¿Qué es un problema?

Un problema es un asunto o un conjunto de cuestiones que se plantean para ser resueltas. La naturaleza de los problemas varía con el ámbito o el contexto: problemas matemáticos, químicos, filosóficos, etc.

Es importante que al abordar un problema se tenga una **descripción simple y precisa del mismo**, de lo contrario resultaría complejo modular, simular, o programar su solución en un ordenador.

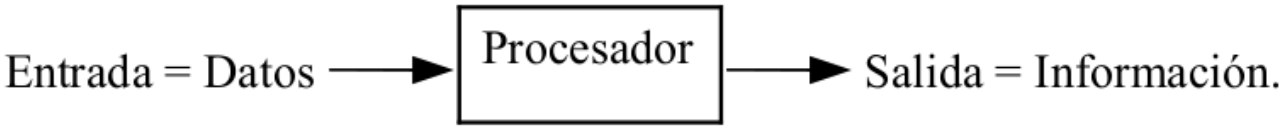
¿Cómo vamos a solucionar los problemas?

Un **programador** es una persona que resuelve problemas, y para llegar a ser un programador eficaz se **necesita aprender a resolver problemas de un modo riguroso y sistemático**:

- *Definición o análisis del problema*: consiste en el estudio detallado del problema. Se debe identificar los datos de entrada, de salida y la descripción del problema.
- *Diseño del algoritmo*: que describe la secuencia ordenada de pasos que conduce a la solución de un problema dado: **algoritmo**.
- *Transformación del algoritmo en un programa (codificación)*: Se expresa el algoritmo como un programa en un **lenguaje de programación**.
- *Ejecución y validación del programa*.

Sistemas de información

Sistema de procesamiento de información es un sistema que transforma datos brutos en información organizada, significativa y útil.



- *Datos*: se refiere a la representación de algún hecho, concepto o entidad real (palabras escritas, números, dibujos etc)
- *Información*: Información implica datos procesados y organizados.
- *Procesador*: Proceso por el que se convierte datos de entrada en información útil.

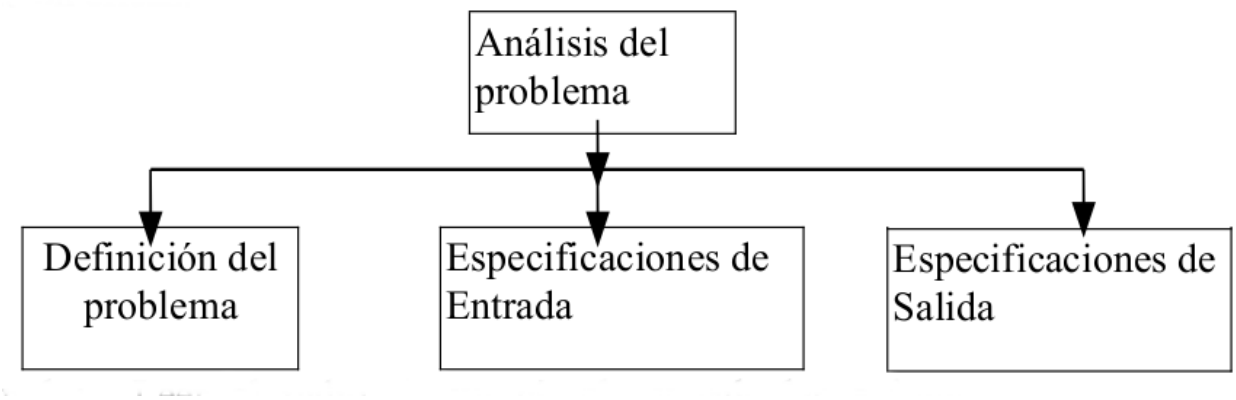
El conjunto de instrucciones que especifican la secuencia de operaciones a realizar, en orden, para resolver un sistema específico o clase de problemas, se denomina **algoritmo**. O sea, un algoritmo es una fórmula para la resolución de un problema.

Cuando el procesador es un ordenador, el algoritmo ha de expresarse de una forma que recibe el nombre de **programa**.

Análisis del problema

El primer paso, análisis del problema, requiere un estudio a fondo del problema y de todo lo que hace falta para poder abordarlo.

El propósito del análisis de un problema es ayudar al programador (Analista) para llegar a una cierta comprensión de la naturaleza del problema. Una buena definición del problema, junto con una descripción detallada de las especificaciones de entrada/salida, son los requisitos más importantes para llegar a una solución eficaz.



Ejemplo de análisis

Nos proponen el siguiente problema:

Leer el radio de un circunferencia y calcular e imprimir su superficie y su longitud.

Análisis

Definición del problema: Tenemos que saber que es el radio de un circunferencia, y saber que es su área y su longitud. Además tenemos que saber cómo calcular el área y la longitud. Por lo tanto necesitamos saber el radio y utilizar las formulas para calcular el área y la longitud.

Especificaciones

Entradas:	Radio de la circunferencia (Variable RADIO).
Salidas:	Superficie de la circunferencia (Variable SUPERFICIE). Longitud de la circunferencia (Variable LONGITUD)
Variables:	RADIO, SUPERFICIE, LONGITUD de tipo REAL.

Los datos de entrada y la información de salida se van a guardar en **variables**, donde se puede guardar datos. Las variables son de distintos **tipos de datos**: entero, real, cadena, booleano,..

Especificaciones del problema

El resultado final del análisis es obtener una serie de documentos (**especificación**) en los cuales quedan totalmente definido el proceso a seguir en la resolución del problema

Diseño de algoritmos

A partir de los requerimientos, resultados del análisis, empieza la etapa de **diseño** donde tenemos que construir un **algoritmo** que resuelva el problema.

Definición de algoritmo

Un **algoritmo** es un conjunto de acciones que especifican la secuencia de operaciones realizar, en orden, para resolver un problema.

Los algoritmos son independientes tanto del lenguaje de programación como del ordenador que los ejecuta.

Las características de los algoritmos 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.

Ejemplo tradicional de un algoritmo: Cambiar la rueda pinchada de un coche.

Etapa de diseño

Aunque en la solución de problemas sencillos parezca evidente la **codificación** en un lenguaje de programación concreto, es aconsejable realizar el **diseño** del algoritmo, a partir del cual se codifique el programa.

La soluciones a problemas más complejos pueden requerir muchos más pasos. Las estrategias seguidas usualmente a la hora de encontrar algoritmos para problemas complejos son:

- **Partición o divide y vencerás**: consiste en dividir un problema grande en unidades más pequeñas que puedan ser resueltas individualmente.
    - Ejemplo: Podemos dividir el problema de limpiar una casa en labores más simple correspondientes a limpiar cada habitación.
- **Resolución por analogía**: Dado un problema, se trata de recordar algún problema similar que ya esté resuelto. Los dos problemas análogos pueden incluso pertenecer áreas de conocimiento totalmente distintas.
    - Ejemplo: El cálculo de la media de las temperaturas de las provincias andaluzas y la media de las notas de los alumnos e una clase se realiza del mismo modo.

La descomposición del problema original en subproblemas más simples y a continuación dividir estos subproblemas en otros mas simples se denomina **diseño descendente (top-down design).** Tras la primera descripción del problema (poco específica), se realiza una siguiente descripción mas detallada con mas pasos concretos. Este proceso se denomina **refinamiento del algoritmo**.

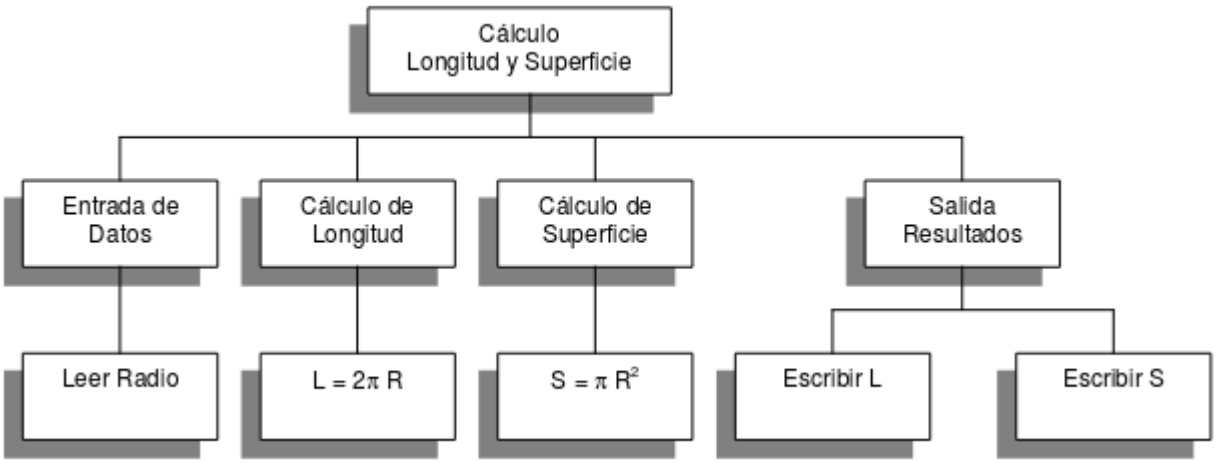
Ejemplo de diseño

Leer el radio de un circunferencia y calcular e imprimir su superficie y su circunferencia.

- Se puede dividir en tres subproblemas más sencillos:
    - Leer Radio
    - Calcular Superficie
    - Calcular Longitud

    - Escribir resultados
- Refinamiento del algoritmo:
    - `Leer Radio`
    - `Superficie <- PI * Radio ^ 2`
    - `Longitud <- 2 * PI * Radio`
    - `Escribir Radio, Longitud, Superficie`

Lo podemos ver en un **diagrama estructurado**:



Herramientas de representación de algoritmos

- Un **diagrama de flujo** es una de las técnicas de representación gráfica de algoritmos más antiguas. Ventajas: permite altos niveles de estructuración y modularización y es fácil de usar. Desventajas: son difíciles de actualizar y se complican cuando el algoritmo es grande.
- El **pseudocódigo**, nos permite una aproximación del algoritmo al lenguaje natural y por tanto un a redacción rápida del mismo.