Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

ANÁLISIS LÉXICO

CONCEPTOS BÁSICOS DEL ANÁLISIS LÉXICO

© Todos los derechos de propiedad intelectual de esta obra pertenecen en exclusiva a la Universidad Europea de Madrid, S.L.U. Queda terminantemente prohibida la reproducción, puesta a disposición del público y en general cualquier otra forma de explotación de toda o parte de la misma.

La utilización no autorizada de esta obra, así como los perjuicios ocasionados en los derechos de propiedad intelectual e industrial de la Universidad Europea de Madrid, S.L.U., darán lugar al ejercicio de las acciones que legalmente le correspondan y, en su caso, a las responsabilidades que de dicho ejercicio se deriven.

Índice

© Universidad Europea de Madrid. Todos los derechos reservados.

Presentación

El objetivo de este tema es entender las tareas que realiza el analizador léxico y cómo las realiza para familiarizarse con él y ser capaces de diseñar esta funcionalidad si fuera necesario.

Aprenderemos a diseñar y sabremos cómo se implementa un analizador léxico.

Los conceptos a trabajar en este tema son:

- El análisis léxico.
- Funciones del analizador léxico y ventajas.
- Definiciones básicas.
- ¿Cómo funciona el analizador léxico?
- Diseño de un analizador léxico.
- Reconocimiento de identificadores.
- Errores léxicos.

Este componente de un compilador se puede utilizar de forma independiente para tratar ficheros en los que sea necesario que se ejecute una acción a partir del reconocimiento de unos símbolos determinados, sin que sea necesario tener en cuenta lo que hay alrededor del símbolo o símbolos que nos interesan.

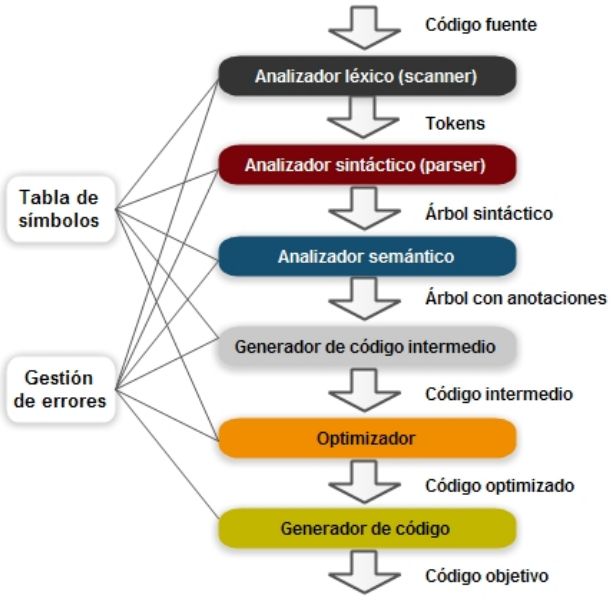
Así, se puede por tanto variar la composición y la estructura del fichero que estamos tratando y nuestro programa seguiría funcionando.

© Universidad Europea de Madrid. Todos los derechos reservados.

El análisis léxico

En este tema vamos a proceder a definir y comprender todas las **tareas que realiza el analizador léxico y que son clave** para el correcto funcionamiento del compilador.

Como vemos en la figura, tiene como entrada el código fuente del lenguaje de programación que acepta el compilador y como salida, proporciona al analizador sintáctico los tokens.



¿Qué es un token?

Es una agrupación de caracteres reconocidos por el analizador léxico que constituyen los símbolos con los que se forman las sentencias del lenguaje.

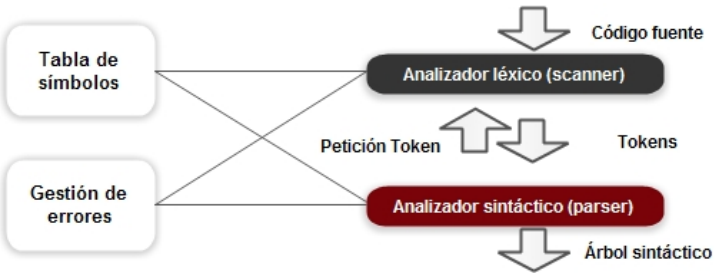
Lo que el analizador léxico devuelve al analizador sintáctico es el **nombre de ese símbolo junto con el valor del atributo** (si ese token lo necesita, ya que no todos los tokens llevan atributo, como por ejemplo: una palabra reservada "if").

© Universidad Europea de Madrid. Todos los derechos reservados.

El análisis léxico

Si observamos con más detalle lo que pasa en el analizador léxico, en su relación con el analizador sintáctico y con la tabla de símbolos, vemos que una vez empieza a leer el código fuente y reconoce el primer token, se lo envía al analizador sintáctico y este, en cuanto lo recibe, le pide el siguiente token para que siga reconociendo la entrada. Por tanto, los tokens son enviados al analizador sintáctico **bajo demanda.**

> Esta forma de funcionar se denomina **"dirigida por el analizador sintáctico"** (en inglés, *parser driven*).



Por otro lado, si reconoce un identificador lo almacena en la tabla de símbolos, y posteriormente, si el analizador sintáctico reconoce que ese identificador lleva asociada **información de tipo** (entero, real, etc.) o de valor, también añade esta información a la mencionada tabla.

En cuanto al sistema de gestión de errores, se encarga de **detectar símbolos que no pertenezcan a la gramática** porque no encajen con ningún patrón. Bien porque haya caracteres inválidos, ejemplo @, o bien porque se escriban mal las palabras reservadas del lenguaje, los identificadores o los números, como 5.25 en lugar de 5,25, pudiendo simplemente no hacer nada o bien informar del tipo de error que se ha cometido. Se puede **minimizar el número de errores** borrando caracteres inválidos, insertando el carácter que falta o remplazando un carácter por otro según sea el caso.

© Universidad Europea de Madrid. Todos los derechos reservados.

Funciones del analizador léxico y sus ventajas

El analizador léxico realiza varias funciones, siendo la fundamental la de **agrupar los caracteres** que va leyendo uno a uno del programa fuente y formar los tokens. Las otras funciones que realiza el analizador léxico son:

© Universidad Europea de Madrid. Todos los derechos reservados.

Ventajas de contar con un analizador léxico:

1ª ventaja	Se simplifica el diseño, puesto que hay una herramienta especializada en el tratamiento del fichero que contiene el código fuente.

© Universidad Europea de Madrid. Todos los derechos reservados.

2ª ventaja	Aumenta la portabilidad del compilador, pudiendo tenerse versiones diferentes para distintos formatos del texto de código fuente (ASCII, EBCDIC, etc.).
3ª ventaja	Mejora la eficiencia al ser una herramienta especializada en el tratamiento de caracteres.
4ª ventaja	Detección de determinados errores fáciles de corregir a este nivel (5.25 por 5,25).

© Universidad Europea de Madrid. Todos los derechos reservados.

Definiciones básicas

Para entender el analizador léxico debemos conocer una serie de conceptos y sus definiciones.

Estos conceptos son:

Token	Como hemos visto anteriormente, es una agrupación de caracteres reconocidos por el analizador léxico que constituyen los símbolos con los que se forman las sentencias del lenguaje y también se les denomina **componentes léxicos**. Constituyen los símbolos terminales de la gramática: ● Palabras reservadas. ● Identificadores. ● Operadores y constantes. ● Símbolos de puntuación y especiales.
Lexema	Es la secuencia de caracteres, ya agrupados, que coinciden con un determinado token, como por ejemplo el nombre de un identificador, o el valor de un número. Un token puede tener uno o infinitos lexemas. Por ejemplo, las palabras reservadas tienen un solo lexema, mientras que los identificadores o los números tienen infinitos.
Patrón	Es la forma de describir los tipos de lexema. Esto se realiza utilizando expresiones regulares.

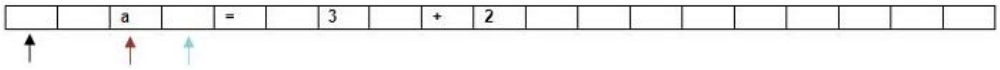
Ejemplos

Token (Componente léxico)	Lexema	Patrón
While	While	While
Suma	+	+
Identificador	a, valor, b	[a-zA-Z]+
Número	5, 3, 25, 56	[0-9]+(\.[0-9]+)?

© Universidad Europea de Madrid. Todos los derechos reservados.

¿Cómo funciona el analizador léxico?

Como vimos en el segundo apartado, el analizador léxico funciona **bajo demanda del analizador sintáctico** cuando le pide el siguiente token. A partir de ese fichero que contiene el código fuente vamos leyendo caracteres que almacenamos en un buffer de entrada. Supongamos el siguiente texto:



Si comenzamos donde está la flecha, según accedemos al fichero, tenemos que eliminar dos caracteres en blanco y posicionarnos en a y leemos ese carácter, devolver ese token, cuando veamos el siguiente espacio en blanco (flecha azul), ver si encaja con alguno de los patrones que hemos definido mediante las expresiones regulares y si es así, pasárselo al analizador sintáctico, que automáticamente nos pedirá que reconozcamos el siguiente token, en este tras eliminar el analizador léxico y el carácter en blanco (donde se encuentra la flecha azul).

Estas expresiones regulares se construyen mediante un autómata finito, pero a modo de introducción un **autómata finito** es un programa que acepta cadenas de un lenguaje definido sobre un alfabeto, y responde si la cadena pertenece al alfabeto o no. Para definir un autómata de una manera formal, necesitamos cinco elementos:

Expresiones regulares

Representan patrones de cadenas de caracteres.

© Universidad Europea de Madrid. Todos los derechos reservados.

Diseño de un analizador léxico

Lo primero que tenemos que hacer para construir un analizador léxico es **diseñarlo,** pudiendo usarse para ello una tabla o un diagrama de transición que representa los estados por los que va pasando el autómata para reconocer un token:

Tabla de transiciones

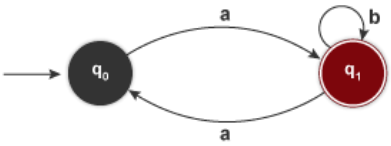
- En las filas colocamos los estados (q) que pertenecen al conjunto de estados Q, en las columnas estarán los símbolos de la entrada (e) y que pertenecen al alfabeto ∑, el estado inicial llevará el siguiente símbolo →, los estados finales llevarán el siguiente símbolo * , y en la posición (fila, columna) tendremos el estado que determina la función f (q, e).

Por tanto el autómata sería:

f	a	b
→ q_0	q_1	q_1
*q_1	q_0	q_1

Diagrama de transiciones

- El estado inicial llevará el símbolo →, en los nodos se mostrarán los estados, los arcos unirán los estados con el símbolo de la entrada, los estados finales tendrán un doble círculo (equivalente al * en la tabla).



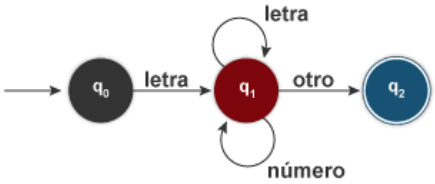
© Universidad Europea de Madrid. Todos los derechos reservados.

Reconocimiento de identificadores

Ahora que sabemos como diseñar un analizador léxico, vamos a ver un ejemplo en el que reconocemos un identificador. Un identificador está formado por al menos una **letra mayúscula o minúscula** (ejemplo: a) seguida de forma opcional por mas letras o números (ej: aa, a1, etc).

Se trata de representar este símbolo mediante un **diagrama de transiciones.**

La forma de representar mediante expresiones regulares cualquier letra mayúscula o minúscula es: **[a-z][A-Z]** y le denominamos **letra**. La forma de representar un número cualquiera es **[0-9]** y le denominamos **número**. Para finalizar se define como **[otro]** cualquier otro símbolo, indicando que ya ha terminado de definirse el identificador. Esto lo representamos mediante un diagrama de transiciones de la siguiente manera:



Empezamos en el estado inicial q_0, y estamos leyendo un fichero con código fuente donde tenemos escrito a2 = b + 5. Empezamos reconociendo una letra, con lo que pasamos del estado q_0, al estado q_1 (hemos reconocido a, de la variable a2). Leemos el siguiente carácter, con lo que entra el 2, y nos quedamos en el mismo estado que ya estábamos q_1 (ya hemos reconocido a2) que todavía no es un estado final.

Leemos el espacio en blanco o bien el signo =, si el analizador elimina los espacios en blanco, por tanto el siguiente carácter es distinto de una letra o un número, con lo que saltamos al estado q_2, que este si es un estado final, y ya se ha reconocido a2, pasándose el identificador al analizador sintáctico, que nada mas recibirlo solicitará otro token al analizador léxico.

© Universidad Europea de Madrid. Todos los derechos reservados.

Implementación de un analizador léxico

Hay varias formas de implementar un analizador léxico:

- **Utilizando un generador de analizadores léxicos**: son herramientas que a partir de las expresiones regulares generan un programa que permite reconocer los tokens o componentes léxicos.

  Estos programas suelen estar escritos en C, donde una de las herramientas es FLEX, o pueden estar escritos en Java, donde las herramientas pueden ser JFLEX o JLEX.

Ventajas	Inconvenientes
Comodidad y rapidez de desarrollo.	Estos programas son ineficientes y dificultad de mantenimiento del código generado.

- **Utilizando un lenguaje de alto nivel**: a partir del diagrama de transiciones y del pseudocódigo correspondiente se programa el analizador léxico (véase un ejemplo en Louden, 2004).

Ventajas	Inconvenientes
Eficiente y compacto (lo que facilita el mantenimiento).	Hay que realizar todo a mano.

- **Utilizando un lenguaje de bajo nivel (ensamblador):**

Ventajas	Inconvenientes
Más eficiente y compacto.	Más difícil de desarrollar.

© Universidad Europea de Madrid. Todos los derechos reservados.

Errores léxicos

Los **errores léxicos** son detectados, cuando durante el proceso de reconocimiento de caracteres, los símbolos que tenemos en la entrada no concuerdan con ningún patrón. Hay que tener en cuenta que hay pocos errores detectables por el analizador léxico, entre ellos están:

Nombres incorrectos de los identificadores	Se debe a que se utilizan caracteres inválidos para ese patrón, como por ejemplo un paréntesis, o se empieza por un número.
Números incorrectos	Debido a que está escrito con caracteres inválidos (puntos en lugar de comas) o no está escrito correctamente.
Palabras reservadas escritas incorrectamente	Se producen errores de ortografía. El problema aquí es cómo distingues entre un identificador y una variable reservada.
Caracteres que no pertenecen al alfabeto del lenguaje	Ejemplos: @, €, ¿, ?, ñ, etc.

La estrategia de recuperación de errores más sencilla es la recuperación en "modo pánico".

El intentar transformaciones para reparar la entrada es muy costoso. Otra estrategia muy aplicada es la siguiente:

- Si cuando se detecta el error ya hemos pasado por un estado de aceptación, se ejecuta la acción correspondiente al estado de aceptación por el que se acaba de pasar y el resto de caracteres se devuelven a la entrada y nos posicionamos en el estado inicial, para empezar a partir de ahí a reconocer el siguiente token.
- Si no hemos pasado por ningún estado de aceptación, se elimina el carácter que no concuerda y aceptamos el siguiente.

Ejemplo -25.j

© Universidad Europea de Madrid. Todos los derechos reservados.

Modo pánico

Básicamente consiste en borrar de forma sucesiva caracteres de la entrada hasta que el analizador léxico es capaz de encontrar un token bien formado.

¿Cuál sería el proceso para tratar el error?

- Anotar el error y el estado.
- Recuperarse. Tenemos varias alternativas: Ignorar, borrar, insertar, remplazar o conmutar.
- Seguir.

Intentar transformaciones

La mayoría de los analizadores que implementan esta estrategia la aplican cuando se corrige el lexema con una sola transformación.

Ejemplo

-25,j

- Eliminamos los dos últimos caracteres quedándonos con *-25*, quedándonos en un estado de aceptación.
- Devolvemos a la entrada *.j*
- Nos poscionamos en el estado inicial.
- En la siguiente lectura tendremos otro error teniendo que descartar *la coma*, volviendo al estado inicial.
- En la siguiente lectura se reconoce el identificador *j*.

© Universidad Europea de Madrid. Todos los derechos reservados.

Resumen

En este tema hemos entendido como funciona un analizador léxico y cómo se relaciona con el analizador sintáctico y con otras estructuras necesarias, como la tabla de símbolos. Para hacernos una idea más completa hemos visto todas las funciones que debe llevar a cabo el analizador léxico con el fichero de entrada que contiene el código fuente. Además, hemos conocido que ventajas aporta a un compilador.

Más adelante hemos comprendido cuáles son los conceptos básicos (token o componente léxico, lexema y patrón), donde hemos aprendido que un token puede tener uno (palabra reservada u operadores) o infinitos (identificadores o números) lexemas.

Por otro lado, hemos entendido cómo funciona el analizador léxico y cómo se construye a partir de una expresión regular el autómata finito que lo soporta. También se ha conocido cómo se diseña un analizador léxico por medio de una tabla o un diagrama de transiciones, representando de esta forma los estados por los que pasa el analizador para reconocer un token.

Posteriormente, hemos aprendido mediante un ejemplo a reconocer un identificador, y esto puede extenderse a los números enteros, números decimales, operadores, comentarios, etc. Además, se han identificado tres formas de implementar estos ejemplos: generador de analizadores léxicos, lenguaje de alto nivel y lenguaje de bajo nivel.

Para finalizar, hemos visto los errores léxicos que se pueden detectar y la forma de tratarlos para así eliminar esos errores o minimizar su impacto en el resto del compilador.

© Universidad Europea de Madrid. Todos los derechos reservados.