

# UA1. Desarrollo de software

Entornos de Desarrollo – 1ºDAW  
I.E.S. Trafalgar

versión 21.02  
Bajo licencia CC BY-NC-SA 4.0



# Contenidos del tema

## 1. Introducción

- 1.1 Mapa conceptual del tema
- 1.2 ¿Qué es un sistema informático?
- 1.3 ¿Qué es el software?

## 2. Programas y aplicaciones informáticas

- 2.1 ¿Qué es una aplicación informática?
- 2.2 Software a medida vs. estándar

## 3. Lenguajes de programación

- 3.1 ¿Qué es un lenguaje de programación?
- 3.2 Tipos de lenguajes de programación
- 3.3 Paradigmas de programación

## 4. El proceso de traducción/compilación

- 4.1 Traductores de código
- 4.1 Interpretación de código
- 4.2 Compilación de código

## 5. Desarrollo de una aplicación

- 5.1 Fases del desarrollo de una aplicación
- 5.2 Documentación
- 5.3 Roles del desarrollo de software
- 5.4 Paradigmas de desarrollo clásicos
- 5.5 Metodologías ágiles

# Contenidos de la sección

## 1. Introducción

- 1.1 Mapa conceptual del tema
- 1.2 ¿Qué es un ordenador?
- 1.3 ¿Qué es el software?

# 1. Introducción

## 1.1 Mapa conceptual del tema



# 1. Introducción

## 1.2 ¿Qué es un ordenador?

- Los primeros ordenadores aparecen a partir de la década de los 40
- Un ordenador acepta **datos de entrada**, los **procesa** y produce unas **salidas** (resultados)
- Se componen de **elementos físicos** (*hardware*) y **lógicos** (*programas, software*)
- Las órdenes de los usuarios deben ser **traducidas** para que las entienda el ordenador

# 1. Introducción

## 1.2 ¿Qué es un ordenador?

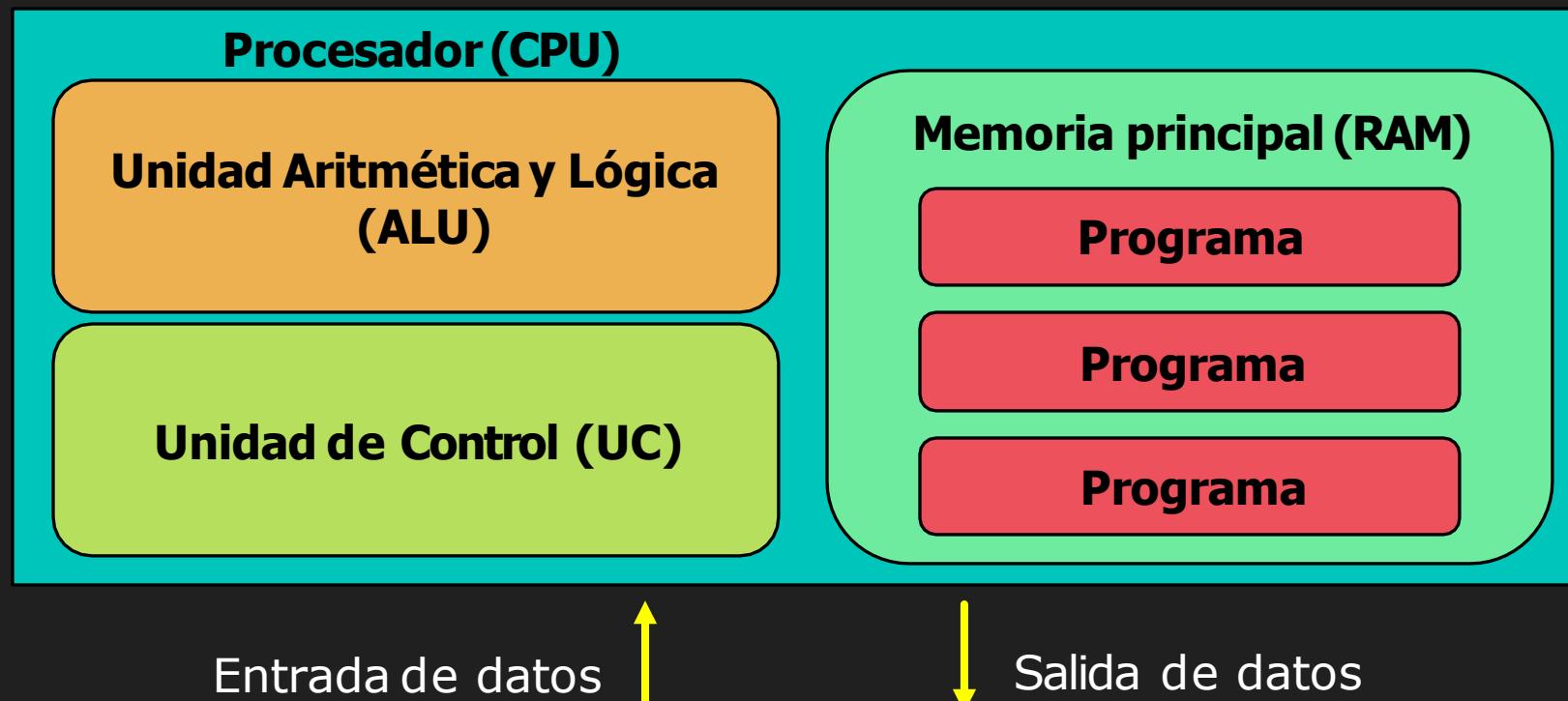


Diagrama de un ordenador

Fuente:

# 1. Introducción

## 1.3 ¿Qué es el software?

- Es la parte **intangible** (o lógica) de un **sistema informático**
- Se **desarrolla** para llevar a cabo una **tarea determinada**
- Se **comunica** con el **hardware** y le dice **qué tiene que hacer**
- Se encarga de **traducir** las **instrucciones de los usuarios**
- El término fue usado por **Charles Babbage** en el siglo XIX
- **Alan Turing** coge el relevo para descifrar la máquina **Enigma**

# 1. Introducción

## 1.3 ¿Qué es el software?

- Algunos **conceptos clave** que conviene tener en cuenta:
  1. El software se **desarrolla**, no se fabrica
  2. El software es **lógico** (intangible), no es un elemento físico
  3. El software **no se estropea** (por lo general), y una copia produce un clon del software original
  4. El software puede desarrollarse a **medida** (aplicaciones desarrolladas a medida)
  5. El software **posibilita el uso** del ordenador
  6. El software se desarrolla usando un **lenguaje de programación** y una **metodología de programación**

# Contenidos de la sección

## 2. Programas y aplicaciones informáticas

- 2.1 ¿Qué es una aplicación informática?
- 2.2 Software a medida vs. estándar

# 2. Programas y aplicaciones informáticas

## 1. ¿Qué es una aplicación informática?

- Un **programa**, o **aplicación informática**, se compone de un **conjunto de instrucciones**
- Una **aplicación** se **desarrolla** usando un **lenguaje de programación**
- Las **instrucciones** le indican al ordenador el **programa** o **pasos** que debe ejecutar
- Si el ordenador **no entiende** alguna instrucción, lo notificará mediante un **mensaje**
- En este ciclo pasaremos de ser **usuarios** a **programadores de aplicaciones**

## 2. Programas y aplicaciones informáticas

### 1. ¿Qué es una aplicación informática?

- Algunos ejemplos de **aplicaciones informáticas**:
  1. Sistemas operativos (Windows, Linux, MacOS)
  2. Aplicaciones de contabilidad y ofimática
  3. Aplicaciones de gestión de bases de datos
  4. Aplicaciones de diseño gráfico
  5. Aplicaciones de correo electrónico
  6. Sistemas de mensajería
  7. Videojuegos

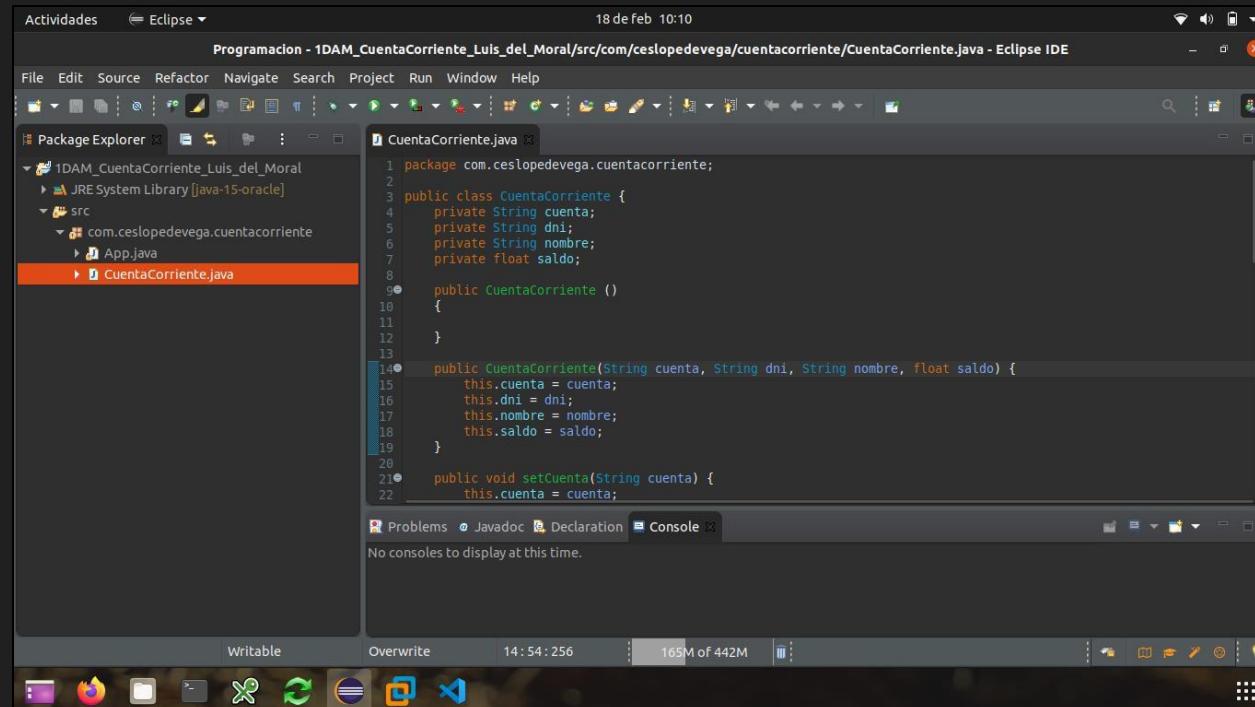
## 2. Programas y aplicaciones informáticas

### 1. ¿Qué es una aplicación informática?

- Algunas definiciones importantes:
  1. **Programa**: conjunto de pasos o instrucciones que indican al ordenador cómo realizar un proceso
  2. **Ejecutar**: consiste en iniciar un programa y ponerlo en ejecución
  3. **Librería**: conjunto de programas y funciones que realizan tareas concretas (BBDD, informes...)
  4. **Entorno de desarrollo (IDE)**: herramienta que facilita y posibilita el desarrollo de software

# 2. Programas y aplicaciones informáticas

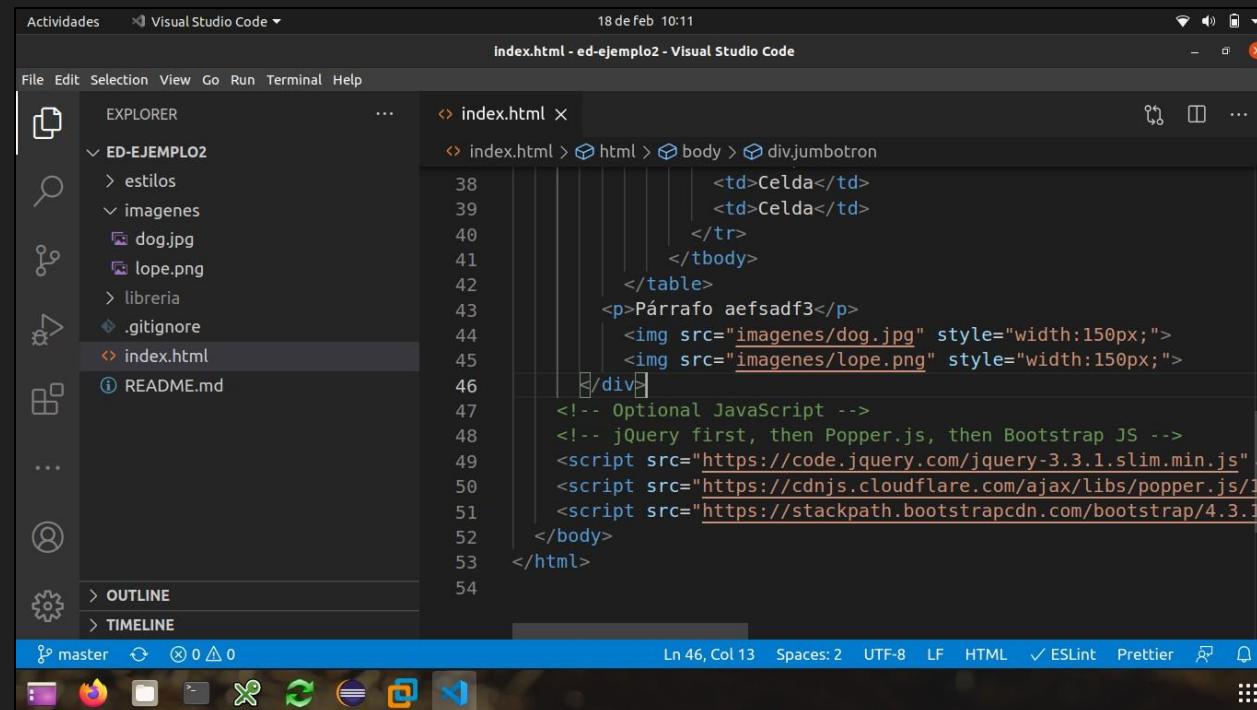
## 2.1 ¿Qué es una aplicación informática?



Entorno de desarrollo Eclipse  
Fuente: [desarrollo propio](#)

# 2. Programas y aplicaciones informáticas

## 2.1 ¿Qué es una aplicación informática?



Entorno de desarrollo Visual Studio Code  
Fuente: [desarrollo propio](#)

# 2. Programas y aplicaciones informáticas

## 2.2 Software a medida vs. estándar

- **Software a medida:**
  - Se desarrolla según las especificaciones o requerimientos de una empresa u organismo
  - Se adecuan a la actividad de dicha empresa u organismo (ejemplo: software propio de facturación)
    - **Características:**
      - Necesita un tiempo de desarrollo (como cualquier software)
      - Se adapta a las necesidades específicas de dicha empresa (puede no ser trasladable a otra empresa)
      - Suele contener errores (como todo software) y necesita de una etapa de adaptación y mantenimiento
      - Es más costoso que el software estándar

# 2. Programas y aplicaciones informáticas

## 2.2 Software a medida vs. estándar

- **Software estándar:**
  - Es un software genérico (válido para cualquier cliente potencial)
  - Resuelve múltiples necesidades, y suele incluir herramientas de configuración
    - **Características:**
      - Se compra ya desarrollado (tan sólo se puede configurar o actualizar llegado el momento)
      - Suele tener menos errores que el software a medida (ha sido mucho más testeado por norma general)
      - Suele ser más barato que el software a medida
      - Suele incluir funciones que nunca serán usadas (y puede carecer de opciones que sí son importantes para la empresa)

# Contenidos de la sección

## 3. Lenguajes de programación

- 3.1 ¿Qué es un lenguaje de programación?
- 3.2 Tipos de lenguajes de programación
- 3.3 Paradigmas de programación

# 3. Lenguajes de programación

## 1. ¿Qué es un lenguaje de programación?

- Se trata de un **lenguaje artificial**, creado por programadores
- Permiten **traducir** las instrucciones de un programa a código comprensible por el ordenador
- Son lenguajes mucho **más complejos y precisos** que el **lenguaje máquina (01010101)**
- Cada instrucción puede dar lugar a muchas instrucciones de lenguaje máquina
- Se componen de un **conjunto de símbolos** (sintaxis) y **reglas** (semántica)

# 3. Lenguajes de programación

## 1. ¿Qué es un lenguaje de programación?

- Algunas definiciones importantes:
  1. **Lenguaje máquina**: lenguaje que sólo es comprensible por el ordenador (**01010101**)
  2. **Programa**: conjunto de instrucciones escritas en un lenguaje de programación
  3. **Instrucción**: cada una de las sentencias u órdenes que forman parte de un programa
  4. **Palabra reservada**: cada uno de los símbolos (léxico) que componen la sintaxis de un lenguaje
  5. **Semántica**: reglas que definen la combinación de los símbolos de un lenguaje de programación

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- Existen diferentes tipos de lenguajes de programación
- Los lenguajes de programación han **evolucionado** a lo largo de los años
- Actualmente, los lenguajes son mas **amigables** y facilitan mucho la tarea del programador
- A veces, la facilidad a la hora de programar implica crear programas más lentos y pesados
- En la actualidad existen infinidad de lenguajes de programación:
  - Fortran, Cobol, Pascal, C, C++, Basic, Visual Basic, C#, Java, JavaScript, PHP, Python...

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Analicemos ahora los tipos de lenguajes de programación más importantes**
  - Lenguaje máquina
  - Lenguaje de medio nivel (ensamblador)
  - Lenguajes de alto nivel

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje máquina:**
  - Posee instrucciones complejas e ininteligibles (**01010101**)
  - Necesita ser traducido (es el único lenguaje que entiende directamente el ordenador)
  - Fue el primer lenguaje usado (muy dependiente del hardware)
  - Difiere para cada procesador (las instrucciones son diferentes de un ordenador a otro)
  - En la actualidad sólo se usa para determinados módulos de un sistema operativo, drivers...

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de medio nivel (ensamblador):**
  - Facilita la labor de programación
  - Se centra en el hardware, pero usa mnemotécnicos (**ADD...**) comprensibles por el programador
  - Hay que compilarlos (traducirlos) para que sean comprensibles por el ordenador
  - Trabaja con los registros del procesador y direcciones de memoria físicas
  - A pesar de estas mejoras, es difícil de entender y de usar

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - Poseen una forma de programar intuitiva y sencilla
  - Son más cercanos a los lenguajes humanos (**IF**, **WHILE**, **DO**...)
  - Incorporan librerías y funciones predeterminadas que ayudan al programador en su labor
  - Suelen ofrecer **frameworks**, que incorporan funciones y componentes que facilitan el desarrollo
  - La mayoría de los lenguajes de programación actuales se engloban en esta categoría
  - Ahora analizaremos las características de algunos de estos lenguajes

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - **Lenguaje C/C++**
    - Uno de los más **potentes** y **utilizados**, ligado a Unix y a Linux
    - Estructurados y **ligados a funciones**
    - Incluyen el concepto de puntero (necesario para gestionar la memoria, pero un concepto complejo)
    - Combinan comandos de alto nivel
    - Programas eficientes, rápidos y pequeños
    - Necesidad de compilar los programas

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- Lenguaje de alto nivel
  - Lenguaje C/C++

```
using namespace std;

int main (int argc, char *argv[])
{
    cout << "Hola mundo" << endl;
    return 0;
}
```

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - **Lenguaje Java**
    - Simplificación de C++ (no admite sobrecarga de operadores ni herencia múltiple)
    - Manejo más eficiente de cadenas de caracteres (String)
    - Lenguaje 100% **Orientado a objetos**
    - Pensado para trabajar con **redes** y **protocolos de red** (TCP/IP, HTTP, HTTPS...)
    - Portable, seguro y permite la programación concurrente
    - Es un lenguaje virtual interpretado

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - **Lenguaje Java**

```
public class HolaMundo {  
    public static void main (String[] args) {  
        System.out.println("Hola mundo");  
    }  
}
```

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - **Lenguaje Python**
    - Lenguaje de alto nivel muy portable
    - Lenguaje interpretado
    - Orientado a objetos
    - Permite incrustarse en otros lenguajes como C/C++
    - Uno de los lenguajes más simples y sencillos de aprender

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - **Lenguaje Python**

```
# Hola mundo en Python
print ("Hola mundo")
```

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - **Lenguaje JavaScript**
    - Se utiliza mucho en programación web (Angular, React...)
    - Se parece mucho a Java, C y C++
    - Es un lenguaje de scripting, seguro y fiable
    - Se ejecuta en el lado del cliente (Frontend)
    - Su código es visible (hay que tener en cuenta aspectos de seguridad)

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - **Lenguaje JavaScript**

```
<script type="text/javascript">
    alert("Hola mundo");
</script>
```

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - **Lenguaje PHP**
    - Lenguaje web de propósito general usado en el servidor (Backend)
    - Se emplea en aplicaciones como Prestashop, WordPress...
    - Es un lenguaje multiplataforma
    - Orientado al desarrollo de webs dinámicas
    - Buena integración con MySQL
    - Orientado a objetos
    - Lenguaje interpretado

# 3. Lenguajes de programación

## 3.2 Tipos de lenguajes de programación

- **Lenguaje de alto nivel**
  - **Lenguaje PHP**

```
<?php  
    echo '<p>Hola mundo</p>';  
?>
```

# 3. Lenguajes de programación

## 3.3 Paradigmas de programación

- Los lenguajes de programación pueden usarse de formas diferentes
- Un paradigma de programación define la forma en la que se desarrolla el programa
- Existen distintos tipos de paradigmas:
  1. Programación estructurada o imperativa
  2. Programación funcional
  3. Programación orientada a objetos
  4. Programación lógica

# 3. Lenguajes de programación

## 3.3 Paradigmas de programación

### ■ Programación estructurada

- Consiste en una secuencia ordenada y organizada de instrucciones
- Es fácil de entender
- Programas sencillos y rápidos
- Posee tres estructuras básicas: secuencia, selección e iteración
- **Inconveniente:** un único bloque de programa (es inmanejable si crece)

# 3. Lenguajes de programación

## 3.3 Paradigmas de programación

- **Programación funcional**
  - Consiste en descomponer el programa en funciones o módulos
  - Programa modular y estructurado
  - **Inconveniente:** el programa puede crecer en gran medida y ser complejo

# 3. Lenguajes de programación

## 3.3 Paradigmas de programación

- **Programación orientada a objetos**
  - Consiste en representar entidades del mundo real
  - Permite reutilizar código
  - Principio de separación de responsabilidades
  - Patrones de diseño y paradigmas avanzados (MVC...)
  - Polimorfismo, herencia y encapsulación

# 3. Lenguajes de programación

## 3.3 Paradigmas de programación

- **Programación lógica**
  - Consiste en representar predicados y relaciones
  - Uso de lógica de predicados de primer orden
  - Muy utilizado en aplicaciones de **inteligencia artificial**

# Contenidos de la sección

## 4. El proceso de traducción/compilación

- 4.1 Traductores de código
- 4.2 Interpretación de código
- 4.3 Compilación de código

# 4. El proceso de traducción/compilación

## 1. Traductores de código

- Un **traductor** traduce un lenguaje de alto nivel a ensamblador o lenguaje máquina
- Existen dos tipos de traductores: **intérpretes** y **compiladores**
- Asimismo, en función de la forma de ejecutar un lenguaje existen los siguientes tipos:
  - Lenguajes compilados
  - Lenguajes interpretados
  - Lenguajes virtuales

# 4. El proceso de traducción/compilación

## 1. Traductores de código

- **Lenguajes compilados**
  - Necesitan un **compilador** para **traducir** el código fuente a **código máquina**
  - Se ejecutan más rápida que los programas interpretados o virtuales
  - Precisan de un **programa enlazador** que permite unir el código objeto con el código objeto de librerías
  - Aunque el código es más **seguro**, no son tan flexibles para modificarlos como los lenguajes interpretados
  - **Ejemplo:** C/C++

# 4. El proceso de traducción/compilación

## 1. Traductores de código

- **Lenguajes interpretados**
  - No generan código objeto
  - El **intérprete** es un programa que tiene que estar cargado en memoria
  - El intérprete “interpreta” cada sentencia del programa y lo ejecuta
  - Las instrucciones se traducen “**on the fly**” (al vuelo), a medida que van ejecutándose
  - **Ejemplo:** PHP

# 4. El proceso de traducción/compilación

## 1. Traductores de código

- **Lenguajes virtuales**
  - Son más **portables** que los lenguajes compilados
  - El código se genera tras la compilación en un **código intermedio** o **bytecode**
  - El código puede ser interpretado por una **máquina virtual** instalada en cualquier equipo
  - Son más lentos pero más versátiles
  - **Ejemplos:** Java

# 4. El proceso de traducción/compilación

## 1. Traductores de código

- **Algunas definiciones importantes**
  - **Código fuente**: código de un programa, escrito por un programador en un lenguaje de programación
  - **Compilar**: proceso que traduce el código fuente en código objeto (comprendible por un ordenador)
  - **Código objeto**: código máquina generado tras la compilación del código fuente
  - **Librería**: código externo que se incluye al programa para proporcionar funcionalidad adicional
  - **Archivo ejecutable**: programa ejecutable que puede ser ejecutado en el ordenador
    - Incluye librerías (tras ser enlazadas), complementos, módulos...

# 4. El proceso de traducción/compilación

## 4.2 Interpretación de código

- Un intérprete **traduce** el código fuente **línea a línea**
- El intérprete tiene que estar en memoria ejecutándose para ejecutar el programa
- El código fuente del programa también se carga en memoria para poder ser interpretado

# 4. El proceso de traducción/compilación

## 4.2 Interpretación de código



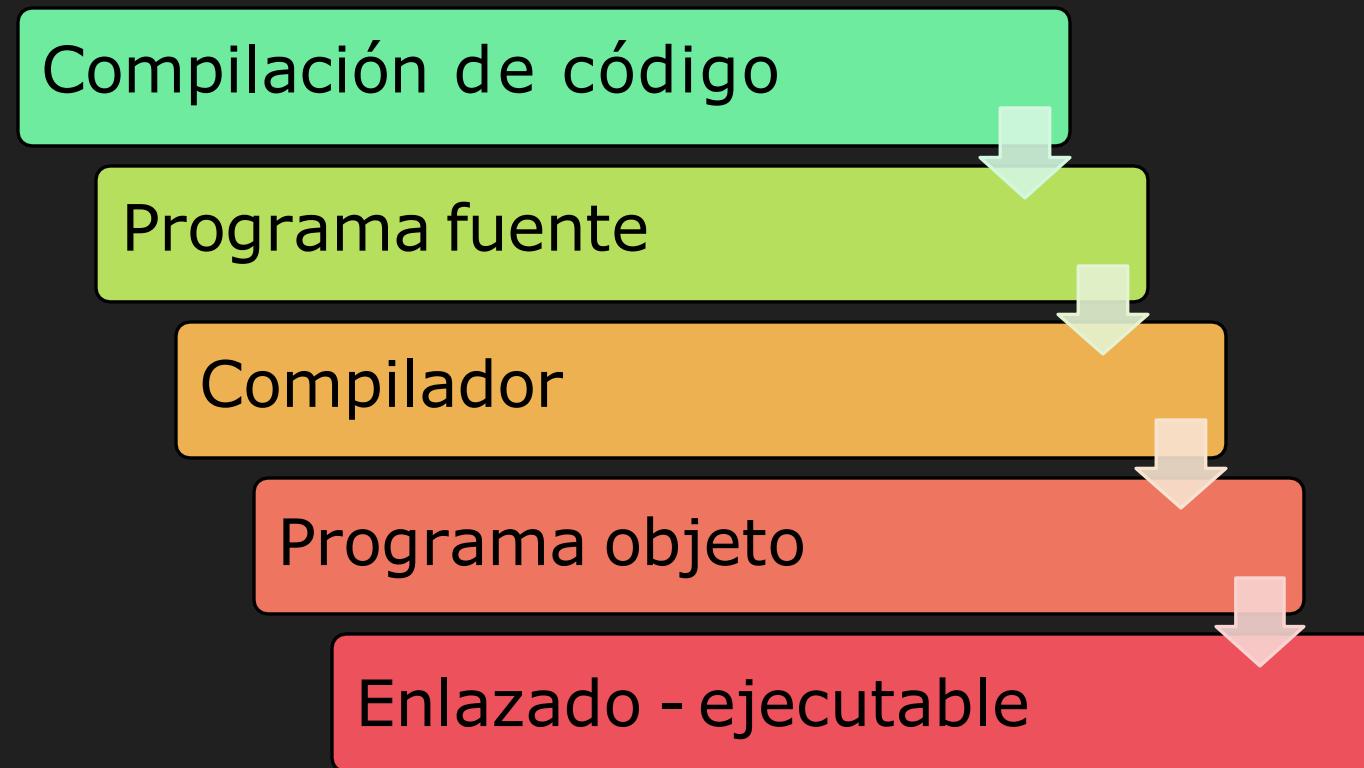
# 4. El proceso de traducción/compilación

## 4.3 Compilación de código

- Un compilador **traduce** código fuente a **código máquina**
- El compilador se instala en cada máquina en la que queramos compilar el programa
- El compilador depende de la arquitectura hardware de la máquina
- El proceso de compilación supone las siguientes fases:
  - **Preprocesado**: se traducen y se ejecutan los comandos de preprocesamiento
  - **Generación de código intermedio**: generación de código máquina
  - **Enlazado**: se enlazan el código objeto con librerías externas

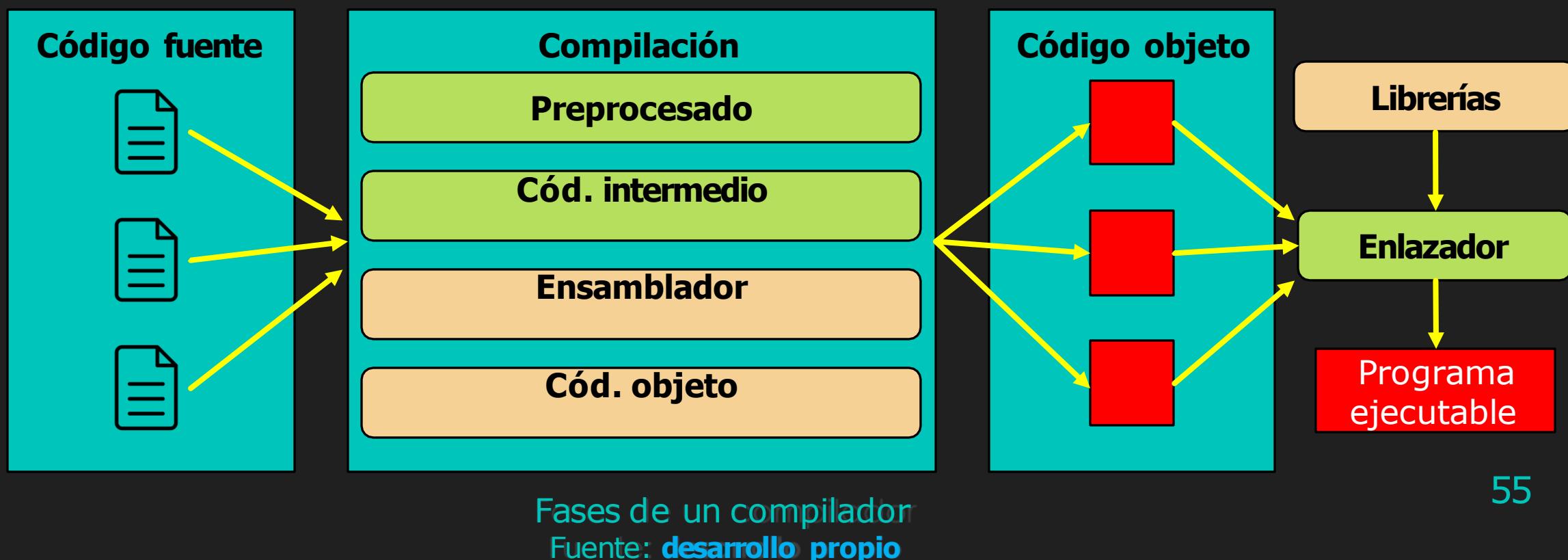
# 4. El proceso de traducción/compilación

## 4.3 Compilación de código



# 4. El proceso de traducción/compilación

## 4.3 Compilación de código



# Contenidos de la sección

## 5. Desarrollo de una aplicación

- 5.1 Fases del desarrollo de una aplicación
- 5.2 Documentación
- 5.3 Roles del desarrollo de software
- 5.4 Paradigmas de desarrollo clásicos
- 5.5 Metodologías ágiles

# 5. Desarrollo de una aplicación

## 1. Fases del desarrollo de una aplicación

- El software se desarrolla siguiendo un **paradigma** o una **metodología de desarrollo**
- Generalmente, los paradigmas suelen tener varias **fases** en común
- A continuación, se analizarán las fases más comunes (estas pueden solaparse):



# 5. Desarrollo de una aplicación

## 1. Fases del desarrollo de una aplicación

- **1. Fase inicial**
  - Planificación del proyecto
  - Estimación de costes (viabilidad del proyecto)
  - Es la fase más **compleja**
  - Precisa de expertos en planificación de proyectos
  - Se desarrollan documentos muy importantes para el proyecto (viabilidad, estimación...)

# 5. Desarrollo de una aplicación

## 1. Fases del desarrollo de una aplicación

### ■ 2. Fase de análisis

- Consiste en **analizar** el problema
- Recopilar, examinar y formular los requisitos del cliente (especificación de requisitos)
- Análisis de restricciones, entrevistas con el cliente y los usuarios finales
- Se genera un **documento vinculante** (a modo de contrato) entre el cliente y el desarrollador

# 5. Desarrollo de una aplicación

## 1. Fases del desarrollo de una aplicación

### ■ 3. Fase de diseño

- Consiste en determinar los **requisitos** generales de la arquitectura de la aplicación
- Se define cada subconjunto de la aplicación
- Los documentos son mucho más técnicos
- Esta fase involucra a los jefes de proyecto, arquitectos de software y analistas
- Los programadores aún no intervienen en esta fase

# 5. Desarrollo de una aplicación

## 1. Fases del desarrollo de una aplicación

- **4. Fase de implementación**
  - Consiste en **implementar** el software usando lenguajes de programación, librerías, frameworks...
  - Se crea documentación muy detallada en la que se incluye y documenta el código fuente
  - Parte del código suele comentarse sobre el propio código fuente generado
  - Se detallan las entradas, salidas, parámetros... de cada uno de los módulos del programa.
  - El detalle es máximo, pensando en el mantenimiento y soporte futuro que tendrá el programa

# 5. Desarrollo de una aplicación

## 1. Fases del desarrollo de una aplicación

- **5. Pruebas**
  - Se realizan pruebas para garantizar que la aplicación se ha programado según las especificaciones
  - A modo preliminar, se pueden considerar dos categorías de pruebas:
    - **Pruebas funcionales:** se prueba que al aplicación hace lo que tiene que hacer (con el cliente)
    - **Pruebas estructurales:** se efectúan pruebas técnicas sobre el sistema (estrés, carga, integración...)
  - En temas posteriores se abordará la **tipología** y el **desarrollo** de las pruebas de software

# 5. Desarrollo de una aplicación

## 1. Fases del desarrollo de una aplicación

- **6. Explotación**
  - Se instala el software en el entorno real de uso
  - Se trabaja con el software de forma cotidiana (nuevas necesidades, incidencias...)
  - Se recogen los errores y las correcciones en un nuevo documento de mantenimiento
  - Los programadores y analistas revisan esos **fallos** para mejorar el software y **aprender** de los errores

# 5. Desarrollo de una aplicación

## 1. Fases del desarrollo de una aplicación

- **7. Mantenimiento**
  - Se realizan **procedimientos correctivos** sobre el programa
  - Siempre hay que tener delante la documentación técnica de la aplicación
  - Las operaciones de mantenimiento se deben documentar para dejar constancia de los cambios

# 5. Desarrollo de una aplicación

## 1. Fases del desarrollo de una aplicación

### ■ 8. Retirada

- El software ha llegado al **final** de su vida útil
- No resulta rentable seguir ampliándolo o manteniéndolo
- Llegados a este punto, el ciclo puede comenzar de nuevo:
  - Comprando un nuevo software
  - Desarrollando un nuevo software (a medida)

# 5. Desarrollo de una aplicación

## 5.2 Documentación

- Como hemos explicado en la sección anterior, la documentación es vital
- En cada fase se generan uno o más documentos
- La documentación es vital para saber cómo usar la aplicación final
- Como mínimo, cada aplicación debe tener estos documentos:
  - **Manual de usuario**: explica cómo usará el usuario la aplicación
  - **Manual técnico**: documentación dirigida a los técnicos (administradores y programadores)
  - **Manual de instalación**: detalla el proceso de instalación de la aplicación

# 5. Desarrollo de una aplicación

## 5.3 Roles del desarrollo de software

- A continuación, detallamos los roles del proceso de desarrollo de software:
  - **Arquitecto de software:**
    - Decide cómo se realiza el proyecto y cómo va a estructurarse
    - Tiene un amplio conocimiento de las tecnologías, los frameworks y las librerías
    - Decide y forma los recursos del desarrollo de un proyecto

# 5. Desarrollo de una aplicación

## 5.3 Roles del desarrollo de software

- A continuación, detallamos los roles del proceso de desarrollo de software:
  - **Jefe de proyecto:**
    - Dirige el curso del proyecto
    - Puede ser un analista con experiencia, un arquitecto o una persona dedicada a ese puesto en exclusividad
    - Debe saber gestionar un equipo y lidiar con los tiempos
    - Trata de manera continua y fluida con el cliente

# 5. Desarrollo de una aplicación

## 5.3 Roles del desarrollo de software

- A continuación, detallamos los roles del proceso de desarrollo de software:
  - **Analista de sistemas:**
    - Esta persona realiza un estudio exhaustivo del problema que va a llevarse a cabo
    - Efectúa el análisis y el diseño de todo el sistema
    - Este perfil requiere mucha experiencia, y también suele involucrarse en reuniones con el cliente

# 5. Desarrollo de una aplicación

## 5.3 Roles del desarrollo de software

- A continuación, detallamos los roles del proceso de desarrollo de software:
  - **Analista programador:**
    - Esta persona realiza un estudio exhaustivo del problema que va a llevarse a cabo
    - Efectúa el análisis y el diseño de todo el sistema
    - Este perfil requiere mucha experiencia, y también suele involucrarse en reuniones con el cliente

# 5. Desarrollo de una aplicación

## 5.3 Roles del desarrollo de software

- A continuación, detallamos los roles del proceso de desarrollo de software:
  - **Programador:**
    - Conoce en profundidad el lenguaje de programación
    - Se encarga de codificar las tareas encomendadas por el analista o el analista programador
    - Su misión es la de codificar y probar los diferentes módulos de la aplicación

# 5. Desarrollo de una aplicación

## 5.4 Paradigmas de desarrollo clásicos

- Los paradigmas de desarrollo clásicos son inflexibles
- Cada etapa debe finalizar para pasar a la siguiente
- ¿Qué pasa si detectamos un error grave de diseño durante la implementación?
- El paradigma más clásico es el **modelo en cascada**
- En esta asignatura trabajaremos con metodologías ágiles (SCRUM)

# 5. Desarrollo de una aplicación

## 5.4 Paradigmas de desarrollo clásicos



El modelo en cascada es muy inflexible  
Cualquier error detectado en una fase muy tardía implica sobrecoste y desperdicios

# 5. Desarrollo de una aplicación

## 5.5 Metodologías ágiles

- En esta asignatura usaremos **metodologías ágiles**
- Estas metodologías responden mejor ante un **cambio** de especificaciones o un error
- Las metodologías ágiles se basan en el **manifiesto** ágil, que valora:
  - A los individuos y su interacción, por encima de los procesos y las herramientas
  - Al software que funciona, por encima de la documentación exhaustiva
  - A la colaboración con el cliente, por encima de la negociación contractual
  - A la respuesta al cambio, por encima del seguimiento de un plan

# Información complementaria

- **Manifiesto ágil:** [enlace](#)
- **Máquina analítica de Babbage:** [enlace](#)
- **Ada Lovelace, primera informática de la historia:** [enlace](#)
- **Los roles de *Frontend* (video):** [enlace](#)
- **Los roles de *Backend* (video):** [enlace](#)
- **Paradigmas de programación (video):** [enlace](#)
- **Metodologías de desarrollo software (video):** [enlace](#)
- **Áreas de la programación (video):** [enlace](#)
- **Lenguajes de programación (video):** [enlace](#)

# Créditos de las imágenes y figuras

## Cliparts e iconos

- **Obtenidos mediante la herramienta web [IconFinder](#)** (según sus disposiciones):
  - Diapositivas 1, 55
  - Según la plataforma IconFinder, dicho material puede usarse libremente (free commercial use)
  - A fecha de edición de este material, todos los cliparts son free for commercial use (sin restricciones)

## Resto de diagramas, gráficas e imágenes

- Se han desarrollado en **PowerPoint** y se han incrustado en esta presentación
- Todos estos materiales se han desarrollado por el autor
- Para el resto de recursos se han especificado sus fabricantes, propietarios o enlaces
- Si no se especifica copyright, el recurso es de desarrollo propio