Introducción a los compiladores

William Cruz-Santos
wdelacruzd@uaemex.mx

Ingeniería en Computación
Universidad Autónoma del Estado de México
Unidad de Competencia I

CENTRO UNIVERSITARIO
XICO
VALLE DE CHALCO

2 de octubre de 2015

Agenda

1. Acerca del curso
2. Breve historia de los lenguajes de programación
3. Fases de un compilador
4. Análisis léxico

Agenda

1. Acerca del curso
2. Breve historia de los lenguajes de programación
3. Fases de un compilador
4. Análisis léxico

Lineamientos del curso

Horario:

- Martes de 12:00-14:30
- Jueves de 12:00-14:30

Evaluación:

- Tareas 20 %
- Exámenes 40 %
- Proyecto 40 %

Bibliografía

- *El libro del dragón púrpura*
- Aho, Lam, Sethi & Ullman
- Existen 9 ejemplares en el acervo de la biblioteca

Estructura del curso

- El curso contempla aspectos teóricos y prácticos
- Se requiere experiencia en lenguajes de programación
- Trabajos escritos (teoría)
- Trabajos de programación (práctica)

Honestidad académica

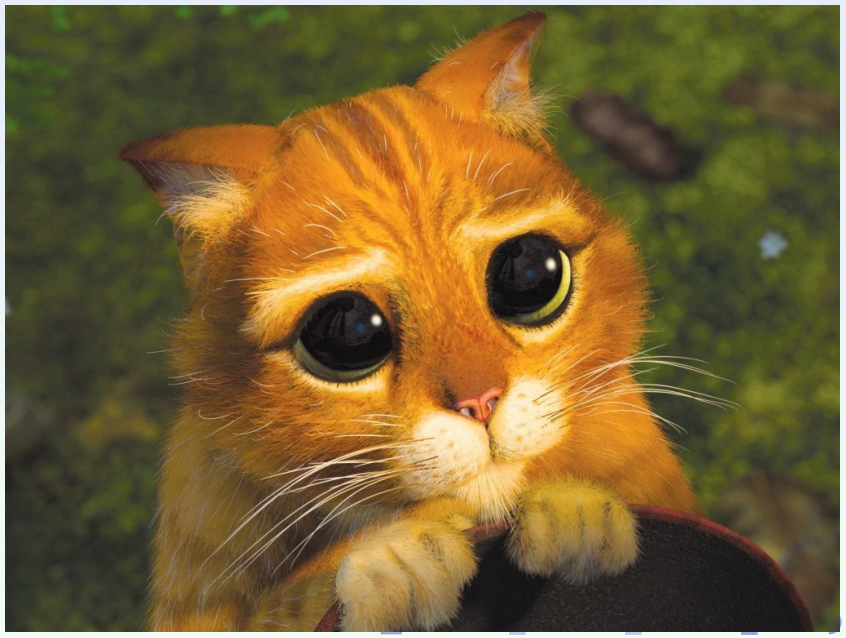
- No use códigos de internet
- No esta prohibida la colaboración pero no se recomienda
- Se penalizará enérgicamente cualquier incidente

Honestidad académica

- No use códigos de internet
- No esta prohibida la colaboración pero no se recomienda
- Se penalizará enérgicamente cualquier incidente

Honestidad académica

- No use códigos de internet
- No esta prohibida la colaboración pero no se recomienda
- Se penalizará enérgicamente cualquier incidente



Agenda

1. Acerca del curso
2. Breve historia de los lenguajes de programación
3. Fases de un compilador
4. Análisis léxico

¿Cómo se implementan los lenguajes?

1. Dos estrategias principales:
   - Interpretados
   - Compilados
2. Los interpretes ejecutan programas pero no generan código ejecutable
3. Los compiladores realizan pre-procesamiento del programa fuente

Historia de los lenguajes de alto nivel

- IBM desarrolla la computadora 704 en 1954
- El costo del software excede el costo del hardware
- La programación se realizó en ensamblador

Figura: Computadora IBM 704 en la NASA (Wikipedia).

Plankalkül

- 1942-45, Konrad Zuse
- Para programar la computadora Z4
- Características: asignaciones, if's, ciclos

Fortran

- 1954-57, J. Backus (*premio Turing 1977*)
- Computación numérica
- Características: modular, compilado

Cobol

- COBOL 1960
- Usado para negocios
- Características: estructuras, archivos, macros

Lisp

- 1960, J. McCarthy (*premio Turing 1971*)
- Computación simbólica e interpretado
- Características: administración de memoria, listas

BASIC

- 1964, J. Kemeny y T. Kurz
- Educativo e interactivo
- Características: estructuras, arreglos, sintaxis sencilla

Pascal

- 1971, Niklaus Wirth (*premio Turing 1984*)
- Educativo y de propósito general
- Características: estructurado, orientado a objetos

C

- 1972, Dennis Ritchie (*premio Turing 1983*)
- Programación de sistemas
- Características: estructurado, usado ampliamente

C++

- 1984, Bjarne Stroustrup
- Lenguaje de propósito general
- Características: estructurado, orientado a objetos, sobrecarga de operadores

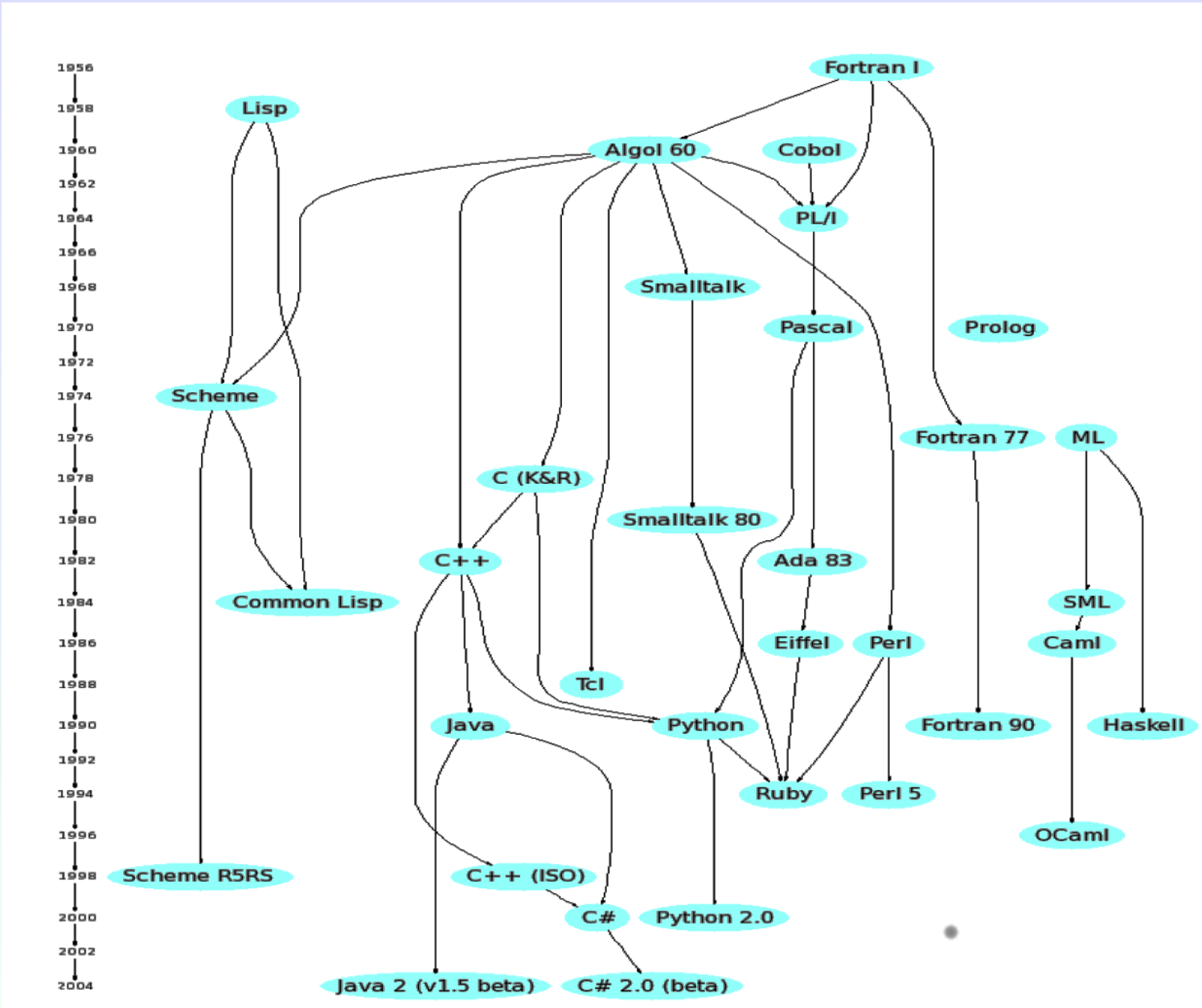
Python

- 1991, G. van Rossum
- Lenguaje interpretado

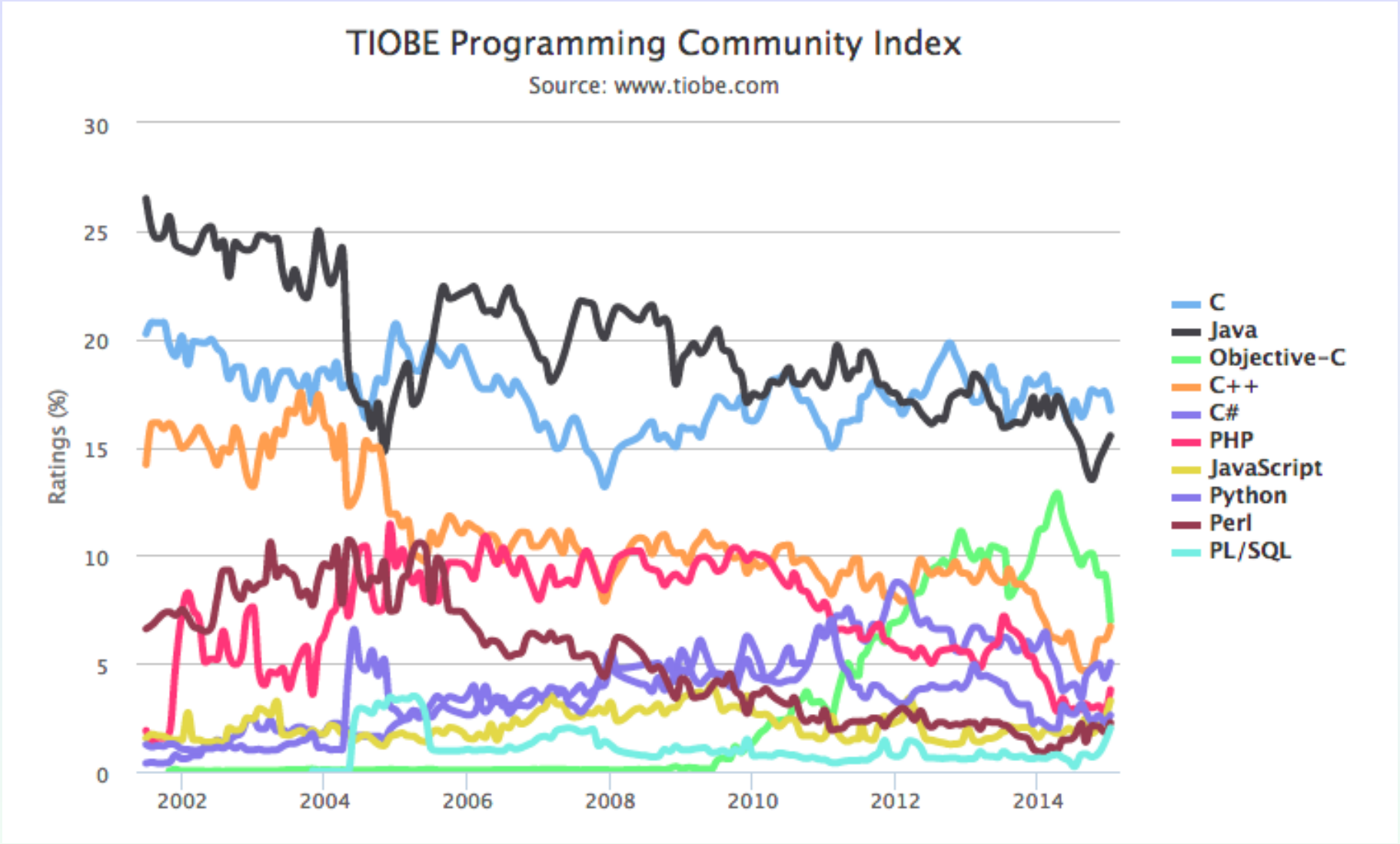
Java

- 1995, Sun Microsystems
- Lenguaje de propósito general
- Basado en C++

Cronología (*http://rigaux.org/*)



Ranking lenguajes de programación



Agenda

1. Acerca del curso
2. Breve historia de los lenguajes de programación
3. Fases de un compilador
4. Análisis léxico

Etapas de un compilador

1. Análisis léxico
2. Análisis sintáctico (*parsing*)
3. Análisis semántico
4. Generación de código intermedio
5. Optimización de código

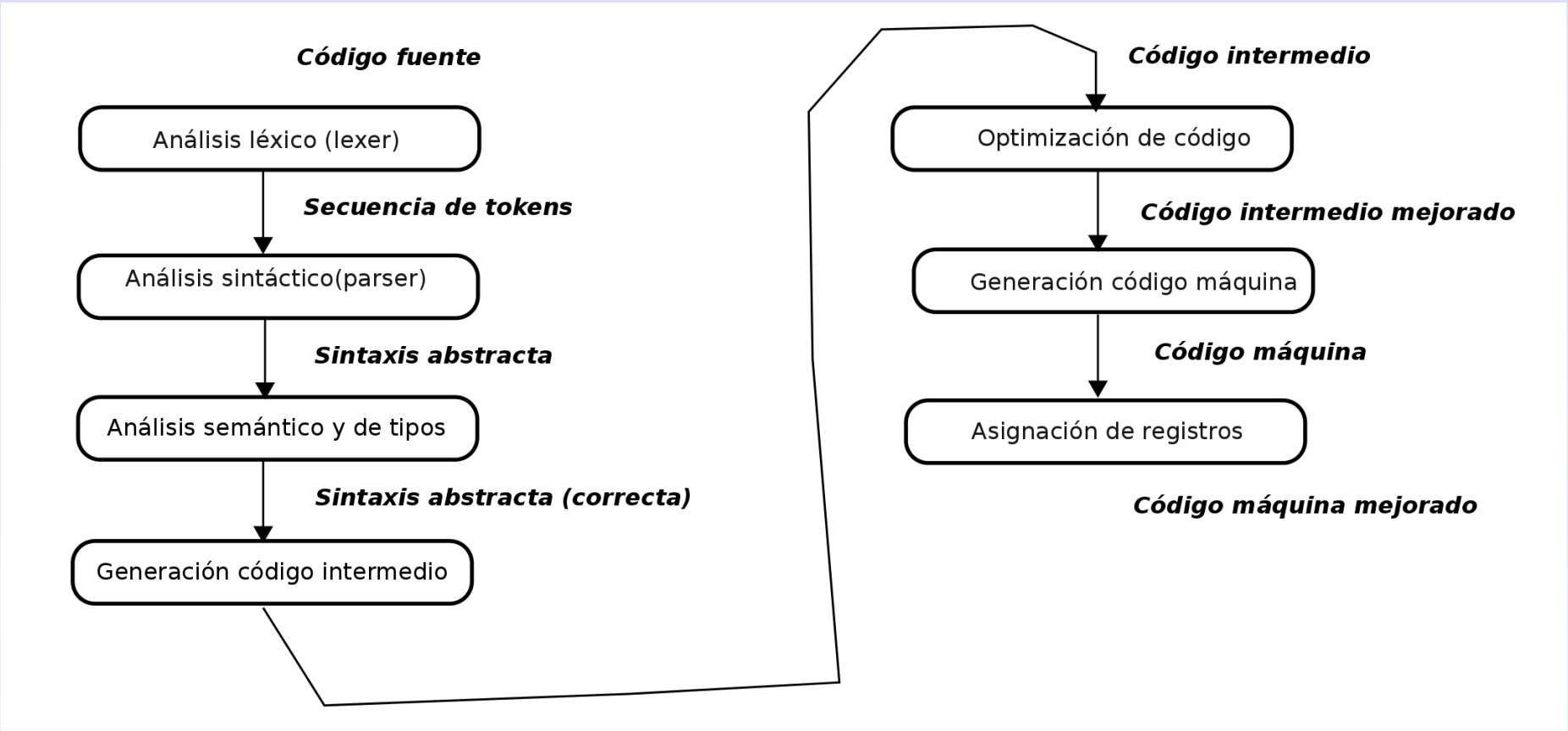


Figura: Fases de un compilador.

Agenda

1. Acerca del curso
2. Breve historia de los lenguajes de programación
3. Fases de un compilador
4. Análisis léxico

Análisis léxico

El *analizador léxico* se encarga de leer los caracteres de entrada del program fuente, los agrupa en *lexemas* y produce como salida una secuencia de *tokens* para cada lexema en el program fuente. El flujo de tokens se envía al *analizador sintáctico* para su análisis.

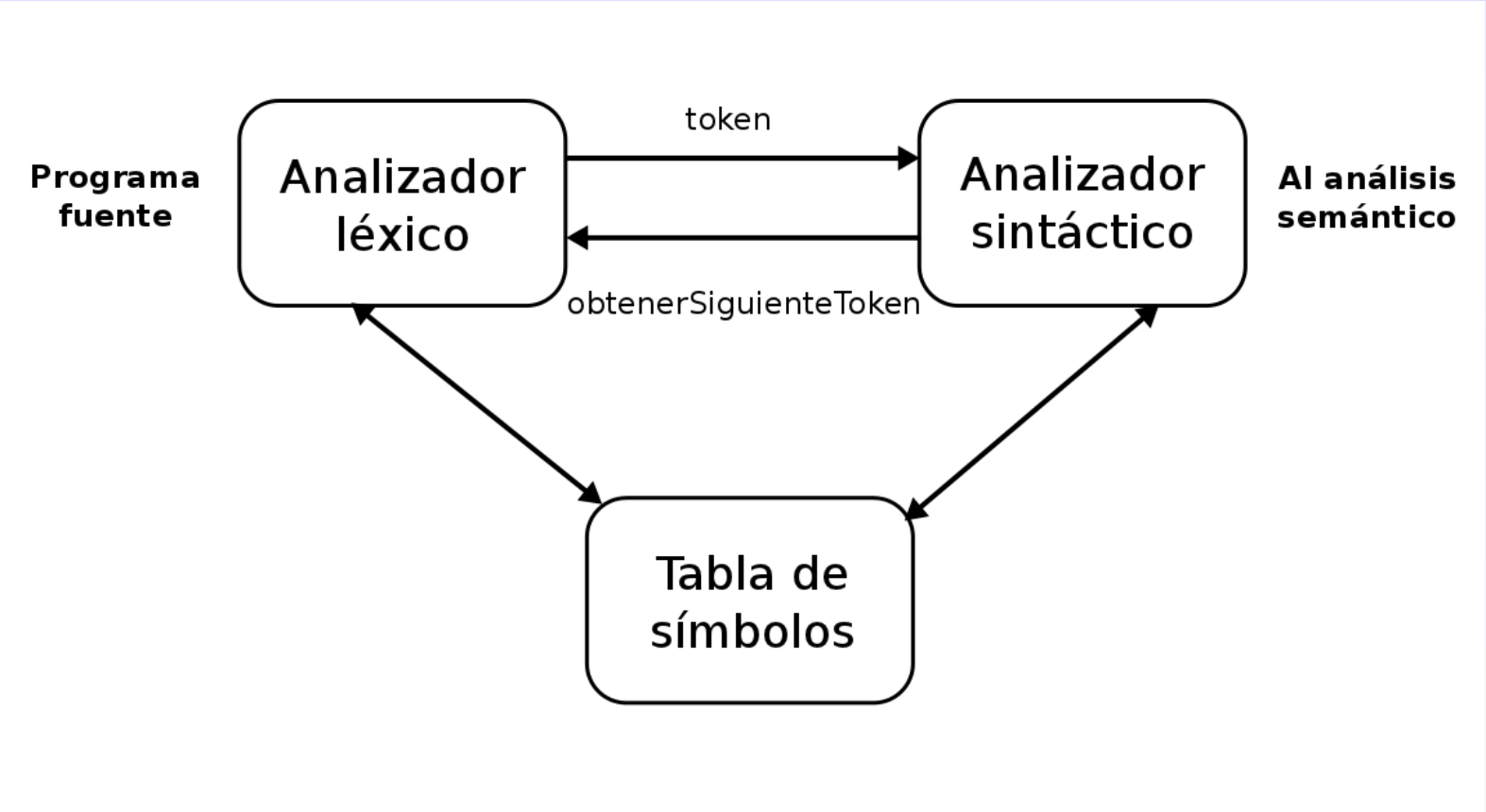


Figura: Interacciones entre el analizador léxico y el analizador sintáctivo.

Tareas de un analizador léxico

- **Escaneo**: consiste en los procesos simples que no requieren la determinación de tokens de la entrada, como la eliminación de comentarios y la compactación de los caracteres de espacio en blanco consecutivos en uno solo.

- **Análisis léxico**: consiste en producir la secuencia de tokens.

Tokens, patrones y lexemas

- **Token**: un token es un par que consiste en un nombre de token y un valor atributo opcional.
- **Patrón**: es una descripción de la forma que pueden tomar los lexemas de un token.
- **Lexema**: es una secuencia de caracteres en el programa fuente, que coinciden con el patrón para un token y que el analizador léxico identifica como una instancia de ese token.

La siguiente tabla muestra algunos ejemplos de tokens:

Token	Descripción informal	Lexemas de ejemplo
if	caracteres **i**, **f**	**if**
else	caracteres **e**,**l**,**s**,**e**	**Else**
comparacion	$<$ o $>$ o $<=$ o $>=$ o $==$ o $!=$	$<=$, $!=$
id	letra seguida por letras y dígitos	**pi**, **puntuacion**, **D2**
numero	cualquier constante numérica	3.14159, 0, 6.02e23
literal	cualquier cosa excepto “ "	“core dumped"

Considere la siguiente instrucción en lenguaje C:

```
printf("Total = %d\n",puntuacion);
```

1. `printf, puntuacion`
   son lexemas correspondientes con el token **id**

2. `"Total = %d\n"`
   es un lexema que coincide con el token **literal**

Clasificación de los tokens:

- Tokens para las palabras claves: **for**, **while**, **if**, **else**, **struct**
- Tokens para los operadores.
- Un token que representa a todos los identificadores.
- Tokens para las constantes, números y cadenas.
- Tokens para los signos de puntuación como paréntesis, coma, y signos.

Considere la siguiente sentencia en Fortran:

```
E = M * C ** 2
```

donde sus tokens y atributos son los siguientes:

<**id**, apuntador a la entrada en la tabla de símbolos para **E** >

<**asigna-op**>

<**id**, apuntador a la entrada en la tabla de símbolos para **M**>

<**mult-op**>

<**id**, apuntador a la entrada en la tabla de símbolos para **C**>

<**exp-op**>

<**numero**, valor entero 2>

Ejercicio 1: Divida el siguiente programa en C++:

```
float cuadroLimitado(x) float x {
          /*devuelve x al cuadrado, pero nunca mas de 100*/
return (x<=-10.0||x>=10.0)?100:x*x;
}
```

en lexemas apropiados.

Ejercicio 2: Divida el siguiente programa en HTML:

```
He aquí una foto de <B>mi casa</B>:
<P><IMG SRC = "casa.gif"><BR>
Vea <A HREF = "masImgs.html">Más imágenes</A> si
le gustó ésa.<P>
```

en lexemas apropiados.

Comentarios finales

- La primera fase de un compilador es el análisis léxico, de tal manera que sea posible identificar los componentes principales en un lenguaje como los tokens.

- En las fases posteriores como la semántica se evaluán las expresiones con respecto a la gramática identificada en el análisis léxico.

- ¿Preguntas?