Introducción al Diseño de Compiladores

BIBLIOGRAFÍA

[AHO] Compilers. Principles, Techniques, and Tools
Aho, Sethi; Adisson-Wesley –1986

[TEU] Compiladores: Conceptos fundamentales.
Teufel ; Addison Wesley - 1995

[SAN] Compiladores. Teoría y construcción.
Sanchís Llorca y Galán Pascual. Paraninfo – 1988

[WIR] Algoritmos + Estructuras de Datos = Programas
Niklaus Wirth . Ediciones del Castillo –1980

[GHE] Conceptos de Lenguajes de Programación
Ghezzi, Jazayeri; Ed. Díaz de Santos -1982-1986

[LEV] Lex &Yacc. Levine; Mason; Brown; O'Reilly & Ass. 1995

CONTENIDOS

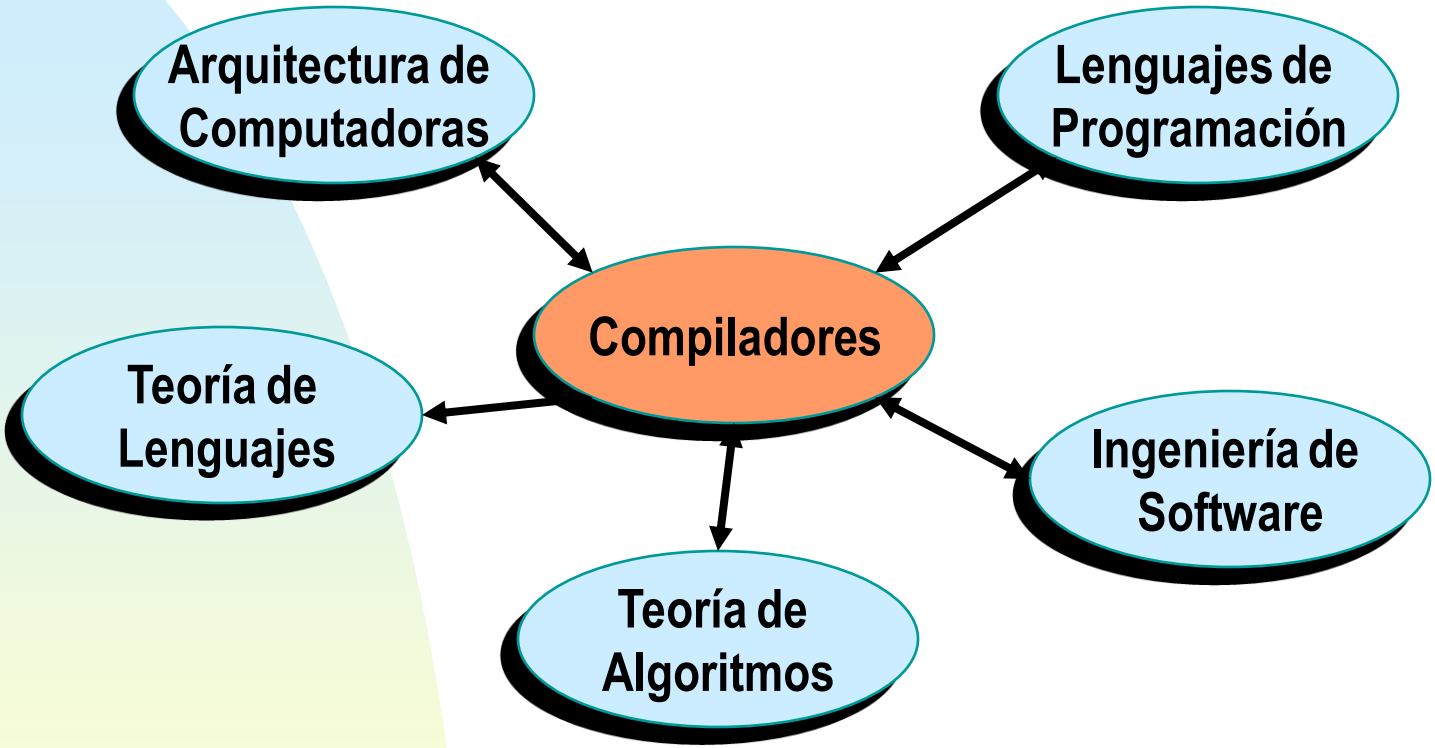
- Tema 1.- Introducción a la Compilación
- Tema 2.- Lenguajes, autómatas y gramáticas
- Tema 3.- Análisis léxico
- Tema 4: Tablas de Símbolos
- Tema 5.- Análisis sintáctico
- Tema 6.- Análisis semántico
- Tema 7.- Principios básicos de la fase de síntesis

PROGRAMA DE PRÁCTICOS

- **Práctica 1: Construcción de Autómatas**
- **Práctica 2: Análisis y transformación de gramáticas. Construcción de un analizador léxico**
- **Práctica 3: Diseño e implementación de un compilador**

INTRODUCCIÓN

Conceptos relacionados



Con algunas técnicas básicas de escritura de compiladores se pueden construir traductores para una gran variedad de lenguajes y máquinas

Compiladores

Un compilador es un programa que
lee un programa en un lenguaje y
lo traduce a un programa equivalente en otro lenguaje,
y además informa al usuario sobre
la presencia de errores en el programa de entrada

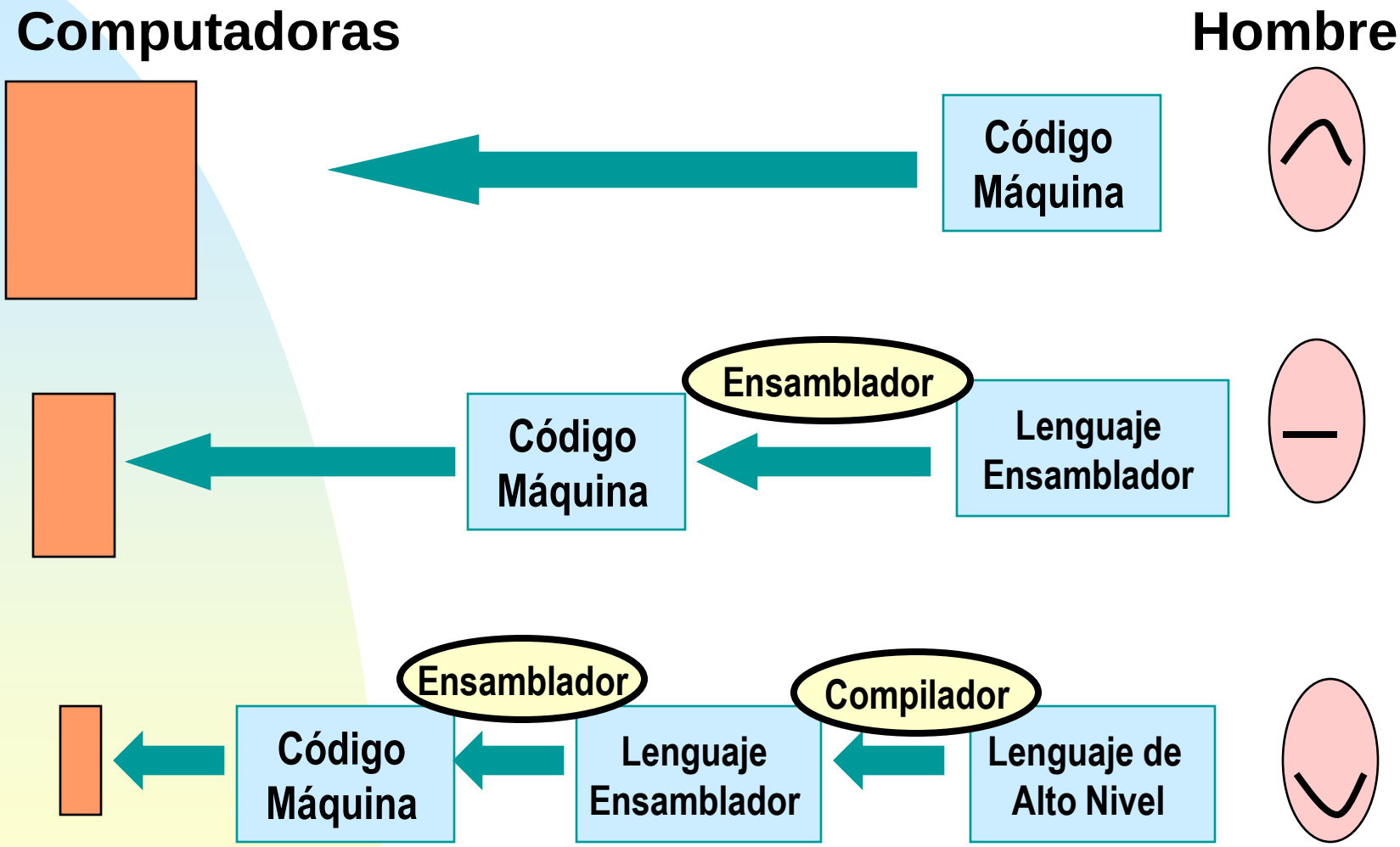
CLASIFICACION GENERAL

- De una pasada o de múltiples pasadas
- De carga y de ejecución
- De depuración o de optimización

HISTORIA

- Experimentación relacionada a traducción de fórmulas
- 1950: difícil escritura
- Primer FORTRAN: 18 años
- Hoy: técnicas sistemáticas, lenguajes de implementación, entornos de programación y herramientas de software

HISTORIA


Computadoras
Hombre
Código Máquina
Ensamblador
Código Máquina
Lenguaje Ensamblador
Ensamblador
Compilador
Código Máquina
Lenguaje Ensamblador
Lenguaje de Alto Nivel

HOY…. Y A FUTURO

- **El Diseño de un compilador surge como resultado de:**
  - **Desarrollo de un nuevo lenguaje de programación**
  - **Adición de extensiones a los ya existentes**
  - **Explotación de las características del hardware**
- **A futuro:**
  - **Extensión para el cómputo paralelo y distribuido**
  - **Explotación de características multimedia (MMX)**

TIPOS DE SISTEMAS DE COMPILACIÓN

- **ENSAMBLADOR**

  Traducen programas escritos en lenguaje ensamblador a código máquina

- **COMPILADOR**

  Traducen programas escritos en lenguaje de alto nivel a código intermedio o a código máquina

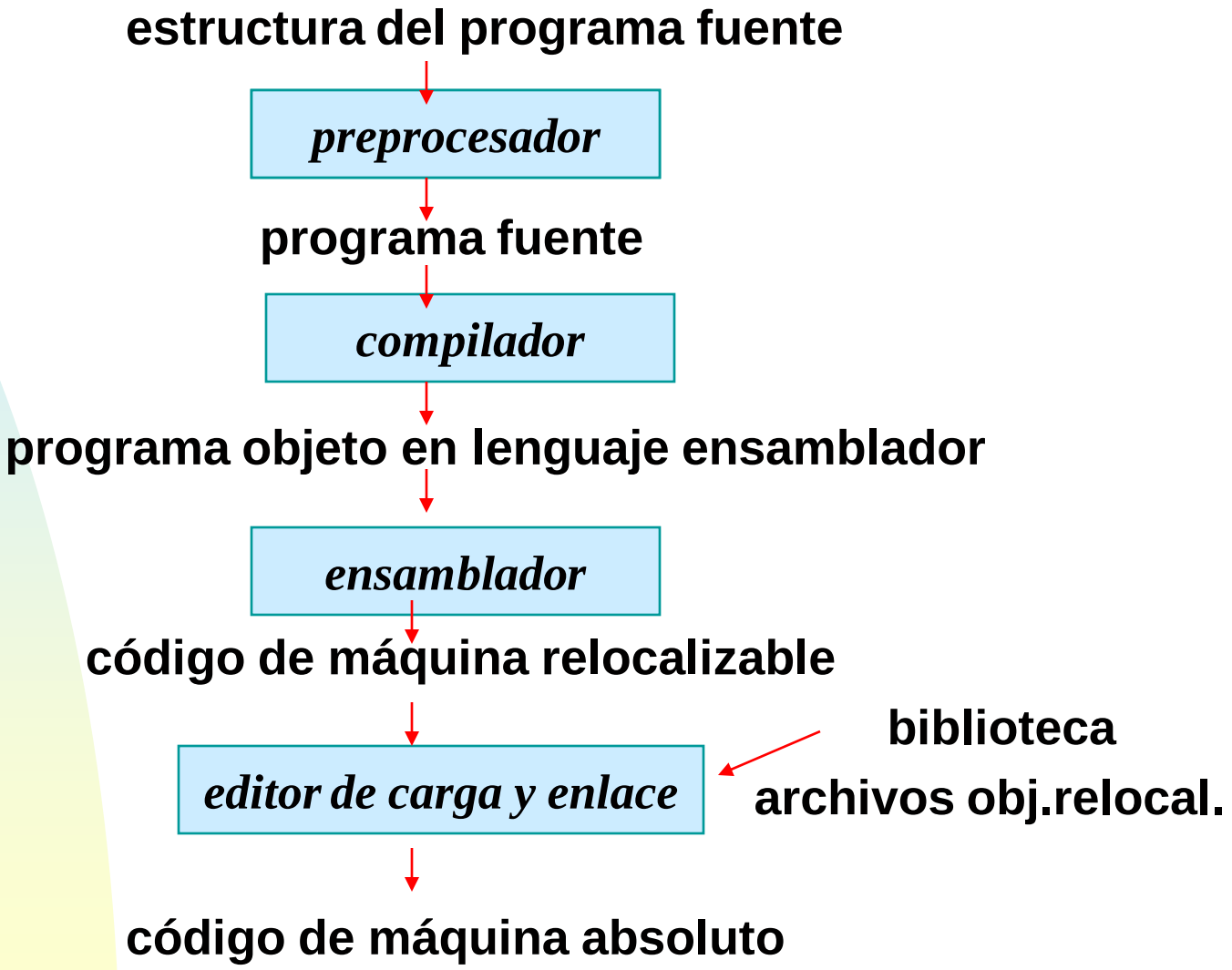
- **INTERPRETE**

  No genera código objeto, analiza y ejecuta directamente cada proposición del Programa Fuente (PF)

- **PREPROCESADOR**

  Sustituyen macros, incluyen archivos o extensión del lenguaje.

SISTEMA PARA PROCESAMIENTO DE UN LENGUAJE



PARTES DE LA COMPILACIÓN

- **ANÁLISIS (Etapa Inicial):**

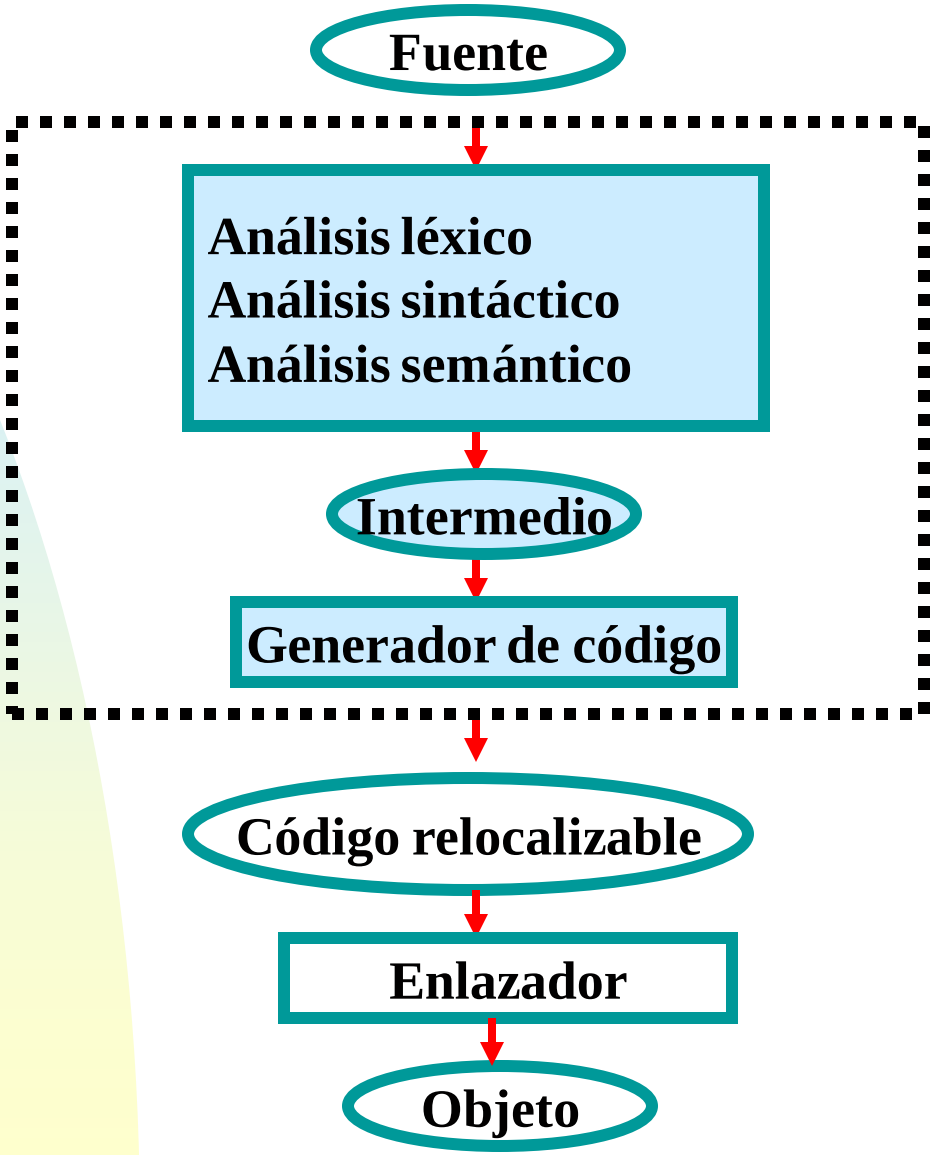
  Divide al PF en sus elementos componentes y crea una representación intermedia.

  Se determinan las operaciones y se registran en una estructura de árbol (ej. árbol sintáctico)

- **SÍNTESIS (Etapa Final):**

  Construye el PO deseado a partir de la representación Intermedia (requiere técnicas más especializadas)

UN AMBIENTE GENERAL DE COMPILACIÓN



ANÁLISIS DEL PROGRAMA FUENTE

- **ANALISIS LINEAL (Léxico- Exploración- Scanner)**

  Se lee el programa como una cadena de izquierda a derecha, se agrupan y se generan *componentes léxicos* o *tokens* (secuencia de caracteres con significado colectivo)

- **ANALISIS JERARQUICO (Sintáctico- Parser)**

  Los componentes léxicos se agrupan en colecciones anidadas con un significado colectivo (frases gramaticales que por lo general se representan mediante *árboles sintácticos*)

- **ANALISIS SEMANTICO**

  Se realizan revisiones para asegurar que los componentes de un programa se ajustan de un modo significativo

EJEMPLO DE ANÁLISIS:

posicion := inicial + velocidad * 60

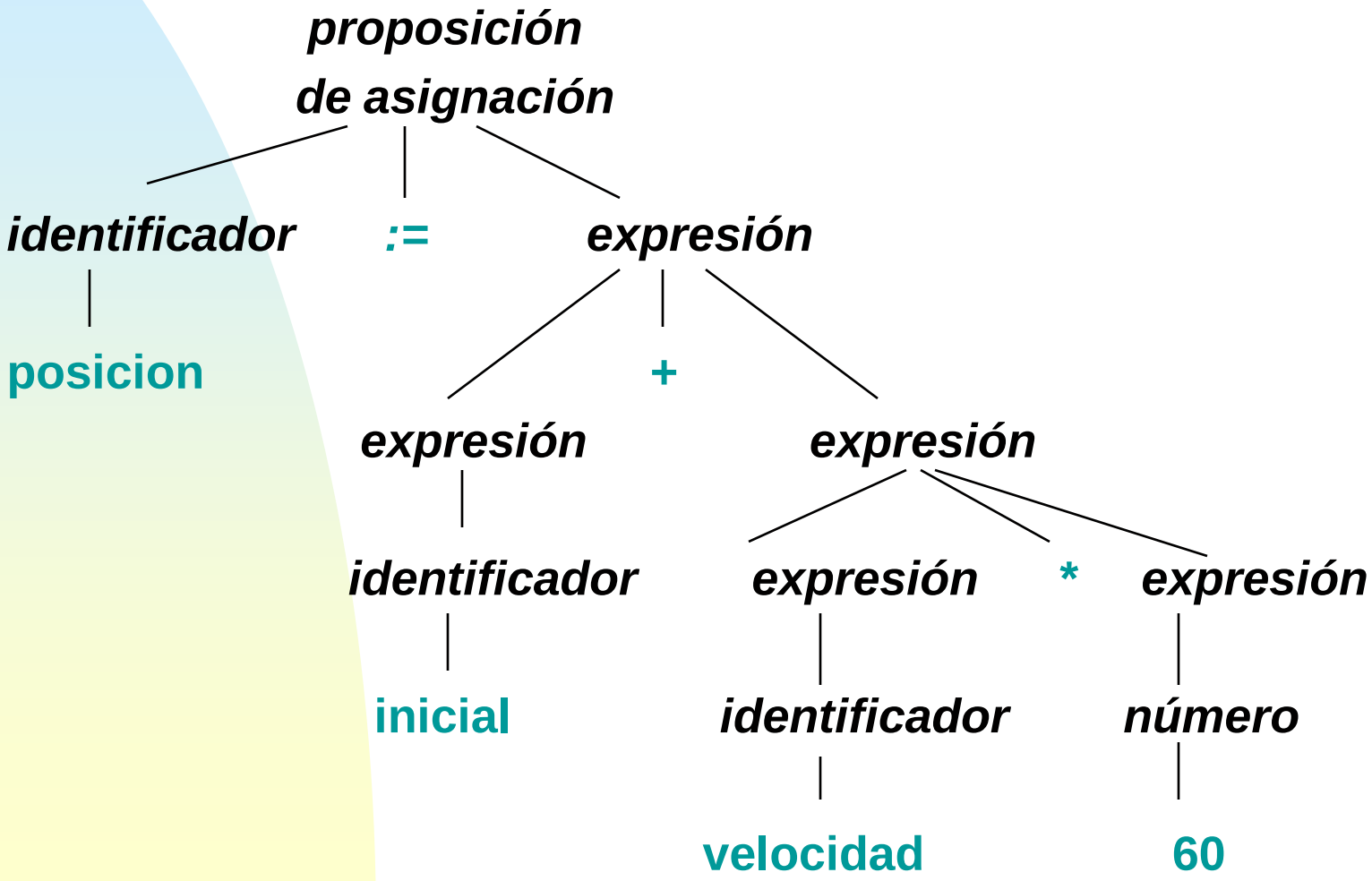
a) Componentes léxicos:

1. El identificador ***posicion***
2. El símbolo de asignación ***:=***
3. El identificador ***inicial***
4. El signo de suma: **+**
5. El identificador ***velocidad***
6. El signo de multiplicación: ***
7. El número ***60***

Los identificadores o nombres reconocidos se organizan en una *tabla de símbolos* que se usará en los pasos siguientes

posicion := inicial + velocidad * 60

b) Análisis sintáctico (árbol de analis. sint.)



posicion := inicial + velocidad * 60

b) Análisis sintáctico (reglas recursivas)

- **Las construcciones léxicas no requieren recursión (ej. Reconocer un identificador) mientras que las sintácticas suelen requerirlas (ej. Emparejamiento de paréntesis o Begin-End)**

- **La estructura jerárquica de un programa normalmente se expresa mediante *reglas recursivas***

  Exp :: ident | nro

  Exp :: Exp + Exp| Exp * Exp | (Exp)

- **Las gramáticas libres de contexto (GLC) son una formalización de reglas recursivas que pueden guiar el análisis sintáctico**

posicion := inicial + velocidad * 60

b) Análisis semántico

- Significado de una unidad gramatical, interpretación
- Traducir la entrada a una forma de representación intermedia
- Análisis y verificación de tipos
- Utiliza la estructura jerárquica del Análisis sintáctico
- El árbol sintáctico permite una representación interna compacta del árbol de análisis sintáctico. Ejemplos: