

# Tema 1. Algoritmos, programas y lenguajes de programación. (I)

## ALGORITMOS

### 1. INTRODUCCIÓN.

Algunos conceptos básicos:

#### DATO

Son representaciones simbólicas de objetos, hechos, instituciones, conocimientos, etc. representados de manera adecuada para su tratamiento por un ordenador.

La palabra proviene del latín datum, forma del verbo dare (dar), que significa *Lo que es dado*.

Ejemplos: 15, Rojo, 25º C, 25 m, 31-10-2011, Febrero, Ana Gómez

#### INFORMACIÓN

Es el resultado de procesar los datos.

Los datos procesados adquieren significado y proporcionan conocimiento.

La información se resume en: Datos + Significado

El principal objetivo de la información es aumentar el conocimiento o reducir la incertidumbre.

Ejemplos

Juan tiene **15** años

**25º C** fue la temperatura de esta mañana.

El edificio mide **25 m** de altura

La fecha de hoy es **31-10-2011**

El mes de nacimiento es **Febrero**.

**María Pérez** es la ganadora del premio.

El color predominante es el **Rojo**.

#### PROBLEMA

El diccionario de la RAE define problema como:

Planteamiento de una situación cuya respuesta desconocida debe obtenerse a través de métodos científicos.

Nuestro objetivo es aprender a resolver problemas mediante un ordenador.

Un programador es una persona que resuelve problemas.

Para llegar a ser un programador eficaz se necesita aprender a resolver problemas de un modo riguroso y sistemático.

La resolución de un problema mediante un ordenador exige al menos los siguientes pasos:

1. Definición del problema.
2. Diseño del algoritmo.
3. Transformación del algoritmo en un programa.
4. Ejecución y validación del programa.

## 2. ALGORITMOS.

Un **algoritmo** se puede definir como el conjunto ordenado y finito de operaciones que permiten hallar la solución a un problema.

La palabra algoritmo proviene de Mohamed al-Khowârizmi, matemático persa que vivió durante el siglo IX y alcanzó gran reputación por el enunciado de las reglas paso a paso para sumar, restar, multiplicar y dividir números decimales.

La traducción al latín del apellido en la palabra *algorismus* derivó posteriormente en Algoritmo.

Un ejemplo de algoritmo en la vida diaria puede ser una receta de cocina.

Una receta de cocina consiste en una serie de pasos detallados a seguir para obtener un plato determinado.

Otro ejemplo: algoritmo para calcular el área y la longitud de una circunferencia.

Obtener el radio de la circunferencia.

Calcular Area =  $\pi r^2$

Calcular Longitud =  $2\pi r$

Mostrar Area y Longitud

**Un algoritmo deberá describir tres operaciones: entrada, proceso y salida.**

En el algoritmo de receta de cocina citado anteriormente se tendrá:

- Entrada: ingredientes y utensilios empleados.
- Proceso: elaboración de la receta en la cocina.
- Salida: plato terminado.

En el algoritmo del cálculo del área y la longitud de la circunferencia:

- Entrada: valor del radio de la circunferencia.
- Proceso: cálculo del área y la longitud.
- Salida: mostrar los resultados (área y longitud).

**Los algoritmos son independientes tanto del lenguaje de programación en que se van a codificar como de la computadora en la que se ejecutan.**

Siguiendo con el ejemplo de la receta de cocina, los pasos se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos a seguir para elaborar el plato serán siempre los mismos sin importar el idioma del cocinero.

En cada problema, el algoritmo se puede expresar en diferentes lenguajes de programación y ejecutarse en ordenadores distintos; sin embargo el algoritmo siempre será el mismo.

**Los algoritmos son más importantes que los lenguajes de programación.**

Tanto el lenguaje como la computadora son los medios para obtener un fin: conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente.

### CARACTERÍSTICAS DE LOS ALGORITMOS.

Las características fundamentales que debe cumplir todo algoritmo son:

- Un algoritmo debe ser **preciso** e indicar el orden de realización de cada paso.
- Un algoritmo debe estar **definido**. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Un algoritmo debe ser **finito**. Si se sigue un algoritmo, se debe terminar en algún momento; o sea debe tener un número finito de pasos.

## ALGUNOS EJEMPLO DE LA VIDA DIARIA.

En general, cualquier actividad de la vida cotidiana se puede describir mediante algoritmos. Para irnos familiarizando con ellos, se desarrollan a continuación algunos ejemplos:

### **Ejemplo 1:** Cómo poner en movimiento un coche

Si la respuesta fuese:

Poner el motor en marcha

- Meter la primera marcha.
- Pisar el acelerador.

Lo más probable es que si seguimos estas instrucciones y el coche ya tenía una marcha metida, al poner el motor en marcha se chocara contra el de delante.

Sería más adecuada la respuesta:

- Pisar el embrague con el pie izquierdo
- Poner punto muerto
- Dar a la llave de contacto para poner el motor en marcha
- Volver a pisar el embrague
- Meter la primera
- Quitar el freno de mano y levantar el pie del embrague suavemente a la vez que con el pie derecho se pisa el acelerador.

Es más correcta porque: primero, desglosa el problema en instrucciones simples y concretas, comprensibles para cualquier persona; y segundo, indica claramente el orden en que se deben ejecutar dichas instrucciones.

Este ejemplo es un caso típico de algoritmo con **estructura secuencial**, caracterizado porque ejecuta de forma imperativa todas y cada una de las órdenes en la secuencia en que están escritas.

### **Ejemplo 2:** Hacer una llamada telefónica desde un teléfono fijo

Solución:

1. - Descolgar el auricular.
2. - Esperar que de señal de línea.
3. - Marcar el número de teléfono deseado
4. - Si esta la persona con la que queremos hablar, conversar con ella; si no está, dejarle un mensaje.
- 5.- Colgar el auricular.

En este algoritmo aparece una novedad en la línea 4 ya que esta instrucción no es imperativa; es decir, no obliga a ejecutar una orden determinada, sino que dependiendo de una condición (que esté la persona buscada), se ejecuta una acción u otra. Diremos pues que esta instrucción tiene una **estructura condicional**.

### **Ejemplo 3:** Veamos ahora los pasos a seguir para comerse un bocadillo.

Solución:

1. - Sacar el bocadillo del bolso o mochila.
2. - Desenvolverlo de forma que podamos comer.
3. - Pegar un bocado y comer.
4. - Si no se ha acabado el bocadillo volver a la instrucción 3.

5. - Tirar el papel que lo envolvía a la papelera.

En este algoritmo podemos ver que el cumplimiento o no de la condición nos lleva a seguir dos caminos diferentes y uno de estos dos caminos nos lleva a la vuelta atrás en el orden de ejecución de las instrucciones. Es decir un grupo de acciones se repite varias veces mientras que se cumpla la condición. Por tanto, este algoritmo contiene una **estructura repetitiva**.

Ejercicio. Escribe los algoritmos que resuelven los siguientes problemas:

1. Hacer un huevo frito.
2. Cambiar la rueda pinchada del coche.

### 3. NOTACIONES PARA EL DISEÑO DE ALGORITMOS.

En los ejemplos anteriores hemos visto que un algoritmo puede ser escrito en un lenguaje narrativo, pero esta descripción suele ser demasiado extensa y, además, ambigua.

Para representar un algoritmo se debe utilizar algún método que permita independizar dicho algoritmo de los lenguajes de programación y, al mismo tiempo, conseguir que sea fácilmente codificable.

Existen diferentes métodos para diseñar algoritmos. Los métodos más usuales son:

- Diagrama de Flujo.
- Pseudocódigo.

#### DIAGRAMA DE FLUJO


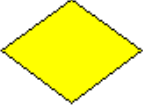








Es una de las técnicas de representación de algoritmos más antigua.

Representan gráficamente la secuencia de operaciones que se han de realizar para resolver un problema.

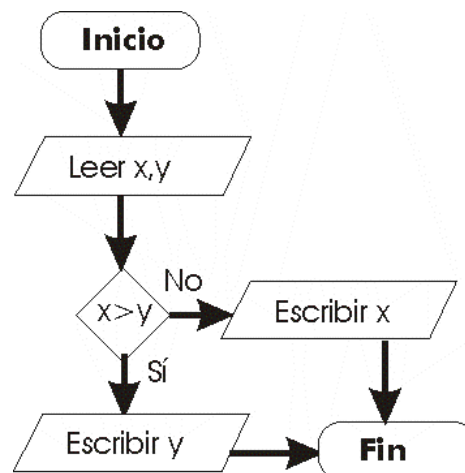
En la actualidad su uso quedado muy limitado por la aparición de otros métodos de diseño más eficaces para la representación y documentación de programas.

No obstante, todo programador debería conocer esta técnica por ser la más básica y por la importancia que ha tenido.

Los símbolos mas utilizados en un diagrama de flujo son:

	<b>Inicio/Final</b> Se utiliza para indicar el inicio y el final de un diagrama; de Inicio sólo puede salir una línea de flujo y al final sólo debe llegar una línea		<b>Decisión</b> Indica la comparación de dos datos y dependiendo del resultado lógico (falso o verdadero) se toma la decisión de seguir un camino del diagrama u otro
	<b>Entrada/Salida</b> Entrada/Salida de datos por cualquier dispositivo (scanner, lector de código de barras, micrófono, parlantes, etc.)		<b>Impresora/Documento.</b> Indica la presentación de uno o varios resultados en forma impresa
	<b>Entrada por teclado.</b> Entrada de datos por teclado. Indica que el computador debe esperar a que el usuario teclee un dato que se guardará en una variable o constante		<b>Pantalla</b> Instrucción de presentación de mensajes o resultados en pantalla
	<b>Acción/Proceso</b> Indica una acción o instrucción general que debe realizarse (operaciones aritméticas, asignaciones, etc.)		<b>Conector Interno</b> Indica el enlace de dos partes de un diagrama dentro de la misma página
	<b>Flujo/Flecas de Dirección</b> Indica el seguimiento lógico del diagrama. También indica el sentido de ejecución de las operaciones		<b>Conector Externo</b> Indica el enlace de dos partes de un diagrama en páginas diferentes

Un ejemplo de diagrama de flujo: algoritmo que calcula el mayor de 2 números.



## PSEUDOCÓDIGO

El pseudocódigo es una herramienta utilizada en el diseño de algoritmos que permite expresar el flujo de ejecución de las instrucciones de una forma clara, sin ambigüedad alguna y usando el lenguaje natural.

El pseudocódigo se concibió para superar las dos principales desventajas del diagrama de flujo: lento de crear y difícil de modificar sin dibujar todo de nuevo.

Es una buena herramienta para el seguimiento de la lógica de un algoritmo y para transformar con facilidad los algoritmos en programas.

Ejemplo: algoritmo que calcula el mayor de 2 números.

```
Inicio
Leer x, y
Si (x > Y)
    Escribir X
Sino
    Escribir Y
Fin
```

Ejemplo: algoritmo que calcule el área y el perímetro de un rectángulo.

```
Inicio
Leer BASE
Leer ALTURA
AREA = BASE * ALTURA
PERIMETRO = (BASE + ALTURA) * 2
Escribir AREA
Escribir PERIMETRO
Fin
```

Ejemplo: Algoritmo que lee una temperatura en grados centígrados y nos calcula y y escribe su valor en grados Reamur y Kelvin.

```
Leer C      // leemos grados centígrados
Reamur = 80*C/100
Kelvin = C+273
Escribir Reamur
Escribir Kelvin
Fin
```

Ejemplo: Algoritmo que lea dos números por teclado y muestre el resultado de la división del mayor por el menor

```
Leer num1
Leer num2
Si(num1 > num2)
{
    mayor = num1
    menor = num2
}
Sino
{
    mayor = num2
    menor = num1
}
Si(menor == 0)
{
    Escribir "No se puede dividir por cero"
}
Sino
    Division = mayor/menor
    Escribir Division
}
```

Ejercicio: Busca en Internet otros métodos para el diseño de algoritmos y descríbelos de forma resumida.