }else{  
          ws.id = a[1]  
          clients.set(a[1], {client: {position: {x: 0, y: 0}, id: a[1]}})  
          ws.send("connected")  
        }  
      }  
    }else{  
      let id = Math.random().toString().slice(2, 8)  
      ws.id = id;  
      clients.set(id, {client: {position: {x: 0, y: 0}, id: id}})
```

# Pruebas

- Para esta etapa es necesario implantar la aplicación en el sistema donde va a funcionar, debe ponerse en marcha y comprobar si su funcionamiento es correcto. Utilizando diferentes datos de prueba se verá si el programa responde a los requerimientos especificados, si se detectan nuevos errores, si éstos son bien gestionados y si la interfaz es amigable. Se trata de poner a prueba nuestro programa para ver su respuesta en situaciones difíciles.



# Explotación y Mantenimiento

- Cuando el programa ya está instalado en el sistema y está siendo de utilidad para los usuarios, decimos que se encuentra en fase de explotación.
- Periódicamente será necesario realizar evaluaciones y, si es necesario, llevar a cabo modificaciones para que el programa se adapte o actualice a nuevas necesidades, pudiendo también corregirse errores no detectados anteriormente. Este proceso recibe el nombre de mantenimiento del software.
- **Mantenimiento del software:** es el proceso de mejora y optimización del software después de su entrega al usuario final. Involucra cambios al software en orden de corregir defectos y dependencias encontradas durante su uso, así como la adición de nuevas funcionalidades para mejorar la usabilidad y aplicabilidad del software.







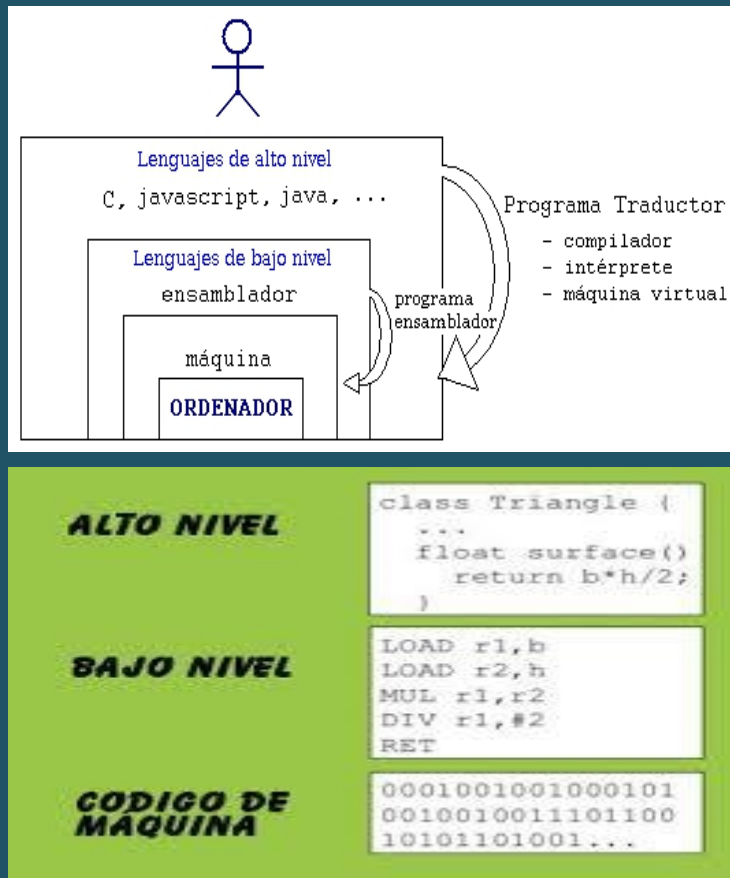
# Tipos de Lenguajes

Cómo clasificarlos

# Conceptos a tener en cuenta

- **Lenguaje de programación:** Conjunto de reglas sintácticas y semánticas, símbolos y palabras especiales establecidas para la construcción de programas. Es un lenguaje artificial, una construcción mental del ser humano para expresar programas.
- **Gramática del lenguaje:** Reglas aplicables al conjunto de símbolos y palabras especiales del lenguaje de programación para la construcción de sentencias correctas.
- **Léxico:** Es el conjunto finito de símbolos y palabras especiales, es el vocabulario del lenguaje.
- **Sintaxis:** Son las posibles combinaciones de los símbolos y palabras especiales. Está relacionada con la forma de los programas.
- **Semántica:** Es el significado de cada construcción del lenguaje, la acción que se llevará a cabo.

# Clasificación según su cercanía al programador



## LENGUAJES DE PROGRAMACIÓN

### ALTO NIVEL vs BAJO NIVEL

Es un lenguaje que entienden **los humanos**.

Son instrucciones **para el procesador**.

```
1 #include <iostream>  
2  
3 using namespace std;  
4  
5 int main() {  
6     cout << "Hola EDteam" << endl;  
7  
8     return 0;  
9  
10 }
```

```
1 COPY    START    2010H  
2 LDX     ZERO  
3 MOVECH  LDCH     STR1, X  
4         STCH     STR2, X  
5         TIX      FOUR  
6         JLT      MOVECH  
7 STR1    BYTE     C'HOLA'  
8 STR2    RESB     4  
9 ZERO    WORD     0  
10 FOUR   WORD     4  
11 END
```

- Está orientado al **software**.
- Utilizan **menos instrucciones** para realizar una acción.
- Te permite programar **aplicaciones y videojuegos**.

- Está orientado al **hardware**.
- Nos ayuda a entender **cómo funcionan las instrucciones** en la computadora.
- Puedes construir **sistemas operativos y núcleos**.

# Clasificación según su cercanía al programador

- **Lenguajes de Bajo Nivel:** son lenguajes totalmente dependientes de la máquina, es decir que el programa que se realiza con este tipo de lenguajes no se pueden migrar o utilizar en otras máquinas.
  - Al estar prácticamente diseñados a medida del hardware, aprovechan al máximo las características del mismo.
  - Dentro de este grupo se encuentran:
    - **El lenguaje máquina:** este lenguaje ordena a la máquina las operaciones fundamentales para su funcionamiento. Consiste en la combinación de 0's y 1's para formar las ordenes entendibles por el hardware de la máquina. Este lenguaje es mucho más rápido que los lenguajes de alto nivel. La desventaja es que son bastantes difíciles de manejar y usar, además de tener códigos fuente enormes donde encontrar un fallo es casi imposible.
    - **El lenguaje ensamblador:** es un derivado del lenguaje máquina y está formado por abreviaturas de letras y números llamadas mnemotécnicos. Con la aparición de este lenguaje se crearon los programas traductores para poder pasar los programas escritos en lenguaje ensamblador a lenguaje máquina. Como ventaja con respecto al código máquina es que los códigos fuentes eran más cortos y los programas creados ocupaban menos memoria. Las desventajas de este lenguaje siguen siendo prácticamente las mismas que las del lenguaje ensamblador, añadiendo la dificultad de tener que aprender un nuevo lenguaje difícil de probar y mantener.

# Clasificación según su cercanía al programador

- **Lenguajes de alto nivel:** son aquellos que se encuentran más cercanos al lenguaje natural que al lenguaje máquina. Se tratan de lenguajes independientes de la arquitectura del ordenador. Por lo que, en principio, un programa escrito en un lenguaje de alto nivel, lo puedes migrar de una máquina a otra sin ningún tipo de problema.
  - Estos lenguajes permiten al programador olvidarse por completo del funcionamiento interno de la máquina/s para la que están diseñando el programa. Tan solo necesitan un traductor que entienda el código fuente como las características de la máquina.
  - Suelen usar tipos de datos para la programación y hay lenguajes de propósito general (C#, Java, Visual Basic, etc.) y de propósito específico (como FORTRAN para trabajos científicos, COBOL para programación financiera y LISP/PROLOG para inteligencia artificial).
- **Lenguajes de Medio nivel:** se trata de un término no aceptado por todos pero sí pueden catalogarse así. Estos lenguajes se encuentran en un punto medio entre los dos anteriores. Dentro de estos lenguajes podría situarse C ya que puede acceder a los registros del sistema, trabajar con direcciones de memoria, todas ellas características de lenguajes de bajo nivel y a la vez realizar operaciones de alto nivel.

# Generaciones

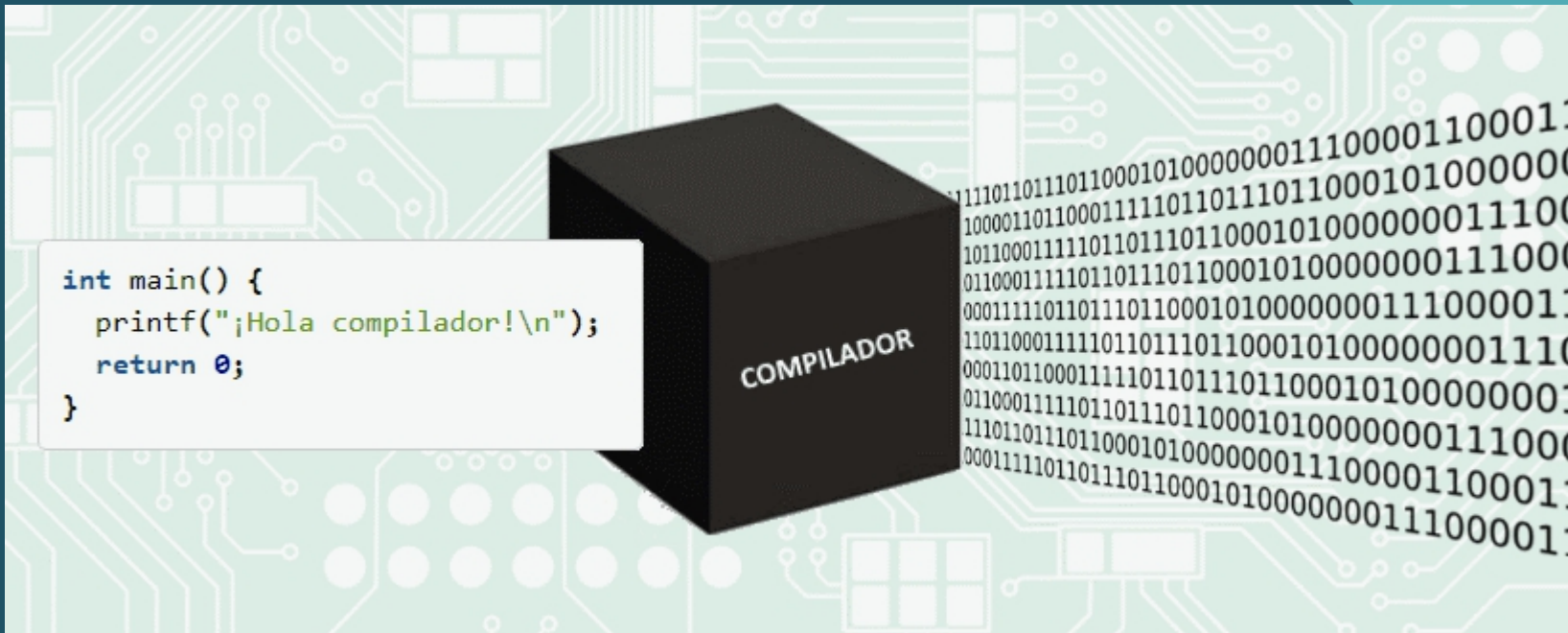
- La evolución de los lenguajes de programación se puede dividir en 5 etapas o generaciones.
  - **Primera generación:** lenguaje maquina.
  - **Segunda generación:** se crearon los primeros lenguajes ensambladores.
  - **Tercera generación:** se crean los primeros lenguajes de alto nivel. Ej. C, Pascal, Cobol...
  - **Cuarta generación.** Son los lenguajes capaces de generar código por si solos, son los llamados RAD, con lo cuales se pueden realizar aplicaciones sin ser un experto en el lenguaje. Aquí también se encuentran los lenguajes orientados a objetos, haciendo posible la reutilización de partes del código para otros programas. Ej. Visual Studio, IntelliJ...
    - Estos lenguajes tienen una estructura lo más parecido al idioma inglés, algunas características son:
      - Acceso a base de datos.
      - Capacidades Gráficas.
      - Generación de código automáticamente.
  - **Quinta generación:** aquí se encuentran los lenguajes orientados a la inteligencia artificial. Ej. LISP

# Compiladores e Intérpretes

¿Cómo se traduce un código?

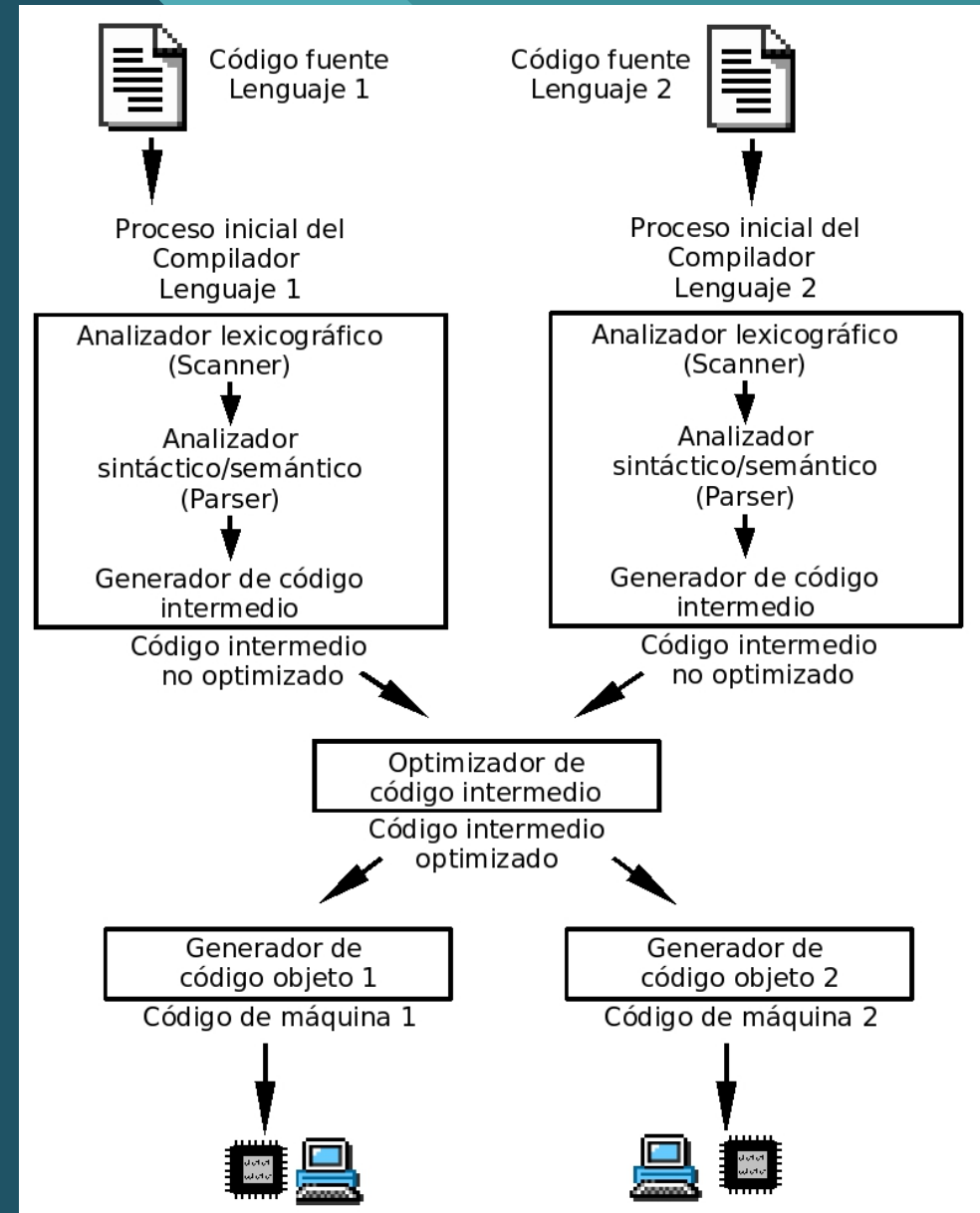
# Proceso de Traducción

- Consiste en transformar el código escrito mediante un lenguaje de programación en código binario o entendible por un ordenador





# Fases de la traducción



# Compilador

- Para obtener un programa ejecutable (en lenguaje máquina) a partir de un lenguaje de alto nivel (compilado), es preciso un proceso de traducción. Se divide en varias fases:
- **A. Compilación**
  - En esta fase se convierte el código fuente a lenguaje máquina, pero no se asignan direcciones absolutas de memoria, pues no se sabe con exactitud a qué lugar va a ir el programa, en su lugar se colocan informaciones para el programa enlazador. Se divide en varias etapas:
    - **1. Preprocesamiento:** no se genera código objeto, básicamente lo que se hace son modificaciones al código fuente (se expanden las macros, pueden eliminarse ciertos módulos de programa -compilación condicional-, pueden añadirse librerías externas, ...). En C las instrucciones para el preprocesador vienen precedidas por # (#include --> incluye un fichero, #define --> (definición de macro) realiza una macrosustitución, #if, #ifdef, #ifndef --> realiza una compilación condicional del bloque).
    - **2. Generación de código intermedio:** se genera un pseudo-código ensamblador, es decir, independiente de la máquina. Es un lenguaje ensamblador muy genérico, no válido para ningún mP, que elimina todas las complejidades de este lenguaje.
    - **3. Generación de código objeto:** el código objeto ya es código máquina, pero ciertas direcciones de memoria todavía no están resueltas y han sido sustituidas por etiquetas (nombres), pues todavía no sabemos en qué lugar de la memoria se va a cargar cada módulo del programa.
  - Entre medias se pueden hacer diversas **optimizaciones**, bien para que se ejecute más rápido, ocupe menos espacio, compilar al lenguaje de oros microprocesadores (compiladores cruzados), obtener el código óptimo para mi microprocesador o hacerlo más compatible con viejos procesadores.

# Compilador

- **B. Enlace**

- Si nuestro programa es muy extenso podemos descomponerlo en módulos que se compilan y prueban por separado, pero para construir el programa completo es necesario enlazarlos. Estos módulos también pueden ser librerías estáticas, que debemos enlazar antes de poder ejecutarlo (las librerías dinámicas se enlazan mientras se ejecuta el programa, es decir, se cargan en la memoria asignada al programa cuando se está ejecutando).

- **C. Ejecución**

- Para ejecutar un programa es necesario cargarlo en memoria. El caso más sencillo es cuando un programa cabe completamente en memoria, si el programa es demasiado grande se tendrá que dividir en segmentos, que se van cargando y descargando según necesidad.

# Compilador

- **Tipos de código.**
- Durante todo este proceso, el código pasa por diferentes estados:
  - **Código Fuente:** es el escrito por los programadores en algún editor de texto. Se escribe usando algún lenguaje de programación de alto nivel y contiene el conjunto de instrucciones necesarias.
  - **Código Objeto:** es el código binario resultado de compilar el código fuente. El código objeto no es directamente inteligible por el ser humano, pero tampoco por la computadora. Es un código intermedio entre el código fuente y el ejecutable y sólo existe si el programa se compila, ya que si se interpreta (traducción línea a línea del código) se traduce y se ejecuta en un solo paso.
  - **Código Ejecutable:** Es el código binario resultante de enlazar los archivos de código objeto con ciertas rutinas y bibliotecas necesarias. El sistema operativo será el encargado de cargar el código ejecutable en memoria RAM y proceder a ejecutarlo. También es conocido como código máquina y ya sí es directamente inteligible por la computadora.
- Hoy en día, con los IDE (entornos de desarrollo integrado) todas estas fases se realizan de manera automática al pulsar un botón. Nosotros lo único que tenemos que hacer es trabajar en un proyecto donde se encuentran codificados todos los programas fuente (en uno o varios módulos).

# Intérprete

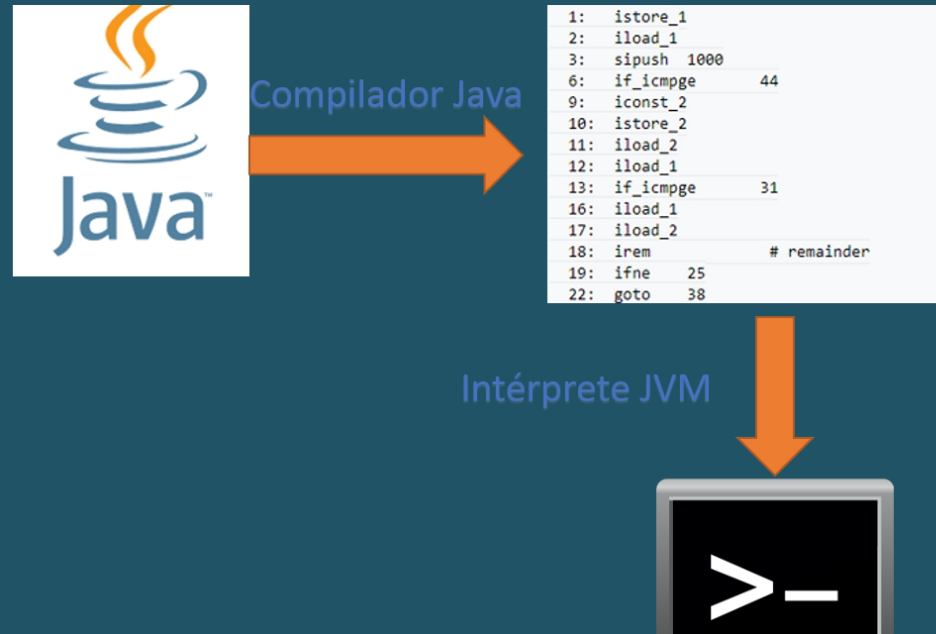
- Es un programa informático capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel.
- Los intérpretes se diferencian de los compiladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción.
- Cada vez que se ejecutan se deben interpretar de nuevo.
- La ejecución del programa por medio de un intérprete es usualmente mucho menos eficiente que la ejecución de un programa compilado. No es eficiente en tiempo porque, o cada instrucción debe pasar por una interpretación en tiempo de ejecución, o como en más recientes implementaciones, el código tiene que ser compilado a una representación intermedia antes de cada ejecución. La máquina virtual es una solución parcial al problema de la eficiencia del tiempo pues la definición del lenguaje intermedio es mucha más cercana al lenguaje de máquina y por lo tanto más fácil de ser traducida en tiempo de ejecución.
- Otra desventaja es la necesidad de un intérprete en la máquina local para poder hacer la ejecución posible

# Lenguajes interpretados y Lenguajes compilados

- **Lenguaje interpretado:** un lenguaje de programación es, por definición, diferente al lenguaje máquina. Por lo tanto, debe traducirse para que el procesador pueda comprenderlo. Un programa escrito en un lenguaje interpretado requiere de un programa auxiliar (el intérprete), que traduce los comandos de los programas según sea necesario.
- **Lenguaje compilado:** un programa escrito en un lenguaje "compilado" se traduce a través de un programa anexo llamado compilador que, a su vez, crea un nuevo archivo independiente que no necesita ningún otro programa para ejecutarse a sí mismo. Este archivo se llama **ejecutable**.
  - Un programa escrito en un lenguaje compilado posee la ventaja de no necesitar un programa anexo para ser ejecutado una vez que ha sido compilado. Además, como sólo es necesaria una traducción, la ejecución se vuelve más rápida. Sin embargo, no es tan flexible como un programa escrito en lenguaje interpretado, ya que cada modificación del archivo fuente (el archivo comprensible para los seres humanos: el archivo a compilar) requiere de la compilación del programa para aplicar los cambios.
  - Por otra parte, un programa compilado tiene la ventaja de garantizar la seguridad del código fuente. En efecto, el lenguaje interpretado, al ser directamente un lenguaje legible, hace que cualquier persona pueda conocer los secretos de fabricación de un programa y, de ese modo, copiar su código o incluso modificarlo. Por lo tanto, existe el riesgo de que los derechos de autor no sean respetados. Por otro lado, ciertas aplicaciones aseguradas necesitan confidencialidad de código para evitar las copias ilegales (transacciones bancarias, pagos en línea, comunicaciones seguras...).

# Lenguajes intermedios

- **Lenguajes intermediarios:** algunos lenguajes pertenecen a ambas categorías (LISP, Java, Python...) dado que el programa escrito en estos lenguajes puede, en ciertos casos, sufrir una fase de compilación intermedia, en un archivo escrito en un lenguaje ininteligible (por lo tanto diferente al archivo fuente) y no ejecutable (requeriría un intérprete o máquina virtual). El objetivo es compilarlos una vez y poder ejecutarlos en distintos sistemas, pues el código generado es independiente del sistema y plataforma, y sí se interpreta o ejecuta por una máquina virtual que sí es dependiente del sistema o plataforma..













# Tipos de Lenguajes





# Lenguajes más usados

- TIOBE, la compañía tras [este índice](#), analiza más de 1.056 millones de líneas de código cada día para clientes de todo el mundo. También monitoriza buscadores y otras fuentes de datos para puntuar los lenguajes de programación según su demanda.

Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		 C	11.83%	-4.12%
2	3	▲	 Python	11.67%	+1.20%
3	2	▼	 Java	11.12%	-2.37%
4	4		 C++	7.13%	+0.01%
5	5		 C#	5.78%	+1.20%
6	6		 Visual Basic	4.62%	+0.50%
7	7		 JavaScript	2.55%	+0.01%
8	14	▲	 Assembly language	2.42%	+1.12%
9	8	▼	 PHP	1.85%	-0.64%
10	10		 SQL	1.80%	+0.04%



“

“Si lo puedes imaginar, lo puedes programar’  
significa que el primer paso es visualizar la  
solución en mi cabeza. Luego, la  
programación me da un lenguaje en el que  
puedo expresar esta idea para que la  
computadora lo entienda y lo ejecute”

- Tu profesor



”

# Recursos

- Twitter: <https://twitter.com/joseluisgonsan>
- GitHub: <https://github.com/joseluisgs>
- Web: <https://joseluisgs.github.io>
- Discord: <https://discord.gg/uv7GcytM>
- Aula Virtual: <https://aulavirtual33.educa.madrid.org/ies.luisvives.leganes/course/view.php?id=245>



# Gracias

José Luis González Sánchez

