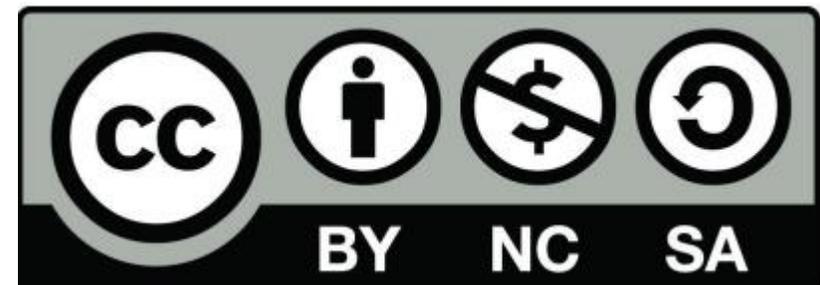


# Tema 1. Problemas, algoritmos y programas

CHRISTOPHER EXPÓSITO IZQUIERDO  
AIRAM EXPÓSITO MÁRQUEZ  
ISRAEL LÓPEZ PLATA  
MARÍA BELÉN MELIÁN BATISTA  
JOSÉ MARCOS MORENO VEGA



# El Software

---

- Son todos aquellos programas que pueden ser ejecutados en un sistema de computación
- Existe una gran variedad de software para realizar innumerables operaciones con el ordenador



# La programación

---

- La programación es el proceso de diseñar, codificar, depurar y mantener el código fuente de programas de ordenador
- Su principal objetivo es la creación de programas con un cometido específico
- Es una rama de la ingeniería que posee una gran complejidad y que requiere una gran especialización por parte del profesional
- Crear un programa exige:
  - Diseño del programa
  - Codificación
  - Testeo

# La programación

---

- Para cualquier ingeniero, el poseer conceptos básicos de programación resulta de gran ayuda en su trabajo
- Proporciona la posibilidad de crear programas para sus propios cometidos
- Ejemplos:
  - Programas de simulación de comportamiento
  - Programas de testeo de posibles hipótesis
  - Programas de ayuda en cálculos, así como representación de la información
- ¡¡Es divertido!!

# La algoritmia

---

- El primer paso para crear un buen software es su fase de **diseño**
- Un programa se forma como un conjunto de algoritmos que resuelven diferentes que se presentan al usuario del software
- Un **algoritmo** se define como el conjunto de instrucciones definidas, ordenadas y finitas, que permiten llevar a cabo una actividad
- El conjunto de instrucciones se debe encontrar ordenado de tal forma que no exista duda sobre que instrucción realizar en cada instante de tiempo

# La algoritmia

---

- La ciencia que realiza el estudio de los algoritmos se le conoce como **algoritmia**
- En la algoritmia se estudian las diferentes formas de creación de algoritmos
- También se analiza la efectividad de los algoritmos diseñados, en términos de eficiencia computacional
- La principal cualidad para la creación de buenos algoritmos es la **INVENTIVA**

# La algoritmia

---

## Ejemplos sencillos de algoritmos

### Arreglar una lámpara

1. Comprobar que la lámpara no funciona
2. ¿Está la lámpara enchufada?
  - 2.a. **No.** Enchufar la lámpara y terminar
  - 2.b. **Si.** Pasar al paso 3
3. ¿Está la bombilla fundida?
  - 3.a. **Si.** Comprar una nueva bombilla y terminar
  - 3.b. **No.** Comprar una nueva lámpara y terminar

### Buscar un partido de fútbol en la TV

1. Poner la TV en el primer canal
2. ¿Están poniendo el partido de fútbol?
  - 2.a. **Si.** Terminar
  - 2.b. **No.**
    - 2.b.1. Pasar al siguiente canal
    - 2.b.2. Ir al paso 2

# La algoritmia

---

- Una forma habitual de representar un algoritmo es mediante **diagramas de flujo**
- Representa el orden que siguen las instrucciones a ejecutar del algoritmo, a través de una línea descendente que empieza en un punto inicial y termina en un punto final
- Permite ver de forma gráfica un algoritmo
- Facilita la comprensión así como el diseño de un algoritmo

# La algoritmia

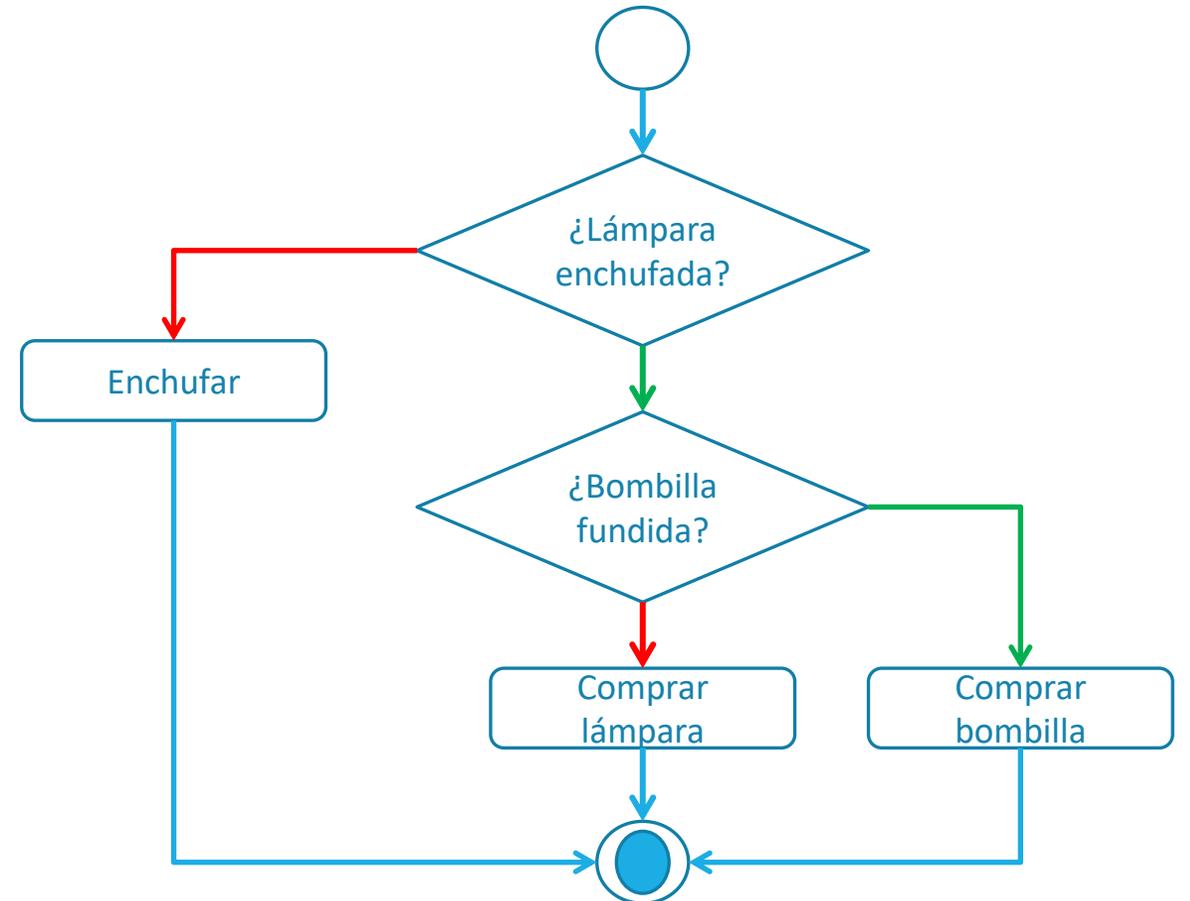
---

- Inicio de flujo 
- Fin de flujo 
- Instrucción del algoritmo 
- Transición entre instrucciones 
- Decisión 

# La algoritmia

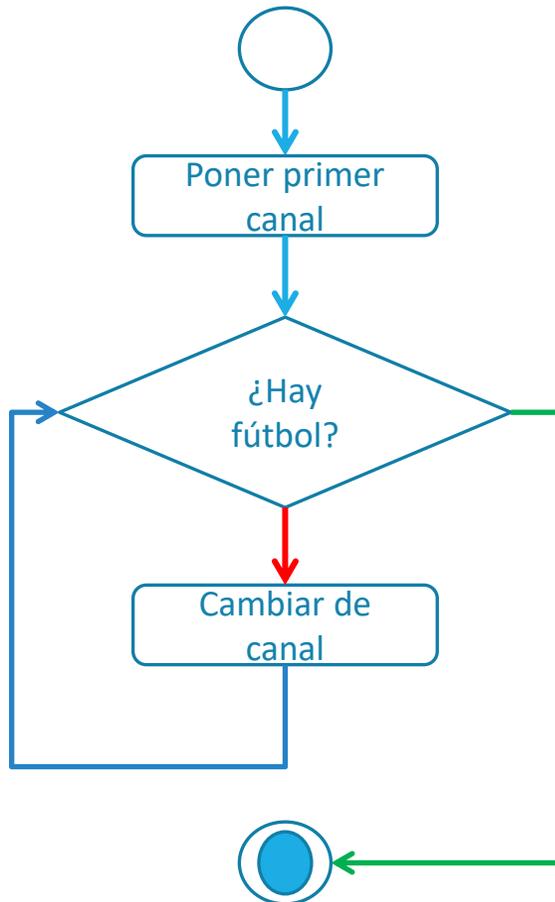
## Arreglar una lámpara

1. Comprobar que la lámpara no funciona
2. ¿Está la lámpara enchufada?
  - 2.a. **No.** Enchufar la lámpara y terminar
  - 2.b. **Si.** Pasar al paso 3
3. ¿Está la bombilla fundida?
  - 3.a. **Si.** Comprar una nueva bombilla y terminar
  - 3.b. **No.** Comprar una nueva lámpara y terminar



# La algoritmia

---



## Buscar un partido de fútbol en la TV

1. Poner la TV en el primer canal
2. ¿Están poniendo el partido de fútbol?
  - 2.a. **Si.** Terminar
  - 2.b. **No.**
    - 2.b.1. Pasar al siguiente canal
    - 2.b.2. Ir al paso 2

# Lenguajes de programación

---

- Una vez se encuentra diseñado un programa, éste debe ser **codificado**
- Para ello se utilizan los **lenguajes de programación**, que permiten proporcionar el conjunto de órdenes de las que se compone el algoritmo diseñado a la máquina
- Existe una gran variedad de lenguajes de programación, adaptados a diferentes necesidades o formas de programar
  - Según el nivel de detalle
  - Según el paradigma utilizado (declarativo, estructurado, orientado a objetos, etc.)
  - Según si se dispone o no de intérprete
  - Etc.



# Tipos de lenguajes

---

- **Lenguaje de bajo nivel**

- Lenguaje cuyas instrucciones son muy cercanas a las que entiende la máquina
- Ensamblador
- **Ventajas:** Rápidos, control total de los recursos del sistema
- **Inconvenientes:** Muy difíciles de dominar

- **Lenguaje de alto nivel**

- El código se asemeja a lo que puede entender un ser humano
- C, C++, Pascal, PHP, Html, Java
- **Ventajas:** Complejos, optimizados, sencillos de aprender
- **Inconvenientes:** No se tiene el control de los recursos del sistema

# Tipos de lenguajes

---

- **Lenguaje interpretado**

- Lenguaje que necesita un **intérprete** para poder ejecutar el programa creado
- Html, PHP, Python, GNU Octave, JavaScript, *Java*
- **Ventajas:** Rápidos, sencillos, multiplataforma
- **Inconvenientes:** Menos posibilidades

- **Lenguaje compilado**

- El código se **compila** previamente para crear el ejecutable
- C, C++, Pascal, COBOL, *Java*
- **Ventajas:** Complejos, optimizados
- **Inconvenientes:** Compilación, dependientes de la plataforma

# Java

---

- Lenguaje compilado, concurrente y orientado a objetos
- A pesar de ser compilado, posee un intérprete que lo hace multiplataforma
- Es uno de los lenguajes con mayor utilización en el mundo
- Posee un gran número de librerías y frameworks



# Java

---

- La primera versión (JDK 1.0) fue lanzada en 1996, por **Sun Microsystems**
- La última versión, llamada Java SE 8, se lanzó en 2014
- Se han lanzado diferentes variantes de Java como:
  - Java SE (Standard Edition). Para aplicaciones de escritorio
  - Java EE (Enterprise Edition). Para aplicaciones web, concretamente en la parte del servidor
  - Java ME (Mobile Edition). Para aplicaciones en dispositivos móviles

# Programa sencillo en Java

---

```
4 public class EjemploHolaMundo {
5
6     /*
7         Punto de entrada del programa Java. Muestra el mensaje
8         "Hola Mundo!" por pantalla.
9     */
10    public static void main(String[] args) {
11        System.out.println("Hola Mundo!");
12    }
13
14 }
```

# Características de Java

---

- Sigue el paradigma de **Programación Orientada a Objetos**, aunque puede usarse a través de programación estructurada
- Es **multiplataforma**, el mismo código se puede ejecutar en diferentes sistemas
- Pensado para el trabajo en red y en remoto
- Optimizado para el uso de la concurrencia
- Fácil de usar. Sintaxis sencilla y muy parecida a otros lenguajes de uso común como C

# Características de Java

---

- Abstrae al programador del uso de la memoria del sistema.
  - No usa punteros
  - Posee un recolector de basura
  
- Con Java se pueden crear programas de diferentes tipos como:
  - Aplicaciones de escritorio
  - Sistemas de servidor
  - Clientes web
  - Aplicaciones móviles (Android)
  - Y mucho mas

# Para utilizar Java

---

- Java es un lenguaje interpretado, por lo que es necesario descargarse el **intérprete Java**
- Java incluye un gran número de librerías que facilitan la labor al programador, se encuentran en el **JDK** (Java Development Kit)
- El intérprete y el JDK se pueden descargar en un solo paquete desde <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
- Para poder programar en Java, ayuda el uso de un Integrated Development Environment (IDE) como NetBeans (<https://netbeans.org/>)
- De cara a la instalación de ambas cosas, debe hacerse en el siguiente orden:
  1. Instalar el JDK
  2. Instalar NetBeans