



Algoritmos y Estructuras de  
Datos

**Problemas y algoritmos**

# Principal material bibliográfico utilizado

- Algoritmos y solución de problemas. Fundamentos de programación. Otoño 2008. Mtro. Luis Edgardo Perez Bernal. Universidad Jesuita de Guadalajara.
- Metodología de Programación, Programación en C, Aplicaciones electrónicas 1/47. Técnicas de Programación 1ª Parte: Metodología de Programación.
- Fundamentos de Programación 2006-2007 – Metodología de la programación. Jorge Sanchez.



# Dato e Información

## **Dato**

- Unidad mínima que por sí sola no tiene significado.
- Representación simbólica de propiedades que poseen entes y/o sucesos.
- Puede referirse a números, letras, símbolos, etc.
- Los datos aisladamente pueden no contener información relevante.

## **Información**

- Conjunto de datos que relacionados entre sí poseen significado.
- Representación simbólica obtenida operando sobre datos.



# Computadora

Máquina electrónica programable (la diferencia por ejemplo de una calculadora).

Máquina electrónica digital y sincrónica.

Procesa datos de entrada y genera resultados de acuerdo a las instrucciones de los programas almacenados en su memoria.



# Utilidad de las computadoras

- Procesar grandes volúmenes de datos.
- Tareas repetitivas.
- Tareas de alta precisión.
- Tareas con cálculos complejos.
- Tareas a alta velocidad.
- Gestión y control de datos administrativos.
- Ciencias: cálculo numérico, estadísticas, simulación, climatología, genética, diagnóstico, encuestas, educación, etc.

# Programa y Sistema

## Programa

- Secuencia de pasos finita expresada en un lenguaje comprensible por una computadora, que resuelve un problema.
- Conjunto de instrucciones o sentencias.

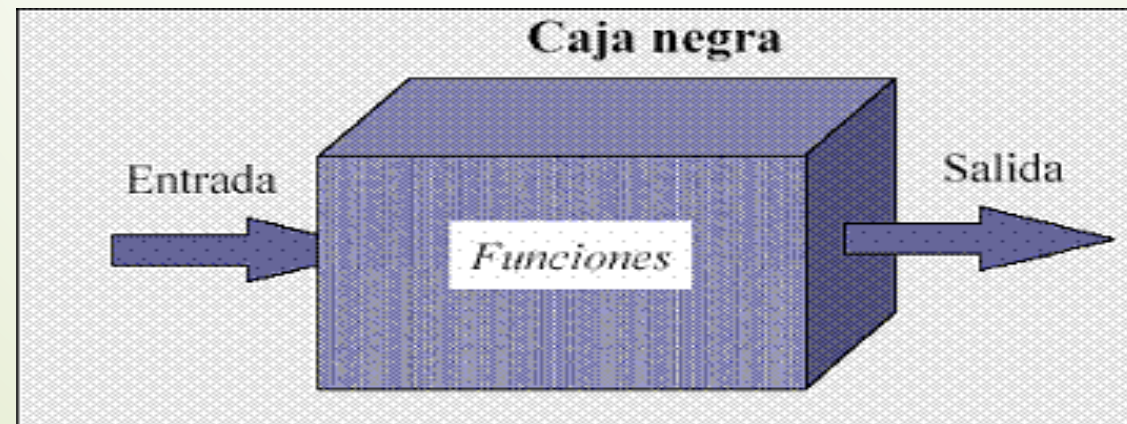
## Sistema

- Conjunto de programas que cumplen una tarea específica, hardware y software que cumplen tareas específicas.
- Conjunto de componentes que cumplen determinada función.



# Entrada, Proceso y Salida

- Entradas: ingresos al sistema, pueden ser recursos materiales, humanos o información.
- Proceso: transforma la entrada en salida (caja blanca, caja negra).
- Salida: resultados de procesar las entradas.





# Lenguaje de programación

- Un lenguaje de programación es aquel elemento dentro de la informática que nos permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes.
- Cada lenguaje de programación utiliza un grupo de símbolos o reglas que tienen un significado.





# Lenguaje de programación

- conjunto de símbolos, reglas sintácticas (forma de escribir) y semánticas (sentido de aquello que se escribe) junto con sus elementos y las expresiones.
- Los lenguajes de programación surgen por la necesidad de automatizar tareas que realiza el usuario de forma repetitiva.



# Representación de la información

- **Lenguajes de alto nivel :**

- cercanos al lenguaje natural de las personas.
- fáciles de aprender y recordar por las personas.
- menores tiempos de desarrollo de aplicaciones.
- independientes del hardware.

- **Lenguajes de bajo nivel:**

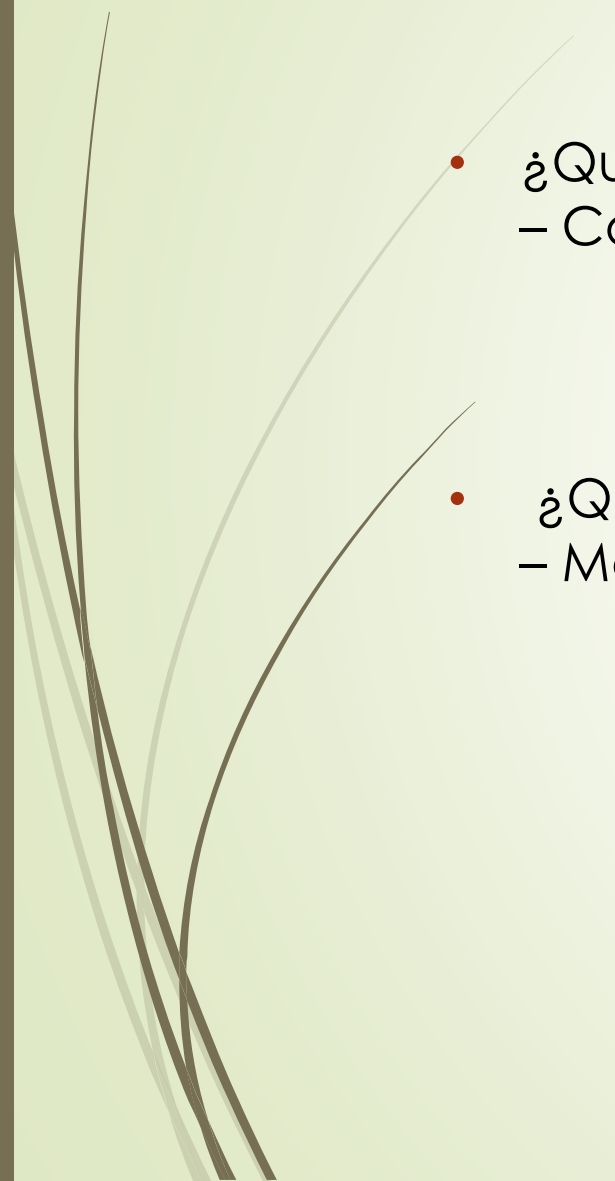
- cercanos al lenguaje de la máquina.
- programadores especializados.
- mayores tiempos de desarrollo de aplicaciones.
- muy dependientes del hardware.

# Problemas y Algoritmos

- La informática se ocupa de los **problemas computables**.
- Se le llama **problema computable** a aquella abstracción de la realidad que tiene representación algorítmica.
- Los **algoritmos** permiten encontrar la solución a problemas computables.
- Intuitivamente las personas efectuamos cotidianamente una serie de pasos, procedimientos o acciones que nos permitan alcanzar algún resultado o resolver un problema (al bañarnos, al desayunar, al ir a la universidad). En realidad todo el tiempo estamos aplicando algoritmos para resolver problemas.



# Resolver problemas

- ¿Qué tipo de problemas se pueden resolver?
    - Computables
  - ¿Qué métodos hay para resolver problemas computables?
    - Metodología de la programación (centrado en los algoritmos)
- 



# Resolver problemas

## **Tipo de operaciones que el computador puede realizar**

- 1 – Ingresar datos – información. Pueden ser varios y de diferente tipo – numéricos y NO numéricos.
- 2 - Realizar cualquier tipo de operación algebraica y ó lógica, efectuando comparaciones y generando ciclos definidos o condicionados.
- 3– Almacenar información en forma transitoria (memoria principal) o permanente (memorias auxiliares) .
- 4 – Exhibir datos y o resultados en diferentes dispositivos como ser en una pantalla, una impresora, etc.

# Resolver problemas

**Tipo de problemas a resolver:** Los problemas que se pueden presentar son en general de los temas más diversos, cuales de ellos se pueden resolver confeccionando un programa?:

- a - calcular una superficie – si - con op. simples y formulas matemáticas.
- b - calcular el importe e imprimir una factura comercial – si
- c - hacer una tortilla de papas y huevos - no – solo hacer una receta y seguirla.
- d - determinar el mayor valor de una serie de 125 números – si
- e - cambiar una goma pinchada en un auto – no
- f - calcular la altura promedio de los alumnos del curso - si
- g – controlar la facturación y entrega de películas de un video club - si
- h - organizar una fiesta - no
- i - resolver un sistema de ecuaciones – si





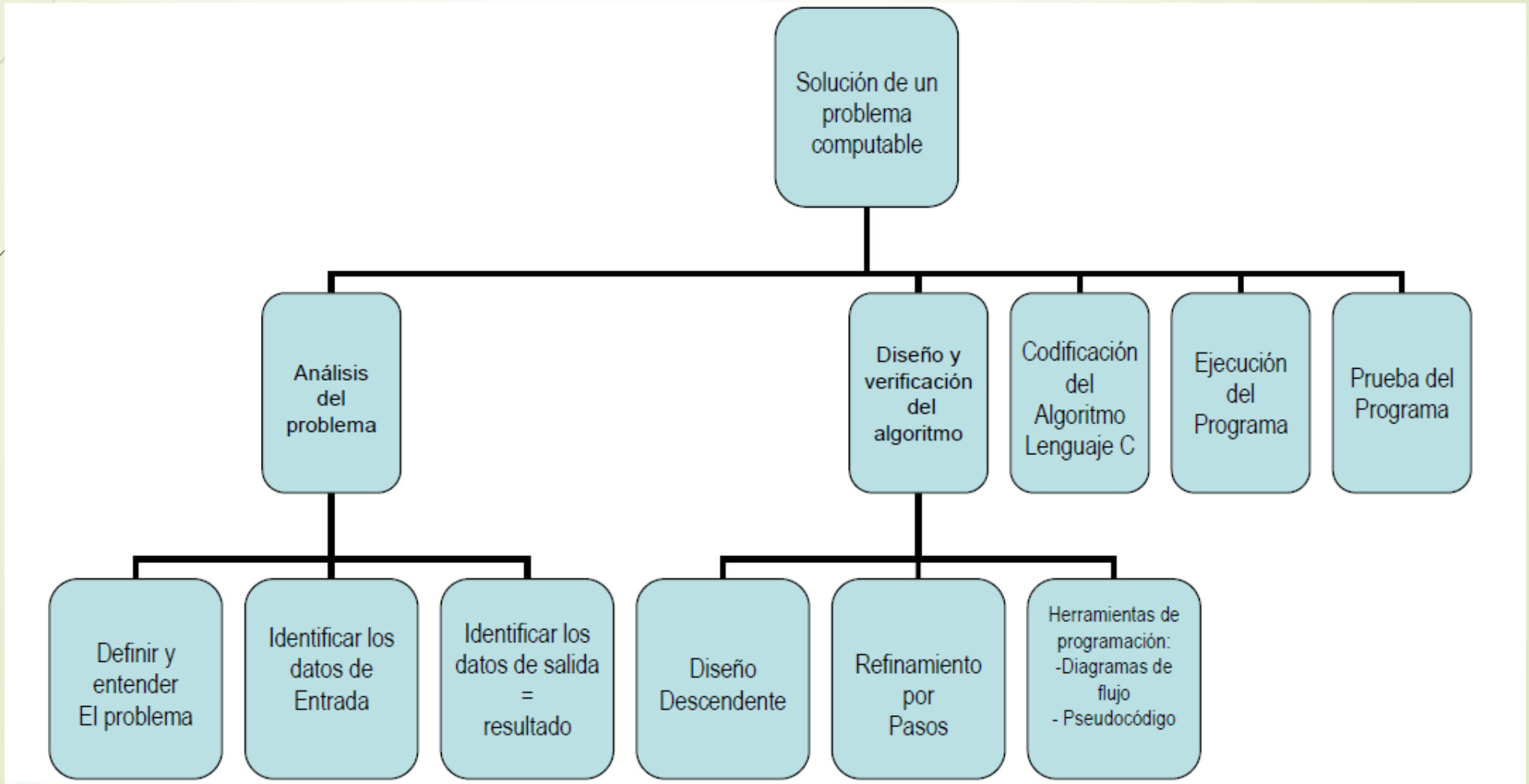
# Metodología para la confección de los programas.

- 1 - Comprensión del problema.
- 2 - Diseño de una estrategia.
- 3 - Desarrollo del algoritmo ( diagrama de lógica ).
- 4 - Prueba del algoritmo, optimización
- 5 - Codificación del algoritmo. ( programa )
- 6 - Ejecución del programa { incluye cargar, compilar , depurar y ejecutar }
- 7 - Evaluación de resultados
- 8 - Implementación, documentación
- 9 - Mantenimiento

# Fases para resolver un problema computable

- Diseño de programas
  - **Análisis** del problema
  - **Diseño** del algoritmo
  - Verificación manual del algoritmo
- En la computadora
  - **Codificación** del algoritmo
  - **Ejecución** del programa
  - **Verificación** del programa
  - **Mantenimiento** (documentación)
- Análisis
- Diseño (descendente, refinamiento paso a paso)
- Codificación
- Ejecución
- Prueba
- Mantenimiento

# Fases para resolver un problema computable

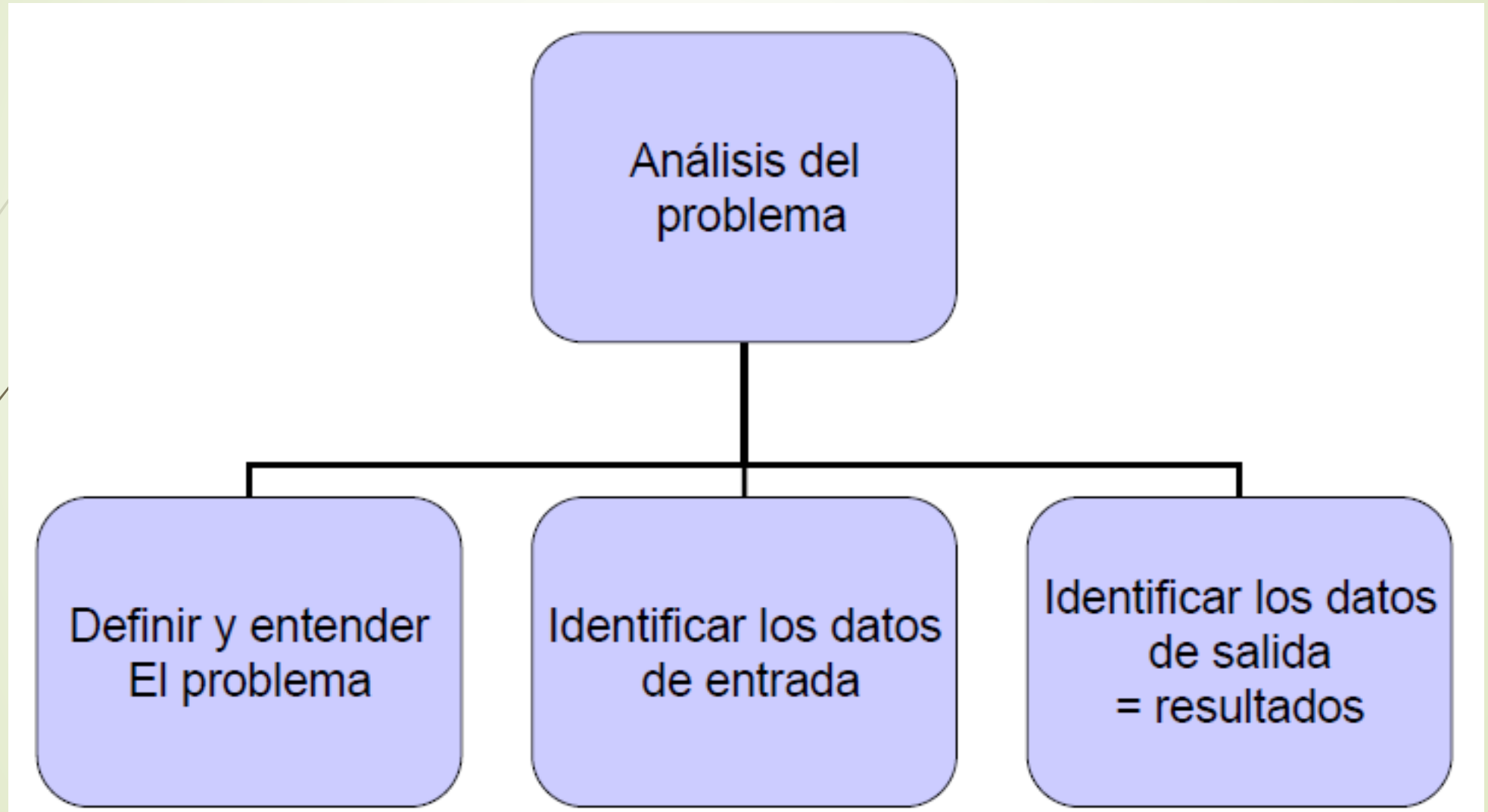




# Análisis del problema

- Es el primer paso a seguir para encontrar la solución a un problema computable es el *análisis del problema*.
- En el *análisis del problema* se requiere del máximo de creatividad e imaginación.
- Debido a que se busca una solución se debe examinar cuidadosamente el problema a fin de **identificar** que tipo de información es necesaria producir. También se deben identificar aquellos elementos de información ofrecidos por el problema y que resulten útiles para obtener la solución al problema.
- Finalmente, un procedimiento para producir los resultados deseados a partir de los datos, es decir, el **algoritmo**.

# Análisis del problema





# Comprensión del problema (que)

- Determinar los resultados a obtener
- Determinar cuales son los datos necesarios para obtener dichos resultados
- Determinar cual es el proceso a realizar con los datos para obtener los resultados



# Comprensión del problema (que)

Cuales son los **resultados que debo proporcionar** ?

- Cuantos son ?
- Debo respetar una secuencia preestablecida ?
- Se han establecido unidades ?

Cuales son los **datos necesarios para resolver el problema** ?

- Son datos explícitos ó implícitos ?
- Cuantos son ?, en que secuencia se presentan ?
- Sin son varios como detecto a los últimos ? Existe una condición de final ?

Qué tipo de **relación, método o proceso debo establecer con los datos?**  
para obtener los resultados pedidos?

- es un proceso matemático ? de evaluación ?
- es un proceso selectivo ? simple ó múltiple ?
- es una combinación y/o repetición de ambos ?
- existen condiciones especiales ó restricciones ?

# Comprensión del problema (que)

## Ejemplo

Calcular el área de un cuadrado cuyos lados tienen como longitud la hipotenusa de un triángulo rectángulo cuyos catetos se dan como datos. Conviene en este caso graficar el problema.

a) Resultados : área del cuadrado de lado  $h$

b) Datos : catetos  $a$  y  $b$ , mayores que cero

c) Proceso : está compuesto por dos cálculos simples, el primero es el de la hipotenusa del triángulo y el segundo el del área del cuadrado.

Restricción / orden : no puedo resolver el área del cuadrado si no calculo previamente su lado.



# Diseño de una estrategia (como)

- Elaboración de un plan general que permita resolver el problema.
- Bosquejo de la solución, haciendo abstracción de los detalles
- Plan sobre el cual nos basaremos para desarrollar el algoritmo en forma gráfica.
- Etapa más creativa y también más dificultosa en la resolución de los problemas .
- Es donde se aprecia la experiencia e intuición de quien resuelve el problema. No se trata de ser siempre original, conviene basarse en problemas anteriores o en experiencias similares.



# Diseño de una estrategia (como)

- En general la estrategia a aplicar consistirá en "dividir el problema original" en una sucesión de problemas mas simples o conocidos, hasta llegar a un tamaño de los problemas que sean perfectamente comprensibles y que luego podríamos ir resolviendo en forma individual.
- Estos pequeños problemas serán partes (módulos/funciones) del prob. inicial.
- Una vez resueltos todos los pequeños problemas y enlazados adecuadamente, tendremos resuelto el problema original (**principio de "divide y reinaras"**).
- **Hablaremos de "tareas" ó "procesos"**.



# Diseño de una estrategia (como)

**a) Acciones externas, de entrada / salida :**

**b) Acciones internas :**

**c) Condición de final :**

**d) Condiciones particulares: propias del problema como ser excepciones, unidades, posibles errores,**

# Diseño de una estrategia (como)

## **a) Acciones externas, de entrada / salida :**

- ingresar datos ( cada dato debe tener asociado un nombre que lo represente)
- informar datos y/ó resultados.

## **b) Acciones internas :**

- Efectuar cálculos aritméticos, pueden ser intermedios ó auxiliares y/o finales
- CONTADOR : cuenta elementos de uno a uno
- ACUMULADOR : efectúa la sumatoria de valores
- Comparar 2 valores y elegir entre 2 caminos posibles.
- Generar ciclos definidos y condicionados.
- Posibilidad de definir y usar funciones que efectúan cálculos específicos, como ser :
  - MAXIMO : halla el máximo de una serie de números.
  - MINIMO : determina el mínimo de una serie de números.
  - ORDENAR : ordena en forma ascendente o descendente una serie de valores





# Diseño de una estrategia (como)

## **c) Condición de final :**

todo programa tiene un final, bajo ciertas condiciones, que debemos especificar. Esta condición solo la aplicaremos en aquellos problemas que tienen más de un dato ó de un juego de datos.

**d) Condiciones particulares: propias del problema como ser excepciones, unidades, posibles errores,**  
evaluación de diversas alternativas ó soluciones posibles, etc.

# Diseño de una estrategia (como)

**Ejemplo :** Determinar la edad promedio de los alumnos del curso. Cuantos alumnos hay? .

1 - COMPRENSION del problema: ( qué ?? )

a) Resultado : edad promedio

b) Datos : Edad de cada alumno y cantidad de alumnos [ la edad debe ser  $> 0$  , hay edad mínima?]

c) Proceso : cálculo del promedio (cociente entre suma de edades y cantidad de alumnos)

2 - ESTRATEGIA ( cómo ?? )

a - Obtengo la cantidad de alumnos y la edad de cada uno

b - Calculo el promedio

c - Informo la edad promedio

d - La edad promedio la considero en años ó años y meses ?



# Algoritmo

- Es un conjunto de pasos a seguir para la solución a un problema.
- Es una serie finita de instrucciones para realizar una tarea.

**“Es un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema.”**

Respecto a la estrategia, en lugar de hablar de procesos o tareas **hablamos de instrucciones**

**Involucra mayor grado de minuciosidad y precisión.**

**NOS ACERCAMOS AL NIVEL DE DETALLE QUE NOS RECLAMA LA COMPUTADORA**



# Algoritmo

**Para que una estrategia pueda llamarse algoritmo:**

- 1 – Que se hallen perfectamente identificados los datos de entrada.
- 2 – Que se produzca al menos un resultado.
- 3 – Que cada instrucción sea clara y precisa, sin ambigüedades.
- 4 – Que se halle compuesto por una cantidad finita de instrucciones.
- 5 – Que cada instrucción sea lo suficientemente básica para que pueda ser resuelta por cualquier computador.

# Características de los algoritmos

- Las características que debe cumplir un algoritmo son:
  - Un algoritmo debe ser **Preciso** e indicar el **orden de realización de cada paso**.
  - Un algoritmo debe ser **Definido**, es decir, **si se sigue un algoritmo dos veces, se debe obtener el mismo resultado**.
  - Un algoritmo debe ser **Finito**, es decir, **si se sigue el algoritmo se debe terminar en algún momento**.



# Otras características de los algoritmos


Debe cumplir con:

- Una secuencia de instrucciones claras y finitas
- Debe ser correcto y debe resolver el problema planteado en todas sus facetas
- Debe ser legible





# Elementos de un algoritmo

- Una definición de la entrada del algoritmo
  - Un lenguaje formal en el que definir el algoritmo
  - Una lista ordenada de sentencias con las que resolver el problema asociado al algoritmo
  - Una definición de la salida del algoritmo
- 



# Diseño del algoritmo

- La solución de un problema complejo puede requerir muchos pasos, es necesario dividir el problema en sub-problemas más sencillos de resolver.
- Este método se denomina *divide y vencerás* y es aplicable a la resolución y escritura de algoritmos y programas para computadora.
- Este método de división de un problema en otros sub-problemas más sencillos se puede expresar para conseguir su solución en una computadora, mediante el método denominado **diseño descendente**.
- El proceso de la rotura de un problema principal en etapas o sub-problemas más sencillos se denomina **refinamiento paso a paso o sucesivos**.

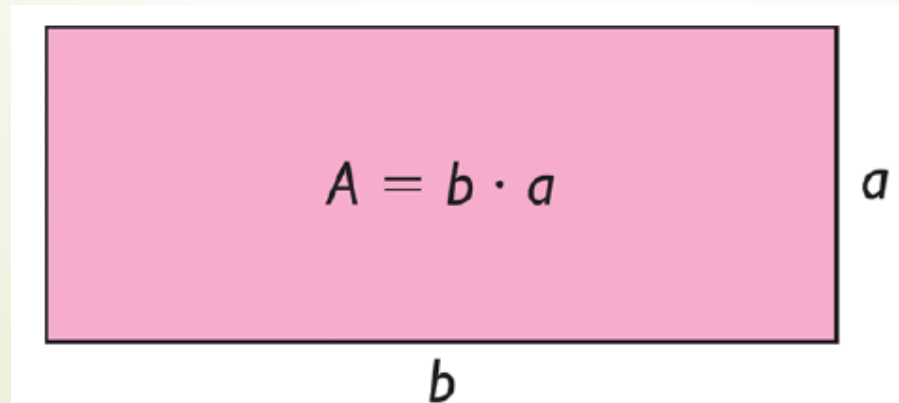
# Ejemplo: calcular el área de un rectángulo

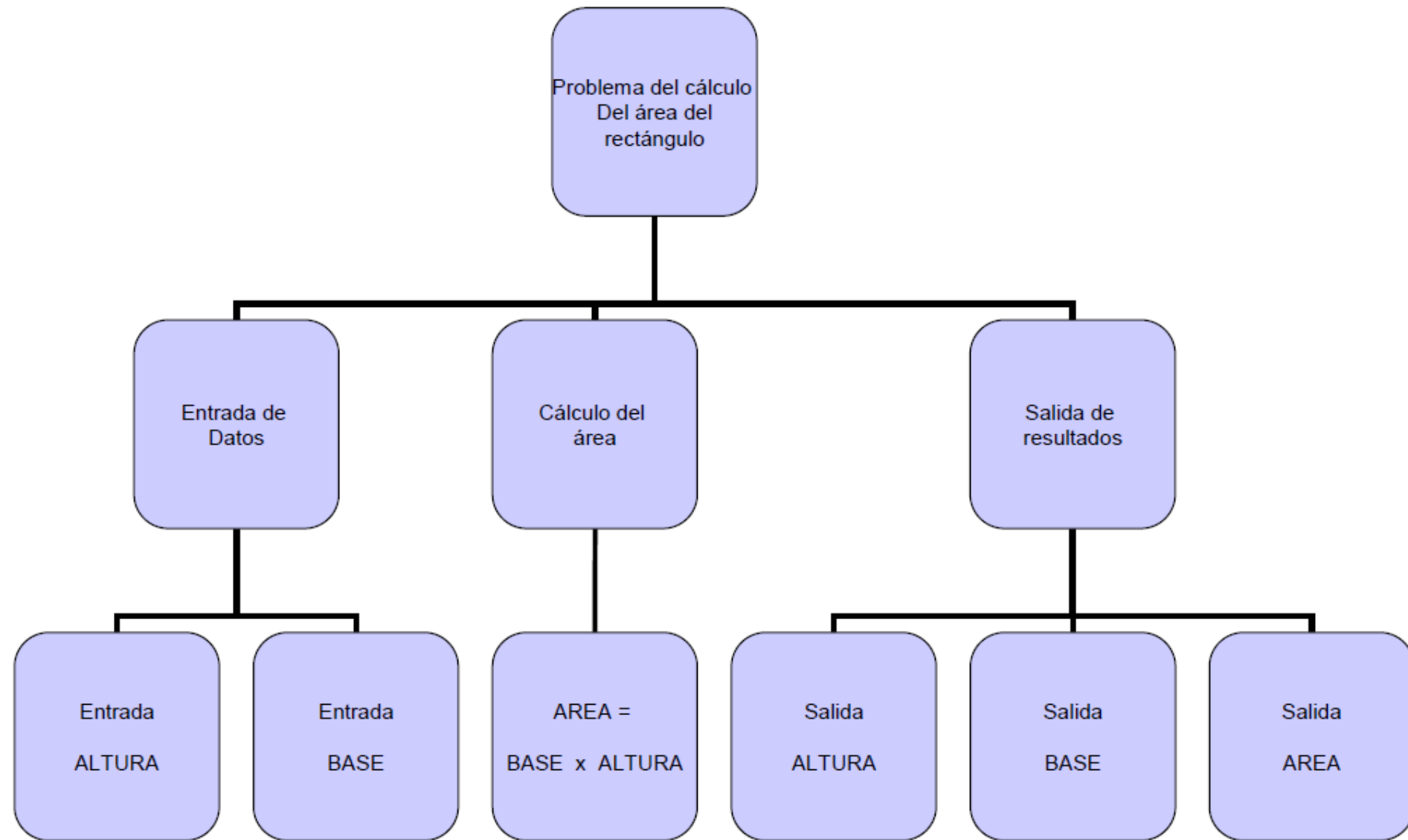
- Análisis del problema
  - El cálculo del área del rectángulo se puede dividir en:

**Entrada** de datos (altura, base)

**Proceso:** Cálculo del área (= base x altura)

**Salida** de datos (base, altura, área)







# Herramientas de programación

- Las herramientas de programación utilizadas como lenguajes algorítmicos son:
  - **Pseudocódigo:** es un lenguaje algorítmico, muy parecido al español pero más conciso que permite la redacción rápida del algoritmo.
  - **Diagramas de flujo:** ha sido la herramienta de programación por excelencia, y aún hoy sigue siendo muy utilizada. Es fácil de diseñar pues el flujo lógico del algoritmo se muestra en un diagrama en lugar de palabras.



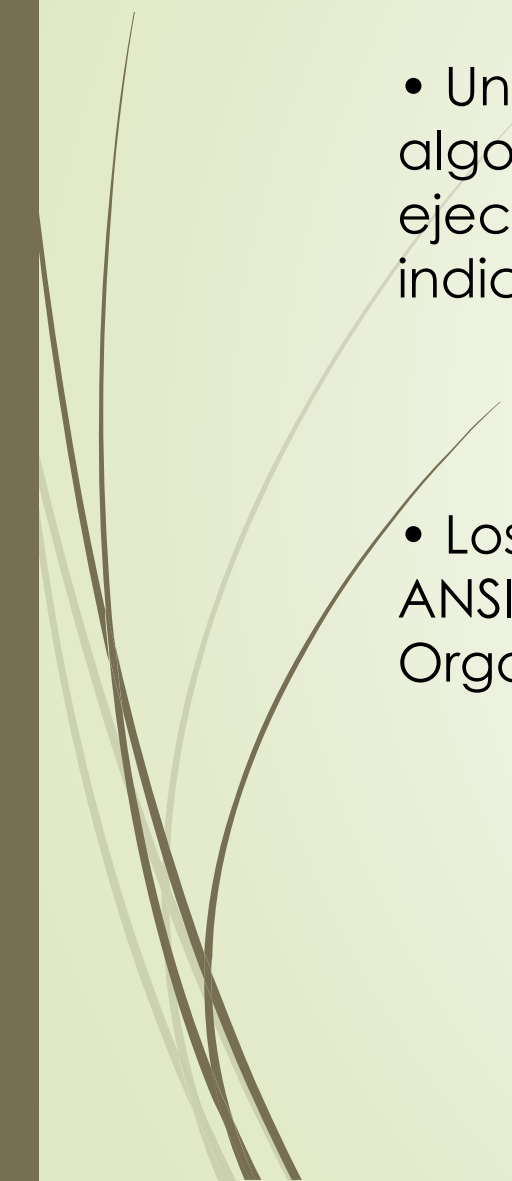
# Pseudocódigo

- Es un lenguaje de *pseudoprogramación*, es decir, muy parecido a un lenguaje de programación.
- El **pseudocódigo** es muy fácil de utilizar, ya que es muy similar al lenguaje coloquial.

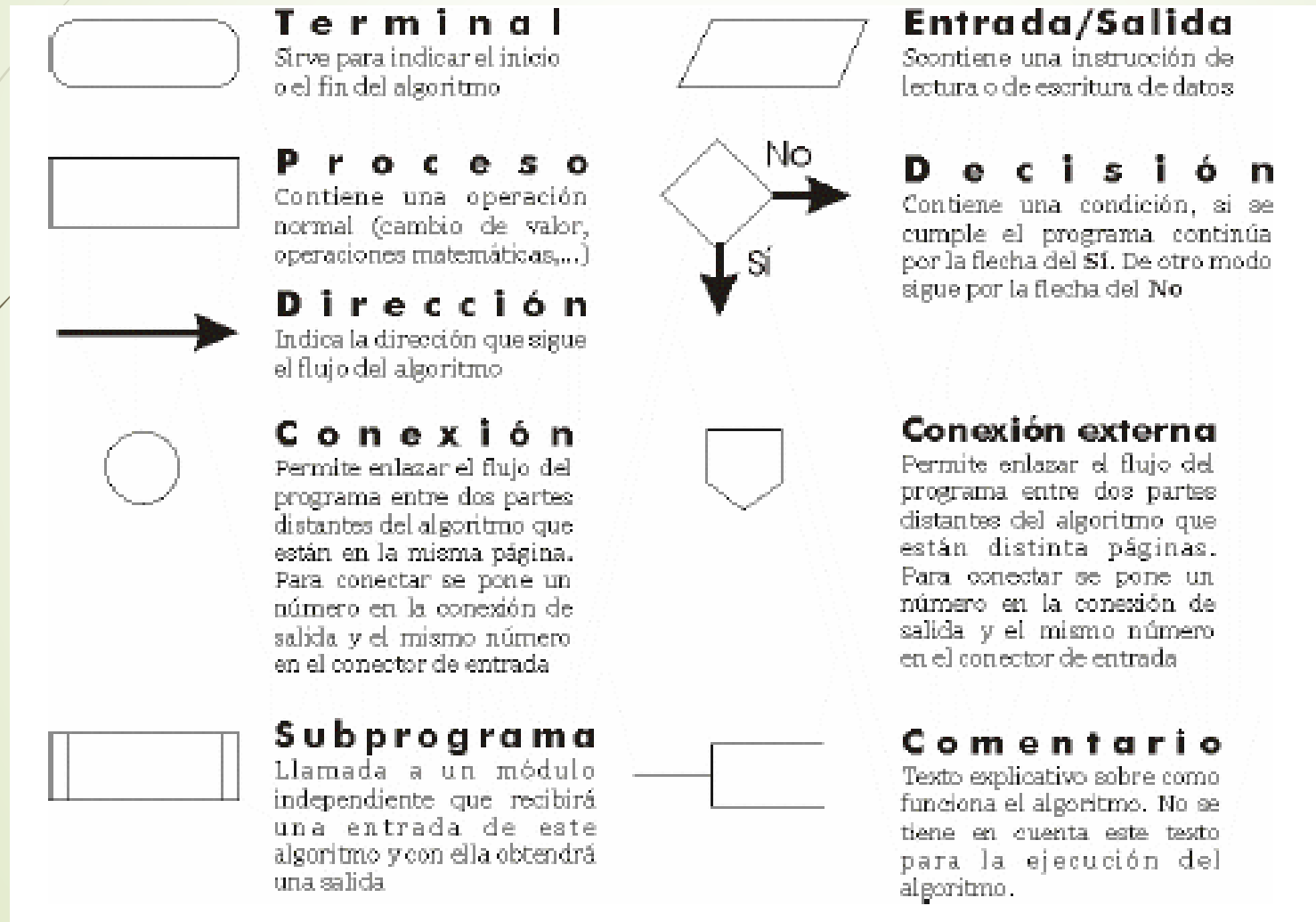




# Diagramas de flujo

- Un diagrama de flujo utiliza símbolos estándar en el que cada paso del algoritmo se visualiza dentro del símbolo y en el orden en que estos pasos se ejecutan, se indica conectándolos con flechas llamadas *líneas de flujo*, ya que indican el flujo lógico del algoritmo.
  - Los símbolos utilizados en los diagramas de flujo han sido estandarizados por la ANSI (American National Institute) y por la ISO (International Standard Organization)
- 

# Símbolos de diagramas de flujo 1



# Variables y Constantes

## a) Variables :

- Elementos que toman valores específicos de algún tipo de datos.
- Lo toman por asignación
- Lo toman por entradas de datos
- Es una representación simbólica de una dirección de memoria donde se guarda un valor (el valor de la variable).

VARIABLE	DIRECCION DE MEMORIA	VALOR
a	3AB2	3
b	3AB6	4
c	3ABA	5

## b) Constantes:

- Almacenan siempre el mismo valor.
- Se usan por ejemplo para patrones de conversión: fuerza de gravedad, pulgadas a cm, Pi, etc.

# Tipos de datos

- Tienen que ver con la clase de datos que una variable/constante pueden alojar.

Tipo de datos	Descripcion	Ejemplo
Entero int	Números enteros Se escriben tal cual Declaracion: int nombre_variable	Edad de una persona en años El numero de hijos Las notas de los alumnos de una comisión La velocidad de un auto
Real float	Números reales Se escriben tal cual con coma Declaración: float nombre_variable	Sueldo de un trabajador El promedio de ventas
Carácter char	Caracteres alfanuméricos Se escriben entre comillas simples Declaración: char nombre_variable	Estado civil de una persona Cada uno de los caracteres ASCII
Lógico o booleano	Verdadero o Falso Se escriben V F	El resultado de la comparación 5 > 10
Cadena (ó texto) char[x]	Conjunto de caracteres Se escriben entre comillas dobles Declaración char[10] nombre_variable	El nombre de una persona El domicilio de una persona Un número de telefono

# Operadores

Tipo de operador	Simbolo	Ejemplo
Asignación	<-	a <- 10 b <- a * c
Aritméticos	+ (suma) - (resta) * (producto) / (división) Mod (resto) Div (resultado entero en división) Exponente	9+(6/3)
De relación / comparación	> (mayor que) < (menor que) >= (mayor o igual que) <= (menor o igual que) <> != (distinto que) == (igual que)	a >= b  Siempre dan como resultado un valor verdadero o falso
Unión de expresiones	Y O Not	(a >= b) Y (a>=c)



# Instrucciones primitivas

- Asignaciones
- Operaciones (+, -, \*, .....
- Identificadores (nombres de variables / constantes).
- Valores (números ó texto encerrado entre comillas).
- Llamadas a subprogramas.

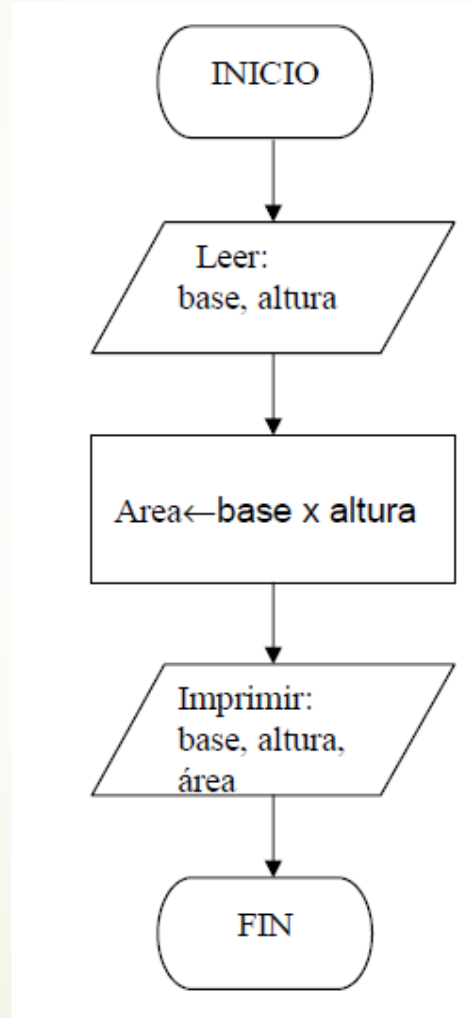




# Entrada / Salida

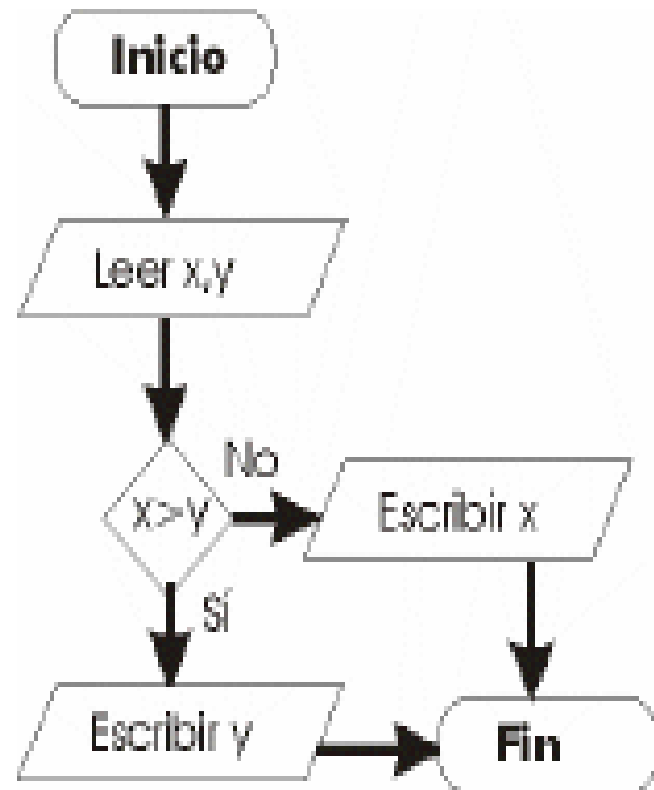
- **Entrada:**
  - Simula la lectura desde un dispositivo (teclado, lector de código de barras etc.)
  - Se hacer mediante la orden Leer.
- **Salida:**
  - Simula la salida a la pantalla, impresora, etc.
  - Se hace mediante la orden Escribir

# Ejemplo: Diagrama de flujo



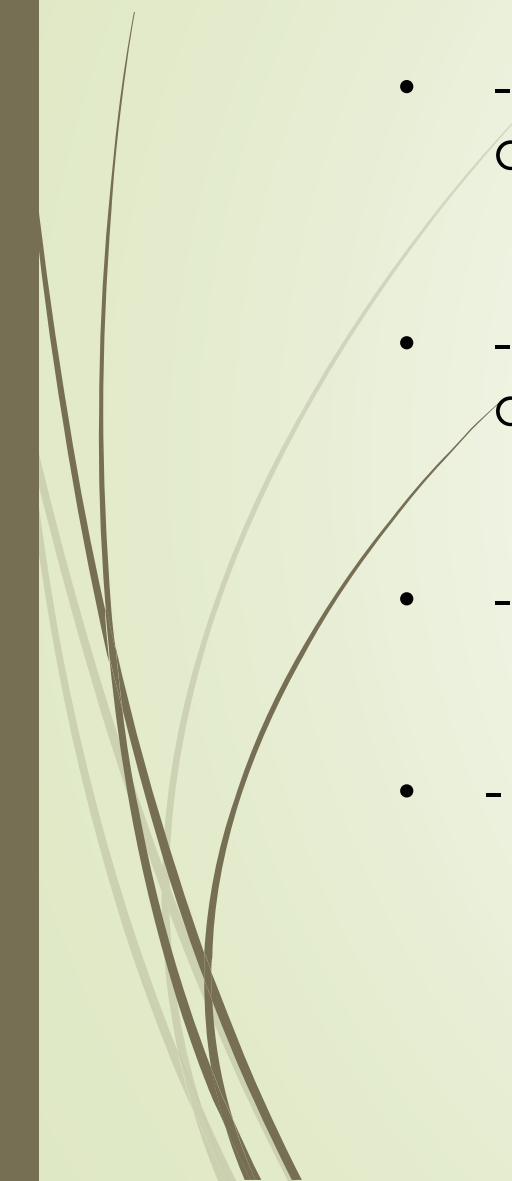
# Ejemplo: Diagrama de flujo

Ejemplo:





# Diagramas de flujo - ventajas

- - permite una sencilla y rápida visualización total del problema y sus distintas alternativas.
  - - es un verdadero medio de comunicación entre quien explica y quien aprende.
  - - es un medio claro y conciso de documentación de los problemas
  - - facilita los posibles cambios y visualiza distintas alternativas posibles
- 



# Prueba de algoritmos

Consiste en la elaboración de un **"juego de prueba"**, que es un conjunto de datos, algo especial, para poder probar el algoritmo elaborado.

Con estos datos efectuamos el seguimiento paso a paso del diagrama y ayudados por una lista con las variables utilizadas, donde registramos los valores que van tomando, podemos determinar los resultados a obtener