PROGRAMACIÓN PROCEDURAL

Lenguajes Procedurales

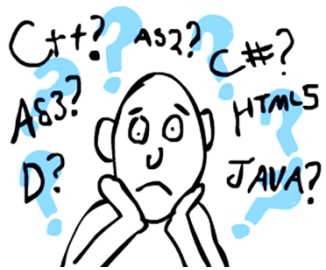
Unidad 1

Lenguaje de Programación

Lenguaje Natural

La comunicación necesita comprensión mutua de cierto conjunto de símbolos y reglas del lenguaje.

Lenguaje de Programación



Lenguaje de Programación

Un **lenguaje de programación** es un sistema notacional para describir **computaciones** en una forma legible tanto para la máquina como para el ser humano

Computar

Se puede interpretar como "*cualquier proceso que puede ser realizado por una computadora*".

No solo cálculos matemáticos sino también manipulación de datos, procesamiento de textos, almacenamiento y recuperación de la información.

Lenguaje de Programación

Objetivo: Construir programas, escritos por personas.

Los programas se ejecutarán sobre una computadora que realizará las tareas descritas.

Utilizar un lenguaje de programación requiere, comprensión mutua por parte de personas y máquinas.

Cuestiones de Diseño

El diseño de lenguajes exige:

- *legibilidad:* por parte del ser humano es un requisito complejo y sutil.
- *abstracción:* permitir concentrarse en un problema al mismo nivel de generalización, dejando de lado los detalles irrelevantes.

El principal objetivo de la abstracción en el diseño de lenguajes de programación *es el control de la complejidad*. Se controla la complejidad elaborando abstracciones que ocultan los detalles cuando es apropiado.

Cuestiones de Diseño

Tipos de Abstracciones

- **Abstracciones de datos**

  Técnica de inventar *nuevos tipos de datos* que sean más adecuados a una aplicación y, por consiguiente, facilitar la escritura del programa. (Ejemplo: uso de struct en lenguaje C).

- **Abstracciones de control**

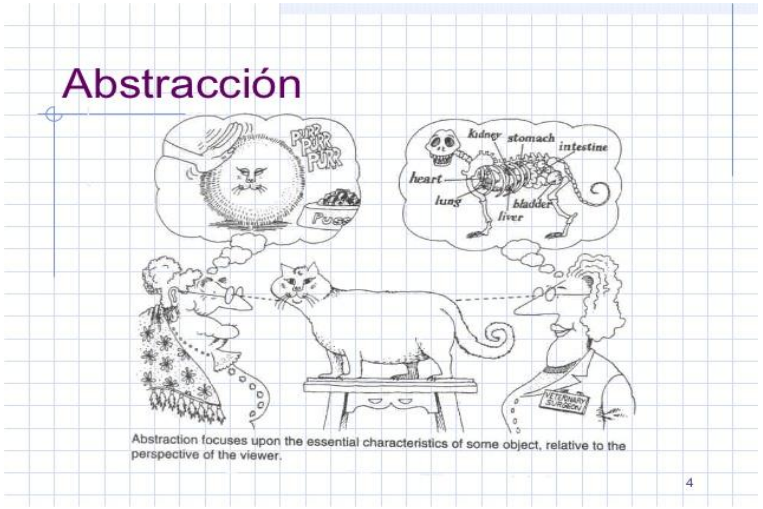
  Resume propiedades de la transferencia de control, es decir, la modificación de la trayectoria de ejecución de un programa en una determinada situación. (Ejemplo: for, if, while, etc, en C).

- **Abstracciones procedimentales**

  Es una secuencia nombrada de *instrucciones* que tienen una función específica y limitada. (Ejemplo: uso de funciones en C).

Abstracciones de datos

- int x; abstracción de datos básica.
- float notas[8]; abstracción de datos estructurada



Abstracciones de control

- a=a+3; abstracción de control básica, resume el cómputo y almacenamiento de un valor en el lugar de almacenamiento de la variable "a"
- sentencias estructuras for, if, while abstracciones de control estructuradas

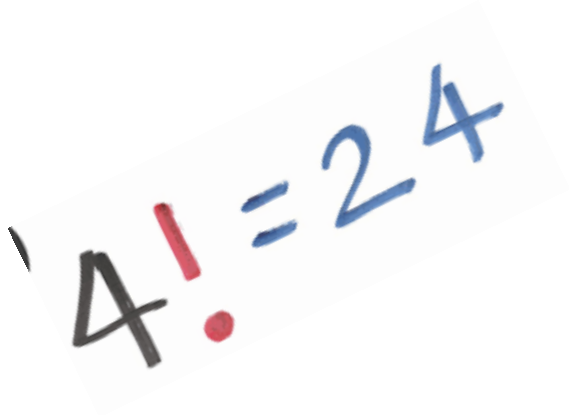
Abstracciones procedimentales

Secuencia nombrada de instrucciones que tienen una función específica.

Ejemplo: las funciones en lenguaje C

```
longint factorial (int x)
{
   int p=1;
   for(i=1; i<=x; i++)
        p=p*i;
   return p;
}

void main()
{ int f,m,n=6;
   m=sqrt(m);
   n= pow(m,3);
   f=factorial(n);
   printf("resultado = %d", f);
}
```



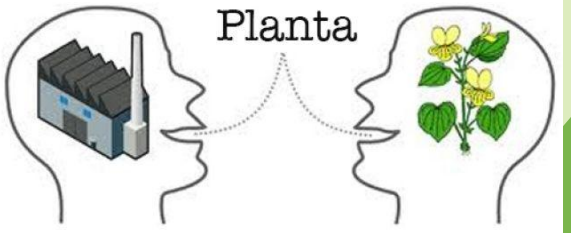
Definición de un lenguaje de programación

La definición de un lenguaje de programación necesita una descripción precisa y completa, además de un **traductor** que permita aceptar el programa en el lenguaje en cuestión y que, lo ejecute directamente o bien lo transforme en una forma adecuada para su ejecución.

Dos partes se pueden distinguir en la definición de un lenguaje:

La ***sintaxis*** estudia las reglas de formación de frases. Las reglas de sintaxis nos dicen cómo se escriben los enunciados, declaraciones y otras construcciones del lenguaje.

La ***semántica,*** sin embargo, hace referencia al significado de esas construcciones.



Computadoras Simuladas por Software y Traducción

¿Cómo se pueden ejecutar programas en lenguajes de alto nivel en computadoras reales con lenguaje de bajo nivel?

Simulación por software o Interpretación

es un software que recibe un programa en lenguaje de alto nivel, lo analiza y lo ejecuta. Para analizar el programa completo, va traduciendo sentencias de código y ejecutándolas si están bien, así

hasta completar el programa origen.

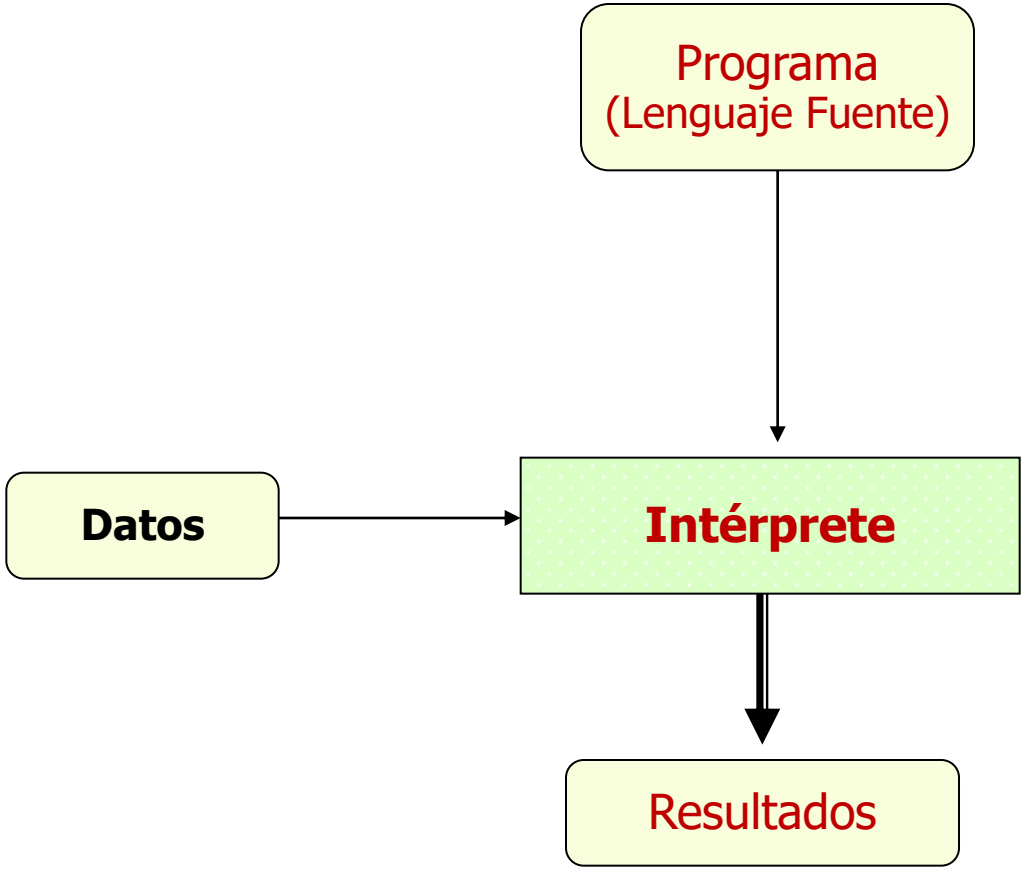
Traductor

Es un "*procesador de lenguajes*" que acepta programas en cierto lenguaje fuente como entrada y produce programas funcionalmente equivalentes en otro lenguaje objeto.

lenguajes interpretados (predomina la interpretación) (LISP, PROLOG, SmallTalk, ML)

lenguajes compilados (prevalece la traducción) (C , C++, Pascal, Fortran, Ada, Cobol)

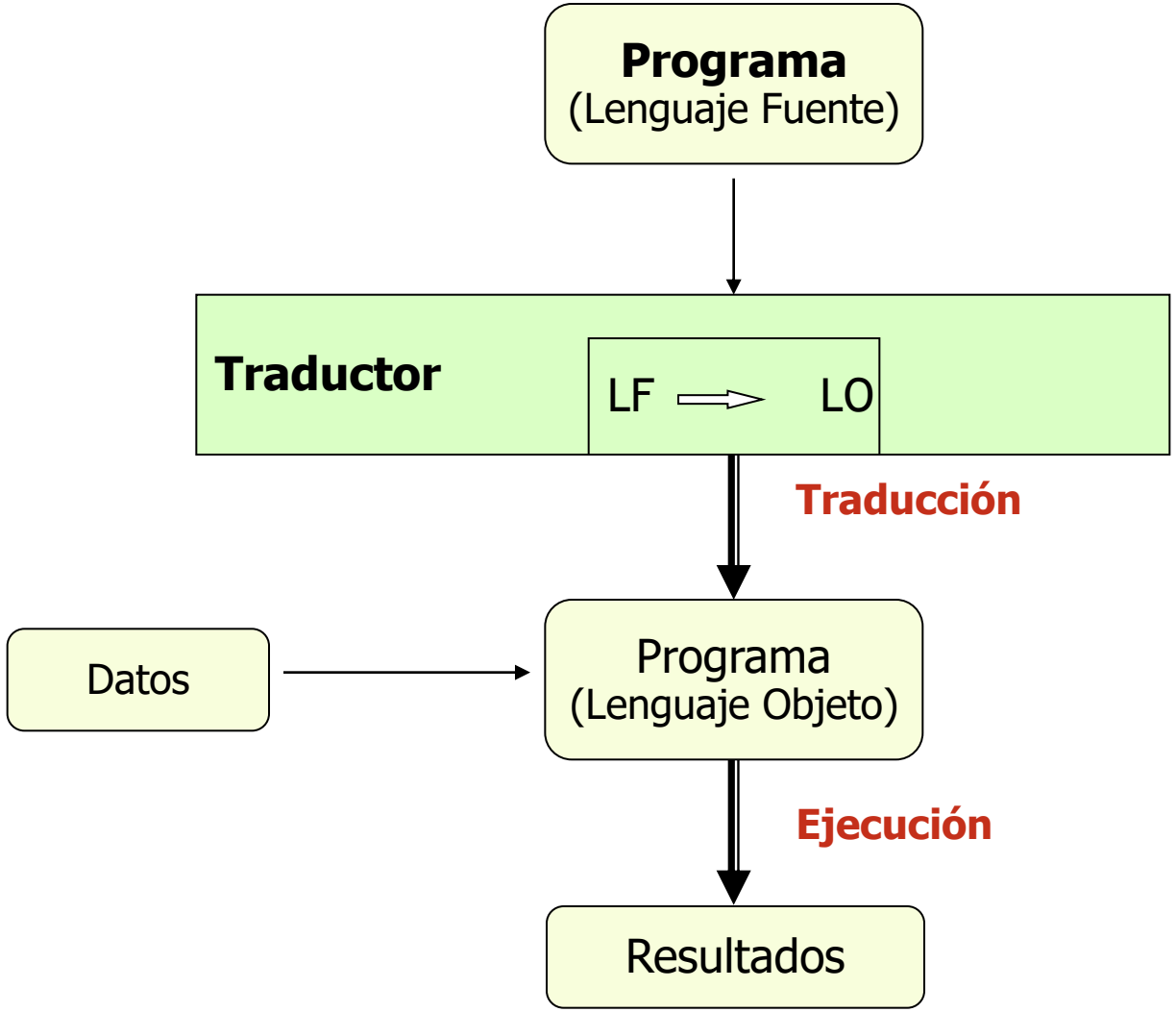
Esquema General de un Intérprete



Traductores (procesadores de lenguajes)

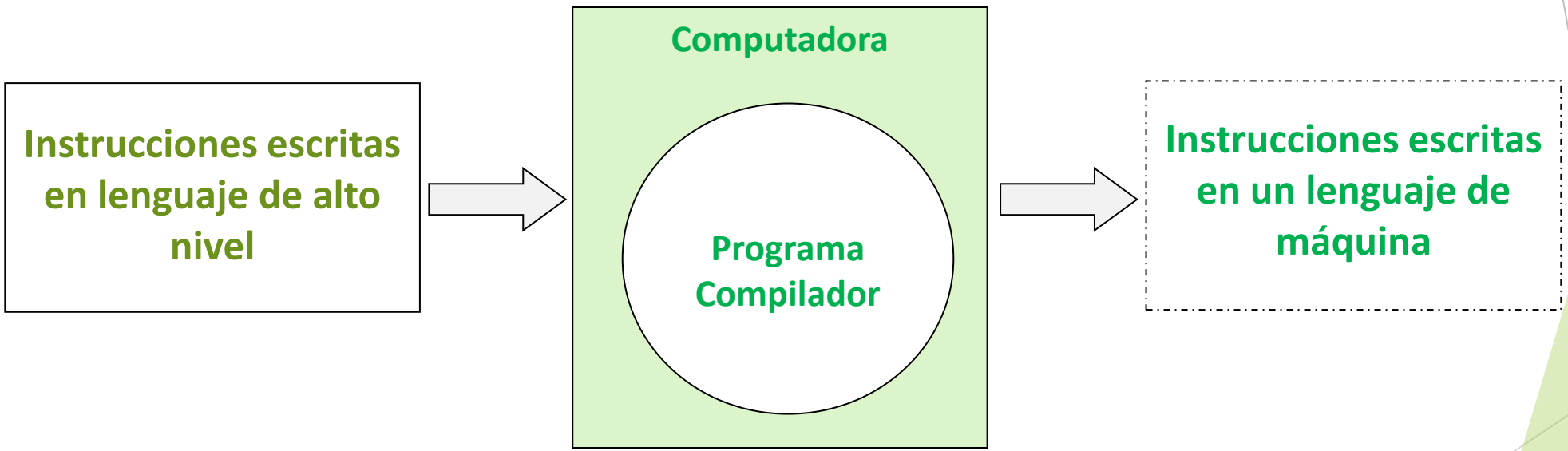
Ensamblador	**Lenguaje ensamblador**	→	**Lenguaje de máquina**
Compilador	**Lenguaje de alto nivel**	→	**Lenguaje de cercano al de máquina**
Cargador o editor de vínculos	**Lenguaje de máquina reubicable (crea un ejecutable con direcciones compatibles)**	→	**código de máquina**
Preprocesador o macroprocesador	**Forma amplia de lenguaje de alto nivel**	→	**Forma standard del lenguaje**

Esquema General de un Traductor

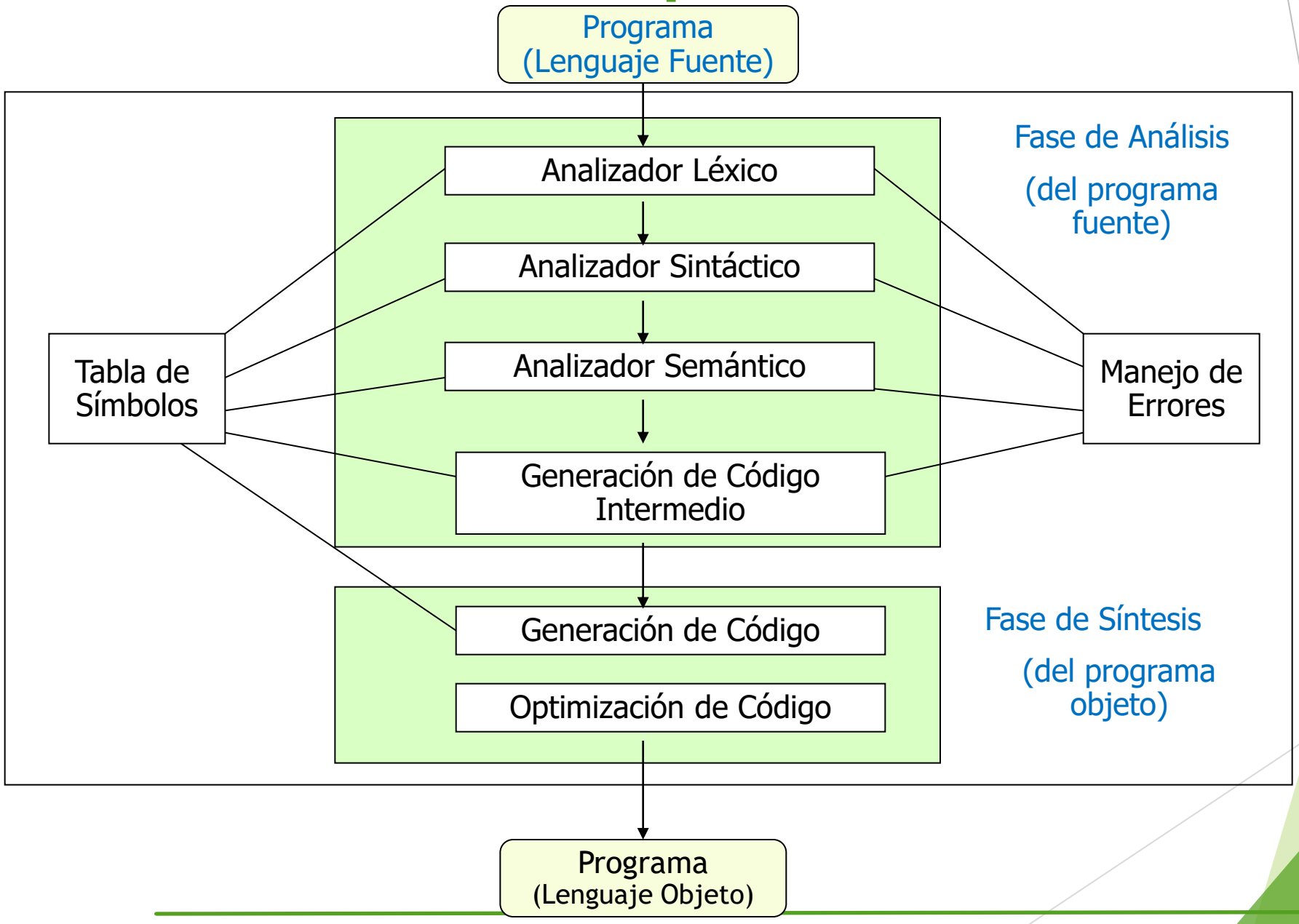


Compilación

Un **compilador es un traductor** que transforma un programa escrito en un lenguaje de alto nivel (lenguaje fuente) en otro programa escrito en un lenguaje de bajo nivel.



Esquema General de un Compilador



Implementación de un lenguaje de programación

- Cuando se implementa un lenguaje de programación, las estructuras de datos y algoritmos que se utilizan en tiempo de ejecución de un programa escrito en ese lenguaje, definen un **modelo de ejecución** definida por la implementación del lenguaje.
- Múltiples decisiones debe tomar el implementador en función de los recursos de hardware y software disponibles en la computadora subyacente y los costos de su uso.

Existen dos principios importantes a tener en cuenta: la *eficiencia* y la *regularidad*.

Eficiencia

Eficiencia de código: un diseño del lenguaje que permite que el traductor genere un **código ejecutable** eficiente. Ej. uso de declaraciones anticipadas de variables (estáticas).

Eficiencia de traducción: un diseño del lenguaje que permite que el código fuente se traduzca **con rapidez** y con un traductor de **tamaño** razonable. Ejemplo: C standard permite un compilador de una sola pasada pues las variables se declaran antes de usarse. C++, el compilador debe realizar una segunda pasada para resolver referencias del identificador.

Eficiencia de implementación: eficiencia con la que se puede escribir un traductor. Está relacionado con la eficiencia de la traducción y con la complejidad de la definición del lenguaje.

Eficiencia de la programación: facilidad para escribir programas en el lenguaje. Entre otras, involucra capacidad de expresión del lenguaje y **capacidad de mantenimiento al programa (localizar y corregir errores).**

Regularidad

La regularidad indica lo bien que se integran las características del lenguaje.

Es una cualidad que implica que hayan pocas restricciones en el uso de sus constructores particulares y en la forma en la que se comportan las características del lenguaje.

Generalidad

Ortogonalidad

Uniformidad

Regularidad

Generalidad: Un lenguaje alcanza generalidad evitando casos especiales en cuanto a disponibilidad o uso de constructores.

La no generalidad esta ligada a restricciones de los constructores dependiendo del contexto en que estén.

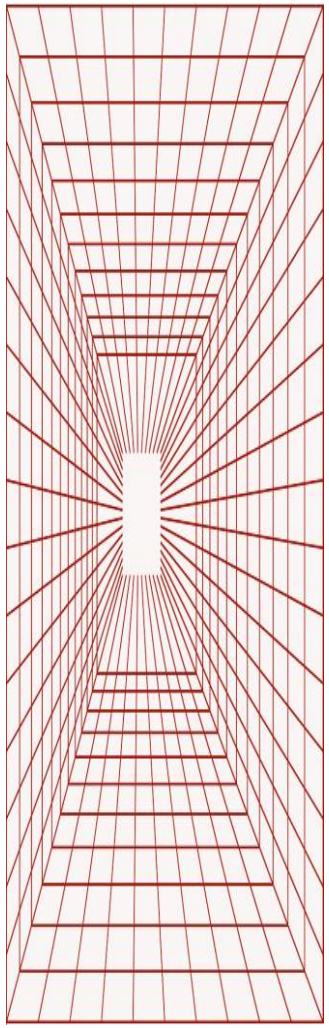
Ejemplos

- En lenguaje C el operador de igualdad == carece de generalidad.

Dos arreglos (o estructuras) no pueden compararse a través de este operador.

- En cuanto a la longitud de los arreglos, C tiene arreglos de longitud variable, mientras que esta generalidad no está presente en Pascal, que admite sólo arreglos de longitud fija.

Regularidad



Ortogonalidad: Se refiere al atributo de combinar varias características de un lenguaje de manera que la combinación tenga significado.

Ejemplos:

- Si un lenguaje provee de **expresiones que pueden devolver un valor** y también posee un **enunciado condicional que compara valores**; estas dos características son ortogonales si se puede usar dentro del enunciado condicional cualquier expresión.

```
if  ((x+y -2) < (x*y))
```

Regularidad

Uniformidad: Hace referencia a la consistencia de la apariencia y comportamiento de los constructores del lenguaje.

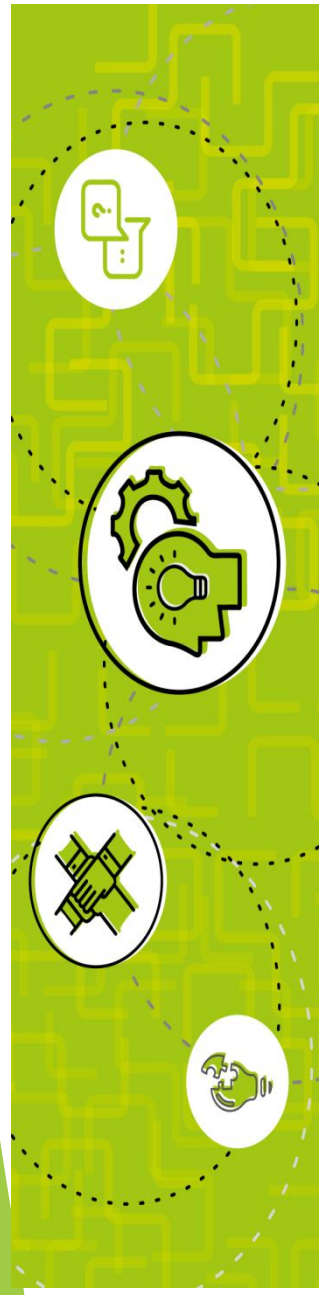
Las no-uniformidades son de dos tipos:

a) dos cosas similares que no parecen serlo, se comportan similarmente, o

b) cosas no similares, se comportan similarmente cuando no debieran.

Ejemplo

En C de la situación b): los operadores & (and bit a bit) y && (and lógico) tienen una apariencia confusamente similar y producen resultados muy distintos.



Paradigma de Programación

Un paradigma de programación "consiste en un método para llevar a cabo cómputos y la forma en la que deben estructurarse y organizarse las tareas que debe realizar un programa".

Clasificación de los Paradigmas

En general, la mayoría de paradigmas son variantes de los dos tipos principales de programación:

- Imperativa
- Declarativa

En la **programación imperativa** se describe paso a paso un conjunto de instrucciones que deben ejecutarse para variar el estado del programa y hallar la solución

En la **programación declarativa** se programa diciendo lo que se quiere resolver a nivel de usuario, pero no las instrucciones necesarias para solucionarlo. Esto último se realizará mediante mecanismos internos de inferencia de información a partir de la descripción realizada.

Programación declarativa

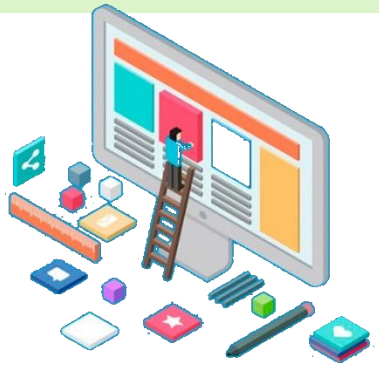
Está basada en describir el problema declarando propiedades y reglas que deben cumplirse, en lugar de instrucciones.. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla. No existen asignaciones destructivas, y las variables son utilizadas con transparencia referencial. Uso de mecanismos matemáticos para optimizar el rendimiento de los programas.
Ejemplos: Lisp y Prolog.

- **Programación funcional**
- **Programación lógica**
- **Programación con restricciones**
- **Programación multiparadigma**
- **Programación reactiva**
- **Lenguaje específico del dominio o DSL**

Programación imperativa o procedural

Es la más usada en general, se basa en dar instrucciones al ordenador de cómo hacer las cosas en forma de algoritmos, en lugar de describir el problema o la solución.
Ejemplos de lenguajes puros de este paradigma serían el C, BASIC o Pascal.

Programación orientada a objetos: encapsula elementos denominados objetos que incluyen tanto variables como funciones.
Ejemplo: C++, C#, Java, Python entre otros, Smalltalk que está completamente orientado a objetos.



Programación dinámica: proceso de dividir problemas en partes pequeñas para analizarlos y resolverlos de forma cercana al óptimo. Este paradigma está basado en el modo de realizar los algoritmos.

Programación orientada a eventos: es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen.
Ejemplo: Visual Basic.

Lenguajes Imperativos

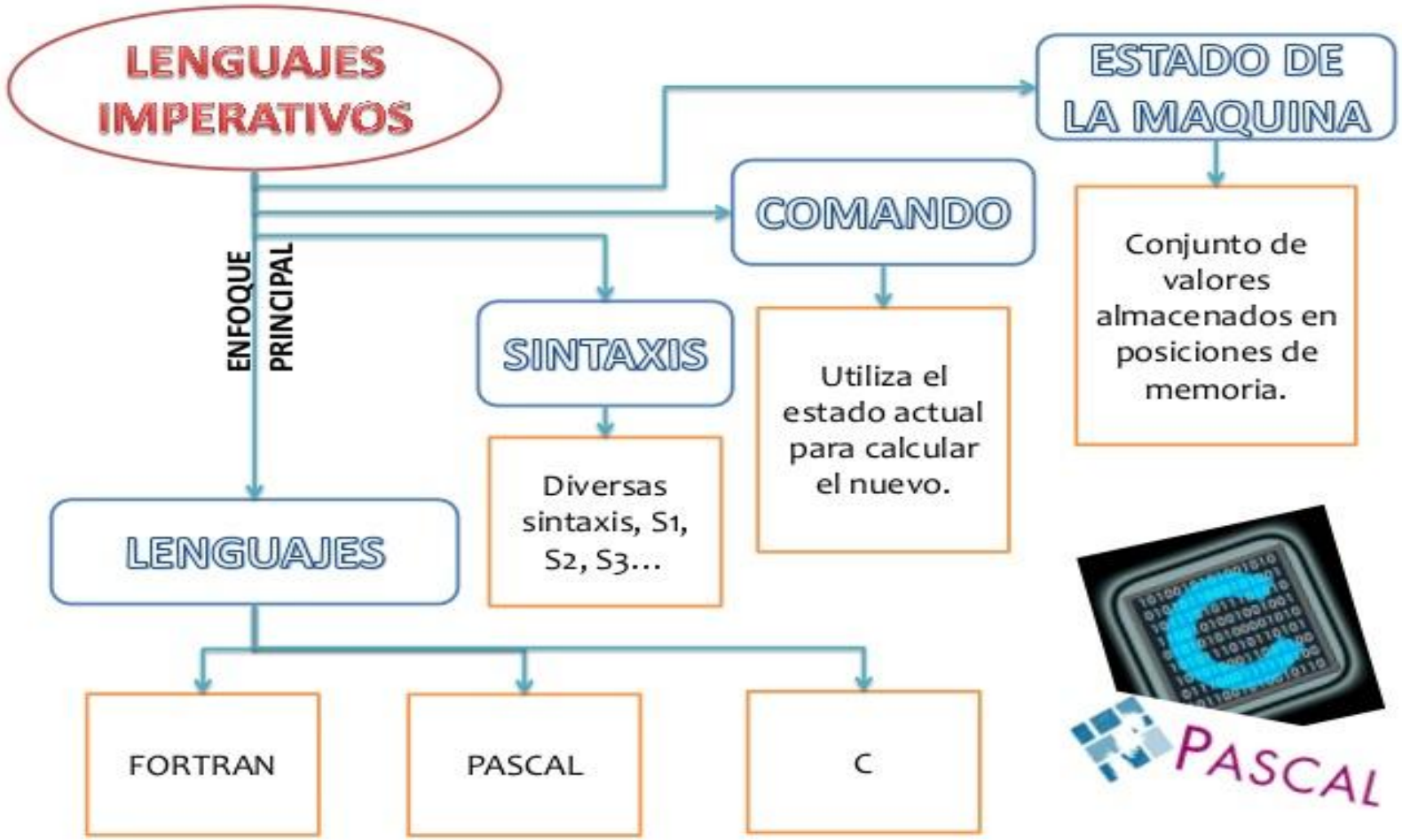
Un lenguaje procedural o imperativo es un lenguaje controlado por mandatos o instrucciones.

El **proceso de ejecución** de programas procedurales por la computadora, tiene lugar a través de una **serie de estados**, cada uno definido por el contenido de la memoria, los registros internos y los almacenes externos durante la ejecución.

- El almacenamiento inicial de estas áreas define el **estado inicial** de la computadora.
- Cada paso en la ejecución transforma este estado en otro nuevo, a través de la modificación de alguna de estas áreas, **transición de estado**.
- Cuando termina la ejecución del programa, **estado final** viene dado por el contenido final de las áreas antes mencionadas.

***El lenguaje C, responde al paradigma imperativo*.**

Lenguajes Imperativos



Lenguaje C

- Los primeros lenguajes de programación se denominaban lenguajes de bajo nivel o lenguajes de máquina (ceros y unos).
- Lenguajes de alto nivel, que distan mucho del lenguaje de máquina y se asemejan más al lenguaje que utilizan las personas para comunicarse (el inglés).
- El lenguaje C fue desarrollado en 1972 por Dennis Ritchie y Ken Thompson en los laboratorios Bell Telephone de AT&T.
- El lenguaje C comienza a desarrollarse como un lenguaje para escribir software y compiladores, específicamente para desarrollar el sistema operativo UNIX, pero su uso se ha generalizado y hoy en día muchos sistemas están escritos en C o en C++.
- El lenguaje C es un lenguaje de propósitos generales y, si bien es un lenguaje de alto nivel, provee sentencias de bajo nivel que permite facilidades similares a las que se encuentran en los lenguajes ensambladores.

Etapas en la obtención de programa ejecutable en lenguaje C

Antes de ejecutar un programa en lenguaje C, existen ciertas etapas que se deben realizar, ya que todas ellas contribuyen a la obtención del programa ejecutable.

Ellas son: edición, compilación, enlace, carga y ejecución.

Etapas para la obtención de un programa ejecutable en C