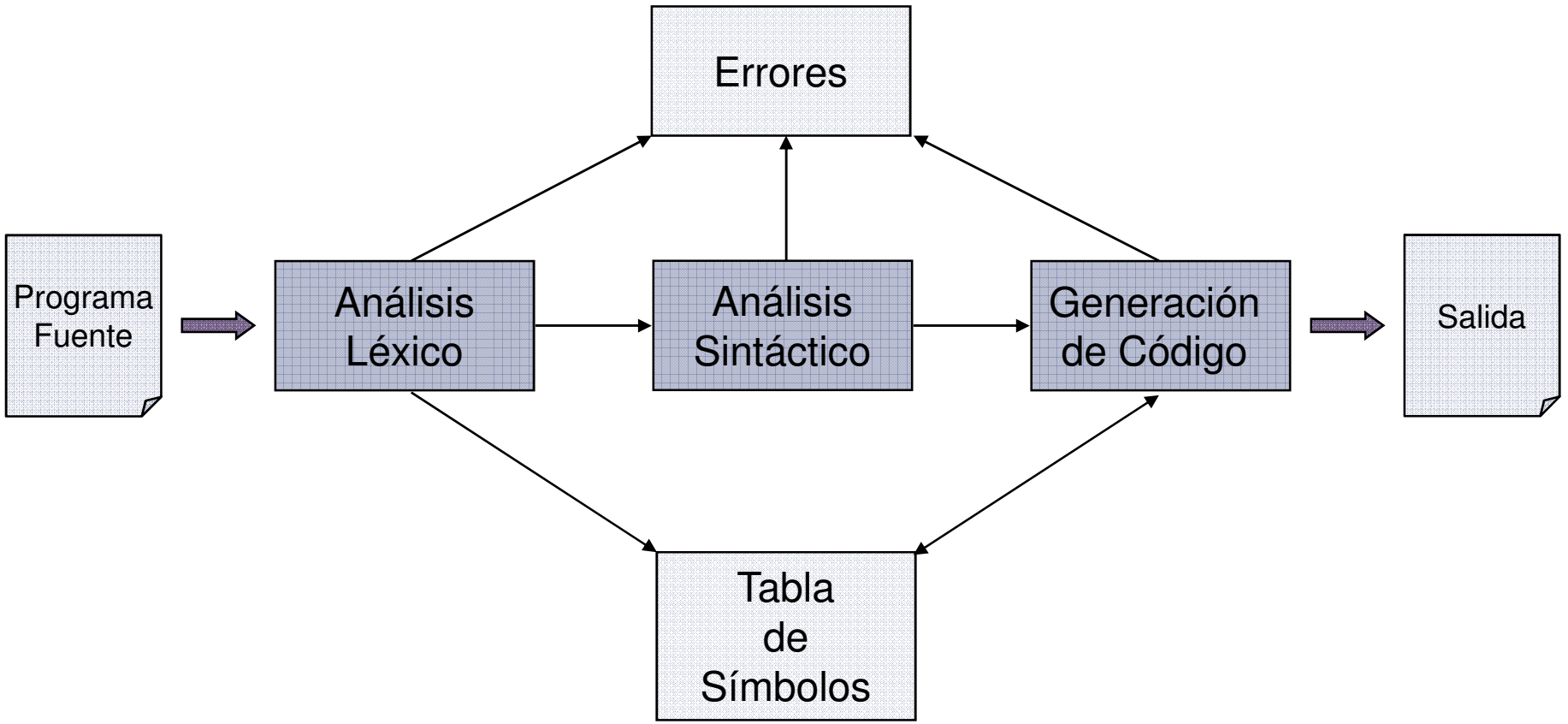
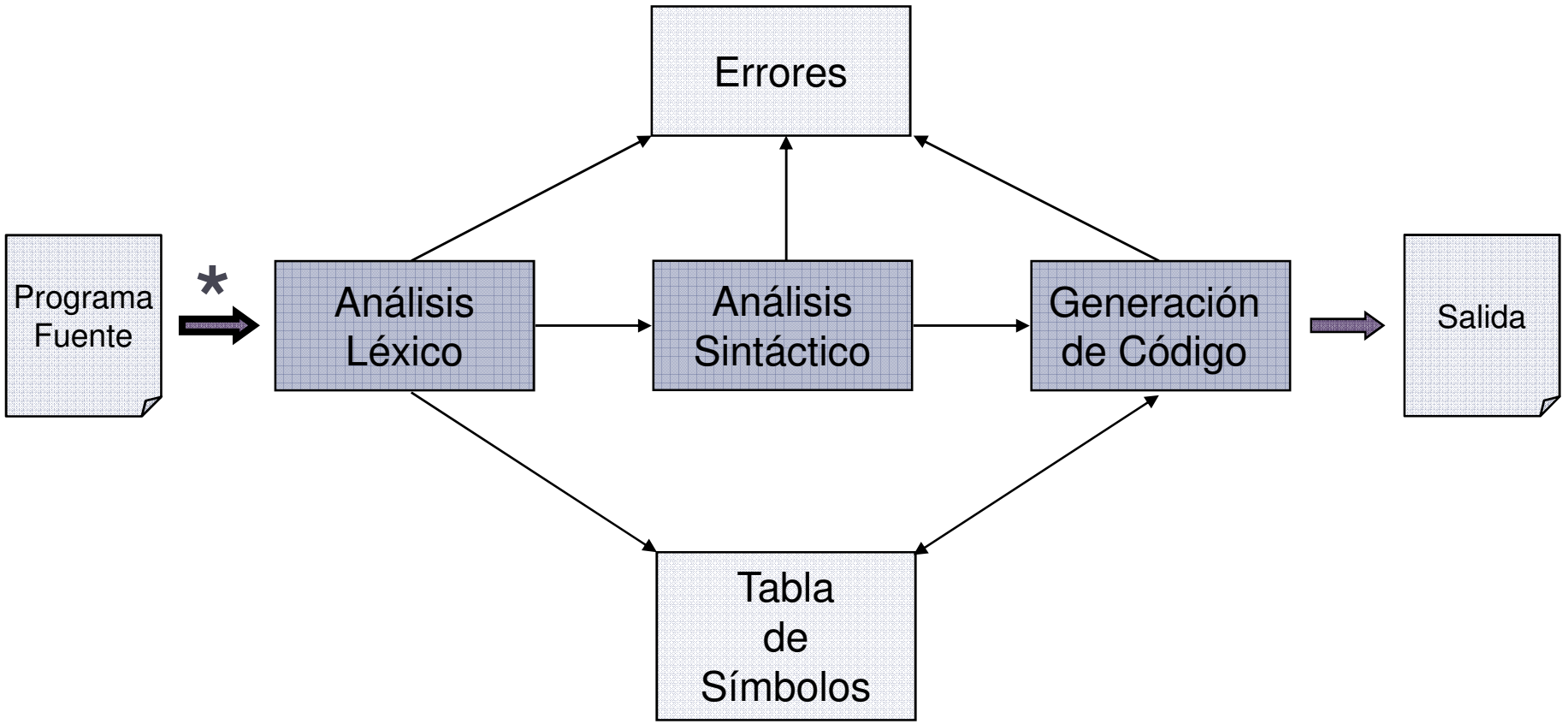
Diseño de Compiladores I

Análisis Léxico

Fases de la Compilación



Fases de la Compilación

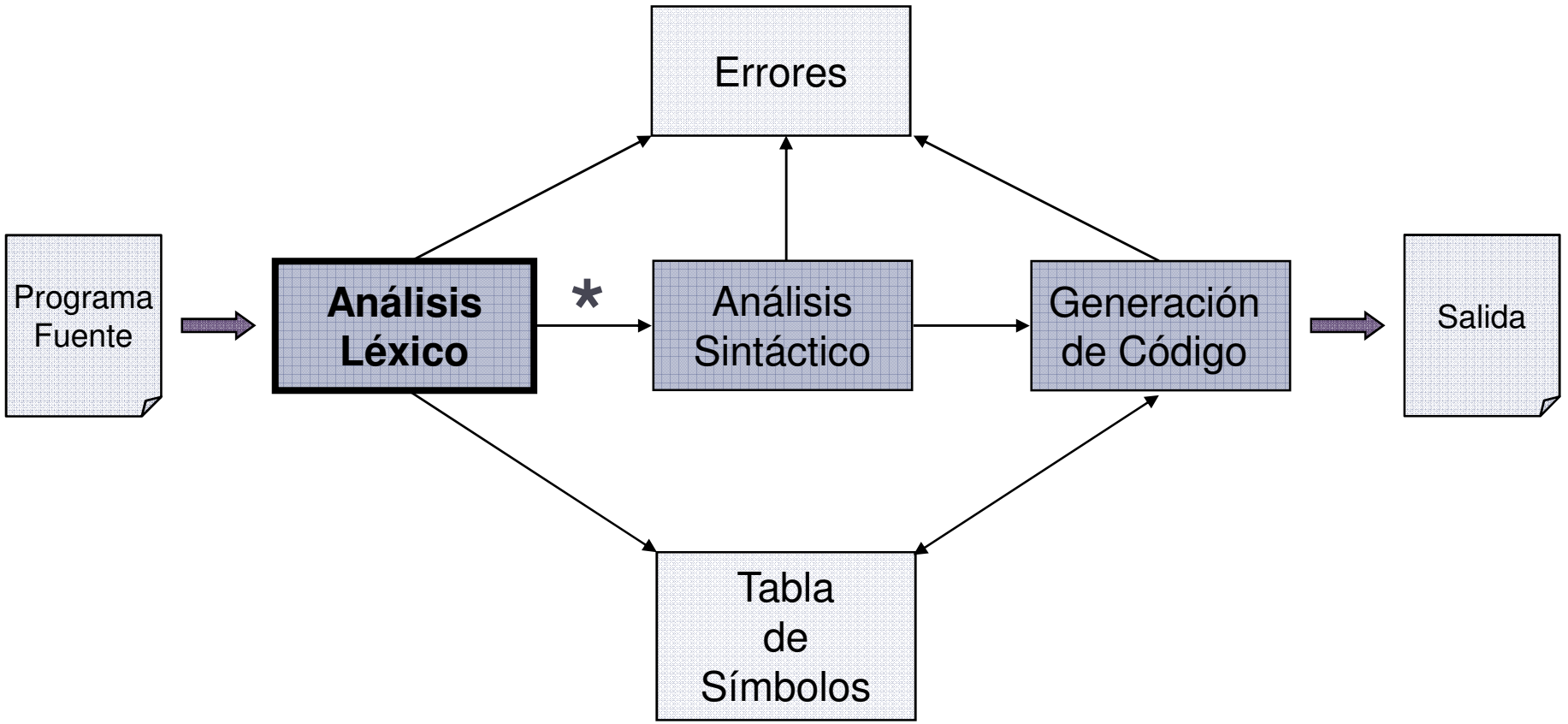


Fases de la Compilación

Puede haber preprocesadores para:

- Eliminar comentarios
- Incluir archivos
- Expandir macros
- Efectuar compilación condicional
- Reemplazar constantes simbólicas

Fases de la Compilación

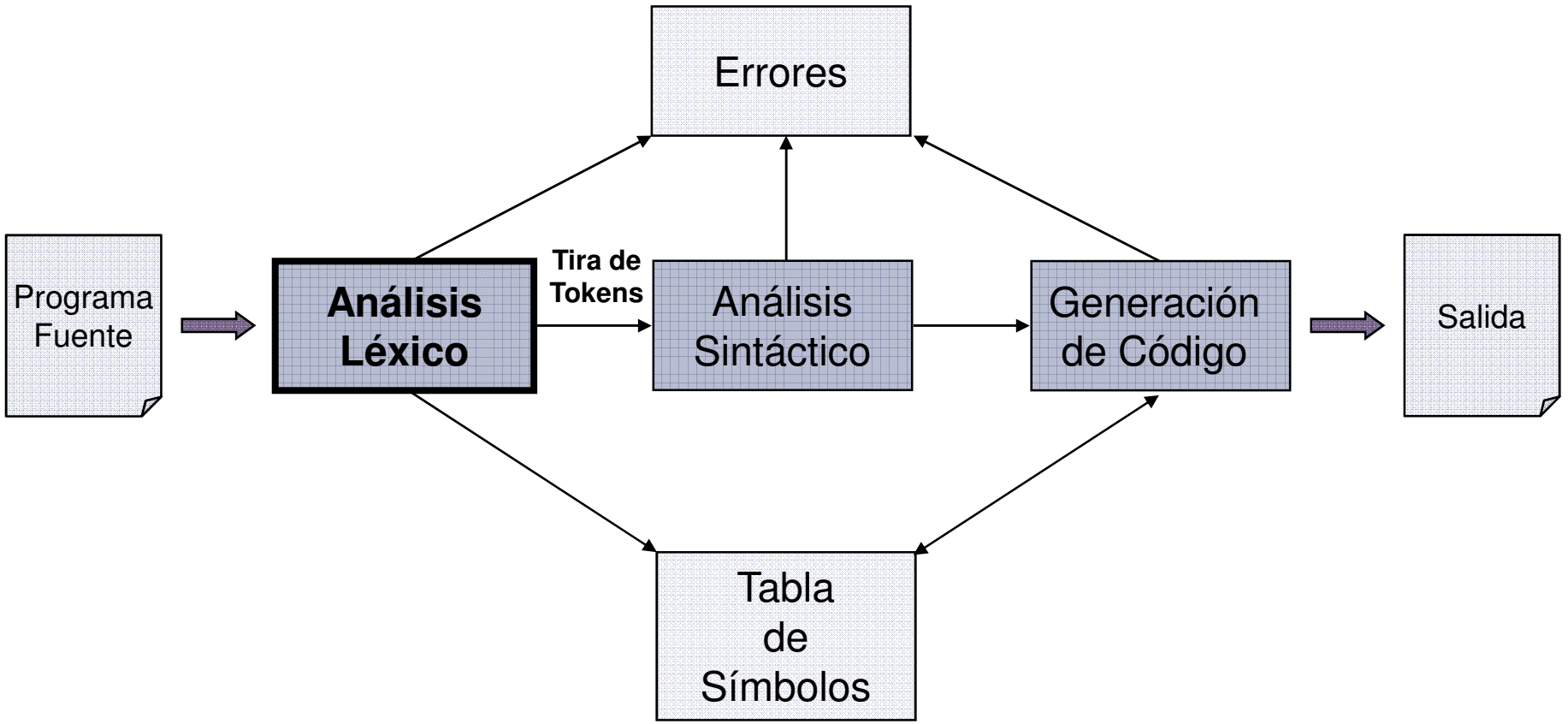


Análisis Léxico

- Lee el programa fuente.
- Agrupa los caracteres en unidades llamadas ***tokens***.

token: Secuencia de caracteres que forman una unidad significativa

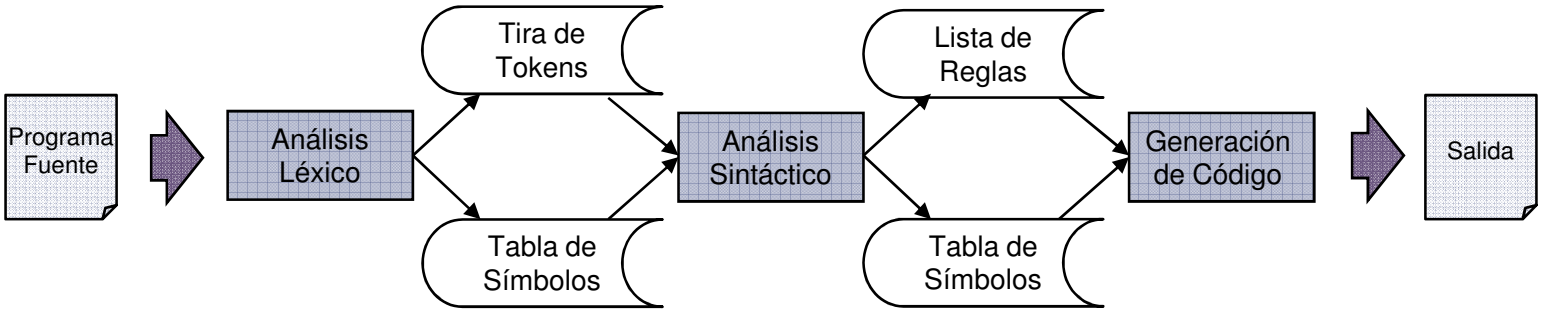
Fases de la Compilación



¿Por qué un Analizador Léxico?

Interacción Analizador Léxico – Analizador Sintáctico – Generador de Código

- Batch



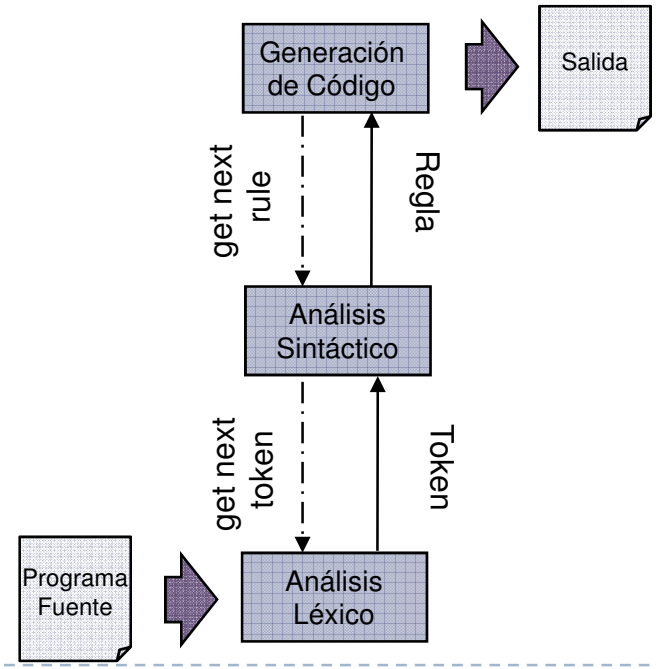
- Concurrente
  - Ejemplo: gcc

c1 ¦ c2 ¦ c3
(AL) (AS) (GC)

Interacción Analizador Léxico – Analizador Sintáctico – Generador de Código

- Monolítico
  - El Generador de Código es el main

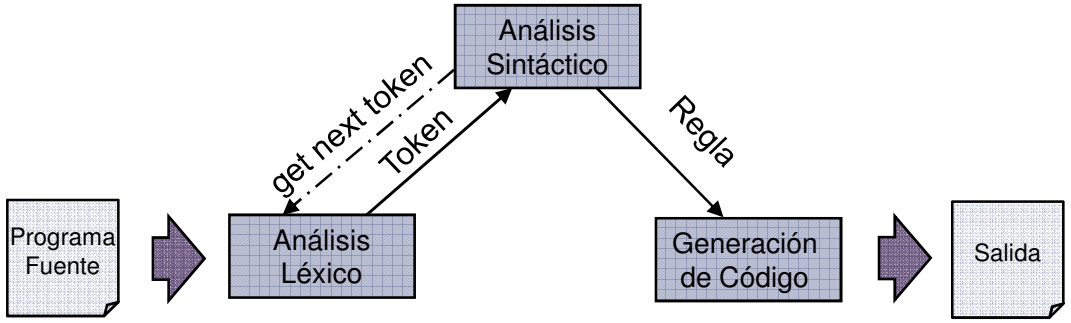
    El A.S. entrega una regla cuando el G.C. se la pide, y el A.L. entrega un token cuando el A.S. se lo pide



Interacción Analizador Léxico – Analizador Sintáctico – Generador de Código

- Monolítico
  - El Parser (A.S.) es el main:

    El Analizador Sintáctico va pidiendo tokens al Léxico, y cuando tiene una regla le pide al G.C. que genere código para ella.



Esta es la opción que utiliza Yacc

Tokens

- Identificadores
- Constantes
- Cadenas de caracteres
- Palabras reservadas
- Operadores
- Comparadores
- etc.

Tokens

- Los tokens se diferencian de la cadena de caracteres que representan.
- La cadena de caracteres es el ***Lexema*** o valor léxico.
  - Existen tokens que se corresponden con un único lexema
    - Ejemplo: Palabra Reservada IF
  - Existen tokens que pueden representar lexemas diferentes
    - Ejemplo: Identificador Plazo, Identificador Tasa

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

Análisis Léxico
Ejemplo

[if] [Plazo] [>=] [30]
  [then] [Tasa] [:=] [Base] [+] [Recargo] [/] [100]
  [else] [Tasa] [:=] [Base]

Tokens

- El Análizador Léxico hace una correspondencia entre cada tipo de token y un número entero.
- El Analizador Léxico entrega al Analizador Sintáctico los números enteros que corresponden a cada tipo de token.

Tokens

Tipo de token	Identificación del tipo de token
ID	27
CTE	28
IF	59
THEN	60
ELSE	61
+	70
/	73
>=	80
:=	85

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

```
[if] [Plazo] [>=] [30]
   [then] [Tasa] [:=] [Base] [+] [Recargo] [/] [100]
   [else] [Tasa] [:=] [Base]
```

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[**59**] [Plazo] [>=] [30]
[then] [Tasa] [:=] [Base] [+] [Recargo] [/] [100]
[else] [Tasa] [:=] [Base]

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[**59**] [**27**] [>=] [30]
[then] [Tasa] [:=] [Base] [+] [Recargo] [/] [100]
[else] [Tasa] [:=] [Base]

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[59] **[27]** **[80]** [30]
[then] [Tasa] [:=] [Base] [+] [Recargo] [/] [100]
[else] [Tasa] [:=] [Base]

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[59] **[27]** **[80]** **[28]**
[then] [Tasa] [:=] [Base] [+] [Recargo] [/] [100]
[else] [Tasa] [:=] [Base]

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[59] [27] [80] [28]
[60] [27] [85] [27] [70] [27] [73] [28]
[61] [27] [85] [27]

Tokens

- Puesto que un token puede representar más de un lexema, el A.L. debe enviar información adicional al A.S., en forma de atributo/s. Esa información será usada por el G.C.
- Por cada token detectado, el A.L. entrega un par:

  <token, atributo>
- En principio, ese atributo sería el lexema.

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[59,'IF'] [27,'Plazo'] [80,'>='] [28,'30']
[60,'THEN'] [27,'Tasa'] [85,':='] [27,'Base'] [70,'+'] [27,'Recargo'] [73,'/'] [28, '100']
[61,'ELSE'] [27,'Tasa'] [85,':='] [27,'Base']

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[59,‘IF’] [27,‘Plazo’] [80,‘>=’] [28,‘30’]
[60,‘THEN’] [27,‘Tasa’] [85,‘:=’] [27,‘Base’] [70,‘+’] [27,’Recargo’]
[73,‘/’] [28,‘100’]
[61,‘ELSE’] [27,’Tasa’] [85,‘:=’] [27,‘Base’]

Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[59] [27,'Plazo'] [80] [28,'30']
[60] [27,'Tasa'] [85] [27,'Base'] [70] [27,'Recargo'] [73] [28,'100']
[61] [27,'Tasa'] [85] [27,'Base']

Atributos de los tokens

- Lexema, nro. de línea en que aparece el token, etc.

- En la práctica, la información adicional para cada token se almacena en una **Tabla de Símbolos**, y el atributo entregado es el puntero o referencia a la entrada correspondiente en la Tabla de Símbolos.

Tabla de Símbolos

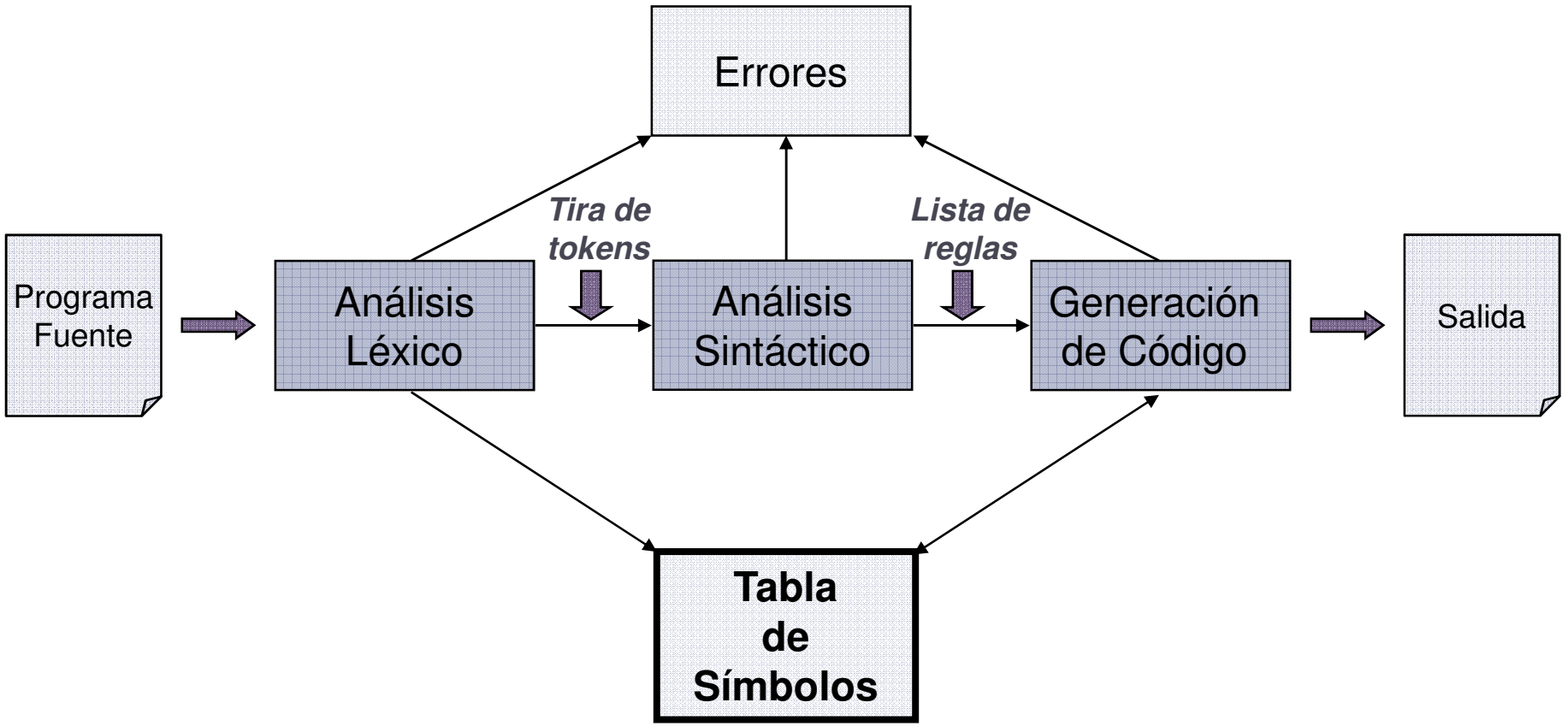
Es una estructura de datos que contiene un registro para cada identificador (y todo otro token que pueda representar más de un lexema) utilizado en el código fuente, con campos que contienen información relevante para ese símbolo (atributos).

Tabla de Símbolos

- Cuando el Análisis Léxico detecta un token de tipo identificador, u otro token que pueda representar más de un lexema, lo ingresa en la Tabla de Símbolos.

  (Esta acción podría hacerse durante el Análisis Sintáctico)
- Durante la Generación de Código se ingresa información para los atributos de los símbolos, y se usa esa información de diversas maneras.
- Durante la Generación de Código puede ser necesario incorporar nuevas entradas a la Tabla de Símbolos.

Fases de la Compilación



Análisis Léxico
Ejemplo

```
if Plazo >= 30
   then Tasa := Base + Recargo / 100
   else Tasa := Base
```

[59] [27] [80] [28] [60] [27] [85] [27] [70] [27]
[73] [28] [61] [27] [85] [27]

IF ID >= CTE THEN ID := ID + ID / CTE ELSE ID := ID

Análisis Léxico: Funciones

Principales:

- Reconocer tokens
- Informar errores léxicos

Secundarias:

- Eliminar comentarios
- Eliminar blancos, tabulaciones, newlines
- Llevar la cuenta de los saltos de línea, para correlacionar los mensajes de error con el programa fuente

Construcción del Analizador Léxico

- Construir un diagrama que represente la estructura de los tokens del programa fuente.
- Convertir el diagrama en un programa.

Autómata Finito

- AF = { Q , S_e , δ , q_0 , F }
  - Q : Conjunto finito de estados
  - S_e : Conjunto finito de símbolos de entrada
  - δ : Función de transición
  - q_0 : Estado inicial
  - F : Conjunto de estados finales

Gramática independiente de Contexto

- Conjunto de símbolos terminales: los tokens.
- Conjunto de no terminales.
- Conjunto de producciones donde cada producción consiste de un no terminal, o lado izquierdo, una flecha y una secuencia de tokens y/o no terminales, o lado derecho.
- Designación de uno de los no terminales como símbolo de inicio.

Gramáticas regulares

- Todo no terminal se define en base a terminales, o en base a un NT y un terminal

  $A \rightarrow a \mid Aa$ (recursiva a izquierda)

  $A \rightarrow a \mid aB$ (recursiva a derecha)

Diagrama de Transición de estados

- Una gramática regular se puede representar mediante un diagrama de transición de estados

  Dado:

  T = {a,b,c} NT = {A,B,C,S}

  Gramática:

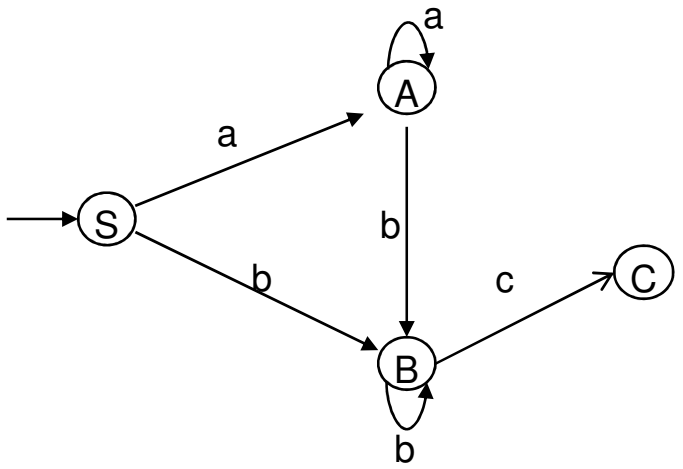
  $A \rightarrow Sa \mid Aa$

  $B \rightarrow Sb \mid Bb \mid Ab$

  $C \rightarrow Bc$

  $S \rightarrow \varepsilon$

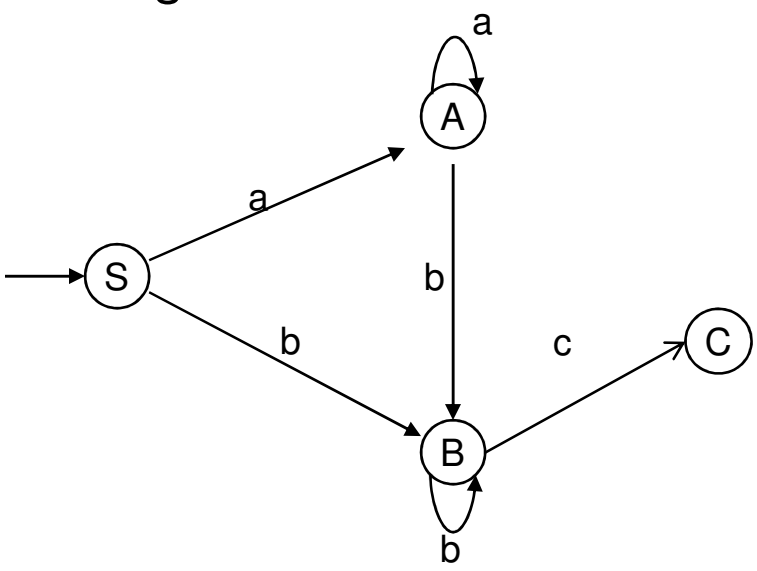
  Diagrama de Transición de Estados



Matriz de Transición de Estados

- El diagrama de transición de estados se puede representar mediante una matriz de transición de estados

Diagrama de Transición de estados



Matriz de Transición de Estados

	a	b	c
S	A	B	
A	A	B	
B		B	C
C			

Analizador Léxico: Construcción

- Identificadores (cadena que comienza con una letra y continúa con letras y/o dígitos)
- Constantes enteras (secuencia de dígitos)
- Comentarios tipo C /* … */
- Operadores aritméticos + - * /
- Comparadores > >= < <= == <>
- Palabras reservadas IF, ELSE, WHILE

Analizador Léxico: Construcción

- Se construye el diagrama de transición de estados que represente la estructura de cada uno de los tokens del lenguaje.

Análisis Léxico: Gramática

- **Identificador**: cadena que comienza con una letra y continúa con letras y/o dígitos

  ID: letra (letra | digito)*

- **Constante entera**: secuencia de dígitos

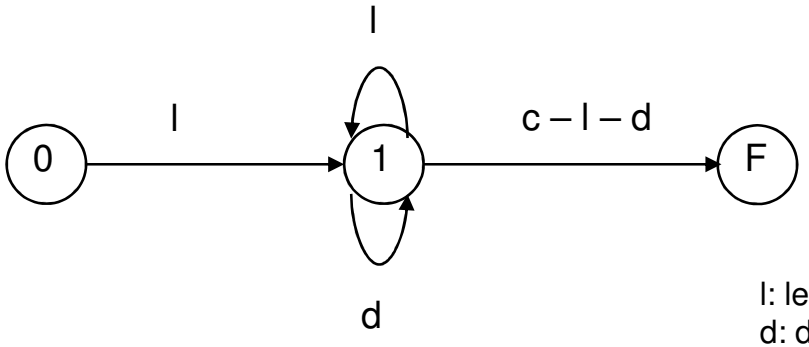
  CTE: digito +

- Reglas Léxicas:
  - <ID> → <letra> | <ID> <letra> | <ID><digito>
  - <CTE> → <digito> | <CTE> <digito>
  - <letra> → a | b | c | … | z
  - <digito> → 0 | 1 | 2 | … | 9

¿Esta gramática es regular?

Identificadores

- **Identificador**: cadena que comienza con una letra y continúa con letras y/o dígitos



l: letra mayúscula o minúscula
d: dígito

- 0: <Inicio>
- 1: <ID>
- F: <Fin>

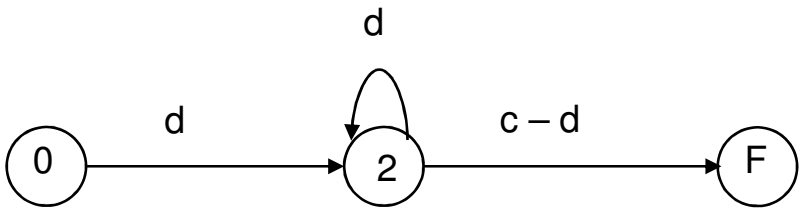
<ID> → <Inicio> l | <ID> l | <ID> d

<Fin> → <ID> otro

(otro es cualquier carácter distinto de l o d)

Constantes

- **Constante entera**: secuencia de dígitos



- 0: <Inicio>
- 2: <CTE>
- F: <Fin>

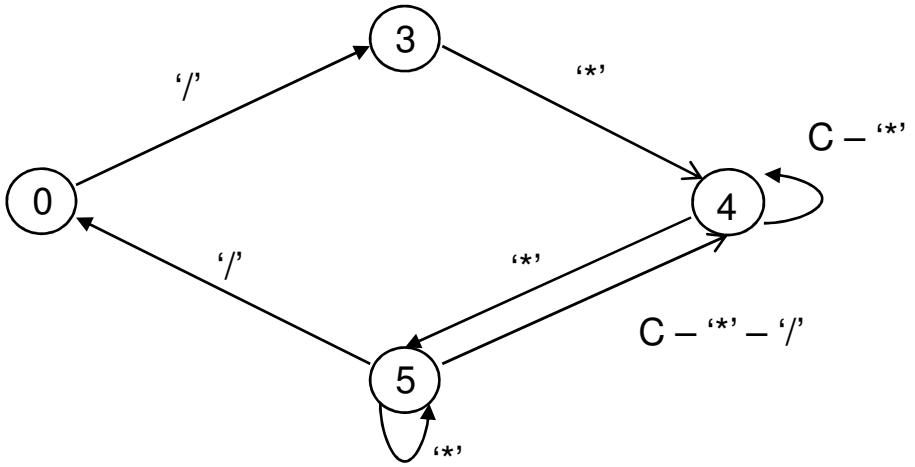
<CTE> → <Inicio> d | <CTE> d

<Fin> → <CTE> otro

(otro es cualquier carácter distinto de d)

Comentarios

- **Comentarios tipo C**: /* … */



- 0: <Inicio>
- 3: <PCom>
- 4: <Com>
- 5: <PFCom>

<PCom> → <Inicio> '/'

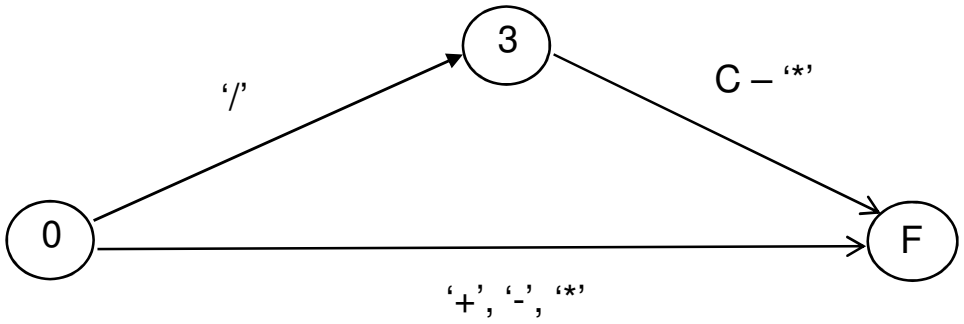
<Com> → <PCom> '*' | <Com> otro1 |
<PFCom> otro2

(otro1 es cualquier carácter distinto de '*' y otro2 cualquier carácter distinto de '*' y '/')

<PFCom> → <Com> '*' | <PFCom> '*'
<Inicio> → <PFCom> '/'

Operadores

- **Operadores aritméticos**: + - * /

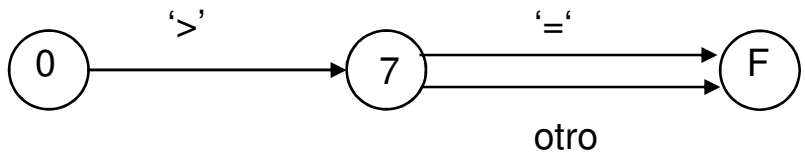


- 0: <Inicio> <PCom> → <Inicio> '/'
- 3: <PCom> <Fin> → <Inicio> '+' | <Inicio> '-' | <Inicio> '*' |
- F: <Fin> <PCom> otro1

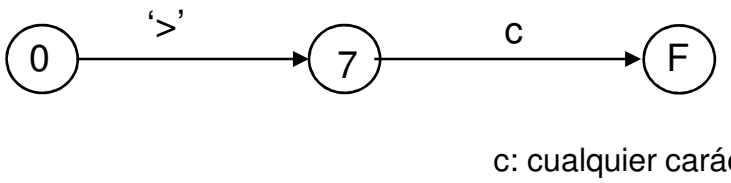
(otro1 es cualquier carácter distinto de '*')

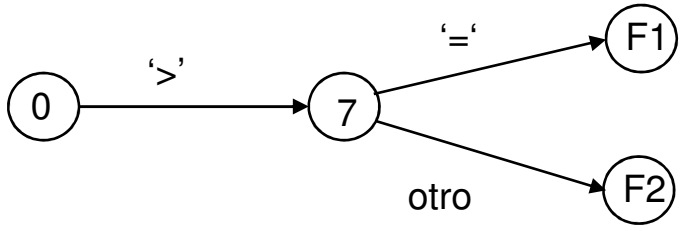
Comparadores

- **Comparadores:** > >= < <= == <>



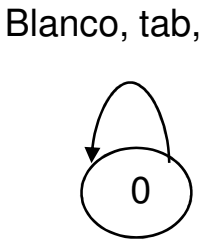
- En Autómatas Finitos la representación debería ser diferente:





Blancos, tabulaciones, saltos de línea

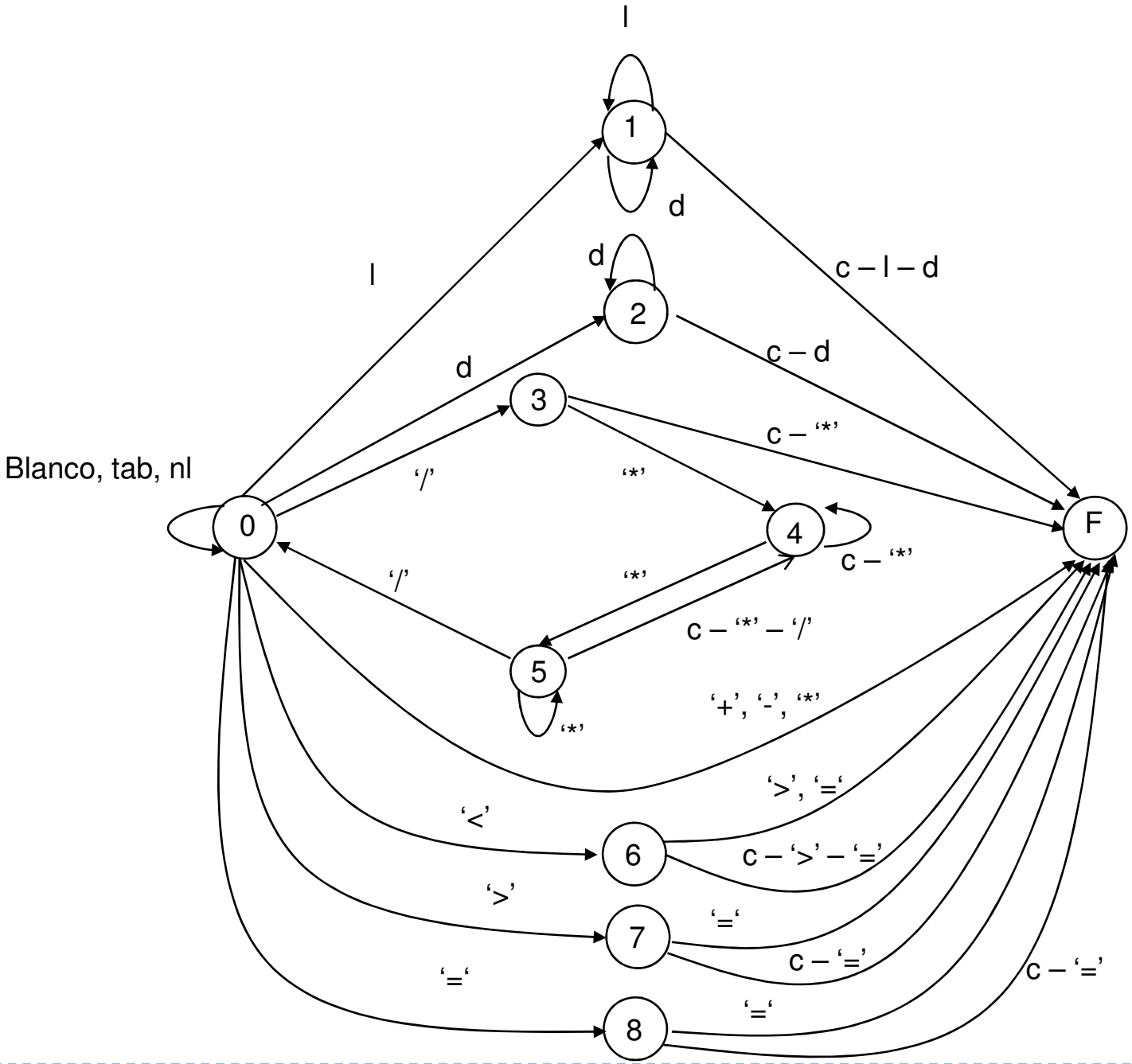
- Eliminar blancos, tabulaciones, saltos de línea



- No se entregan al Analizador Sintáctico

  ¿Nunca?

Autómata para el Analizador Léxico


l
1
d
l
d
2
c – l – d
d
c – d
3
c – ‘*’
Blanco, tab, nl
‘/’
‘*’
0
4
F
c – ‘*’
‘/’
‘*’
c – ‘*’ – ‘/’
5
‘+’, ‘-’, ‘*’
‘*’
‘>’, ‘=‘
‘<’
6
c – ‘>’ – ‘=’
‘>’
‘=‘
7
c – ‘=’
‘=‘
c – ‘=’
‘=‘
8

Matriz de Transición de Estados

	l	d	/	*	+	-	=	<	>	otro	BL tab nl	$
0	1	2	3	F	F	F	8	6	7	F	0	F
1	1	1	F	F	F	F	F	F	F	F	F	F
2	F	2	F	F	F	F	F	F	F	F	F	F
3	F	F	F	4	F	F	F	F	F	F	F	F
4	4	4	4	5	4	4	4	4	4	4	4	F
5	4	4	0	5	4	4	4	4	4	4	4	F
6	F	F	F	F	F	F	F	F	F	F	F	F
7	F	F	F	F	F	F	F	F	F	F	F	F
8	F	F	F	F	F	F	F	F	F	F	F	F

¿Qué significa llegar al estado final?

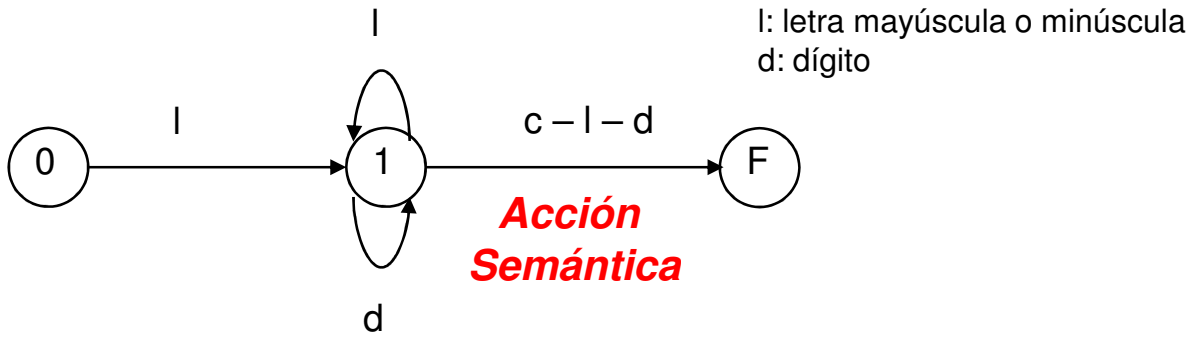
Analizador Léxico: Tokens

- Identificadores (cadena que comienza con una letra y continúa con letras y/o dígitos)
- Constantes enteras (secuencia de dígitos
- Comentarios tipo C /* … */
- Operadores aritméticos + - * /
- Comparadores > >= < <= == <>
- Palabras reservadas (secuencia de letras mayúsculas)

¿Y las palabras reservadas?

Palabras Reservadas

- **Palabra reservada:** Secuencia de letras mayúsculas

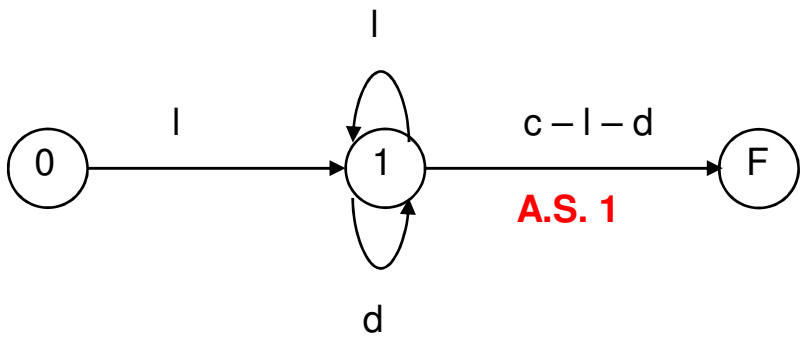


  - ¿Identificador o palabra reservada?

Acciones Semánticas

- Fragmentos de código en el lenguaje del compilador.
- Se asocian a las transiciones.
- Impiden el crecimiento del número de estados del autómata.

Identificadores y Palabras Reservadas

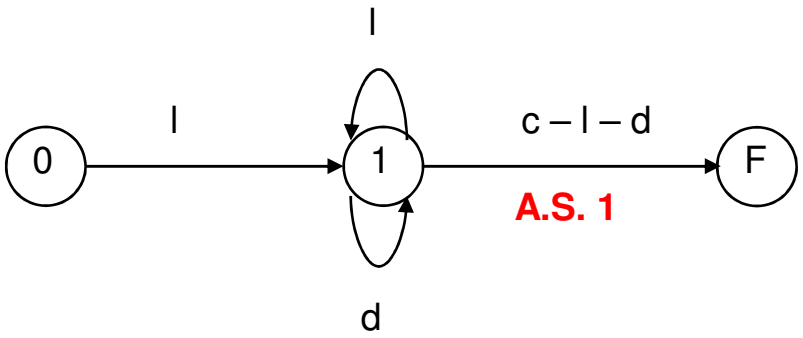


Almacenamiento de las palabras reservadas:

- Tabla de palabras reservadas
- Tabla de Símbolos

- Acción Semántica 1:
  - Devolver a la entrada el último carácter leído
  - Buscar en la TPR
    - Si está, devolver la Palabra Reservada
    - Si no está,
      - Buscar en la TS
        - Si está, devolver ID + Punt TS
        - Si no está,
          - Alta en la TS
          - Devolver ID + Punt TS

Identificadores y Palabras Reservadas

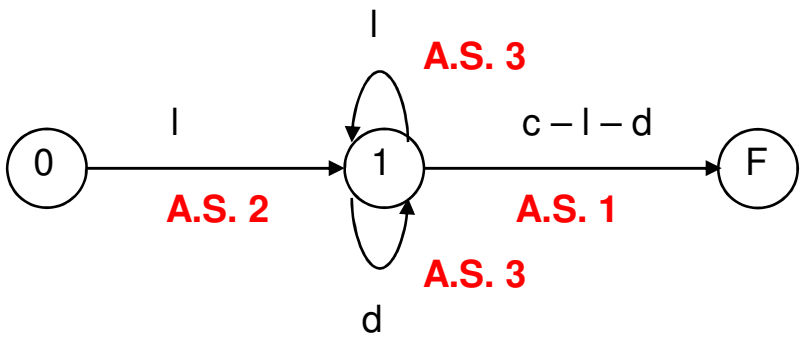


Almacenamiento de las palabras reservadas:

- Tabla de Símbolos

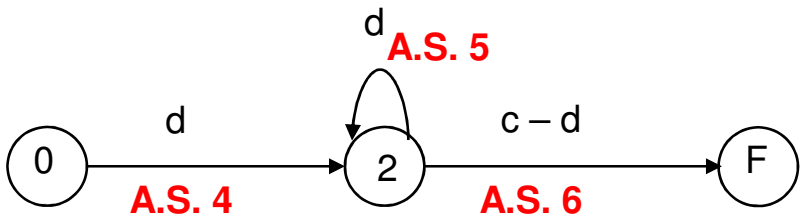
- Acción Semántica 1:
  - Devolver a la entrada el último carácter leído
  - Buscar en la TS
    - Si está,
      - Si es PR, devolver la Palabra Reservada
      - Si no, Devolver ID + Punt TS
    - Si no está,
      - Alta en la TS
      - Devolver ID + Punt TS

Acciones Semánticas



- Acción Semántica 2:
  - Inicializar string (se reserva la máxima longitud permitida para identificadores)
  - Agregar letra al string
- Acción Semántica 3:
  - Agregar letra o dígito al string

Constantes - Acciones Semánticas

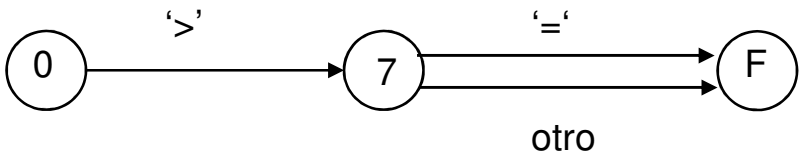


- Acción Semántica 4:
  - Inicializar string para la constante
  - Agregar dígito al string
- Acción Semántica 5:
  - Agregar dígito al string
- Acción Semántica 6:
  - Devolver a la entrada el último carácter leído
  - Verificar rango de la constante
  - Alta en la TS
  - Devolver CTE + Punt TS

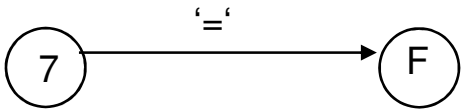
Constantes

- Formas de reconocerlas
  - Un solo token
  - Diferentes tokens
- Almacenamiento
  - Tabla de Símbolos
  - Tabla de Constantes y Literales

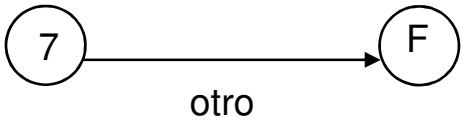
Comparadores



- ¿Qué diferencia hay entre una y otra transición?

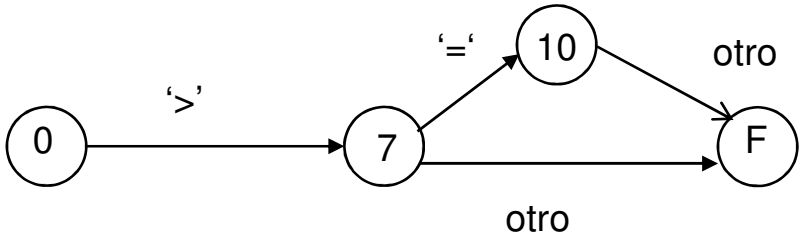


El carácter '=' se consume.



El carácter 'otro' no se consume.

- ¿Cómo se distingue una situación de la otra?
  - Solución 1:



  - Solución 2: *ACCIONES SEMÁNTICAS*

Acciones semánticas

- ¿Cómo se asocian las acciones semánticas con las transiciones?

Matriz de Transición de Estados

(Contiene la "letra chica" del Análisis Léxico)

Matriz de Transición de Estados

	l	d	/	*	+	-	=	<	>	otro	BL tab nl	$
0	1 AS2	2 AS4	3	F	F	F	8	6	7	F	0	F
1	1 AS3	1 AS3	F AS1	F AS1	F AS1	F AS1	F AS1	F AS1	F AS1	F AS1	F AS1	F AS1
2	F	2	F	F	F	F	F	F	F	F	F	F
3	F	F	F	4	F	F	F	F	F	F	F	F
4	4	4	4	5	4	4	4	4	4	4	4	F
5	4	4	0	5	4	4	4	4	4	4	4	F
6	F	F	F	F	F	F	F	F	F	F	F	F
7	F	F	F	F	F	F	F	F	F	F	F	F
8	F	F	F	F	F	F	F	F	F	F	F	F

Programación

- Para cada carácter leído, mapear con las columnas de la matriz
  - l → columna 0
  - d → columna 1
  - '/' → columna 2
  - …
  - Si el carácter leído no corresponde a ninguna columna, debe informarse "Carácter Inválido"

Programación

- Definir matriz de estados
  - nuevo_estado [9][12] = {1,2,3,-1,-1,8,6,…}

Programación

- Definir matriz de acciones semánticas
  - accion_sem [9][12] = {AS2, AS4, … }

Implementación:

Matriz de punteros a función o equivalente

Programación

- Código para el Análisis Léxico:
  - Dados:
    - estado: estado actual
    - entrada: símbolo leído
  - Acciones:
    - accion_sem[estado][entrada]

      ejecuta la acción semántica
    - estado = nuevo_estado[estado][entrada]

      actualiza el estado actual

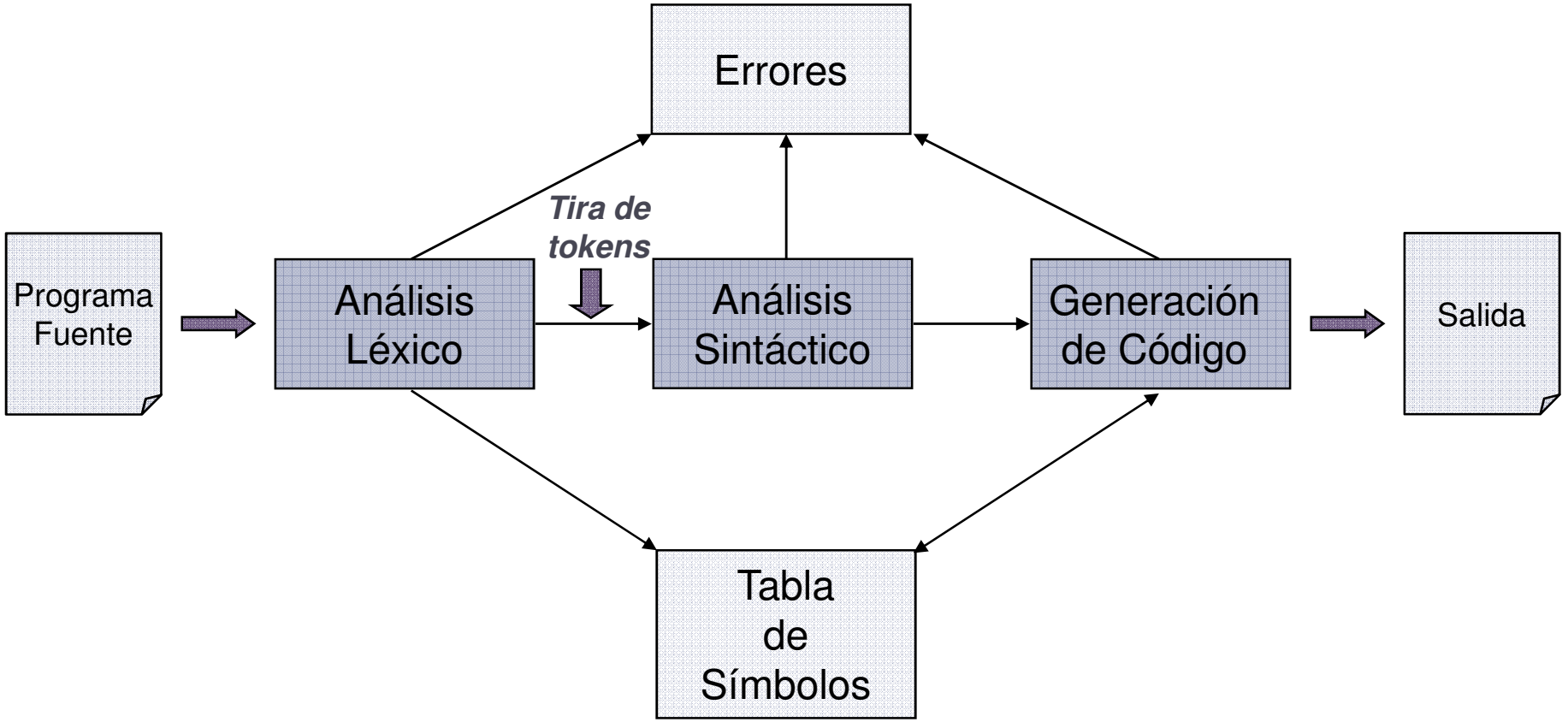
Errores Léxicos

- En el mapeo de los símbolos de entrada, se pueden detectar caracteres inválidos.
- Si en un estado del autómata, el carácter de entrada no coincide con ninguno de los arcos de salida, se trata de un error.
- Constante fuera de rango.
- etc.

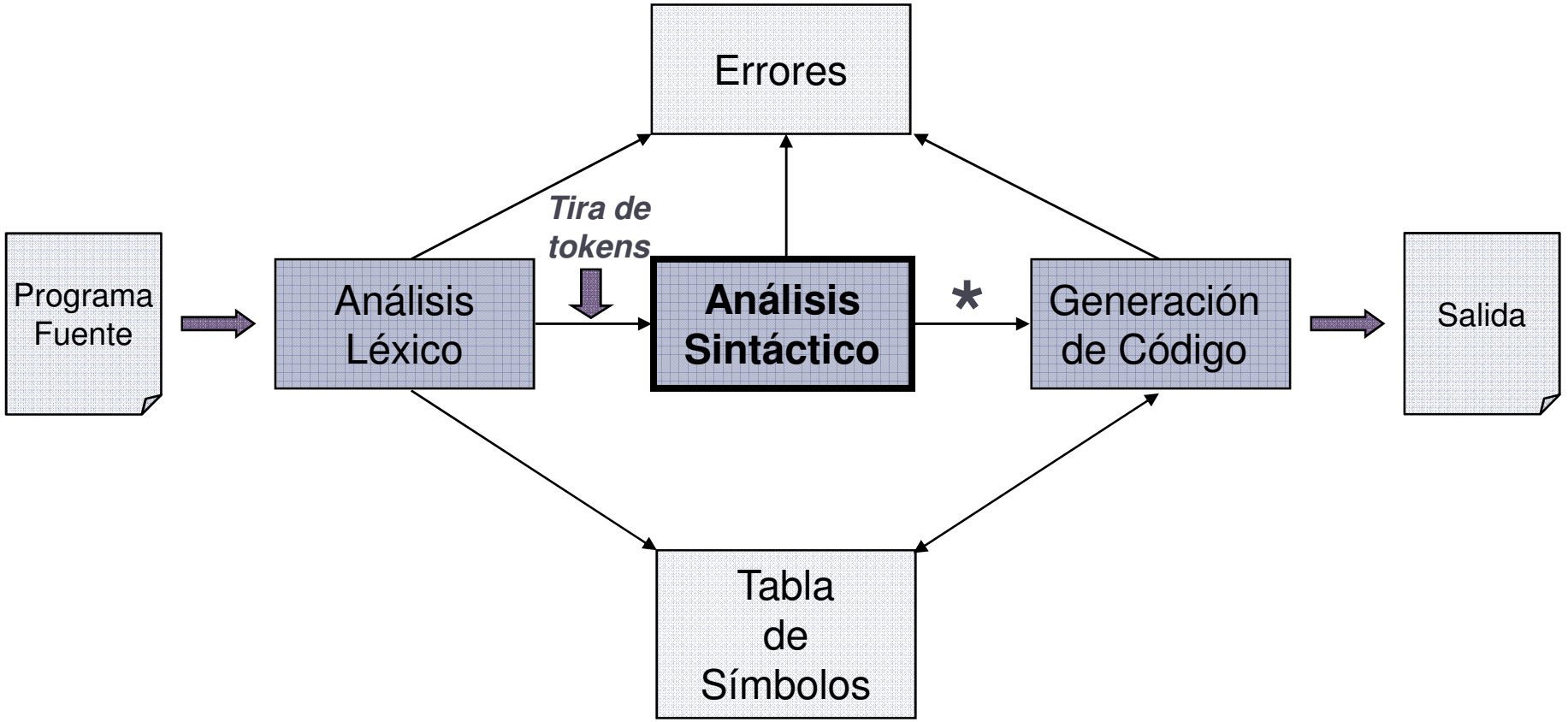
Errores Léxicos

- Acciones frente a un error:
  - Modo pánico: borrar caracteres hasta que aparezca un token conocido
    - Ej. Constante fuera de rango.
  - Inserción / Borrado / Reemplazo
    - Borrar un carácter de la entrada
    - Insertar un carácter omitido
    - Sustituir un carácter por otro
    - Transponer dos caracteres adyacentes

Fases de la Compilación



Fases de la Compilación



Ejercicios

- Indicar si en las siguientes sentencias hay errores léxicos
- (considerar el lenguaje presentado en clase)
  - Ab12cd + 12x
  - FI (x > 234.3)
  - X * 32768
  - W = 40000
  - xyZw+*123

Ejercicios

- Dibujar el fragmento del autómata de un Analizador Léxico que reconozca:
  - Constantes enteras con valores entre -32767 y 32768
  - Operadores aritméticos +, -, * y /

Ejercicios

- Se tiene un lenguaje donde, tanto los identificadores como las palabras reservadas se escriben indistintamente en mayúsculas o minúsculas, pero los identificadores comienzan con "_".

  Dibujar el fragmento de autómata finito del Analizador Léxico que reconozca identificadores y palabras reservadas.