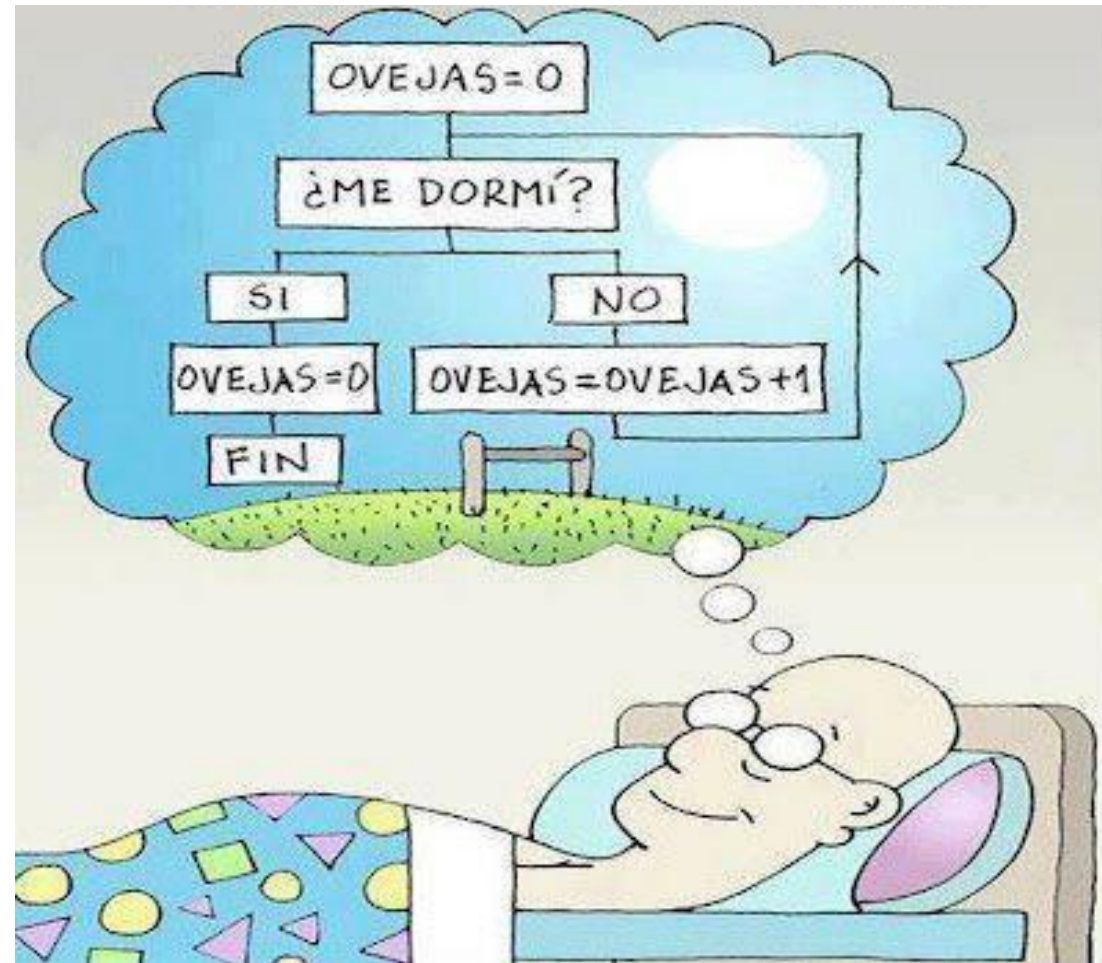


# Módulo Técnicas de Programación



## Capacidades Profesionales a las que contribuye el Módulo de Técnicas de Programación

- Interpretar las **especificaciones** de diseño o requisitos de las asignaciones a programar
- Comprendiendo en su contexto inmediato **cuál es el problema** a resolver
- Determinar el **alcance del problema** y convalidar su interpretación a fin de identificar aspectos faltantes.
- **Desarrollar algoritmos** que dan soluciones a los problemas asignados o derivados de los mismos.

## Prácticas formativas de Carácter Profesionalizante

- Práctica de resolución de una situación problemática, real o simulada de acuerdo a especificaciones de diseño, **desarrollando algoritmos** que den **solución a problemas** específicos.

## Contenido

- Introducción
- Elementos Informáticos
- Diseño de Algoritmos
- Desarrollo de Programas
- Niveles de lenguajes de programación

## Introducción

### SIGLO XVIII



### SIGLO XIX





## Introducción

### SIGLO XX

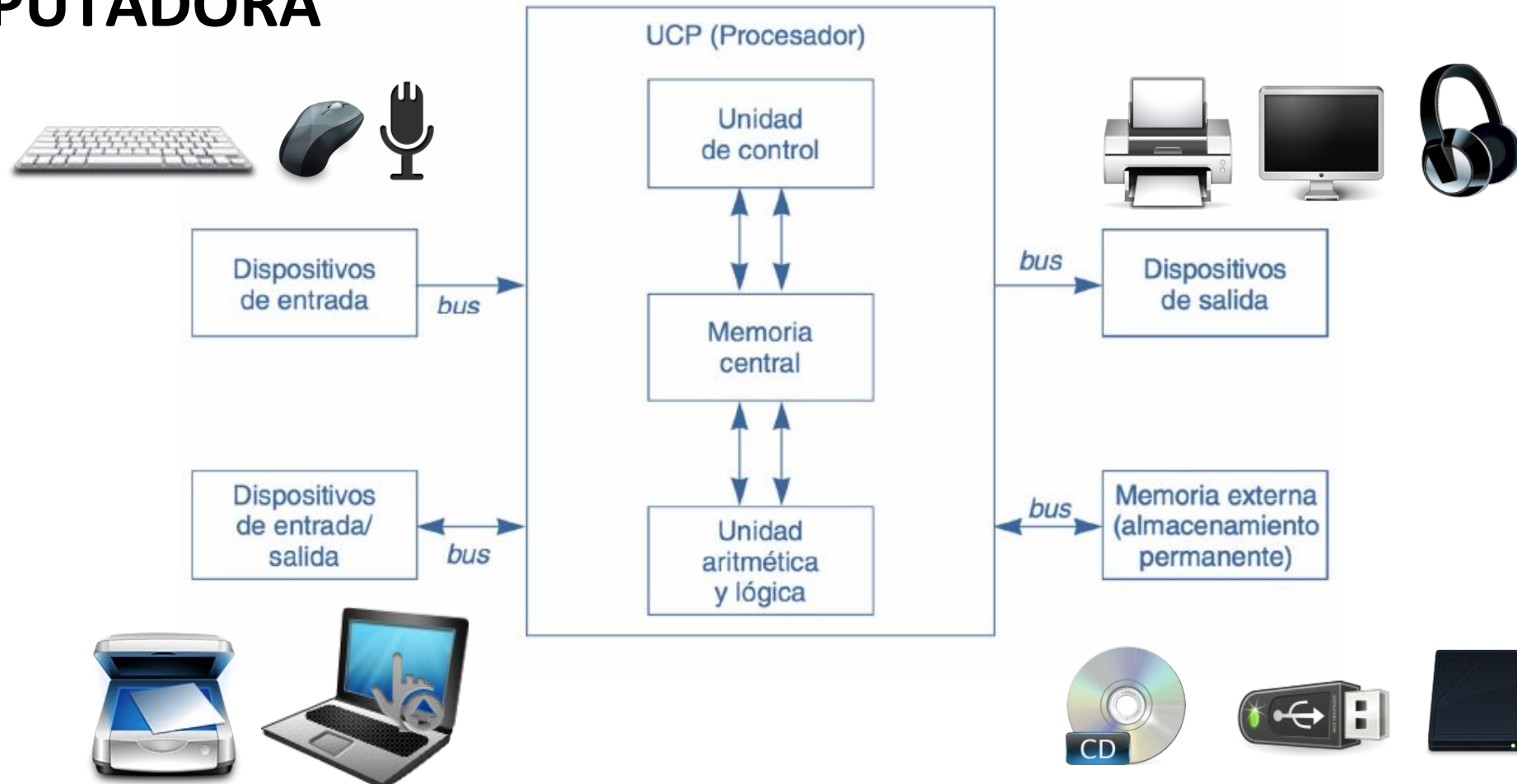


### SIGLO XXI



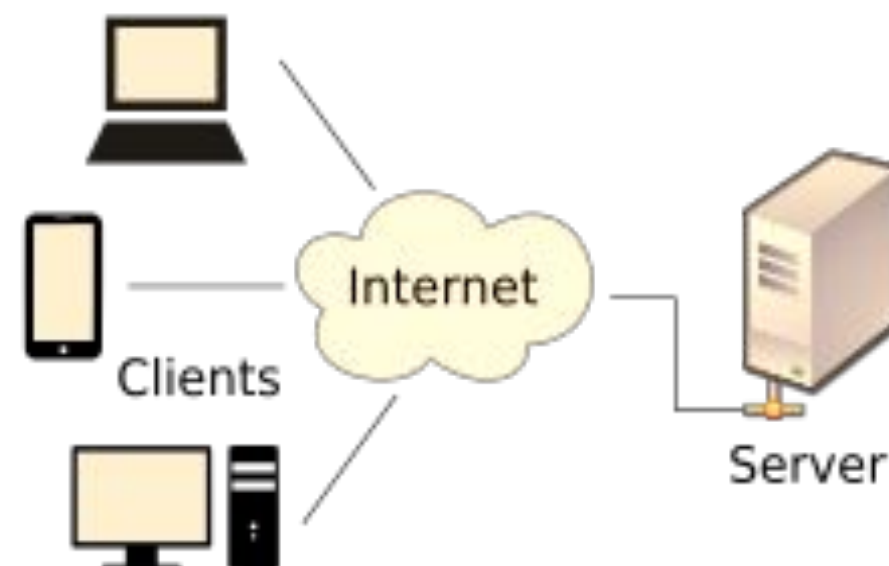
## Elementos Informáticos

# COMPUTADORA



## Elementos Informáticos

### REDES

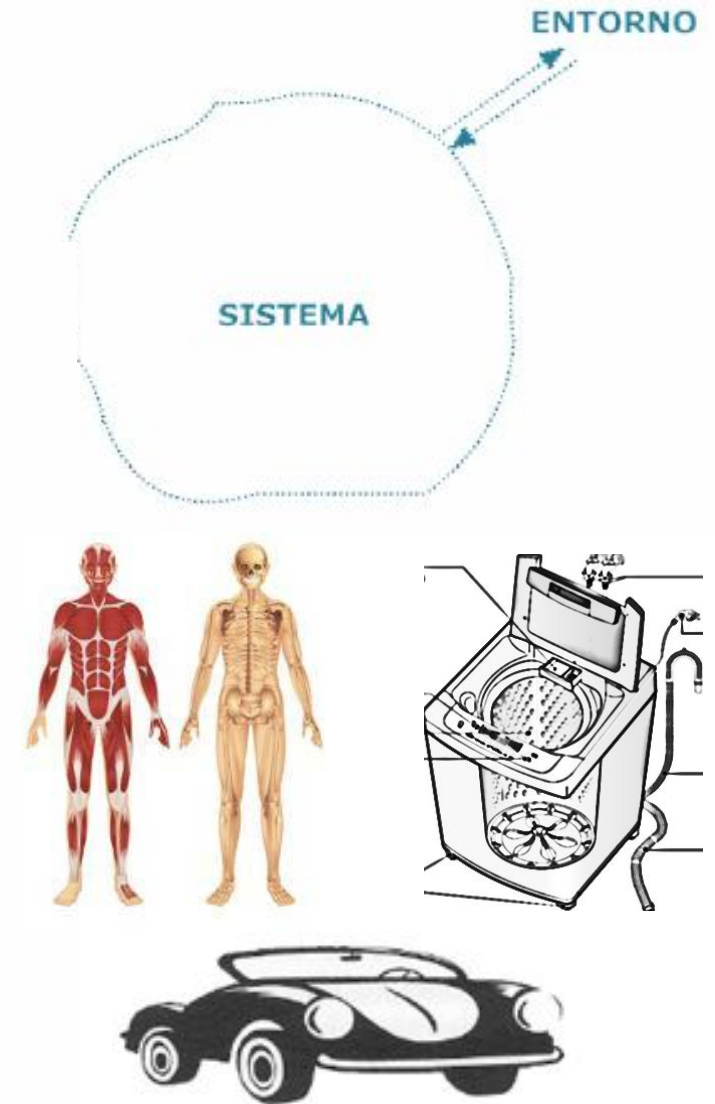
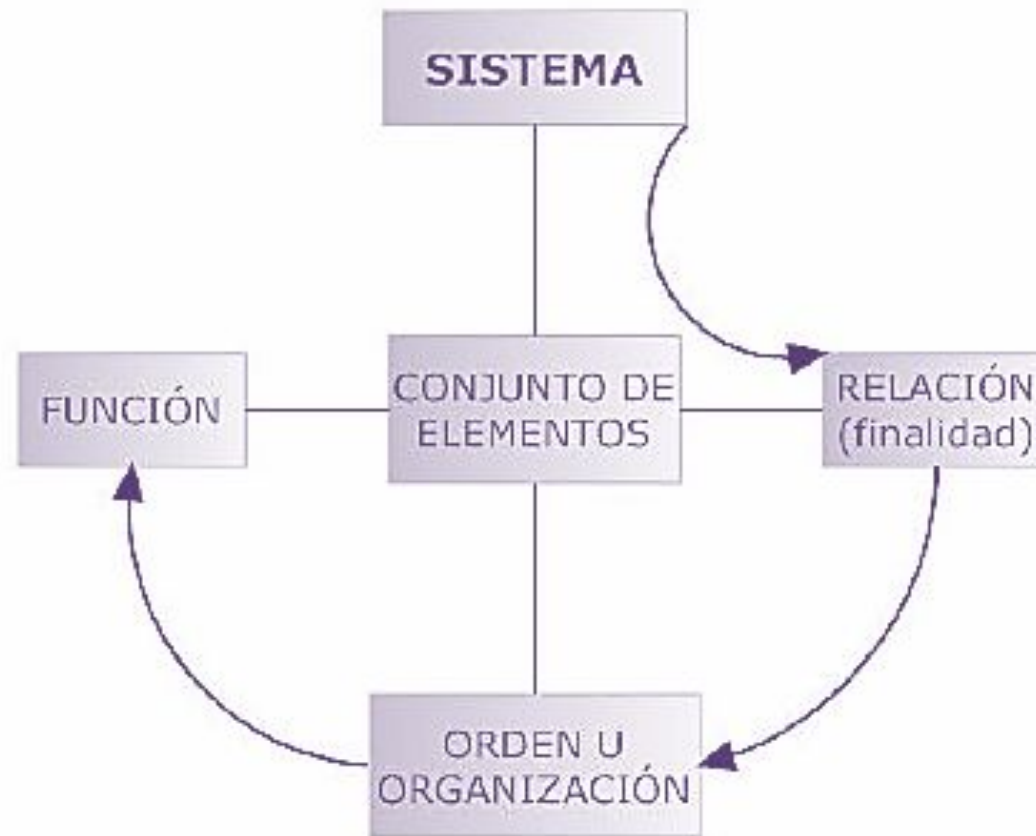




## Elementos Informáticos

### SISTEMAS

*Teoría General  
de los Sistemas  
Bertalanffy*



## Elementos Informáticos

### SISTEMAS

Otras características:

- ✓ **Límite:** Concreto o Simbólico.
- ✓ **Depósitos:** Permanentes o Transitorios.
- ✓ **Canales.**
- ✓ **Subsistemas.**

#### Niveles y subsistemas del automóvil.



## Elementos Informáticos

# INTERCAMBIO ENTRE SISTEMAS



## SISTEMAS TECNOLÓGICOS

*“Son diseñados por los seres humanos para que cumplan con una finalidad específica.”*

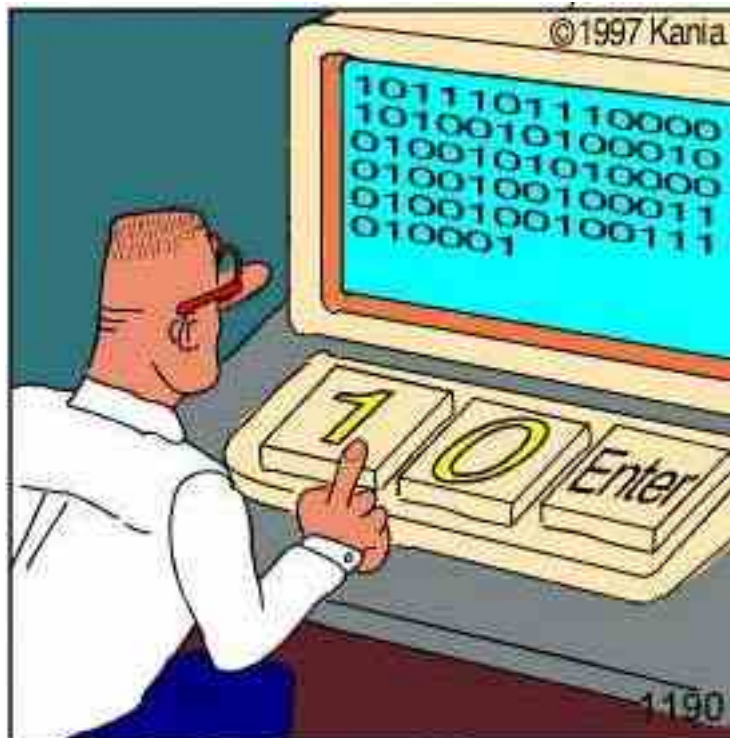


*“Sistemas de:  
Procesamiento  
de Materia, de  
Procesamiento  
de Energía, de  
Información.”*

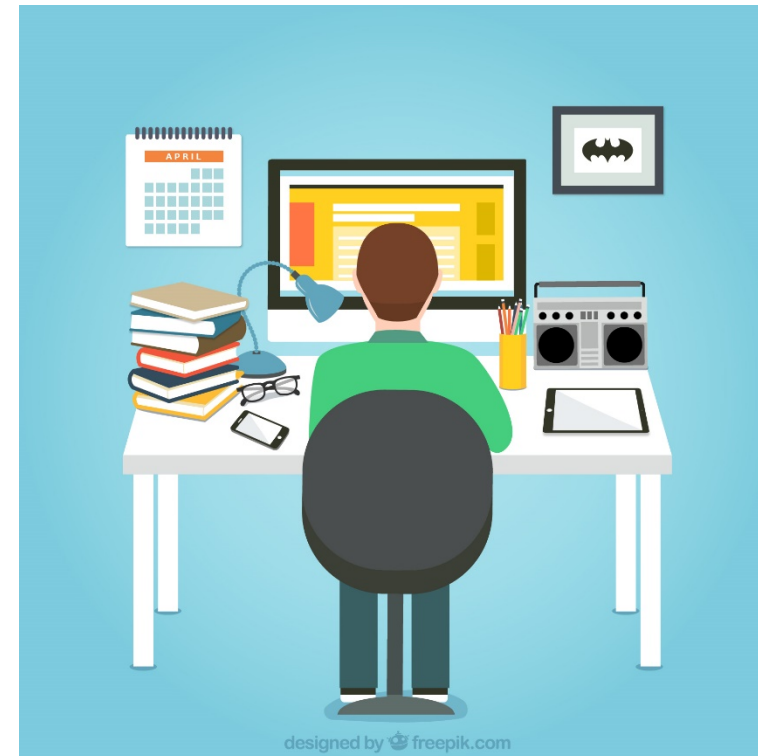
## Elementos Informáticos

# PROGRAMAS <- LENGUAJES DE PROGRAMACIÓN

ANTES



AHORA



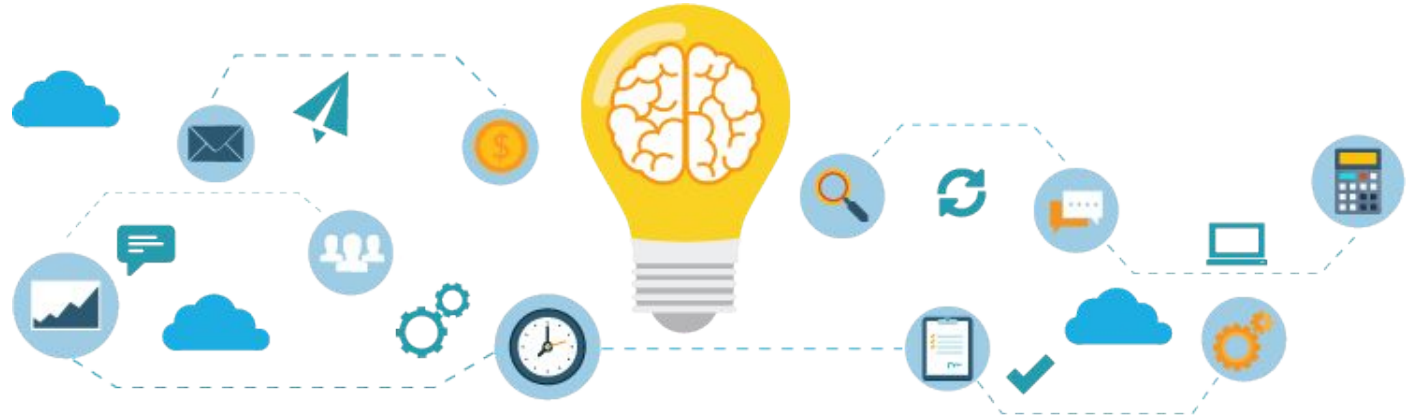


## Elementos Informáticos

### SOFTWARE

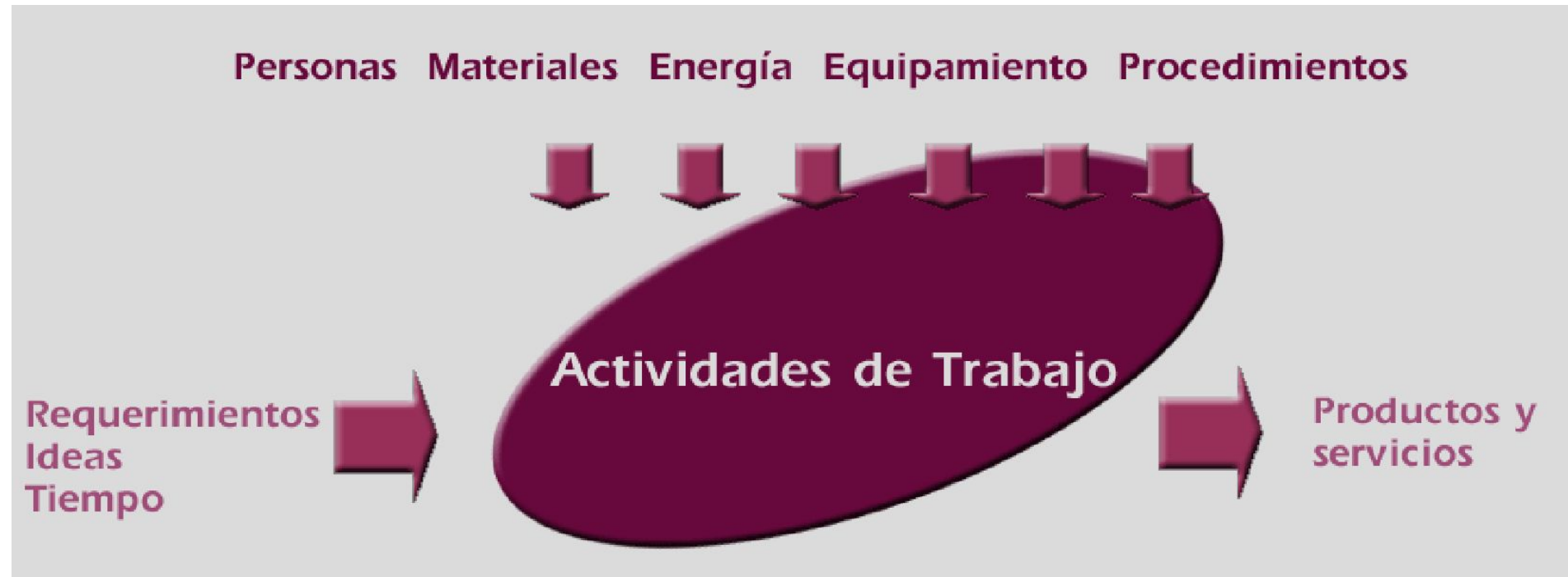
*Conjunto de:*

- *Programas*
- *Procedimientos*
- *Reglas*
- *Documentación*
- *Datos*



## Elementos Informáticos

# CONSTRUCCIÓN DEL SOFTWARE



***PROCESO DE CONSTRUCCIÓN DEL SOFTWARE***

## ACTIVIDADES DEL PROCESO DE CONSTRUCCIÓN DEL SOFTWARE



## Elementos Informáticos

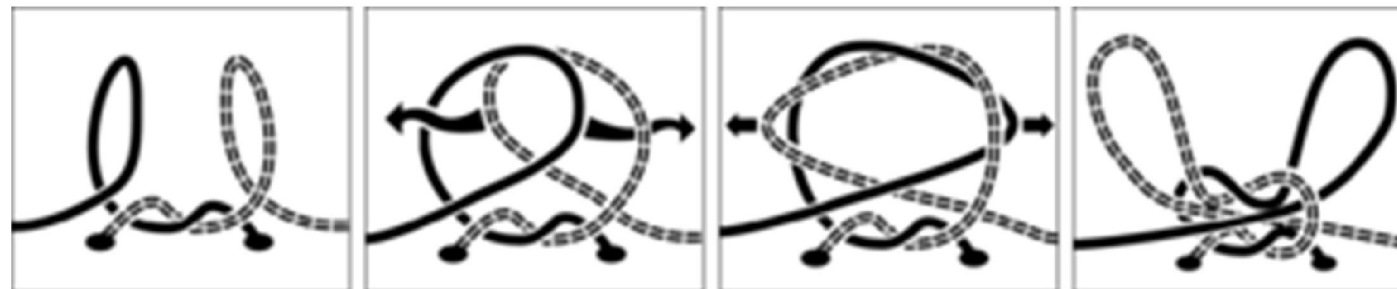
# CONSTRUCCIÓN DEL SOFTWARE





## Diseño de Algoritmos

### ALGORITMO -> PROGRAMA



## Diseño de Algoritmos

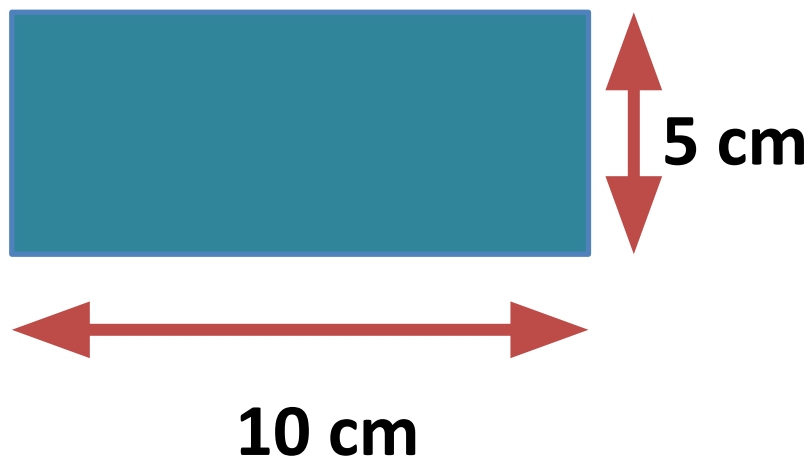
### ALGORITMO



- ✓ Un algoritmo debe estar **específicamente definido**.
- ✓ Un algoritmo debe ser **finito**.
- ✓ Un algoritmo debe ser **correcto**.
- ✓ Un algoritmo es **independiente** del lenguaje y del medio.

## Diseño de Algoritmos

### EJEMPLO ELEMENTOS DE UN ALGORITMO



Obtención del área de un rectángulo:

Altura: 5 cm

Base: 10 cm

**1. Salida:** área

Fórmula del área: base x altura.

**2. Entradas de datos:**

Dato de Entrada 1: altura: 5 cm


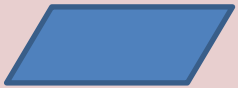
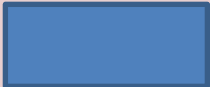


Dato de Entrada 2: base: 10 cm

**3. Proceso:**

$\text{área} = \text{base} * \text{altura} \rightarrow \text{área} = 50$

## Diseño de Algoritmos

### REPRESENTACIÓN DE ALGORITMOS → Diagrama de Flujo

Nombre del Símbolo	Representación del Símbolo	Función
Inicio/Final		Representa el comienzo o la finalización de un algoritmo.
Entrada		Representa la entrada de datos por medio de un dispositivo de entrada.
Proceso		Representa una operación como puede ser una operación matemática, una declaración de una variable o una asignación de una variable a un valor, etc.
Decisión		Representa una condición: realiza comparaciones entre distintos valores. Al evaluar dicha condición puede dar como resultado dos posibles valores: verdadero o falso, en función de que si esa condición se cumple o no.
Salida		Representa la salida de datos o resultados. Sirve para representar los mensajes que el sistema le muestra al usuario por pantalla.



## Diseño de Algoritmos

# REPRESENTACIÓN DE ALGORITMOS → Pseudocódigo

```
INICIO restaEnteros
    IMPRIMIR "Ingrese el primer número"
    IMPRIMIR "Ingrese el segundo número"
    ENTERO resta = numero1-numero2
    IMPRIMIR "El resultado de la resta es" + resta
FIN
```

# VALIDACIÓN DE ALGORITMOS

### ✓ Pruebas de Escritorio

- Sirve para validar un algoritmo, utilizando datos reales.
- Consiste en hacer seguimiento de un algoritmo **recorriendo las sentencias secuencialmente**, simulando el funcionamiento de la computadora.
- Se debe identificar las variables de **entrada**, cuáles son las variables **auxiliares** y cuáles son las variables de **salida**.
- A medida que se van recorriendo las líneas se anotan en una **tabla auxiliar** los valores que van tomando las variables.

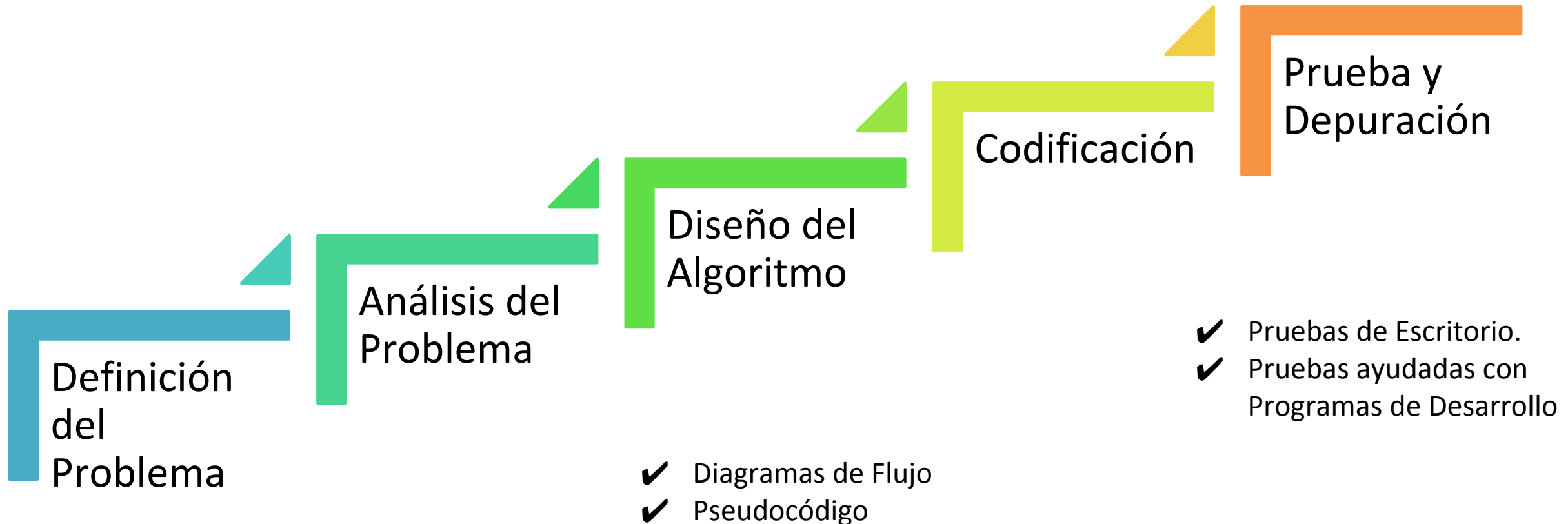
Ejemplo para:  
**restaEnteros**

Entrada	
Numero1 (entero)	Numero2 (entero)
10	2
200	50

Salida
Resta (entero)
10-2 = 8
200-50=150

## Diseño de Algoritmos

### CONSTRUCCIÓN DE UN PROGRAMA:



# Niveles de lenguajes de programación



Los lenguajes de programación son desarrollados para dar instrucciones a una computadora, hay niveles de lenguajes de programación que facilitan esta tarea.



## *Lenguajes de bajo nivel*

Lenguaje que entiende directamente la computadora, por lo tanto, se utiliza solo 0 y 1.

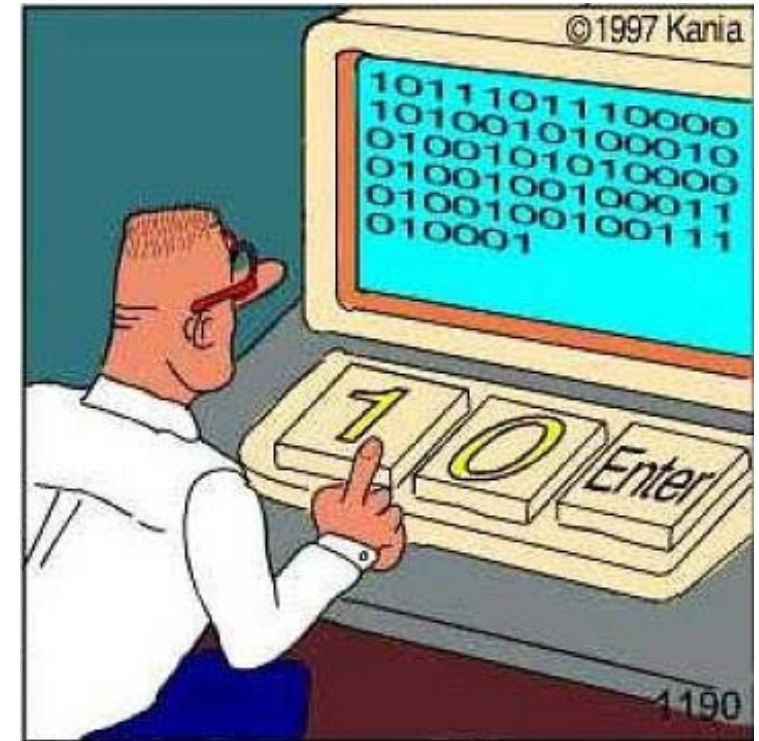
Podemos clasificar este nivel en dos tipos:

**Lenguaje máquina:** instrucciones formador por 0 y 1 que se ejecuta directamente en el [CPU](#) ([Procesador](#)) del equipo.

**Lenguaje ensamblador:** este derivado del lenguaje máquina, sin embargo, utiliza letras y números para las instrucciones.

Es necesario utilizar un compilador para interpretar las instrucciones del programador, el lenguaje no es cercano al humano.

Ejemplos: Lenguaje ensamblador.



## *Lenguajes de medio nivel*


Son lenguajes que permiten una mayor abstracción, pero manteniendo algunas características del lenguaje de bajo nivel.

El código es enviado a un compilador que lo convierte al lenguaje máquina.

Por ejemplo: Lenguaje C.

C puede acceder a registros del sistema y direcciones de memoria, todas propias de lenguajes de bajo nivel.

Debido a sus características, estos lenguajes se pueden situar entre los de bajo nivel y alto nivel, aunque no es muy aceptado.



```
/* the standard library. It provides input and
 * to the program.
 */
#include <stdio.h>

/*
 * Function (method) definition. This outputs
 * standard output to the specified file.
 */
void sayHello() {
    // printf() in the specified text
    // formatting options
    printf("Hello, world!\n");
}

/*
```

## *Lenguajes de alto nivel*

Es más cercano al lenguaje humano, manejando conceptos, tipos de datos, etc. sin importar la computadora.

Es indispensable utilizar un interprete o compilador que traduzca las instrucciones al lenguaje máquina.

Son lenguajes independientes de una computadora, puedes migrar tu código a otro y seguir trabajando sin problemas.

### **Ventajas:**

- Mejor comprensión del lenguaje
- Independiente del equipo



