Diseño de Compiladores I

Trabajo Práctico 1 - Análisis Léxico

Trabajo Práctico 1

- La entrega se hará en forma conjunta con el Trabajo Práctico Nro. 2
- Fecha de Entrega: 25-09-2017

Trabajo Práctico 1- Objetivo

Desarrollar un Analizador Léxico

Trabajo Práctico 1- Objetivo

El Analizador Léxico debe reconocer:

- Identificadores cuyos nombres pueden tener hasta 15 caracteres de longitud. El primer carácter debe ser una letra, y el resto pueden ser letras, dígitos y "_". Los identificadores con longitud mayor serán truncados y esto se informará como Warning. Las letras utilizadas en los nombres de identificador pueden ser mayúsculas o minúsculas, y el lenguaje NO será case sensitive. Entonces, el identificador MyVariable, será igual a myvariable.

- Palabras reservadas (en mayúsculas):

IF, THEN, ELSE, END_IF, BEGIN, END, OUT

Trabajo Práctico 1- Objetivo

El Analizador Léxico debe reconocer:

- Constantes correspondientes a los temas particulares asignados a cada grupo.
- **Nota**: Para aquellos grupos cuyos temas asignados incluyan constantes tales que, el rango de uno de los tipos esté incluido en el rango del otro, considerar que, una constante incluida en el rango más acotado, deberá ser considerada del tipo más chico.

  Ejemplos:

  INT y LONG. La constante 123 será de tipo INT.

  UINT y ULONG. La constante 25 será de tipo UINT.

  FLOAT y DOUBLE. La constante 1.234 será de tipo FLOAT.

Trabajo Práctico 1- Objetivo

El Analizador Léxico debe reconocer:

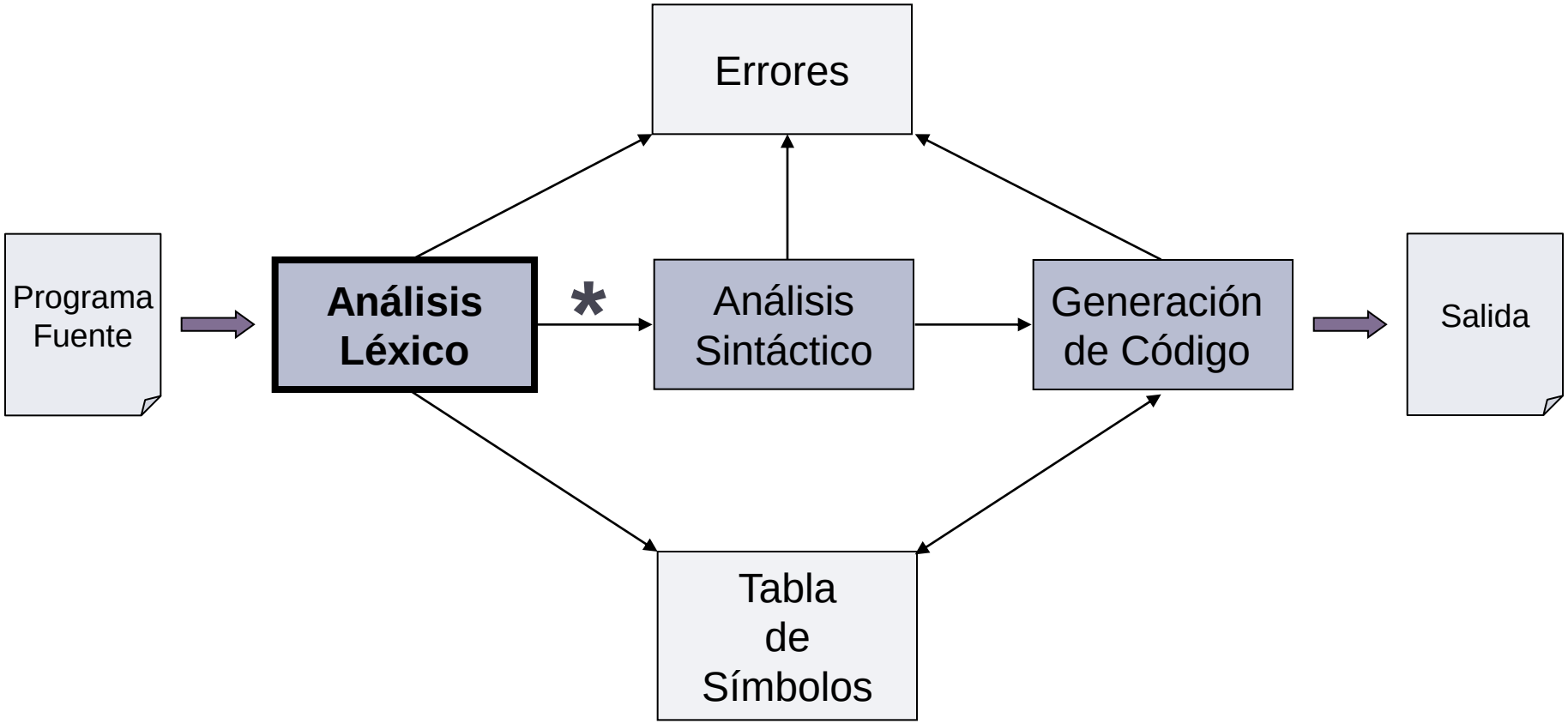
- Operadores aritméticos: “**+**”,“**-**” ,“*****”,“*/*”.
- Operador de asignación: “**=**”
- Comparadores: “**>=**”,“**<=**”,“**>**”,“**<**”,“**==**”,“**<>**”
- “**(**” “**)**” “**,**” “**:**” y “**.**”
- Cadenas de caracteres correspondientes al tema particular de cada grupo.

Trabajo Práctico 1- Objetivo

El Analizador Léxico debe eliminar de la entrada:

- Comentarios correspondientes al tema particular de cada grupo.
- Caracteres en blanco, tabulaciones y saltos de línea, que pueden aparecer en cualquier lugar de una sentencia.

Fases de la Compilación



Análisis Léxico

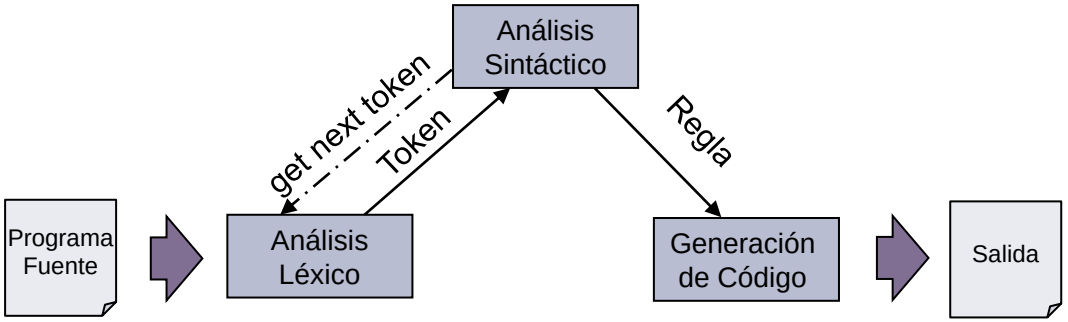
- Lee el programa fuente.
- Agrupa los caracteres en unidades llamadas ***tokens***.

token: Secuencia de caracteres que forman una unidad significativa

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

Trabajo Práctico 1- Especificaciones

Se sugiere la implementación de un **consumidor de tokens** que invoque al Analizador Léxico solicitándole tokens. En el trabajo práctico 2, esta funcionalidad estará a cargo del Analizador Sintáctico.

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

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

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 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

Contendrá un registro para cada identificador, constante y cadena de caracteres que aparezcan en el código fuente, con campos para registrar la información relevante para cada símbolo (atributos).

Es aconsejable la implementación de esta tabla con un mecanismo eficiente para buscar y recuperar información en ella.

Se debe implementar con una estructura dinámica, que permita ingresar tantos símbolos como sea necesario.

Construcción del Analizador Léxico

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

Trabajo Práctico 1- Objetivo

El Analizador Léxico debe reconocer:

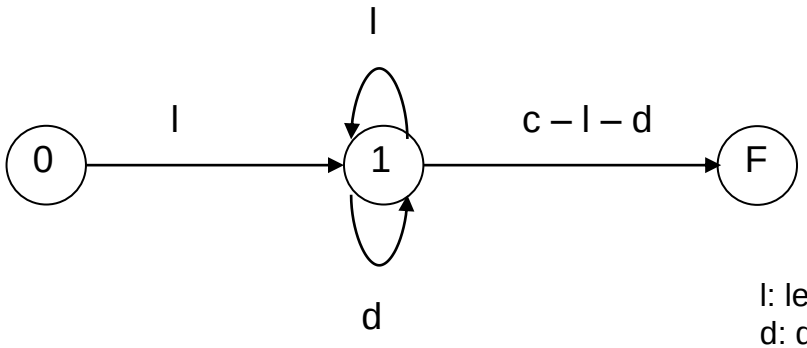
- Identificadores cuyos nombres pueden tener hasta 15 caracteres de longitud. El primer carácter debe ser una letra, y el resto pueden ser letras, dígitos y "_". Los identificadores con longitud mayor serán truncados y esto se informará como Warning. Las letras utilizadas en los nombres de identificador pueden ser mayúsculas o minúsculas, y el lenguaje NO será case sensitive. Entonces, el identificador MyVariable, será igual a myvariable.

- Palabras reservadas (en mayúsculas):

IF, THEN, ELSE, END_IF, BEGIN, END, OUT

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>

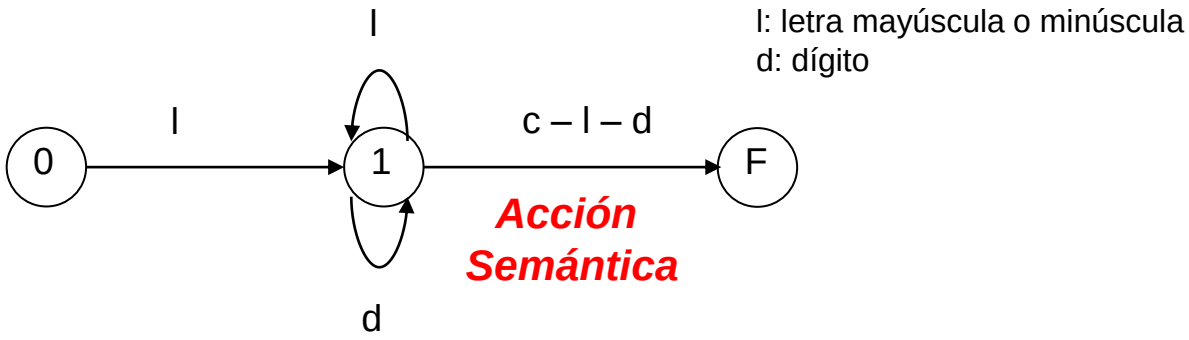
<ID> → <Inicio> l | <ID> l | <ID> d

<Fin> → <ID> otro

(otro es cualquier carácter distinto de l o d)

Palabras Reservadas

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

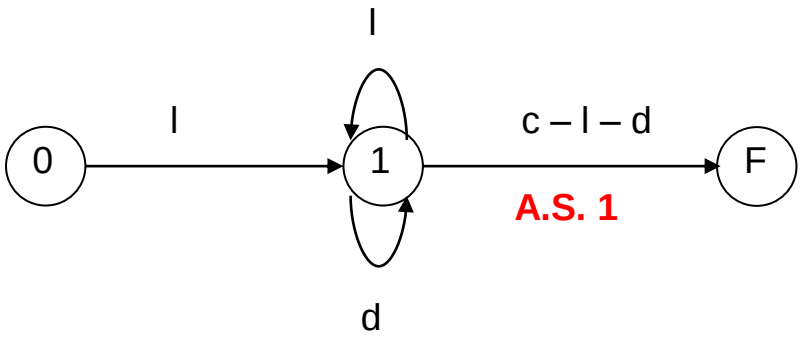


- ¿Identificador o palabra reservada?

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

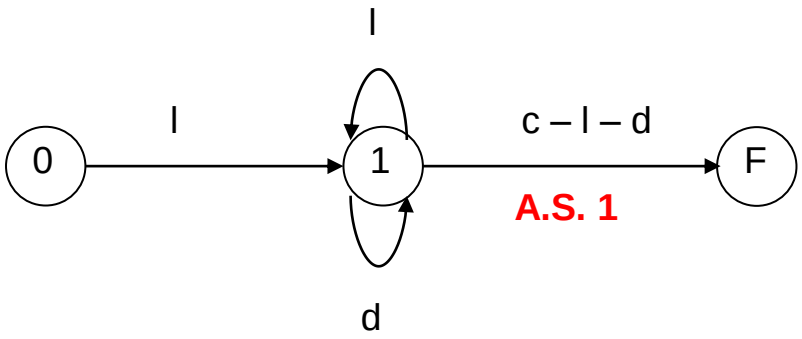


Almacenamiento de las palabras reservadas:

- 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

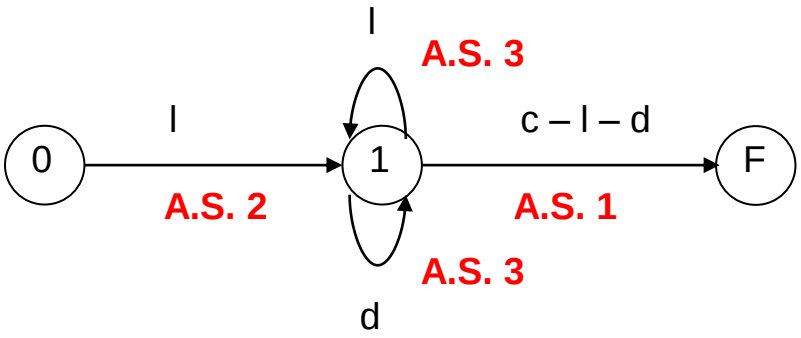


Almacenamiento de las palabras reservadas:

- 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

Trabajo Práctico 1- Objetivo

El Analizador Léxico debe reconocer:

- Constantes correspondientes a los temas particulares asignados a cada grupo.
- ***Nota***: Para aquellos grupos cuyos temas asignados incluyan constantes tales que, el rango de uno de los tipos esté incluido en el rango del otro, considerar que, una constante incluida en el rango más acotado, deberá ser considerada del tipo más chico.

  Ejemplos:

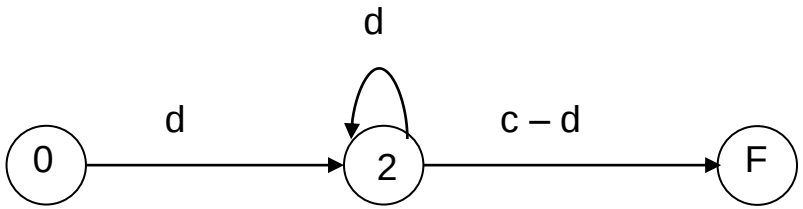
  INT y LONG. La constante 123 será de tipo INT.

  UINT y ULONG. La constante 25 será de tipo UINT.

  FLOAT y DOUBLE. La constante 1.234 será de tipo FLOAT.

Constantes

- **Constante entera**: secuencia de dígitos



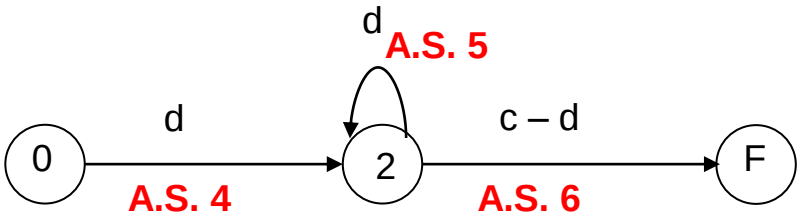
- 0: <Inicio>
- 2: <CTE>
- F: <Fin>

<CTE> → <Inicio> d | <CTE> d

<Fin> → <CTE> otro

(otro es cualquier carácter distinto de d)

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

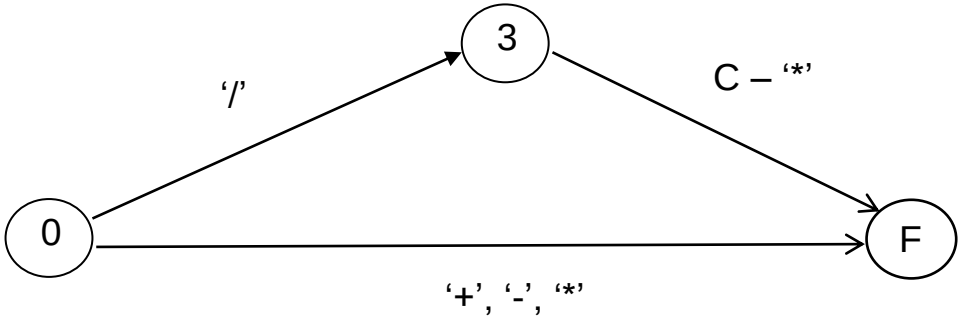
Trabajo Práctico 1- Objetivo

El Analizador Léxico debe reconocer:

- Operadores aritméticos: “**+**”,“**-**” ,“*****”,“*/*”.
- Operador de asignación: “**=**”
- Comparadores: “**>=**”,“**<=**”,“**>**”,“**<**”,“**==**”,“**<>**”
- “**(**” “**)**” “**,**” “**:**” y “**.**”
- Cadenas de caracteres correspondientes al tema particular de cada grupo.

Operadores

- **Operadores aritméticos**: + - * /



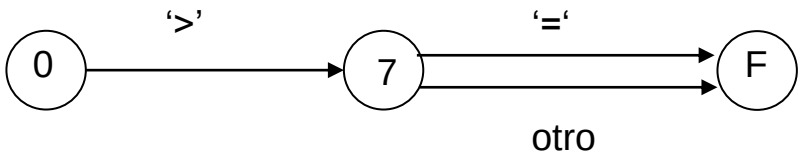
- 0: <Inicio>
- 3: <PCom>
- F: <Fin>

<PCom> → <Inicio> '/'

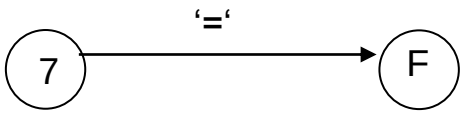
<Fin> → <Inicio> '+' | <Inicio> '-' | <Inicio> '*' |
<PCom> otro1

(otro1 es cualquier carácter distinto de '*')

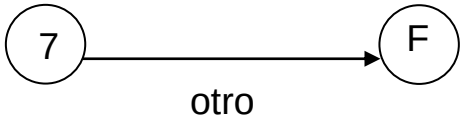
Comparadores



- ¿Qué diferencia hay entre una y otra transición?

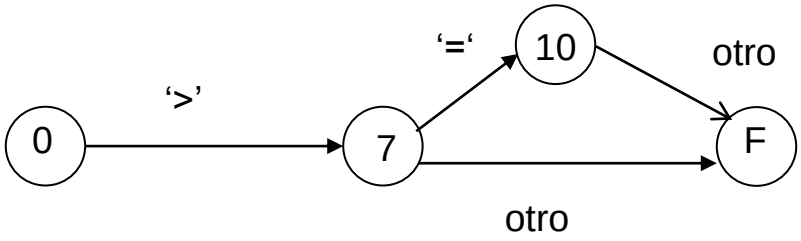


El carácter '=' se consume.



El carácter 'otro' no se consume.

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



  - Solución 2: *ACCIONES SEMÁNTICAS*

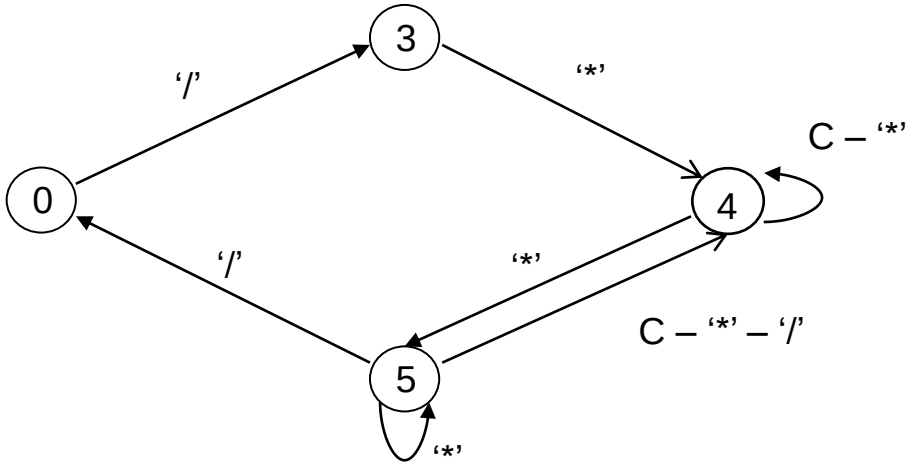
Trabajo Práctico 1- Objetivo

El Analizador Léxico debe eliminar de la entrada:

- Comentarios correspondientes al tema particular de cada grupo.
- Caracteres en blanco, tabulaciones y saltos de línea, que pueden aparecer en cualquier lugar de una sentencia.

Comentarios

- **Comentarios tipo C**: /* … */



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

<PCom> → <Inicio> '/'

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

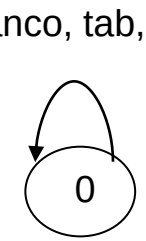
(otro1 es cualquier carácter distinto de '*' y otro2 cualquier carácter distinto de '*' y '/')

<PFCom> → <Com> '*' | <PFCom> '*'

<Inicio> → <PFCom> '/'

Blancos, tabulaciones, saltos de línea

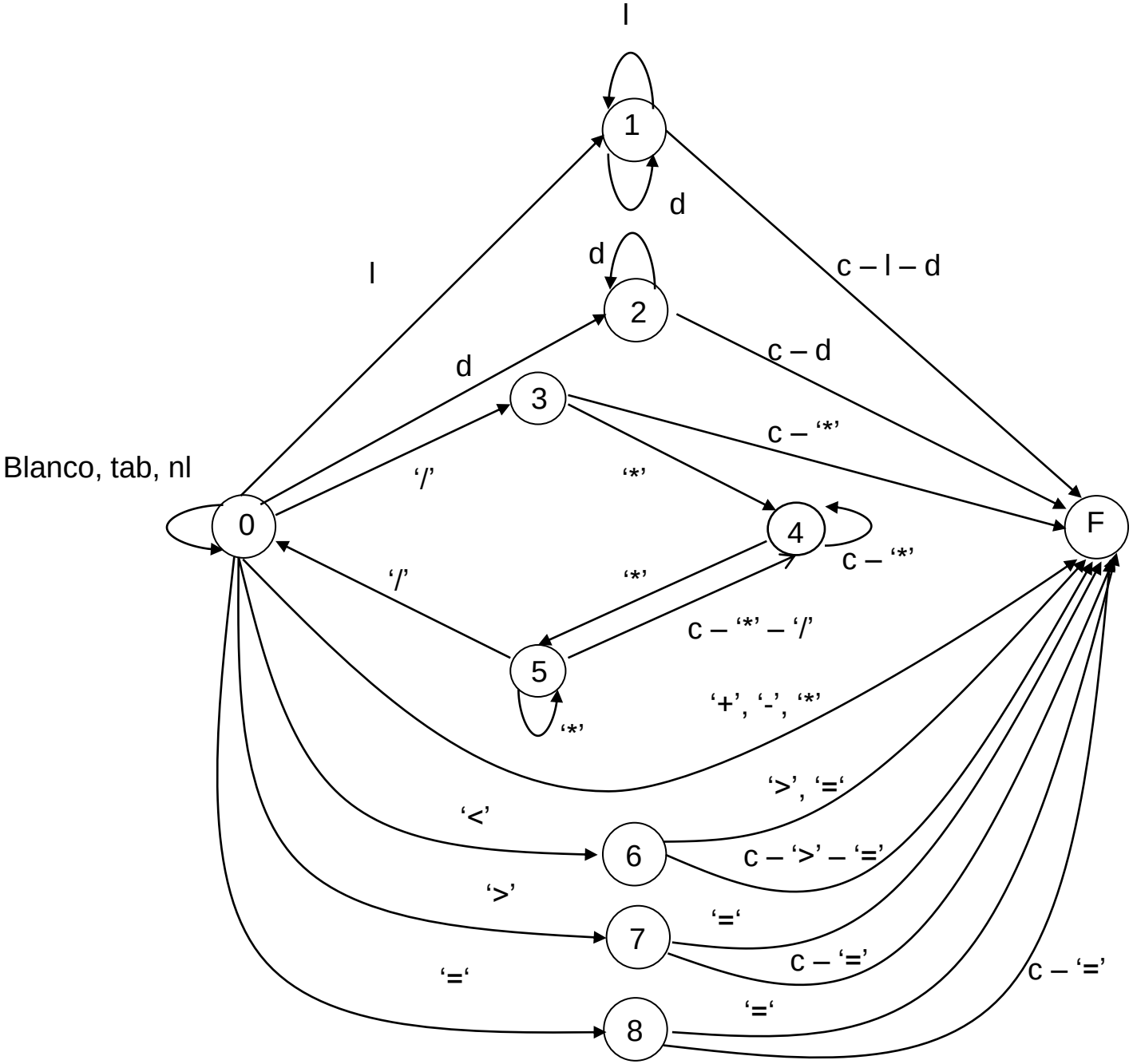
- Eliminar blancos, tabulaciones, saltos de línea



- 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

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,…}
- 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

Programación

- Matriz de transición de estados

  nuevoestado[9][11] = {1,2,3,-1,…}

- Matriz de acciones semánticas

  Int (*AS[9][11])()= {AS1, AS2, AS3, … }

- En el código del A.Léxico

  // ejecutar acción semántica:

  (*AS[estado][entrada])();

  // actualizar estado:

  Estado = nuevoestado[estado][entrada];

Programación

```
public abstract class AccionSemantica {
        public abstract boolean ejecutar(…);
}

public class InicializarBuffer extends AccionSemantica {
@Override
public boolean ejecutar(…) {
…
return true;
}
}
```

Programación

```
public AnalizadorLexico(){
                        int matEstados[][];
                        AccionSemantica accionesSemanticas[][];
…
// ejecutar accion semántica
accion = accionesSemanticas[estadoActual][caracterActual];
resultado = accion.ejecutar(…);

//actualizar estado;
estadoActual = matEstados[estadoActual][caracterActual];
```

Trabajo Práctico 1- Especificaciones

- El Analizador Léxico deberá leer un código fuente, identificando e informando:
  - Tokens detectados en el código fuente. Por ejemplo:

    Palabra reservada **IF**

    (

    Identificador **varx**

    +

    Constante entera **25**

    Palabra reservada **ELSE**

    etc.
  - Errores léxicos detectados en el código fuente, indicando: nro. de línea y descripción del error. Por ejemplo:

    Línea 24: Constante entera fuera del rango permitido
  - Contenidos de la Tabla de Símbolos.

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.

Trabajo Práctico 1- Especificaciones

- El código fuente **DEBE SER LEÍDO DESDE UN ARCHIVO**, y el nombre **DEBE PODER SER ELEGIDO** por el usuario del compilador.

  Se sugiere que la dirección del archivo pueda ser leída como primer argumento de la aplicación en la línea de comandos.

- La numeración de las líneas de código debe comenzar en 1. Si se implementa una interfaz que permite mostrar o editar el código fuente, incluir alguna manera de identificar el número de cada línea del código.

Trabajo Práctico 1- Especificaciones

- Para la programación se podrá elegir el lenguaje. Para esta elección, tener en cuenta que el analizador léxico se integrará luego a un Parser (Analizador Sintáctico) generado utilizando una herramienta tipo Yacc. Por lo tanto, es necesario asegurarse la disponibilidad de dicha herramienta para el lenguaje elegido.

- El Analizador Léxico deberá implementarse mediante una matriz de transición de estados y una matriz de acciones semánticas, de modo que cada cambio de estado y acción semántica asociada, sólo dependa del estado actual y el carácter leído.

Trabajo Práctico 1- Especificaciones

- Implementar una Tabla de Símbolos donde se almacenarán identificadores, constantes, y cadenas de caracteres. Es requisito para la aprobación del trabajo, que la tabla sea implementada con una estructura dinámica.

- La aplicación deberá mostrar, además de tokens y errores léxicos, los contenidos de La Tabla de Símbolos.

  Esta información puede mostrarse en una interfaz o en archivos de texto generados con este contenido.

Trabajo Práctico 1- Especificaciones

Se sugiere la implementación de un **consumidor de tokens** que invoque al Analizador Léxico solicitándole tokens. En el trabajo práctico 2, esta funcionalidad estará a cargo del Analizador Sintáctico.

Trabajo Práctico 1- Consideraciones

- **Para aquellos grupos que tienen asignados tipos de datos que pueden llevar signo, la distinción del uso del símbolo '-' como operador aritmético o signo de una constante, se postergará hasta el trabajo práctico Nro. 2.**

Trabajo Práctico 1- Entrega

- La forma de entrega (correo electrónico, medio físico, etc.) se pactará con el docente asignado al grupo.

- El material entregado debe incluir:
  - Ejecutable del compilador (**debe poder ejecutarse en una máquina virtual que será provista por la cátedra**)
  - Código fuente completo del compilador
  - Casos de prueba
  - Informe

Temas particulares

- **Enteros**: Constantes enteras con valores entre -2^{15} y $2^{15} - 1$.

  Se debe incorporar a la lista de palabras reservadas la palabra **INT**.

- **Enteros sin signo:** Constantes enteras con valores entre 0 y $2^{16} - 1$.

  Se debe incorporar a la lista de palabras reservadas la palabra **UINT**.

- **Enteros largos:** Constantes enteras con valores entre -2^{31} y $2^{31} - 1$.

  Se debe incorporar a la lista de palabras reservadas la palabra **LONG**.

- **Enteros largos sin signo:** Constantes enteras con valores entre 0 y $2^{32} - 1$.

  Se debe incorporar a la lista de palabras reservadas la palabra **ULONG**.

Temas particulares

- **Flotantes:** Números reales con signo y parte exponencial. El exponente comienza con la letra E (mayúscula o minúscula) y puede tener signo. La ausencia de signo implica positivo. La parte exponencial puede estar ausente. El símbolo decimal es la coma "**,**".

  Ejemplos válidos: 1. ,6 -1,2 3,e–5 2,E+34 2,5E1 15, 0,

  Considerar el rango 1,17549435E-38 < x< 3,40282347E38 (incluir el 0.0)

  Se debe incorporar a la lista de palabras reservadas la palabra **FLOAT**.

- **Dobles:** Números reales con signo y parte exponencial. El exponente comienza con la letra E (mayúscula o minúscula) y puede tener signo. La ausencia de signo implica positivo. La parte exponencial puede estar ausente. El símbolo decimal es la coma "**,**".

  Ejemplos válidos: 1. ,6 -1,2 3,E–5 2,e+34 2,5E1 13, 0,

  Considerar rango 2,2250738585072014E-308 < x < 1,7976931348623157E308 (incluir el 0.0)

  Se debe incorporar a la lista de palabras reservadas la palabra **DOUBLE**.

Temas particulares

- Incorporar a la lista de palabras reservadas las palabras **WHILE y DO**.
- Incorporar a la lista de palabras reservadas las palabras **DO** y **UNTIL**.
- Incorporar a la lista de palabras reservadas las palabras **FROM, TO** y **BY**.
- Incorporar a la lista de palabras reservadas las palabras **SWITCH** y **CASE**.
- Incorporar a la lista de palabras reservadas la palabra **LET**
- Se definirá en el trabajo práctico 3.
- Incorporar a la lista de palabras reservadas las palabras **FUNCTION, RETURN y MOVE**.
- Se definirá en el trabajo práctico 2
- Se definirá en el trabajo práctico 2
- Se definirá en el trabajo práctico 2

Temas particulares

- **Comentarios de 1 línea:** Comentarios que comiencen con "**" y terminen con el fin de línea.
- **Comentarios multilínea:** Comentarios que comiencen con "**{** " y terminen con "**}**" (estos comentarios pueden ocupar más de una línea).
- **Cadenas de 1 línea:** Cadenas de caracteres que comiencen y terminen con " **'** " (estas cadenas no pueden ocupar más de una línea).
- **Cadenas multilínea:** Cadenas de caracteres que comiencen con y terminen con " **"** " . Estas cadenas pueden ocupar más de una línea, y en dicho caso, al final de cada línea, excepto la última, deben aparecer 3 puntos suspensivos " **…** ". (En la Tabla de símbolos se guardará la cadena sin los puntos suspensivos, y sin el salto de línea.