

Sesión 1

Análisis Léxico

Expresiones Regulares

Antonio Moratilla Ocaña

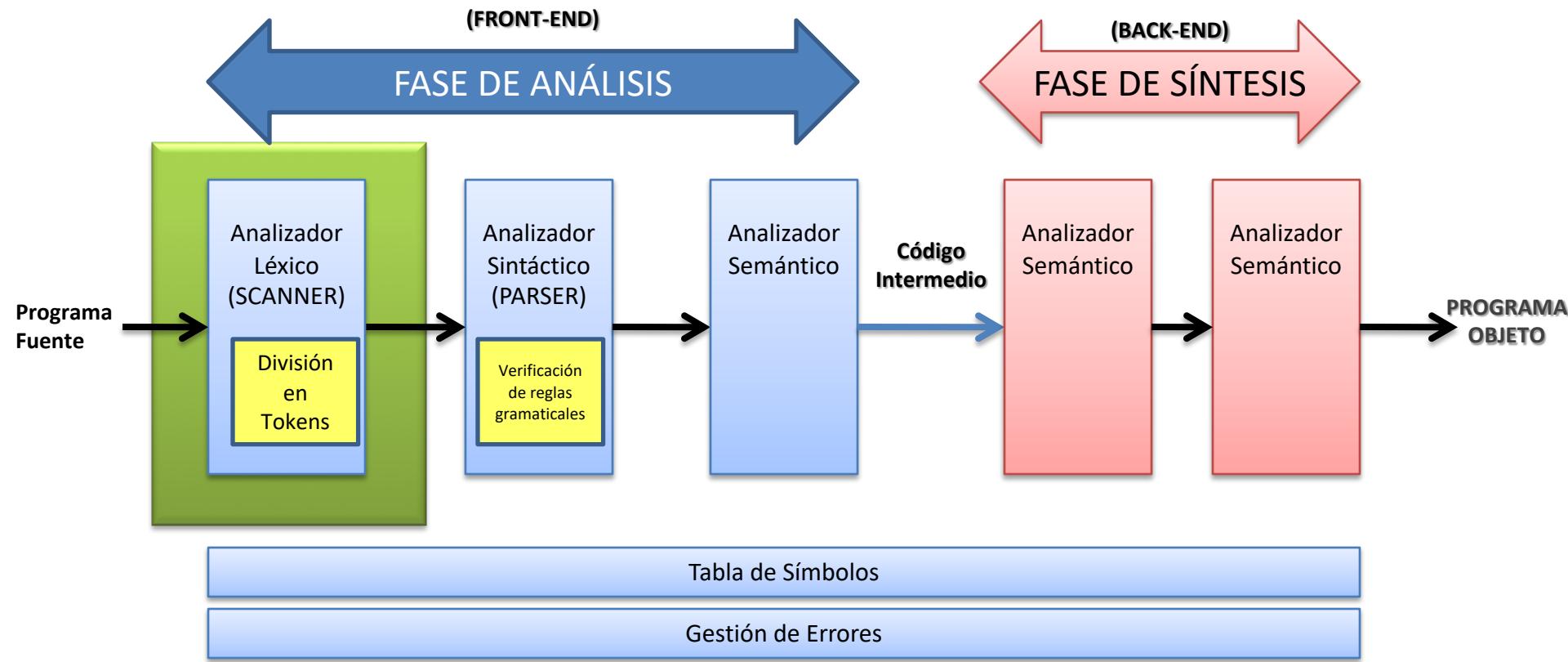


Resumen del tema

- Objetivo:
 - Introducción a la estructura y funcionamiento del Analizador Léxico
 - Introducción a las Expresiones Regulares



Posición en el diagrama



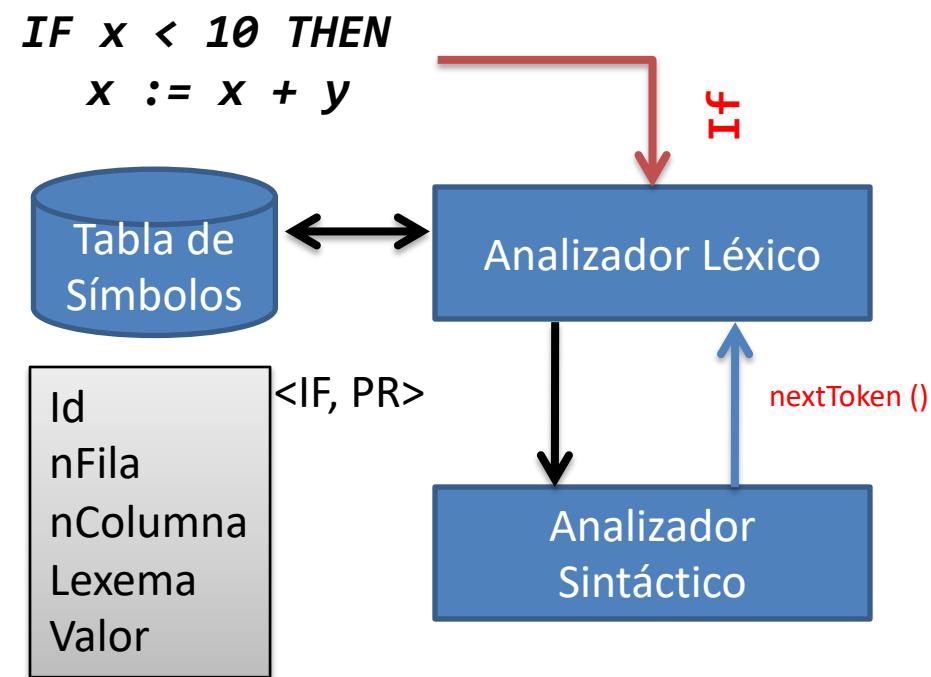
¿Qué es un Analizador Léxico?

Un ANALIZADOR LÉXICO o SCANNER es un programa capaz de descomponer una entrada de caracteres (generalmente contenidas en un fichero) en una secuencia ordenada de *tokens*.

El analizador léxico responde bajo demanda a las solicitudes de *siguiente token* que le va haciendo el analizado sintáctico.

Cada vez que éste último hace una solicitud al primero, el analizador léxico consume un número de caracteres de la entrada (de uno en uno) y retorna un *artefacto computacional* que representa el siguiente token en la entrada.

En lo venidero utilizaremos el término token para referirnos a dicho artefacto



Conceptos básicos

- **Token** (o componente léxico): Secuencia de caracteres con significado sintáctico propio.
- **Lexema**: Secuencia de caracteres cuya estructura se corresponde con el patrón de un token.
- **Patrón**: Regla que describe los lexemas correspondientes a un token.

Componente Léxico	Ejemplo de Lexema	Descripción del patrón	observaciones
Identificador	x, y, valor, x2	<letra>(<letra> <dígito>)*	Identificadores
Cadena	“cadena”	Caracteres entre “y “	Constantes cadena



Tareas del Analizador Léxico

1. Reconocer los componentes léxicos en la entrada.
 2. Eliminar los espacios en blanco, los caracteres de tabulación, los saltos de línea y de página y otros caracteres propios del dispositivo de entrada, que no tengan relevancia en el lenguaje.
 3. Eliminar los comentarios de entrada.
 4. Detectar errores léxicos.
-
- En lenguajes de programación, además:
 - Reconocer los identificadores de variable, tipo, constantes, etc. y guardarlos en la tabla de símbolos.
 - Relacionar los mensajes de error del compilador con el lugar en el que aparecen en el programa fuente (línea, columna, etc.).



Necesidad de un Analizador Léxico

- Al comparar las expresiones

a $\emptyset := \emptyset b \emptyset * \emptyset 7 \emptyset + \emptyset 4 \emptyset;$
a $\emptyset \emptyset \emptyset := \emptyset \emptyset b * 7 + 4 \emptyset \emptyset \emptyset;$

- La estructura de las dos expresiones es equivalente,
- La posición de los caracteres que las componen, aunque siguen el mismo orden, son diferentes.

Si las distintas fases del compilador tuvieran que trabajar con los caracteres directamente sería más complicado descubrir la estructura de un programa.



Componentes Léxicos

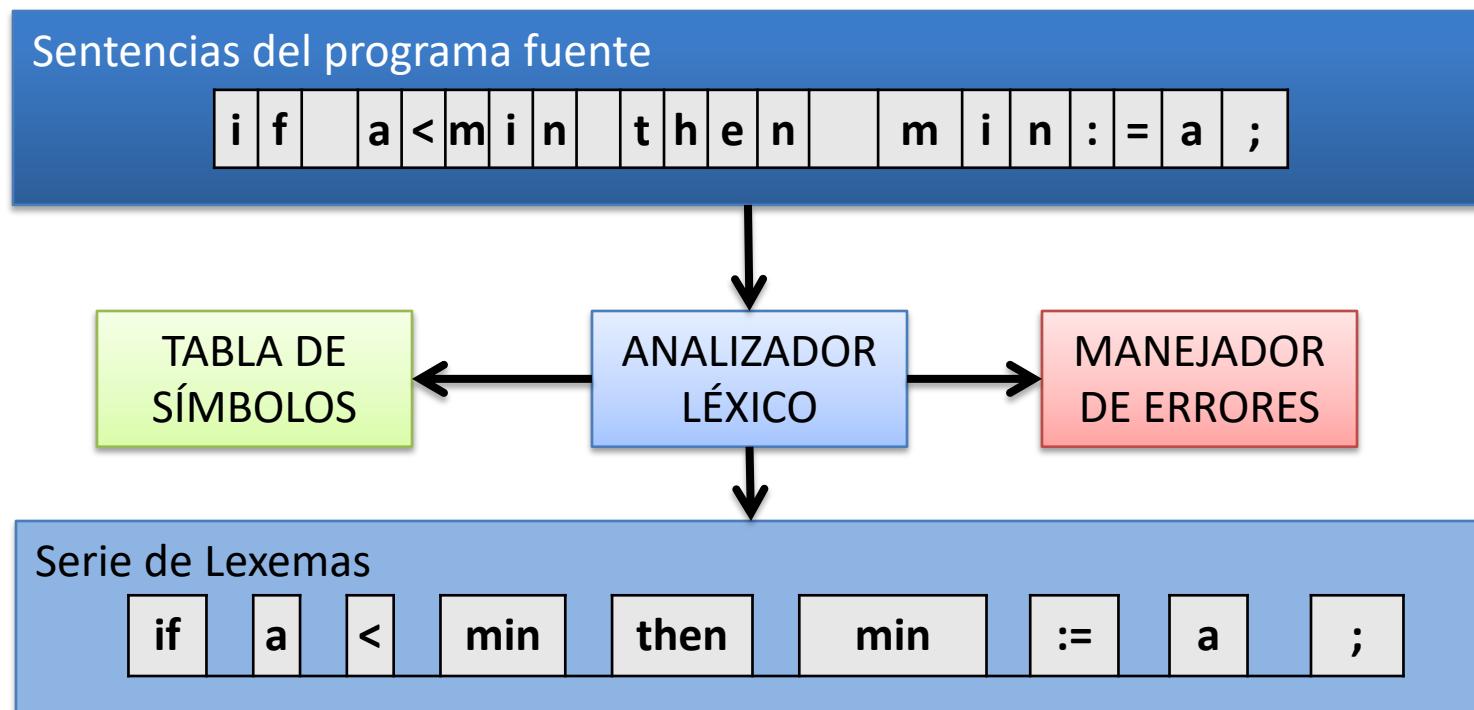
- En la mayoría de los lenguajes de programación, se consideran componentes léxicos (tokens):
 - Palabras reservadas
 - Operadores (de comparación, asignación, lógicos, aritméticos ...)
 - Identificadores
 - Constantes
 - Signos de puntuación (paréntesis, punto y coma ...)
 - Marcas de comienzo y fin de bloque
- Los delimitadores no serán considerados, en general, tokens.
- Cuando un patrón puede concordar con más de un lexema es necesario conocer información adicional:
 - Esta información se almacena como atributos del token.



Descripción de un Analizador Léxico

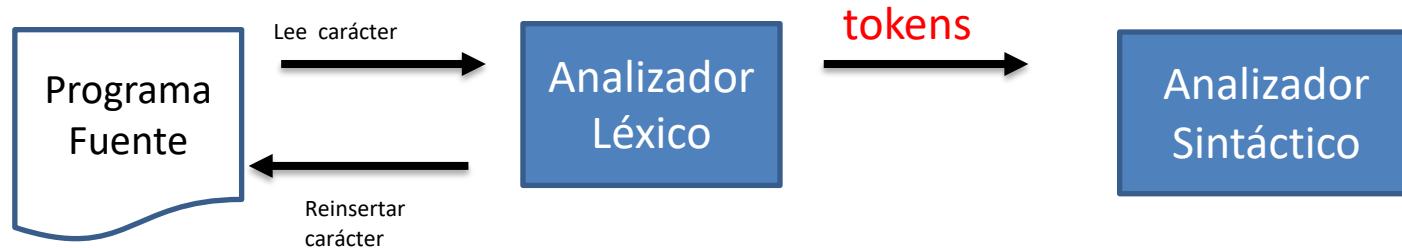
El Análisis Léxico realiza un análisis de los caracteres

- Parte de éstos y por medio de patrones reconoce los lexemas
- Envía al analizador sintáctico el componente léxico y sus atributos
- Puede hacer tareas adicionales: eliminar blancos, control líneas ...



El Analizador Léxico como interfaz

- El Analizador Léxico y el Analizador Sintáctico forman un par productor-consumidor.
 - En algunas situaciones, el analizador léxico tiene que leer algunos caracteres por adelantado para decidir de qué token se trata.



Ejemplo

Para la sentencia :

IF x < 10 THEN x := x + y

El analizador lexicográfico daría como resultado inicialmente la siguiente cadena de caracteres:

<79> <12> <28> <13> <80> <12> <65> <12> <34> <12>

O lo que es lo mismo:

If identificador op_menor literal_entero then
identificador asignación identificador op_suma
identificador.

PERO: aquí falta algo... Lo que se está leyendo!!!



Ejemplo

El resultado final sería:

```
<79,-> <12,32>    <28,->    <13,10>    <80,-> <12,32> <65,->
<12,32> <34,-> <12,33>
```

Es decir:

```
<if,-> <identificador,32> <op_menor,-> <literal_entero,10> <then,->
<identificador,32> <asignación,-> <identificador,32> <op_suma,->
<identificador,33>
```

¿De dónde sale el '32' de <identificador,32>? : conexión con tablas de símbolos y analizador sintáctico, estamos avanzando mucho...



Atributos

- Los identificadores tienen asociados como atributos el lugar de la tabla de símbolos donde se han guardado:
 $< \text{identificador}, 32 >$
 - A veces el analizador sintáctico es el único que maneja la tabla de símbolos, en ese caso llevan asociado el propio lexema:
 $< \text{identificador}, x >$
- Los literales tienen asociados como atributos el propio lexema:
 $< \text{literal_entero}, 10 >$



Errores léxicos

- El analizador léxico rechaza texto con caracteres ilegales (no recogidos en el alfabeto) o combinaciones ilegales. Ejemplos:
 - “ñ”, “é”(caracteres que no pertenecen al alfabeto del lenguaje)
 - “:=”, “:::(no coinciden con ningún patrón de los tokens posibles)
- Se debe mostrar un mensaje de error claro y exacto.
 - En vez de...
 - Error 124
 - Falta declaración
 - Error en la línea 85
 - Se ha producido un error
 - Sería mejor...

```
int número = 1 ;
```

^ERROR 124: línea 85, columna 6, carácter no válido



Recuperación de Errores léxicos

- Posibles acciones del analizador léxico para detectar errores, recuperarse y seguir trabajando:
 - Modo de pánico: Ignorar los caracteres no válidos hasta un carácter en el cual se considera que podría empezar un lexema correcto.
 - Distancia mínima de corrección de un error, i.e., recuperación “inteligente” con correcciones como:
 - Borrar los caracteres extraños.
 - Insertar un carácter que pudiera faltar.
 - Reemplazar un carácter incorrecto por uno correcto.
 - Comutar las posiciones de dos caracteres adyacentes.



Organización

- Muy bonito todo, pero...
 - ¿Cómo se organiza la operación?
 - ¿En qué nos basamos?
 - ¿Hay que programar todo de golpe cada vez?
 - Vayamos por partes: Empecemos con qué es una palabra, de qué está compuesta y cómo sabemos si es correcta o no.



Conceptos básicos

- **Alfabeto :** Un alfabeto es un conjunto finito no vacío de símbolos.
 - $\Sigma_1 = \{0, 1\}$
 - $\Sigma_2 = \{a, b\}$
 - $\Sigma_3 = \{na, pa, bra, la\}$
 - $\Sigma_4 = \{\langle\text{HTML}\rangle, \langle/\text{HTML}\rangle, \langle\text{BODY}\rangle, \langle/\text{BODY}\rangle, \dots\}$
- Usamos meta-símbolos (tal como {}, =, y la coma) para escribir sobre lo que hablamos. Desde el contexto siempre deberá estar claro, si se trata de un símbolo del alfabeto o si se trata de un meta-símbolo.
- Usamos subíndices para distinguir diferentes alfabetos.
- Usamos normalmente las minúsculas como símbolos del alfabeto $\Sigma = \{a, \dots, z\}$, en los ejemplos normalmente letras desde el principio del alfabeto.
- Cardinalidad del alfabeto (número de elementos del alfabeto):
$$|\Sigma| > 0, |\Sigma| < \infty$$



Conceptos básicos

- **Palabras o Cadena:** Una secuencia finita de símbolos de un alfabeto es una palabra sobre dicho alfabeto.
 - $\Sigma_1 = 0, 1, 00, 01, 11, 000, 1001101$
 - $\Sigma_2 = a, aa, abb, ababa$
 - $\Sigma_3 = napa, palabra$
 - $\Sigma_4 = <\text{HTML}> <\text{BODY}> </\text{BODY}> </\text{HTML}>$
- Escribimos la palabra vacía, es decir, la palabra que no contiene ningún símbolo, como ϵ .
- Usamos normalmente letras minúsculas para anotar palabras, preferiblemente desde el final del alfabeto.
- El símbolo ϵ no pertenece a ningún alfabeto.
- La longitud de una palabra sobre un alfabeto es el número de símbolos que contiene.
- Si se puede dividir todas las palabras sobre un alfabeto solamente de una manera en sus símbolos, se llama **alfabeto libre**.
- El conjunto de todas las palabras que se pueden formar sobre un alfabeto más la palabra vacía se llama el universo del alfabeto $W(\Sigma)$.
- Si el alfabeto es libre (un generador libre), escribimos Σ^* por $W(\Sigma)$.



Conceptos básicos

• Palabras (II)

Podemos concatenar palabras, sean w, v y u palabras en Σ^* .

- $w.v = wv$, es decir, usamos el $.$ como símbolo de concatenación, pero muchas veces obviamos de él (igual como se suele hacer con el \cdot de la multiplicación).
- " $\varepsilon w = w = w\varepsilon$ ", es decir, ε se comporta como el elemento neutro (o elemento de identidad) respecto a la concatenación.
- $w.v \neq v.w$ para cualquier w y v
si $w = abc$ y $v = dec \Rightarrow wv = abcdec \neq decabc = vw$
- $(w.v).u = w.(v.u)$ para cualquier palabras w, v y u
- Si concatenamos siempre la misma palabra w , obtenemos potencias de w .

$$w^0 = \varepsilon \quad w^1 = w \quad w^2 = ww, \quad w^3 = www \quad \dots$$



Conceptos básicos

- Lenguajes:
 - A vista de pájaro: Un lenguaje está formado por un conjunto de palabras de un alfabeto: cuidado, no todas las palabras, sólo algunas.
 - Esto, para la siguiente sesión... Vamos paso a paso...



Calcular si una palabra es correcta

- El primer objetivo, muy modesto pero importante, es **saber si una entrada de un fichero de datos** (ya sean estos datos el código fuente de un programa o cualquier otra cosa) **es una palabra válida o no.**
- Iniciaremos el viaje con las **Expresiones Regulares**, la más sencilla de las herramientas para esta labor.



Especificación de Analizadores Léxicos

Expresiones Regulares Una expresión regular utiliza los términos del alfabeto de terminales operados a través de operaciones con una semántica específica.

Una expresión regular sobre un conjunto T es un conjunto $\text{ExpReg} = (T, |, \cdot, *)$ que cumple las siguientes propiedades:

- \emptyset es una expresión regular que define el lenguaje $L = \emptyset$
- ϵ (cadena vacía) es una expresión regular que define el lenguaje

$$L(\epsilon) = \{\epsilon\}$$

- Cualquier símbolo $a \in T$ es una expresión regular que define el lenguaje
$$L(a) = \{a\}$$
- Si x, y son dos expresiones regulares, $x \cdot y$ es una expresión regular que define
$$L(x \cdot y) = L(x) L(y)$$
- Si x, y son dos expresiones regulares, $x | y$ es una expresión regular que define
$$L(x | y) = L(x) \cup L(y)$$

- Si x es una expresión regular x^* es una expresión regular que define

$$L(x^*) = \bigcup_{i=0}^{\infty} L(x)^i$$

- Si x es una expresión regular x^+ es una expresión regular que define el lenguaje
$$L(x^+) = \bigcup_{i=1}^{\infty} L(x)^i$$

- No existen más expresiones regulares



Especificación de Analizadores Léxicos

- **Expresiones regulares**

- Si r y s son cadenas que pertenecen respectivamente a los lenguajes $L(r)$ y $L(s)$, son expresiones regulares:

$$\rightarrow r|s \equiv L(r) \cup L(s)$$

$$\rightarrow rs \equiv L(r)L(s)$$

$$\rightarrow r^* \equiv (L(r))^*$$

- Abreviaturas:

$$\rightarrow 0 \text{ ó más} \rightarrow a^*$$

$$\rightarrow 1 \text{ ó más} \rightarrow aa^* \equiv a^+$$

$$\rightarrow 0 \text{ ó 1} \rightarrow \epsilon | a \equiv a?$$

$$\rightarrow \text{clases de caracteres} \rightarrow a | b | c \equiv [abc] . \text{ También: } a | b | c | \dots | z \equiv [a-z]$$

$$\rightarrow \text{Los paréntesis agrupan subexpresiones: } (a | b | c)^*$$



Especificación de Analizadores Léxicos

- **Expresiones regulares . Ejemplos.**

→ $0(0|1)0^*$: 010, 000, 010000, 01, etc.

→ $a(ab)^+b^*$: aab, aababb, aabbbbb, aababababbb, etc.

→ [1-9]?0 : 0, 10, 20, 30, . . . , 90

→ [a-zA-Z]

→ [0-9]+

→ [a-zA-Z]([a-zA-Z] | [0-9])*

→ [0-9]+(. [0-9]+)?



Ejercicios

- Genere las expresiones regulares que permitan saber si una cadena cumple que:
 1. Pertenece a una de las siguientes palabras reservadas: if, else, then, for, while.
 2. Es un número entero par.
 3. Es un número decimal.
 4. Son cadenas de texto en un lenguaje de programación
 5. Son secuencias de uno o mas dígitos que no contienen dos dígitos pares consecutivos.
 6. Es un teléfono español
 7. Define un array como {1.1,3.33,2,9} donde no se conoce el número de elementos del array, que puede variar desde cero hasta infinito.



Fuentes

- Para la elaboración de estas transparencias se han utilizado:
 - Transparencias de cursos previos (elaboradas por los profesores Dr. D. Salvador Sánchez, Dr. D. José Luis Cuadrado).
 - Libros de referencia.

