

Juan Antonio Pérez Álvarez

DATE: 19 - Feb - 2025

Traductores de Lenguajes II - Julio Esteban Valdez López

## Capítulo 2.1 - El análisis Léxico.

Es la primera fase de un compilador y tiene como objetivo leer el código fuente y convertirlo en una secuencia de tokens que luego serán utilizados por el analizador sintáctico.

### Funciones del Analizador Léxico:

- 1.- Eliminar espacios en blanco y comentarios para que el compilador solo procese información relevante.
- 2.- Reconocer las palabras Clave, identificadores, operadores y simbolos del lenguaje de programación.
- 3.- Clasificar tokens según su tipo:

Palabras Clave como if, While.

Id's como x "y"

Operadores como + " y - "

Ejemplo: → int x = 10;

El analizador Léxico generará los siguientes tokens.

int → Palabra Clave

x → identificador

= → Operador de Asignación

10 → Número entero

; → Punto y coma.

Relación con el analizador sintáctico:

19-01-2025 Juan Antonio Pérez Juarez

Traductores de Lenguajes II - Tito Esteban Valdez Lopez

El analizador léxico simplifica la estructura del código para que el analizador sintáctico pueda trabajar con tokens en lugar de caracteres individuales.

También ayuda a manejar errores tempranos en el código:

Herramientas para el Análisis Léxico:

Algunos compiladores usan herramientas como Lex o Flex para generar analizadores léxicos automáticos a partir de expresiones regulares que definen los Tokens.

Capítulo 2.2 - Tokens, patrones y Lexemas

Tokens: Son las unidades básicas reconocidas por el analizador léxico. Representan categorías de elementos del lenguaje, como:

- Palabras Clave
- Identificadores
- Operadores
- Símbolos de Puntuación

Patrones: Cada Token tiene un patrón, que es una descripción de los códigos de caracteres que pueden formar parte de esa categoría.

- Los identificadores pueden definirse con una regla: una letra seguida de letras o números

## Traductores de Lenguajes II - Julio Esteban Ibáñez López

- Los números enteros pueden definirse como: uno o más cifras numéricas

Lexemas: Es la secuencia concreta de caracteres en el código fuente que coincide con un patrón y se clasifica en un token.

Ejemplo:  $x = 42;$

$x$  → identificador.

= → Operador de Asignación;

"42" → Número Entero.

;" → Delimitador.

Los tokens, son las categorías generales

Los patrones, describen qué estructura debe tener un lexema para ser reconocido como token.

Los lexemas, son las instancias concretas en el código fuente

## Capítulo 2.3 - Expresiones regulares.

Una expresión regular (ER) es una notación que describe conjuntos de cadenas de texto, permitiendo reconocer lexemas en el código fuente.

Ejemplo: [a-zA-Z][a-zA-Z0-9]\*

Esta ER describe un identificador, es decir una cadena que comienza con una letra y puede contener letras y números

## Operaciones de los LR

Las expresiones regulares utilizan operadores para definir patrones complejos.

### Concatenación:

Si "a" y "b" son expresiones regulares, entonces "ab" significa que "a" debe ser seguido por "b".

### Unión:

Si "a" y "b" son expresiones regulares, entonces  $a|b$  o  $a \cup b$  representan la unión de ambos conjuntos.

### Cerradura de Kleene:

Si "a" es una expresión regular, entonces  $a^*$  representa cero o más repeticiones de "a".

### Cerradura positiva:

Si "a" es una expresión regular, entonces  $a^+$  representa una o más repeticiones de "a".

### Opcional:

Si "a" es una expresión regular, entonces  $a^?$  representa cero o una aparición de "a".

Los expresiones regulares pueden convertirse en automatas finitos que son estructuras matemáticas usadas para reconocer patrones de texto en el código fuente.