Análisis Léxico

El **análisis léxico** es la primera etapa del proceso de compilación. Su objetivo es leer el código fuente carácter por carácter, identificar las unidades básicas llamadas **lexemas** y producir una secuencia de **tokens** que serán utilizados por el analizador sintáctico.

Dos métodos para construir un analizador léxico (escáner)

Existen dos formas principales de crear un analizador léxico:

1. **De forma manual:**
   Se parte de un diagrama que representa cómo se ven los lexemas. Luego, se escribe el código que sigue este diagrama para reconocer cada lexema y devolver el token correspondiente, junto con información adicional si es necesario.

2. **Usando un generador de léxico:**
   En este método se describen los patrones de los lexemas y se introducen en un programa que genera automáticamente el escáner.
   Los generadores más conocidos son **Lex** y su versión moderna **Flex**.

Aunque el uso de generadores facilita la construcción, los analizadores léxicos hechos **a mano** suelen tener **mayor velocidad** y **mejor manejo de errores**, por lo que los compiladores profesionales suelen implementarlos manualmente.

El papel del analizador léxico

El **analizador sintáctico (parser)** llama al **analizador léxico (lexer)** cada vez que necesita el siguiente token.
El lexer también realiza tareas adicionales como **eliminar espacios en blanco y comentarios**.

Una vez que el análisis léxico termina, el compilador **ya no trabaja con caracteres individuales**, sino con **tokens**, que son las unidades significativas del programa fuente.

Análisis léxico vs. análisis sintáctico

Separar el análisis léxico del sintáctico se hace por razones de **ingeniería de software**:

1. **Simplicidad de diseño:** cada tarea tiene una función clara y modular.
2. **Eficiencia:** al estar separado, se pueden aplicar técnicas específicas que optimizan el proceso.
3. **Portabilidad:** el lexer es el único módulo que se comunica directamente con el texto fuente, facilitando la adaptación a otros entornos.

Tokens, patrones y lexemas

- **Token:** es un par `<nombre, atributo>`. El analizador sintáctico usa estos tokens.
  El atributo puede contener datos adicionales, como un valor numérico o una referencia a una tabla.
- **Patrón:** describe las cadenas de caracteres que forman los lexemas de un token.
  Ejemplo: una letra seguida de letras o dígitos.
- **Lexema:** es la secuencia real de caracteres en el código fuente que coincide con el patrón de un token.

Clases comunes de tokens:

1. Palabras clave (ej. `if`, `while`).
2. Operadores o clases de operadores (`+`, `-`, `==`).

3. Identificadores (nombres de variables o tipos).
4. Constantes (numéricas, reales, cadenas).
5. Símbolos de puntuación (`;`, `,`, `(`, `)`).

Atributos para tokens

Los tokens pueden tener **atributos adicionales** para dar más información.
Por ejemplo:

- Los tokens de **palabras clave** no los necesitan, porque su nombre los identifica completamente.
- Los tokens de **constantes** sí los requieren, para guardar su **valor numérico o literal**.
- Los **identificadores** necesitan un atributo que indique su posición en la **tabla de símbolos**.

El analizador léxico no construye directamente la tabla de símbolos, pero sí genera pares `<lexema, token>` que luego serán utilizados por el analizador sintáctico para formar una tabla más completa.

Errores léxicos

A veces el lexer no encuentra un patrón que coincida con la entrada y se **detiene**.
La acción más simple es **abortar la compilación** mostrando el **número de línea y posición del error**.

Sin embargo, es preferible que el compilador **intente recuperarse**, por ejemplo:

- Saltando hasta el final de una declaración (por ejemplo, después de un `;`).
- Haciendo pequeñas correcciones para poder continuar con el análisis.