# UD1: Desarrollo del software

Entornos de desarrollo

Mª Carmen Safont Richarte

# INDICE



# 1. INTRODUCCIÓN

## CONTENIDOS

# 1. INTRODUCCIÓN

## 1.1. ¿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.

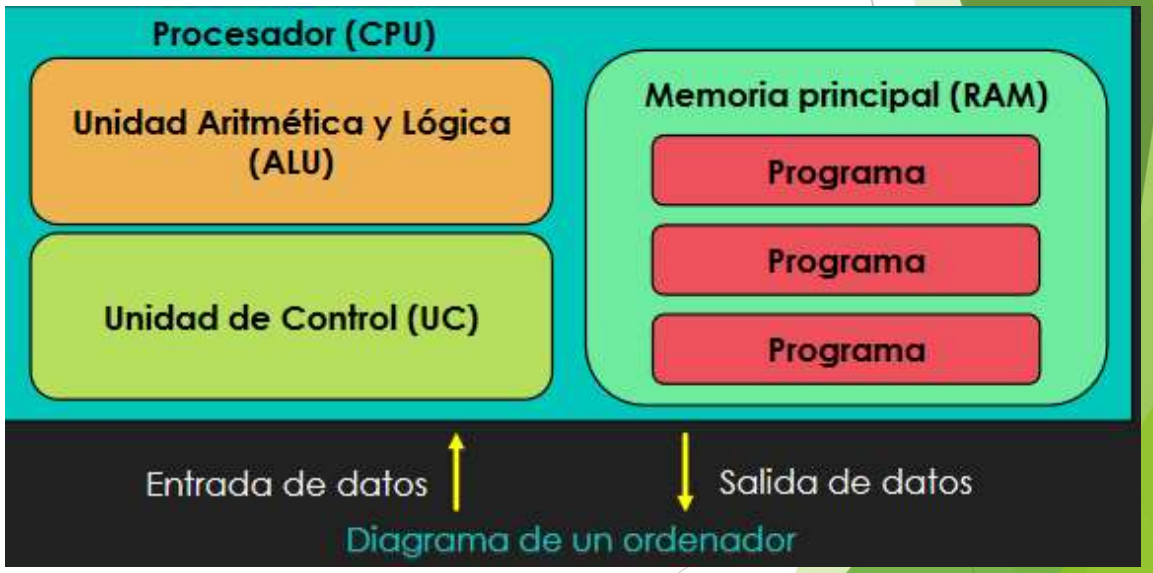
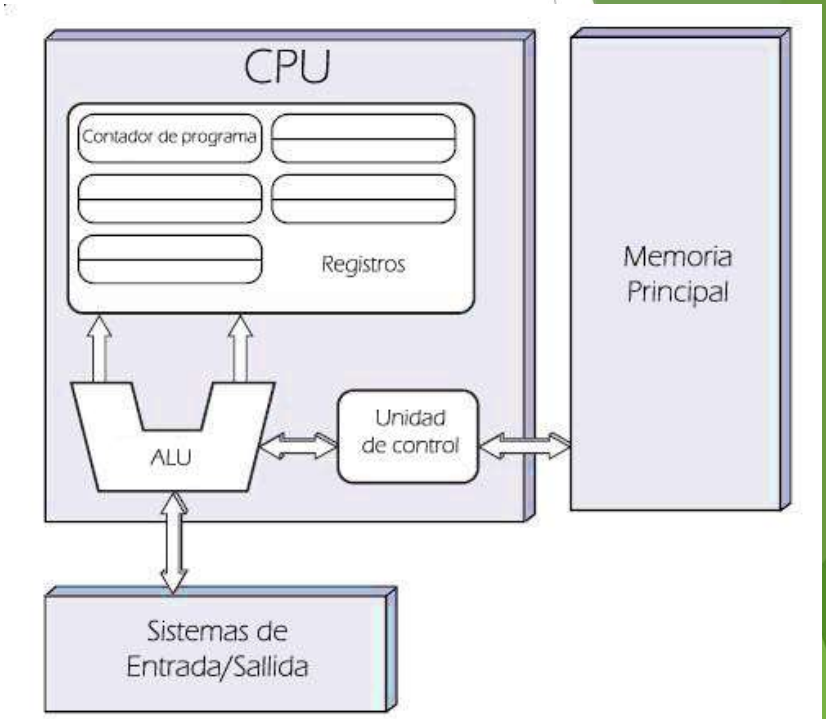


Diagrama de un ordenador

# 1. INTRODUCCIÓN

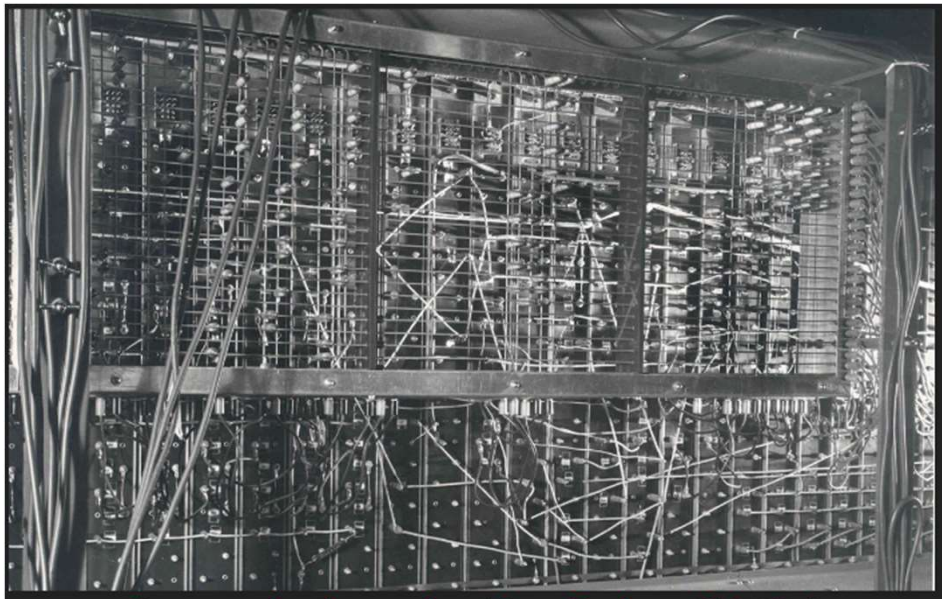
## Arquitectura Von Neumann

- Sistemas de entrada/salida: para recibir y comunicar información
- Memoria Principal: almacenan las instrucciones a ejecutar y los datos
- Unidad de Control: decodifica las instrucciones y su orden de ejecución
- ALU: Ejecuta las instrucciones

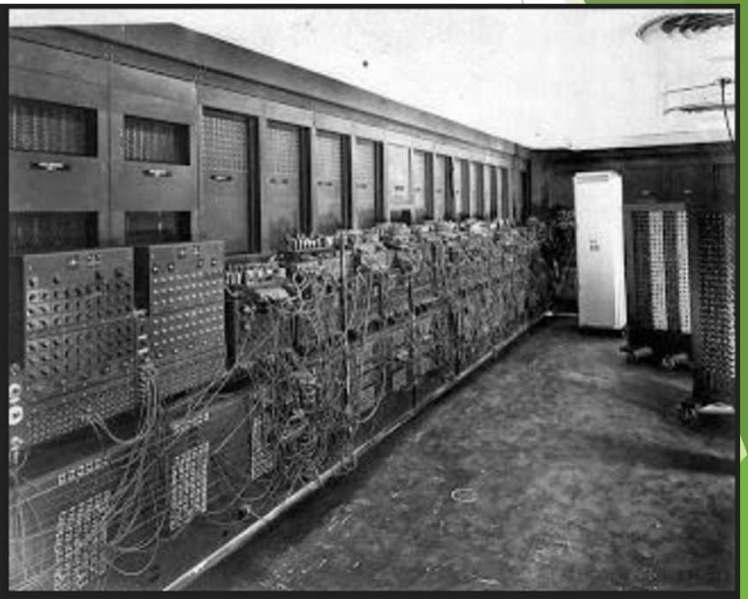


# 1. INTRODUCCIÓN

## 1.1.1. Evolución



Computadora de Alan Turing (1940s)
Fuente: **Autodesk**



ENIAC (1940s)
Fuente: **Helisulbaran**

# 1. INTRODUCCIÓN

## 1.1.1. Evolución

IBM Mainframe 701 (1950s)
Fuente: Time.graphics

IBM PC (1980s)
Fuente: Wikipedia

# 1. INTRODUCCIÓN

## 1.1.1. Evolución

IBM Mainframe 701 (1990s)
Fuente: **NeXTstation**

HP Pavilion x2 (2010-2015)
Fuente: **El País**

# 1. INTRODUCCIÓN

## 1.2. ¿Qué es el software?

Definición formal:

*Es el conjunto de los programas de cómputo, procedimientos, reglas, documentación y datos asociados, que forman parte de las operaciones de un sistema de computación.*

Definición coloquial:

*Entendemos por Software al conjunto de* ***componentes lógicos*** *de un ordenador que permiten la realización de tareas específicas.*

# 1. INTRODUCCIÓN

## 1.2. ¿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

**Conceptos clave a 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**.

# 1. INTRODUCCIÓN

## 1.2.1. Tipos de software

- Software de Sistema: SSOO, controladores, servidores, herramientas de diagnóstico, etc.
- Software de programación: editores de texto, compiladores, intérpretes, depuradores, IDEs, etc.
- Software de aplicación: aplicaciones ofimáticas, bases de datos, videojuegos, software empresarial, software de diseño, etc.

# 1. INTRODUCCIÓN

## 1.2.1. Tipos de software

El SO actúa de intermediario, coordinando el HW y las aplicaciones en funcionamiento.

Las aplicaciones consumen recursos (hardware): Tiempo de CPU para su ejecución, espacio de almacenamiento en memoria RAM, gestión de interrupciones, gestión de dispositivos de entrada/salida.

Las aplicaciones están escritas en lenguaje humano pero los ordenadores solo saben interpretar señales eléctricas o lógicas (0 y 1). Por tanto, es necesario un proceso de traducción:

# 1. INTRODUCCIÓN

## 1.2.2. Tipos de código

### Código Fuente

- Es el conjunto de sentencias (entendibles por el programador) que indica los pasos (instrucciones) a seguir por un ordenador para realizar una tarea (programa o una parte del mismo).
- Está escrito en un lenguaje de programación específico, elegido por el programador (C, C++, C#, Java, Perl, Python, PHP, …).
- NO es directamente ejecutable por el ordenador DEBE ser traducido (compilado) a lenguaje máquina o código objeto.
- Suele estar almacenado en un fichero de texto por lo que se puede abrir con cualquier editor de texto.

# 1. INTRODUCCIÓN

## 1.2.2. Tipos de código

### Código Objeto

- Conjunto de instrucciones y datos escritos en un lenguaje que entiende directamente el ordenador (pero no por el ser humano
- Puede estar escrito en lenguaje máquina o bytecode.
- Es el código resultado de una compilación/interpretación del código fuente.
- Puede estar distribuido en diversos archivos.
- Un enlazador (Linker) se encarga de juntar el código objeto y obtener el programa ejecutable

# 1. INTRODUCCIÓN

## 1.2.2. Tipos de código

### Código Ejecutable

- Es el conjunto de instrucciones compiladas y enlazadas, listas para ser ejecutadas por una máquina (código máquina).
- Es el código que ejecutan los usuarios del sistema y es específico para una plataforma concreta: Windows, Linux, MacOS, …
- El SO será el encargado de cargar este código en la memoria RAM y proceder a ejecutarlo.

# 1. INTRODUCCIÓN

## 1.3. ¿Qué es una aplicación informática/programa?

- 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**

# 1. INTRODUCCIÓN

## 1.3. ¿Qué es una aplicación informática/programa?

**Ejemplos:**

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

# 1. INTRODUCCIÓN

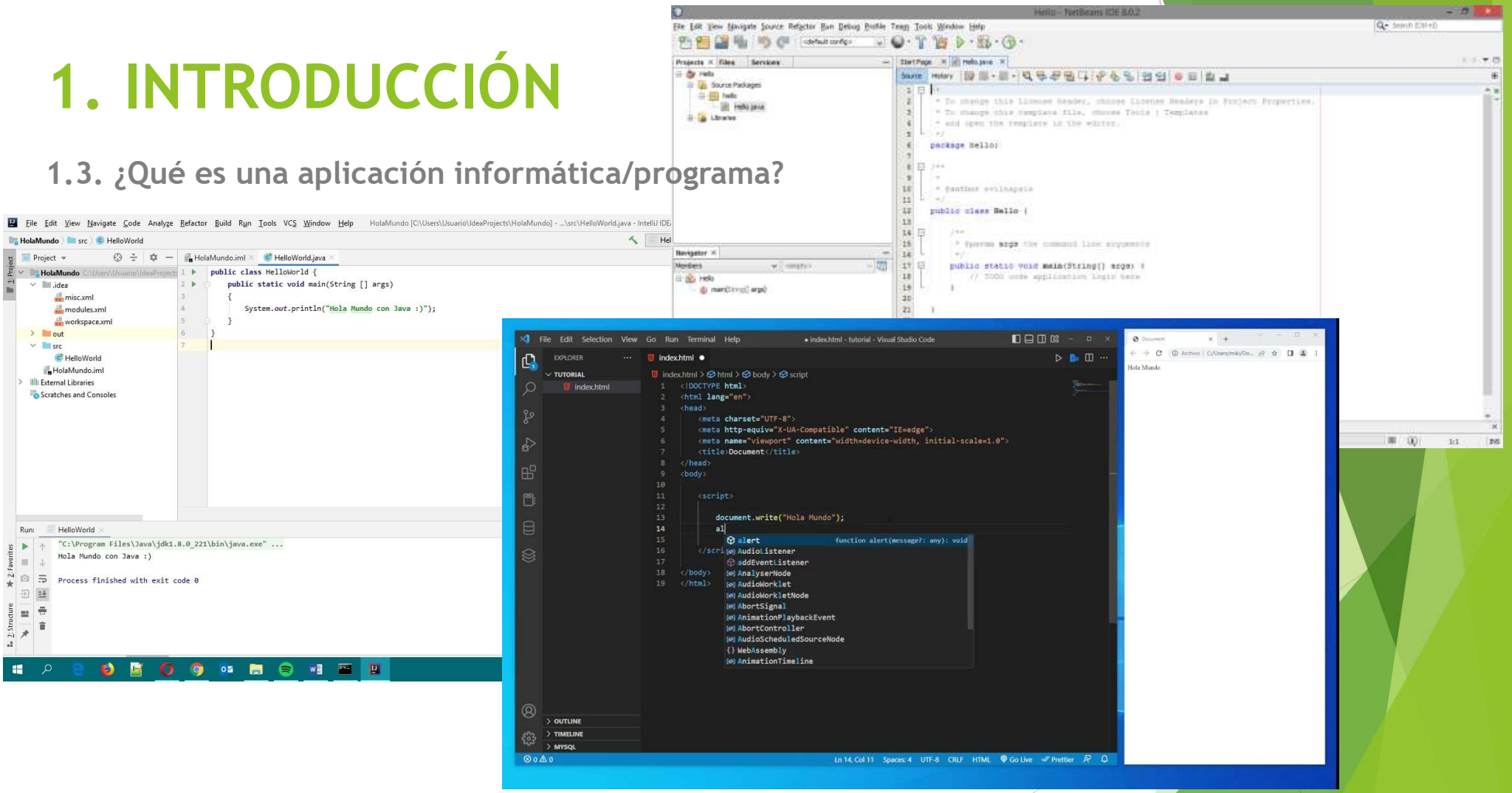
## 1.3. ¿Qué es una aplicación informática/programa?

### Conceptos 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

# 1. INTRODUCCIÓN

## 1.3. ¿Qué es una aplicación informática/programa?



# 2. LENGUAJES DE PROGRAMACIÓN

## CONTENIDOS

2.1. ¿Qué es un lenguaje de programación?

2.2. Tipos de lenguajes de programación

2.3. Paradigmas de programación

2.4. Generación de código: Proceso de traducción/compilación

# 2. LENGUAJES DE PROGRAMACIÓN

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

**Definición:**

*Un lenguaje de programación es un **lenguaje formal** que proporciona al programador la capacidad de escribir una serie de instrucciones en forma de **algoritmos**, con el fin de controlar el comportamiento lógico de un sistema informático.*

# 2. LENGUAJES DE PROGRAMACIÓN

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

### Conceptos a tener en cuenta:

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

# 2. LENGUAJES DE PROGRAMACIÓN

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

- Existen diferentes tipos de lenguajes de programación y diferentes formas de clasificarlos.
- Los lenguajes de programación han ***evolucionado*** a lo largo de los años.
- Actualmente, los lenguajes son más ***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…

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.2. Tipos de lenguajes de programación

### Clasificación según el nivel

LENGUAJE MÁQUINA


LENGUAJE ENSAMBLADOR

(o de bajo nivel)


LENGUAJE DE ALTO NIVEL

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.2. 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…

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.2. Tipos de lenguajes de programación

**LENGUAJE BAJO 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

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.2. Tipos de lenguajes de programación

**LENGUAJE 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.

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.2. Tipos de lenguajes de programación

### Ahora analizaremos las características de algunos de estos lenguajes

```
using namespace std;

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

**Lenguaje C/C++**

- Uno de los más **potentes** y **utilizados**, ligado a Unix y a Linux.
- Estructura y **ligado 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.

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

**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

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.2. Tipos de lenguajes de programación

### Ahora analizaremos las características de algunos de estos lenguajes

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

**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

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

**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)

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

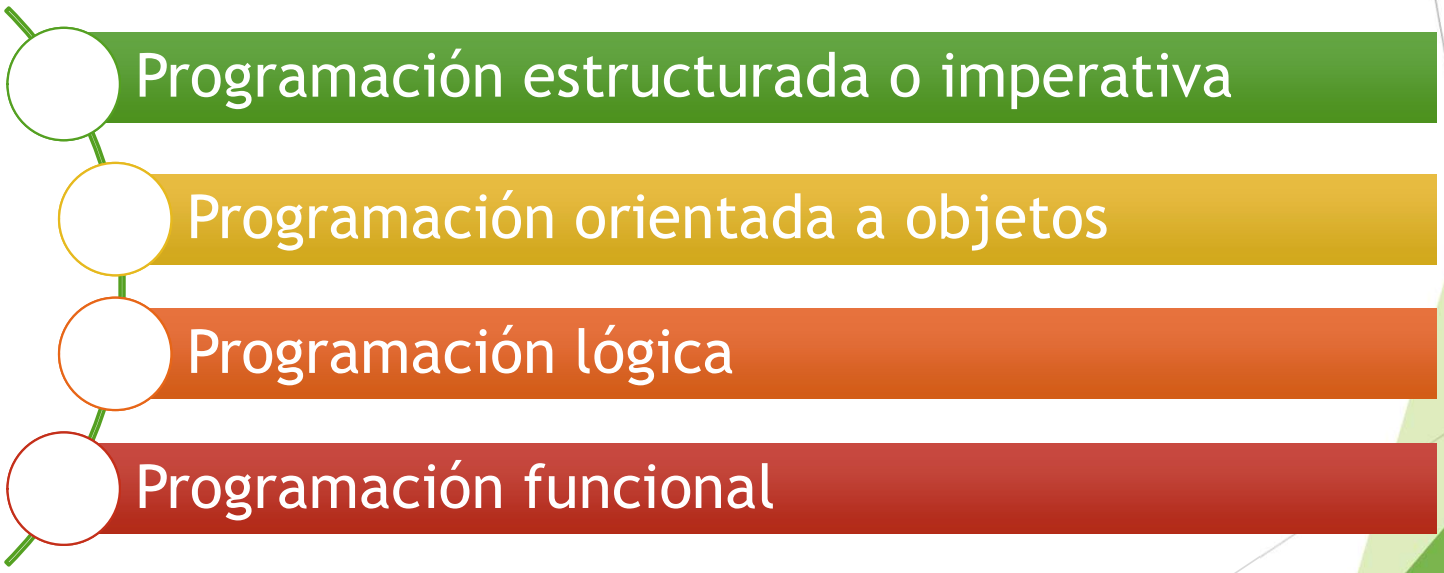
**PHP**

- 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
- Lenguaje interpretado.

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.3. 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



# 2. LENGUAJES DE PROGRAMACIÓN

## 2.3. Paradigmas de programación

### Programación estructurada o imperativa

- 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)

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.3. 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

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.3. 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**

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.3. 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.

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.4. Generación de código: proceso de traducción/compilación

Como hemos mencionado, la tarea de programar implica pasar de un código fuente entendible por humanos a un código máquina.

Esta transformación se realiza mediante un traductor, que puede ser un **compilador** o un **intérprete**, según el lenguaje de programación empleado.

- En un lenguaje compilado, el código fuente es transformado a lenguaje máquina y posteriormente ejecutado en la máquina.
- En un lenguaje interpretado el código fuente es transformado a un lenguaje máquina a medida que es ejecutado.

Así mismo, en función de la forma de ejecutar un lenguaje existen los lenguajes compilados, interpretados o virtuales.

# 2. LENGUAJES DE PROGRAMACIÓN

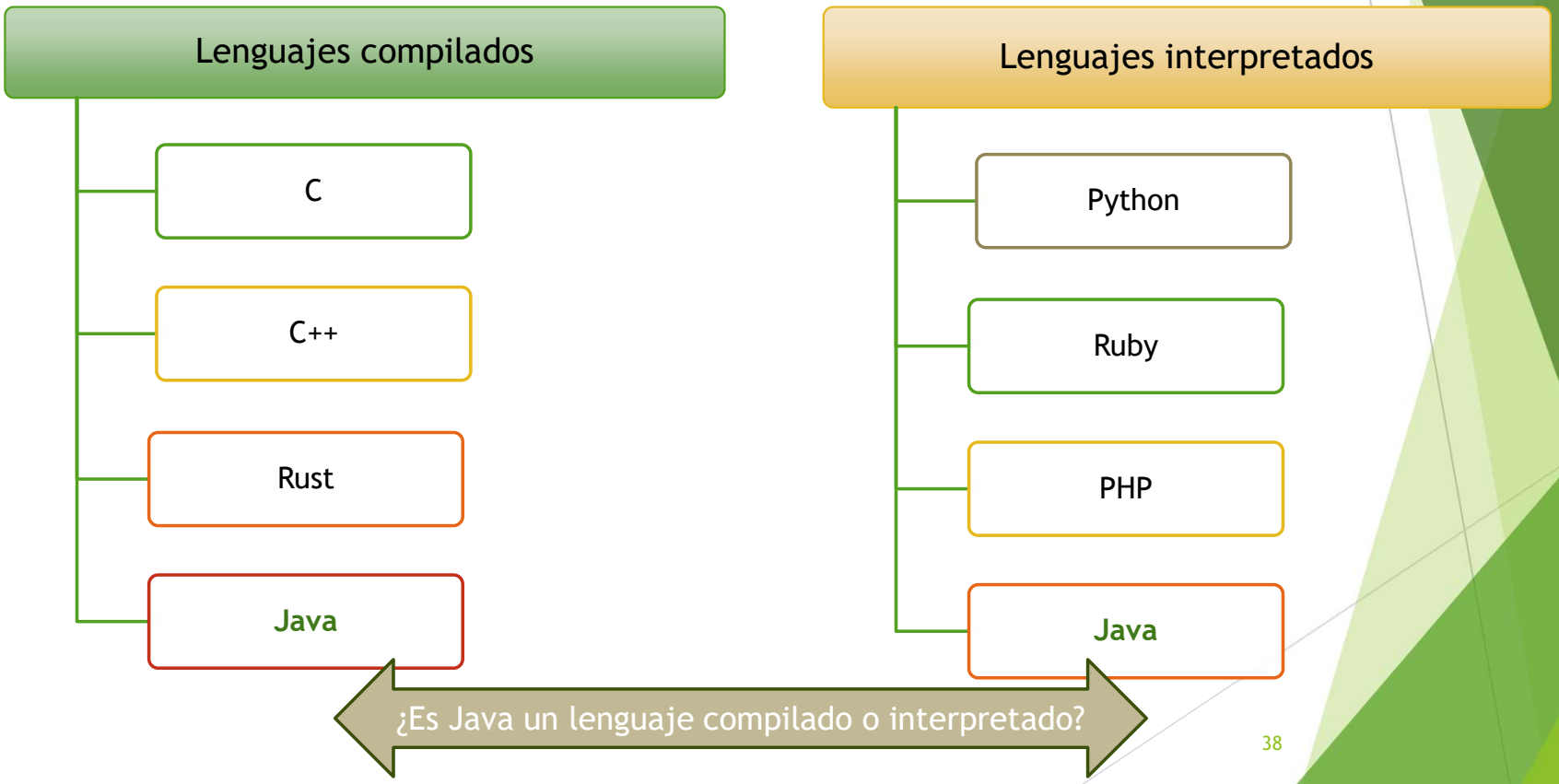
## 2.4. Generación de código: proceso de traducción/compilación

### 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 (comprensible 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…

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.4. Generación de código: proceso de traducción/compilación



# 2. LENGUAJES DE PROGRAMACIÓN

## 2.4.1. 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++

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.4.1. Lenguajes compilados

Un compilador traduce código fuente a código máquina

El compilador se instala en cada máquina en la que queramos compilar el programa

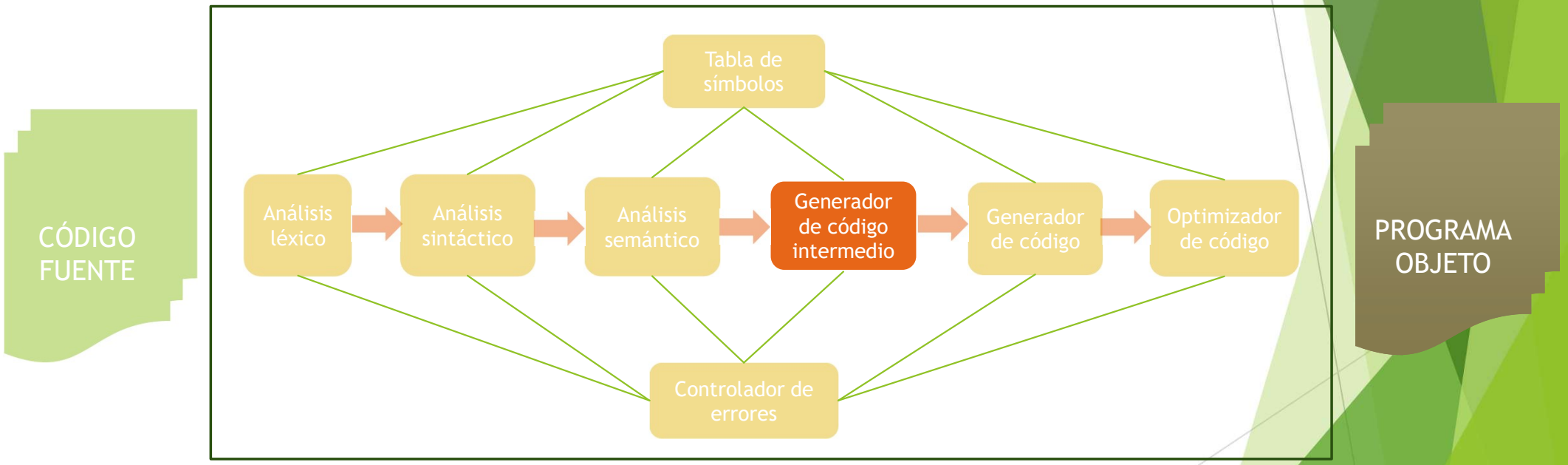
El compilador depende de la arquitectura hardware de la máquina

El proceso de compilación supone las siguientes fases:

- **Preprocesado**: se comprueban el léxico, la sintaxis y la semántica del código.
- **Generación de código intermedio**: generación de código máquina
- **Enlazado**: se enlazan el código objeto con librerías externas

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.4.1. Lenguajes compilados



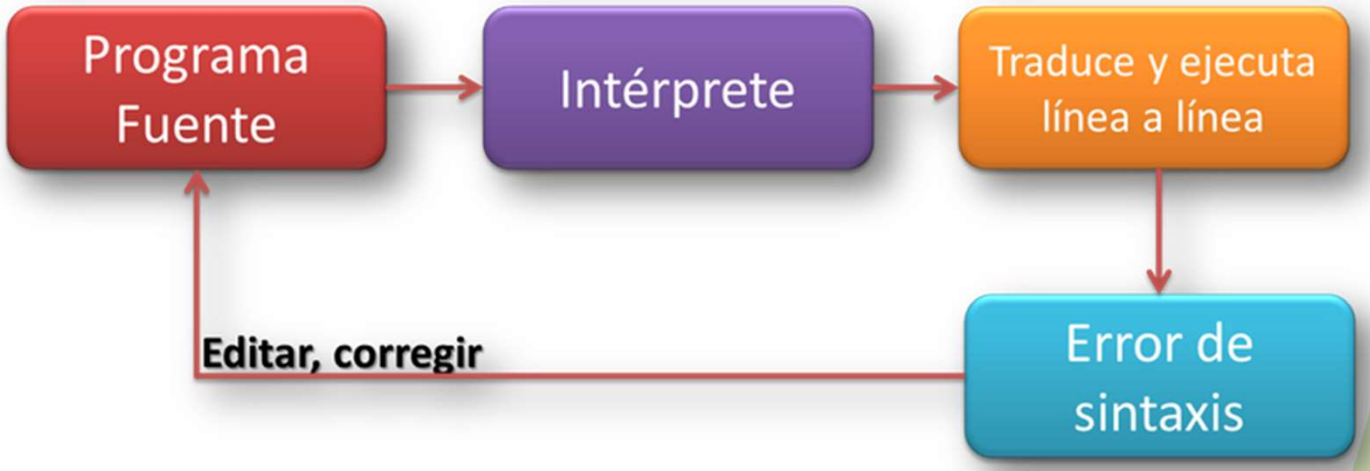
# 2. LENGUAJES DE PROGRAMACIÓN

## 2.4.1. Lenguajes interpretados

- El intérprete tiene que estar en memoria ejecutándose para ejecutar el programa.
- El código fuente del programa también ser carga en memoria para poder ser interpretado.
- Un intérprete traduce el código fuente línea a línea, sin generar código objeto, es decir, "interpreta" cada sentencia del programa y lo ejecuta.
- Ejemplo: PHP

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.4.1. Lenguajes interpretados



# 2. LENGUAJES DE PROGRAMACIÓN

## 2.4.1. 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

# 2. LENGUAJES DE PROGRAMACIÓN

## 2.4.1. Lenguajes virtuales