Metodología de Desarrollo de Programas

Prof. Judith Barrios Albornoz

Departamento de Computación

Escuela de Ingeniería de Sistemas

Facultad de Ingeniería

Universidad de Los Andes

Semestre A_2013

Este material toma como base el curso original Prof. Narciso. Año 2000- 2006

¡ ATENCIÓN !

El ***software*** proporciona la inteligencia que permite a los componentes del ***hardware*** procesar y distribuir los datos y la información

Lenguajes de Programación

- ***Lenguajes de programación***: Permiten escribir programas comprensibles por la computadora
  - Algunos son comprensibles de forma directa por la computadora.
  - Otros requieren pasos intermedios de traducción para ser comprendidos por la computadora

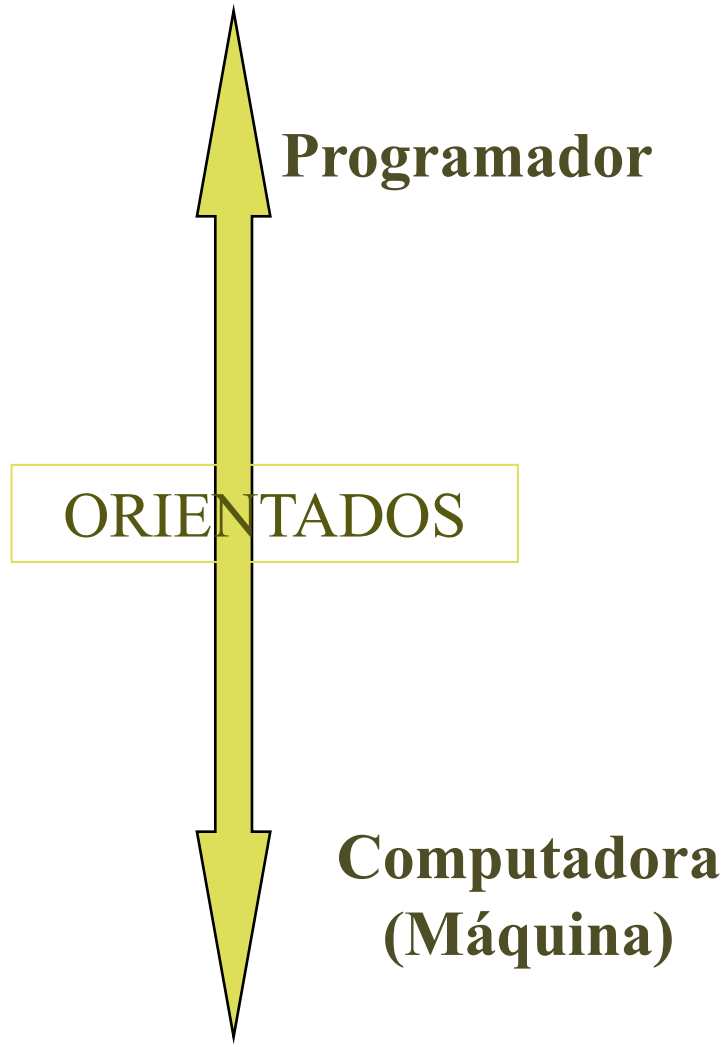
- **Clasificación**:
  - **Lenguaje de máquina**
  - **Lenguaje de bajo nivel o ensamblador**
  - **Lenguaje de alto nivel**

Lenguajes de Programación

Lenguaje de **alto nivel**

Lenguaje **ensamblador**

Lenguaje de **máquina**



Lenguaje de Máquina

- Lenguaje **básico**, propio de cada computadora - está relacionado con el diseño del *hardware* de la misma (***dependiente de la máquina***)
  - Por lo general consisten en cadenas de números al final reducidos a ceros y unos (sistema numérico binario)

- **Operaciones**:
  - Cargar – llevar a memoria
  - Almacenar - guardar
  - Sumar
  - Restar

Lenguaje de Máquina

Ejemplo:

Código de operación	Dirección	Significado
00010101	10000001	**Cargar** contenido de la dir. 129 en Acumulador
00010111	10000010	**Sumar** contenido de la dir. 130 al Acumulador
00010110	10000011	**Almacenar** contenido del Acumulador en la dir. 131

Lenguaje Ensamblador

- Consiste en *abreviaturas* similares al inglés, llamadas instrucciones ***mnemotécnicas***, que permiten representar las operaciones elementales de la computadora ***(dependiente de la máquina)***

Ejemplo:

Código de operación	Dirección	Instrucción en lenguaje ensamblador
00010101	10000001	**LOAD** A
00010111	10000010	**ADD** B
00010110	10000011	**STORE** C

Lenguaje Ensamblador- *bajo nivel*

- La computadora no entiende directamente lenguaje ensamblador por lo que un **programa escrito en este lenguaje tiene que ser traducido a lenguaje de máquina por un programa llamado un *ensamblador*** para que pueda ser ejecutado por la computadora

Ejemplo:

lenguaje	Código de operación	Dirección	Instrucción en ensamblador
	00010101	10000001	**LOAD** A
	00010111	10000010	**ADD** B
	00010110	10000011	**STORE** C

Lenguaje Ensamblador- *bajo nivel*

- **Requieren que** el **programador** tenga un **buen conocimiento de la arquitectura de la computadora**
- Como estos **lenguajes ensambladores son *dependientes* de la máquina**:
  - todo programa escrito en un lenguaje ensamblador particular tendrá que ser reescrito si se va a ejecutar en otro tipo de computadora

Lenguaje de Alto Nivel

Permite a los programadores escribir instrucciones en un lenguaje mas familiar para ellos y que contiene notaciones matemáticas comúnmente utilizadas ***(independiente de la máquina)***

Ejemplo:

Código de operación	Dirección	Instrucción en lenguaje ensamblador	Instrucción en lenguaje de alto nivel
00010101	10000001	LOAD A	
00010111	10000010	ADD B	
00010110	10000011	STORE C	**C = A + B**

La programación es más sencilla ya que no se necesita conocimiento de la arquitectura de la computadora para escribir los programas

Lenguaje de Alto Nivel

La computadora no entiende directamente lenguaje de alto nivel - igual que con los lenguajes ensambladores -

un programa escrito en este lenguaje tiene que ser traducido a lenguaje de máquina por un programa llamado un *compilador*

para que pueda ser ejecutado por la computadora

Un programa escrito en un lenguaje de alto nivel particular **(que tiene una versión estandarizada)** puede ser **ejecutado en cualquier otra computadora** -

lenguajes de alto nivel son independientes de la máquina

Lenguaje de Alto Nivel

Ejemplos *de lenguajes de alto nivel*:

- FORTRAN
- ALGOL
- COBOL
- BASIC
- PL/I
- PROLOG
- LISP
- SMALLTALK
- PASCAL
- ADA

- **C**
- Turbo C
- Turbo Basic
- Turbo Pascal
- C++
- Visual C
- Java
- C#

Lenguajes de Programación

Elementos de un lenguaje de programación

Un ***sub-lenguaje*** para **definir los datos**

- ¿Qué datos tenemos o necesitamos?
- ¿Cómo los llamamos?
- ¿Cómo son? (qué tipo y/o estructura tienen)
- ¿Qué se puede hacer con ellos?

Un ***sub-lenguaje*** para **definir los algoritmos**

- ¿Qué hacemos con los datos? procesamiento
- ¿En qué orden (cuándo lo hacemos)?
- ¿Cuántas veces?

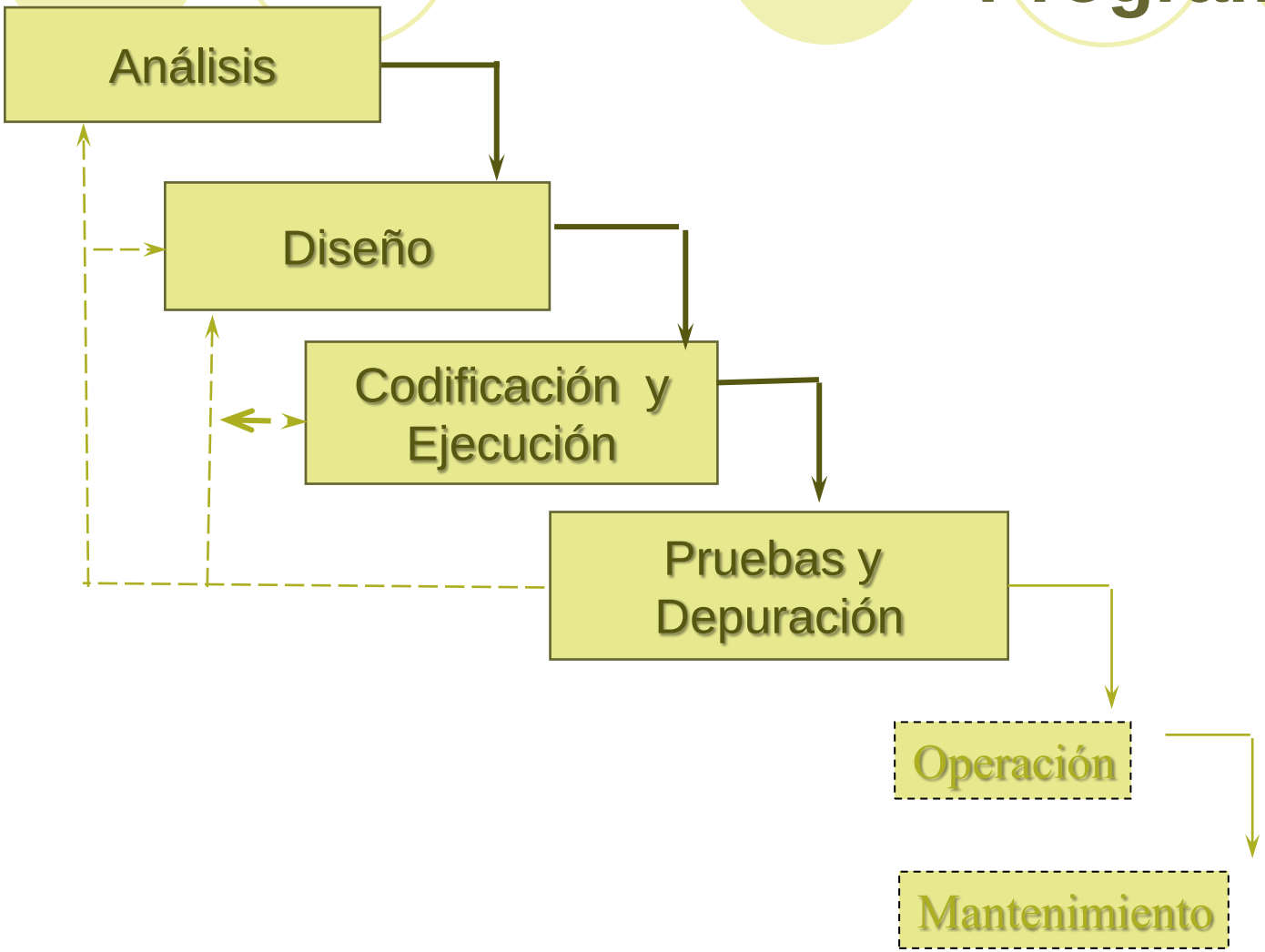
Lenguajes de Programación

¡ Atención- Importante !

- **Cada CPU tiene su propio lenguaje de máquina interno**.
  - La programación a este nivel se realiza en el lenguaje ensamblador específico de la computadora.
  - Cada instrucción en lenguaje ensamblador corresponde a una instrucción en lenguaje de máquina

- Si existe una **estandarización para un lenguaje de alto nivel**, cualquier programa escrito usando este estándar debe poder **ejecutarse** en cualquier computadora **después de compilarlo**.
  - Esto se le conoce como ***portabilidad*** de programas

Metodología de Desarrollo de Programas



Metodología de Desarrollo de Programas

Análisis del problema (**E- P - S**)

- Describir los datos que se requieren para resolver el problema o sobre los cuales se basa la solución del problema - entradas (*entrada al programa*) – usuario/otros programas
- Establecer el proceso - qué se debe hacer con los datos - sobre los datos y cuáles son los resultados que se producen (salida del programa)

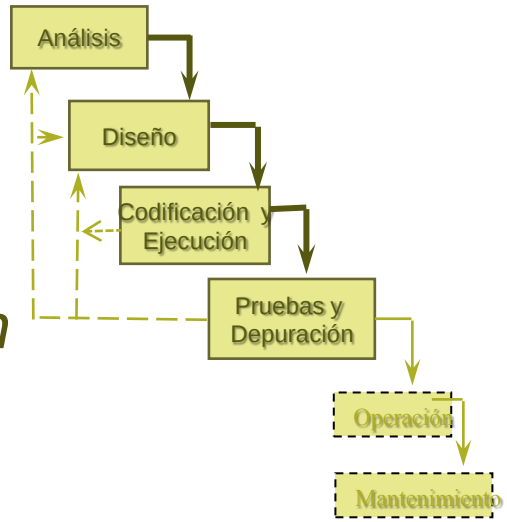
Identificar y comprender el problema

Metodología de Desarrollo de Programas

Diseño de la solución – *algoritmo*

Algoritmo: Descripción de una **secuencia finita y ordenada de pasos** – sin ambigüedades – que conducen a la **solución** de **un problema** específico

- Herramientas de diseño
  - **Pseudocódigo**
  - Diagrama de flujos (*programación estructurada*)
  - Círculos y canales de mensaje (*programación orientada a objetos*)

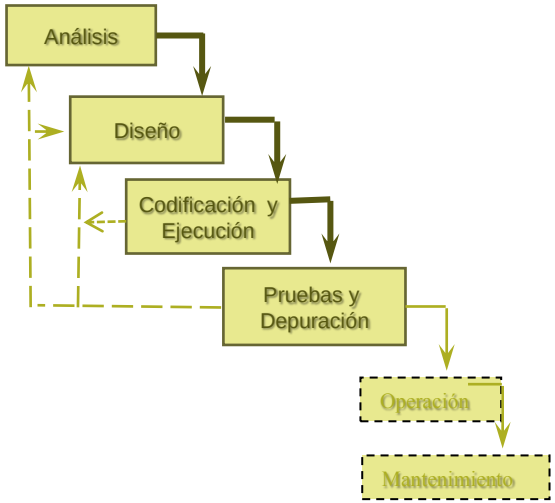


Metodología de Desarrollo de Programas

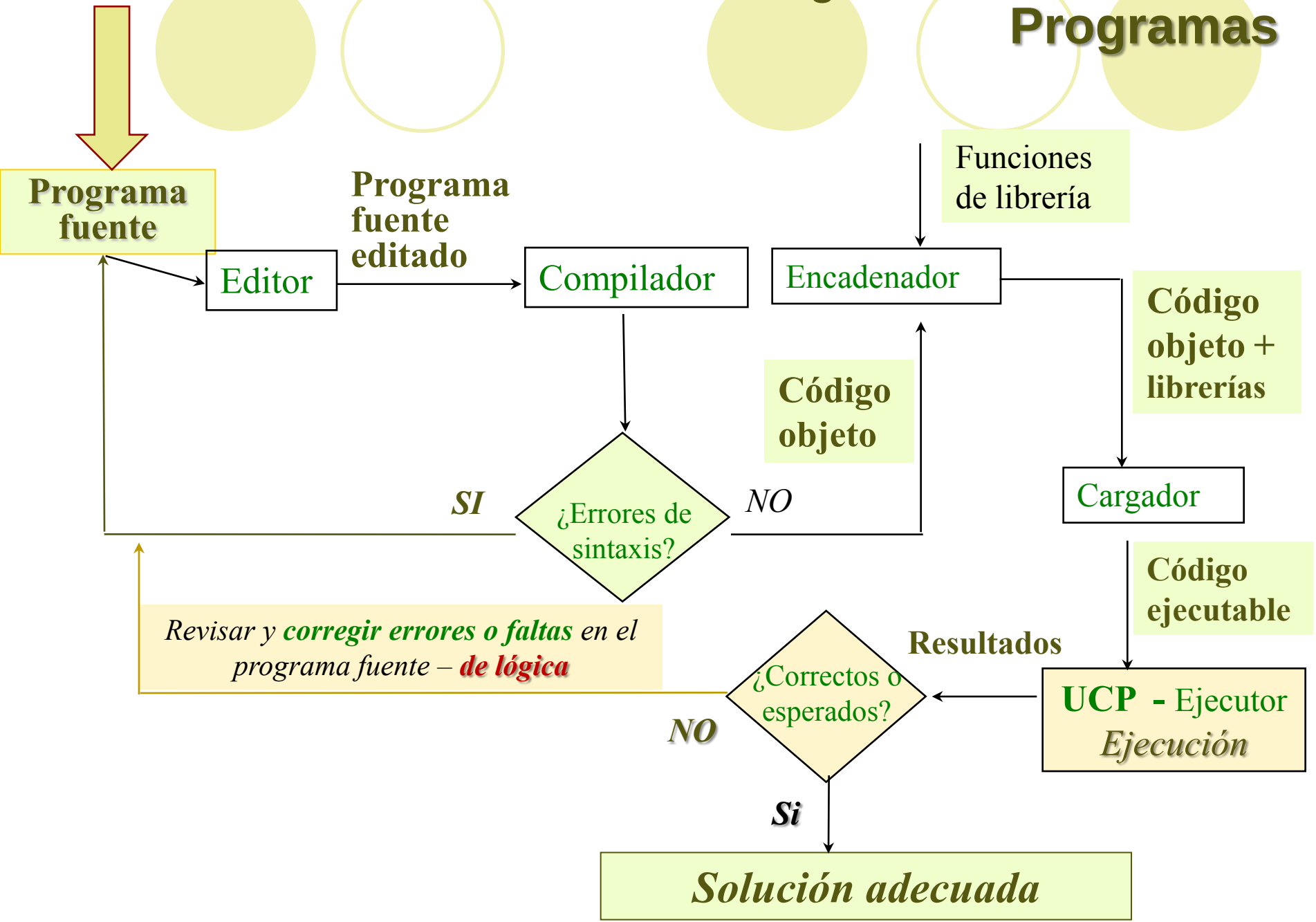
Codificación: **Traducción** del **algoritmo** – diseño - a un programa escrito en un **lenguaje** de **programación**

- **producción** del **código fuente**

  edición, compilación de las instrucciones del lenguaje

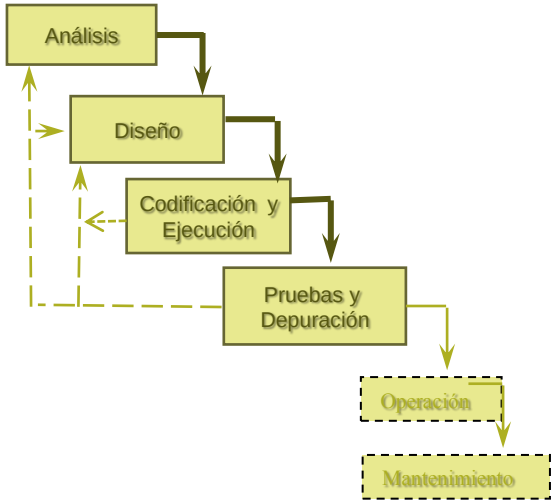


Metodología de Desarrollo de Programas



Metodología de Desarrollo de Programas

Ejecución : Ejecución del **código ejecutable** (*código ya en lenguaje de máquina*) del programa – se realiza una instrucción a la vez - bajo el control del **UCP**



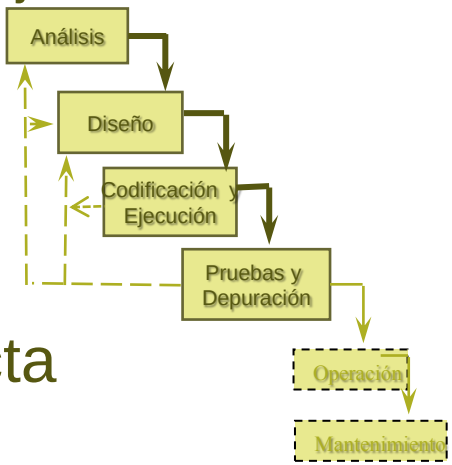
Metodología de Desarrollo de Programas

Pruebas y depuración :

Detección, identificación y eliminación de errores

Errores de sintaxis: Violan las reglas del lenguaje de programación

El compilador los localiza e identifica



Errores de lógica: ***Equivocaciones*** – *faltas o excesos* - que causan que el programa se ejecute de forma incorrecta o inesperada

Resultados NO son CORRECTOS - NO son los ESPERADOS

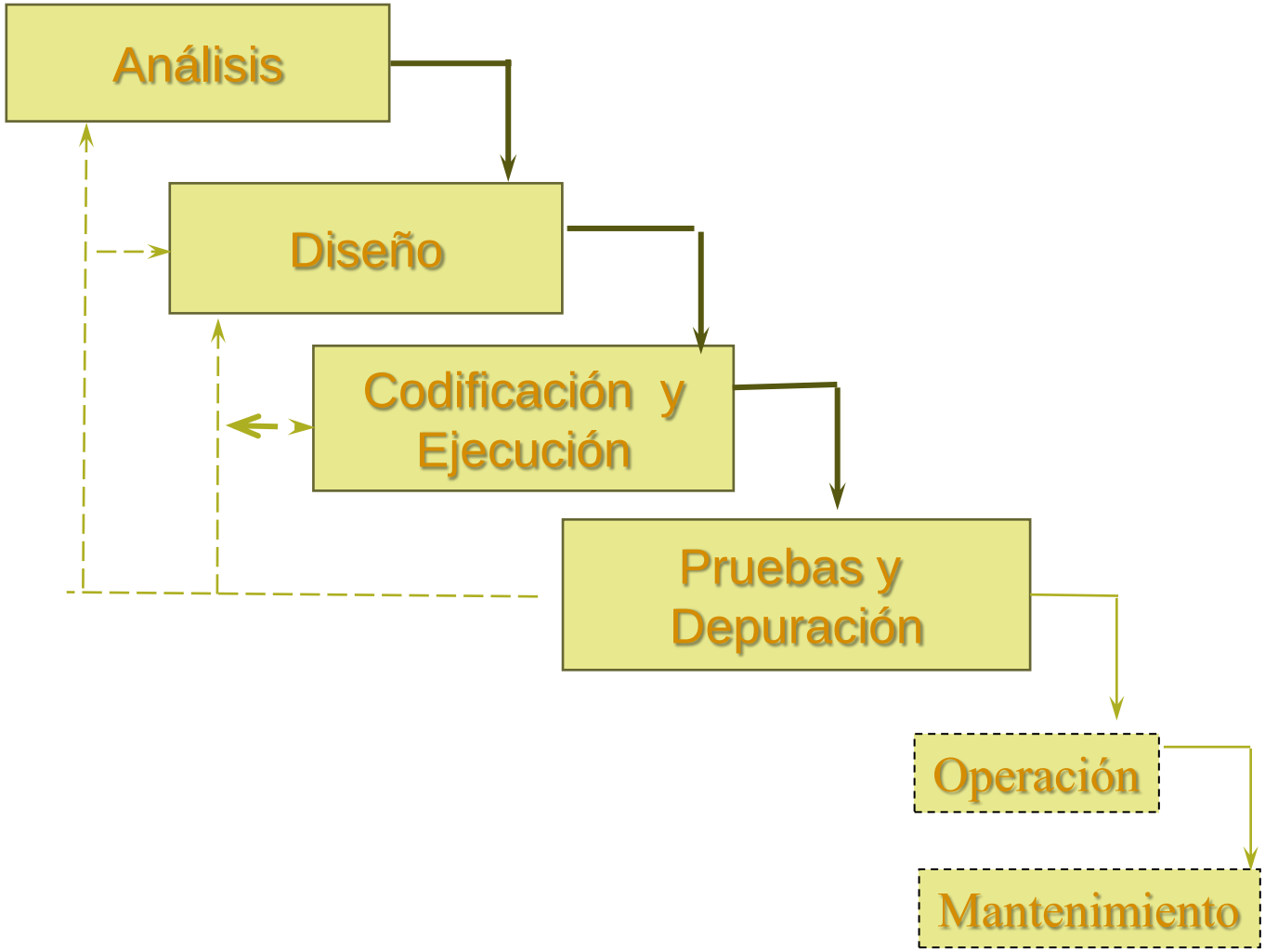
Metodología de Desarrollo de Programas

Puesta en operación: Instalación del *hardware* y *software*, capacitación de usuarios

Mantenimiento del programa:

Permite corregir defectos menores, añadir una mayor funcionalidad, ya sea en respuesta a las demandas del mercado o a las peticiones del usuario – *aparecen después que el programa esta instalado y operando*

Metodología de Desarrollo de Programas



Metodología – *detallada-* de Desarrollo de Programas **Análisis E-P-S**

Identificar y comprender el problema ….

Especificaciones de **E**ntrada*:* datos / elementos que se requieren para la solución del problema

- ¿ Qué y cuántos datos se requieren?
- ¿ En qué orden se introducirán ?
- ¿Qué tipos de datos?
- ¿ Cuáles datos de entrada son válidos ? ¿Rangos?

Metodología – *detallada-* de Desarrollo de Programas Análisis E-P-S

Identificar y comprender el problema ….

Definición del Proceso: *Operaciones, transformaciones o cálculos* que son necesarios para encontrar la solución del problema – involucran los datos definidos en la entrada -

- ¿ Qué tipo de ecuaciones ?
- ¿ Cuántas ecuaciones ?

Metodología – *detallada-* de Desarrollo de Programas

Análisis E-P-S

Identificar y comprender el problema ….

Especificaciones de **S**alida*:*

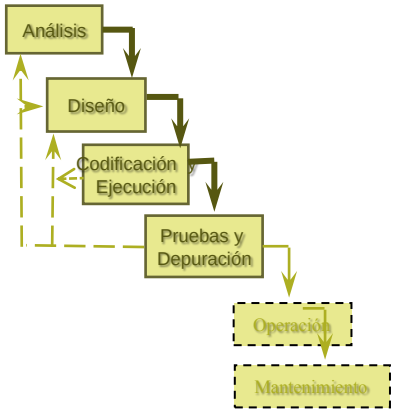
Resultados finales de los cálculos

- ¿ Cuáles son los datos de salida ?
- ¿ Cuántos datos de salida se producirán ?
- ¿ Qué precisión debe tener la salida -resultados ?
- ¿ Cómo se presentan los resultados? Por pantalla? Impresos?

Metodología *– detallada-* de Desarrollo de Programas

- **Diseño del algoritmo** (*lógica de solución*)
  - 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

secuencia finita y ordenada de pasos …



Ejemplo : Realizar el análisis **E-P-S** y diseñar un **algoritmo para calcular el área de superficie de un paralelepípedo** de dimensiones **l (*largo*), a (*ancho*) y h (*altura*)**

Análisis E-P-S

- **Entrada**: **tres números reales l** (largo), **a** (ancho) y **h** (altura)
- **Proceso**:
  - Calcular el área del paralelepípedo

AREA = 2 x (l x a + l x h + a x h)

- **Salida**: **número real** que representa el área (**AREA**)

Diseño del *algoritmo*

Algoritmo Área

0. Inicio
1. Escribir("Largo del paralelepípedo = ")
2. Leer el valor de **l**
3. Escribir("Ancho del paralelepípedo = ")
4. Leer el valor de **a**
5. Escribir("Altura del paralelepípedo = ")
6. Leer el valor de **h**
7. **AREA** = 2 x (l x a + l x h + a x h)
8. **Escribir**("Área del paralelepípedo =", **AREA**)
9. Fin

Ejercicios

Realizar el análisis **E-P-S** y **diseñar un algoritmo** para resolver los siguientes problemas:

- **Cambiar un billete de Bs. 100 en sencillo. El cambio puede incluir billetes de Bs. 50, 20, 10 o 5, según corresponda**
- Comprar una entrada para un concierto en el estadio Metropolitano
- Calcular el promedio de notas de una materia que tiene 4 parciales.
- **Calcular la nota definitiva de una materia que tiene 4 parciales, donde los dos primeros tiene un peso del 45% sobre el total, el tercero tiene un peso del 35% y el último 20%**

- **Cambiar el caucho a un carro**

Algoritmo pinchazo

0. *Inicio*

1. Si (*el gato del carro está dañado o no está*) o (*no hay repuesto*) o (*no está en buen estado*) o (*no sabe cambiar un caucho*)

Solicitar ayuda

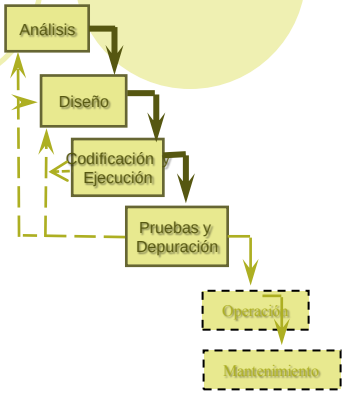
sino

Sacar el gato, las herramientas y el caucho de repuesto
Colocar el gato
Aflojar los tornillos del caucho
Levantar el carro con el gato
Sacar los tornillos del caucho
Quitar el caucho
Poner el caucho de repuesto
Poner los tornillos
Apretar un poco los tornillos
Bajar el gato
Terminar de apretar los tornillos
Guardar el gato, herramientas y el caucho dañado

fin_si

2. *Fin del algoritmo*

Metodología de Desarrollo de Programas
Codificación



Traducir el algoritmo en un programa escrito en un lenguaje de programación de alto nivel ---- en **C**

programa fuente o código fuente

Los pasos del algoritmo se expresan como sentencias o proposiciones de lenguaje

Programa

Secuencia de sentencias - ellas especifican las operaciones – tareas o acciones que debe ejecutar la computadora

ordenes

Ejemplo : Realizar el análisis E-P-S y **diseñar un algoritmo** para calcular el área de superficie de un paralelepípedo de dimensiones **l** (*largo*), **a** (*ancho*) y **h** (*altura*)

Diseño del algoritmo

```
Algoritmo Area
0. Inicio

1. Escribir("Largo del paralelepipedo = ")
2. Leer el valor de l
3. Escribir("Ancho del paralelepipedo = ")
4. Leer el valor de a
5. Escribir("Altura del paralelepipedo = ")
6. Leer el valor de h
7. AREA = 2 x (l x a + l x h + a x h)
8. Escribir("Area del paralelepipedo =",
           AREA)
9. Fin
```

Codificación

```
float Area()
{
    float l, h,a, AREA;

printf ("Largo del paralelepípedo =  \n");
scanf  ("%f", &l);
printf ("Ancho del paralelepípedo= \n");
scanf ("%f", &a);
printf ("Altura del paralelepípedo= \n");
scanf ("%f", &h);
AREA = 2 * (l * a + l * h + a * h);
printf ("Área de superficie del
      paralelepípedo =  %f" , AREA);
}
```

Metodología de Desarrollo de Programas
Codificación

Tipos básicos de sentencias

•Sentencias de **entrada/salida**: Sentencias de transferencia de datos e información entre dispositivos de E/S (teclado, impresora, discos, etc.) y la memoria principal

•Sentencias **aritmético-lógicas**: Sentencias que ejecutan operaciones aritméticas (suma, resta, multiplicación, etc.) o lógicas (y *lógico*, o *lógico*, negación)

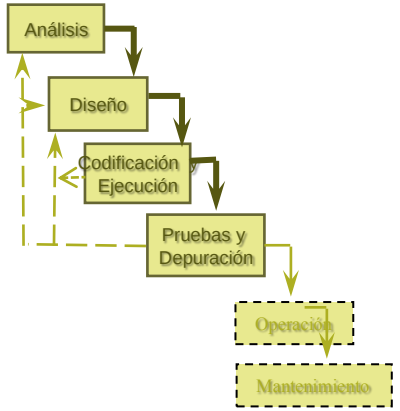
Sentencias de **decisión o selección**: Sentencias que permiten la selección de tareas alternativas en función de los resultados de diferentes expresiones condicionales

Sentencias de **repetición o lazos**: Sentencias que permiten la repetición de secuencias de sentencias un número determinado o indeterminado de veces

Metodología de Desarrollo de Programas
Ejecución del programa

- El **programa fuente** es introducido a la computadora utilizando un programa llamado **editor**

- Una vez editado, el **programa fuente es traducido por el compilador** a un programa escrito en lenguaje de máquina (**código objeto**), siempre y cuando el programa no tenga **errores de sintaxis** (errores de gramática)



Ejemplo

```
if (a < b          /* Falta un paréntesis que cierra */
```

Metodología de Desarrollo de Programas
Ejecución del programa

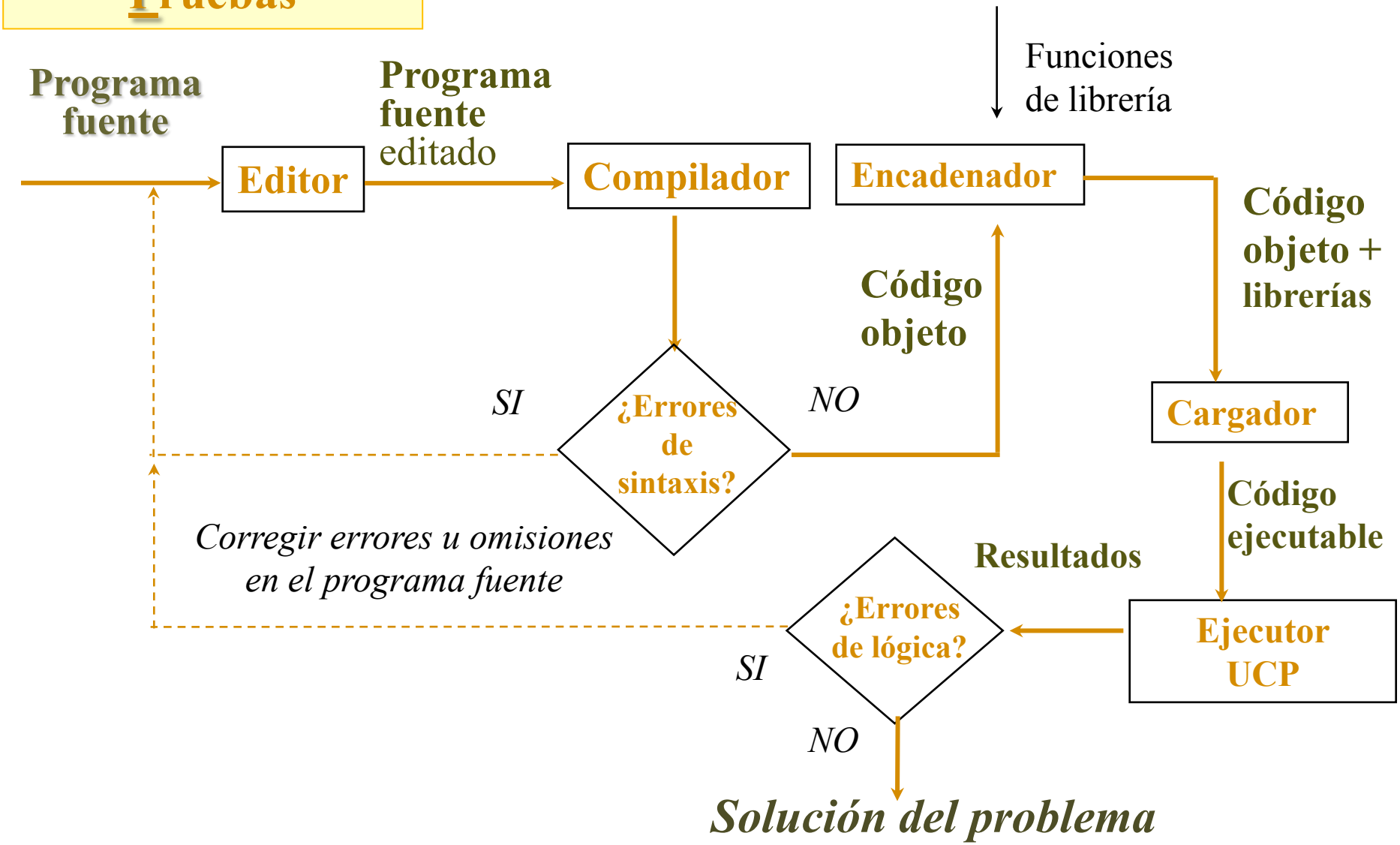
- Si el programa fuente tiene errores de sintaxis no se genera código objeto
  - Los errores deben ser **corregidos** usando el editor y luego se el programa fuente se debe **volver a compilar**
- Cuando el **programa** está sintácticamente **correcto**
  - **el código objeto es encadenado con las funciones de librería** (*otros programas especiales precodificados*) las cuales son requeridas usando un programa llamado **encadenador**

Ejecución del programa

- El código objeto compilado y encadenado es **cargado en la memoria principal** para su ejecución por un programa llamado cargador
- El código objeto compilado, enlazado y cargado (**código ejecutable**) es ejecutado con los datos de entrada

Metodología de Desarrollo de Programas

Análisis - Diseño
Codificación -
Pruebas



Pruebas del programa

Comprobar que el programa realiza las tareas para las cuales ha sido diseñado y produce el resultado correcto y esperado

Si el programa tiene **errores de lógica** (*errores en el método de solución por lo que las salidas obtenidas no corresponden con las salidas esperadas*)

Ejemplo

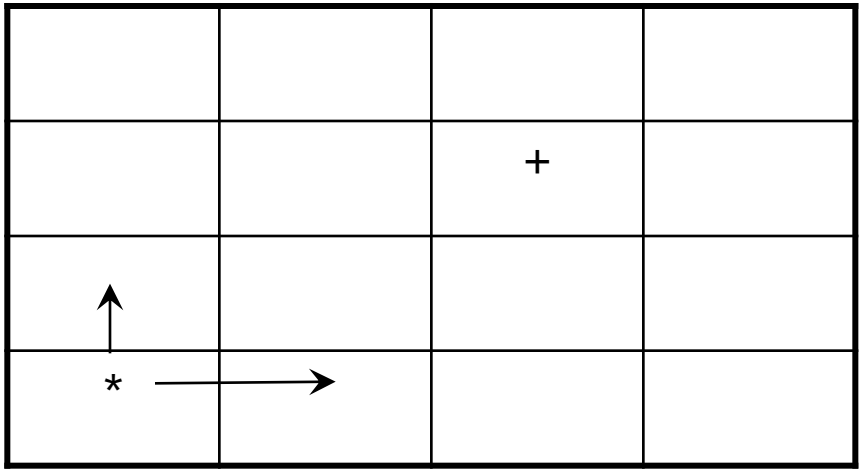
```
b = 0;
c = 5/b;          /* División entre cero */
```

Estos errores deben ser corregidos en el programa fuente usando el editor, el programa se debe volver a compilar

Ejercicio: dar ordenes a un Robot – casos **A** y **B**

- Movimientos:
  - Arriba
  - Abajo
  - Derecha
  - Izquierda
- desplazamientos

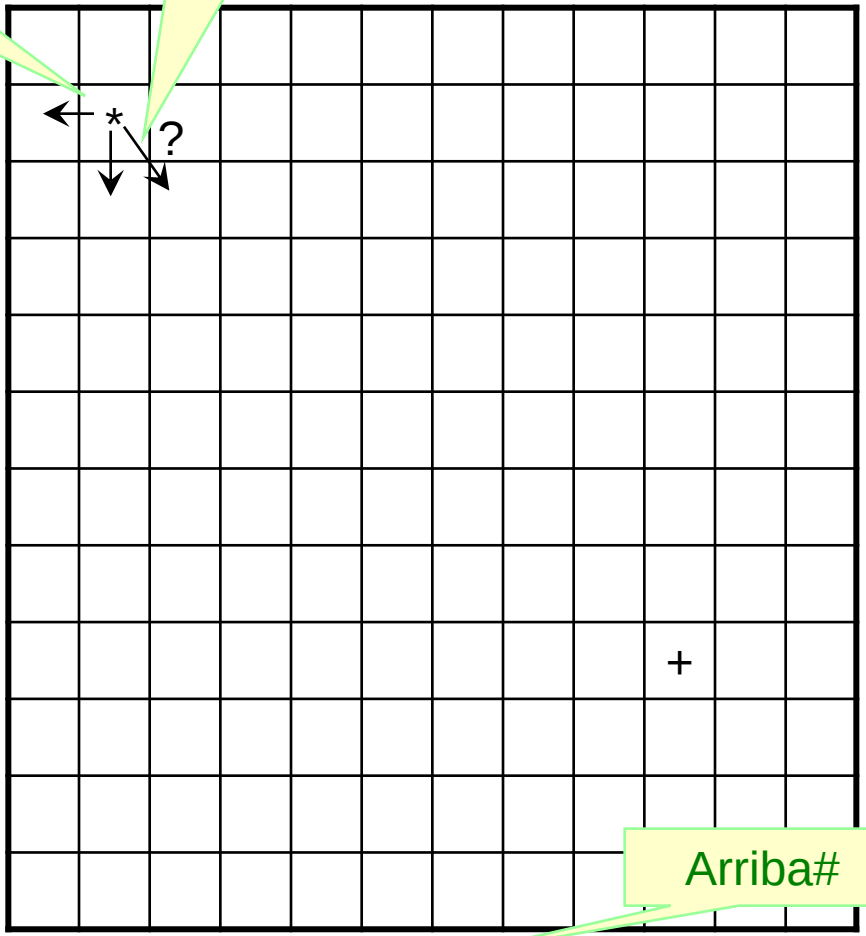
A



N, S, E, O, NE, NO, SE y SO

diagonal

B



Arriba#

Ampliar el juego de instrucciones