# INSTITUTO TECNOLOGICO DE PACHUCA

*"El hombre alimenta el ingenio en contacto con la Ciencia"*

# "TABLA DE TOKENS"

## INGENIERIA EN SISTEMAS COMPUTACIONALES

**NOMBRE DE LA ASIGNATURA**

LENGUAJES Y ATOMATA 1

**INTEGRANTES DEL EQUIPO**

Daniel Flores Bautista

Sergio Enrique Torres Martínez

Víctor Gabriel Gutiérrez Hernández

Mariana Mendoza Gutiérrez

Reyes Hernández Reyes

**PROFESOR DE LA MATERIA**

Ing. Baume Lazcano Rodolfo

**PACHUCA, HIDALGO, 14 DE MAYO DEL 2024**

## Definición del Lenguaje

Antes de construir el analizador léxico, necesitas una comprensión clara del lenguaje que vas a analizar. Esto incluye definir las reglas gramaticales y los elementos que componen tu lenguaje. Algunos componentes comunes incluyen:

**Palabras clave**: Reservadas por el lenguaje (por ejemplo, if, else, while).

**Identificadores:** Nombres de variables, funciones, etc.

**Operadores**: Aritméticos (+, -, *, /), relacionales (==, !=, >, <), lógicos (&&, ||, !).

## Especificación de Tokens

Define los tokens que el analizador léxico reconocerá. Un token es una categoría de lexemas que el analizador léxico identifica y clasifica. Aquí hay una lista típica de tokens:

| Tokens | Ejemplos |
|---|---|
| Palabras reservadas | if, else, while, for |
| Identificador | Nombres de variables y funciones. |
| Operador Aritmético | +, -, *, / |
| Identificador entero | [0-9] |
| Tokens de error | ¿? |
| Comentario: | // comentario de una línea |

## Definición de Patrones Regulares

Este código es un ejemplo de definición de un analizador léxico (lexer) en una herramienta de generación de analizadores léxicos, como JFlex. Un analizador léxico es una parte fundamental de un compilador o intérprete, responsable de dividir el código fuente en tokens, que son las unidades léxicas mínimas con significado.

```
package codigo; // Declaración del paquete "codigo"
import static codigo.Tokens.*; // Importación de los tokens definidos en el archivo Tokens.java
%%
%class Lexer // Declaración del nombre de la clase Lexer
%type Tokens // Declaración del tipo de los tokens
L=[a-zA-Z_]+ // Definición de una expresión regular para identificar letras y guiones bajos como nombres de variables o palabras clave
D=[0-9]+ // Definición de una expresión regular para identificar números enteros
espacio=[*,\t,\r,\n]+ // Definición de una expresión regular para identificar espacios en blanco
%{
    public String lexeme; // Declaración de una variable "lexeme" para almacenar el lexema actual
%}
%%

// Definición de las reglas de tokenización:
int | // Palabra clave 'int'
if | // Palabra clave 'if'
else | // Palabra clave 'else'
while {lexeme=yytext(); return Reservadas;} // Palabra clave 'while'. Se asigna el lexema actual a la variable 'lexeme' y se retorna el

{espacio} {/ignore/} // Espacios en blanco. Se ignoran.
"//".* {/Ignore/} // Comentarios de una línea. Se ignoran.
"=" {return Igual;} // Asignación. Se retorna el token 'Igual'
"+" {return Suma;} // Operador de suma. Se retorna el token 'Suma'
"-" {return Resta;} // Operador de resta. Se retorna el token 'Resta'
"*" {return Multiplicacion;} // Operador de multiplicación. Se retorna el token 'Multiplicacion'
"/" {return Division;} // Operador de división. Se retorna el token 'Division'
{L}({L}|{D})* {lexeme=yytext(); return Identificador;} // Identificador. Se asigna el lexema actual a la variable 'lexeme' y se retorna e
("(-"{D}+")"|{D}+ {lexeme=yytext(); return Numero;} // Número entero. Se asigna el lexema actual a la variable 'lexeme' y se retorna el
. {return ERROR;} // Cualquier otro carácter que no se ajuste a ninguna de las reglas anteriores. Se retorna el token 'ERROR'
```

Este código define un analizador léxico básico para un lenguaje de programación java, capaz de reconocer palabras clave, operadores, identificadores y números enteros, ignorando los espacios en blanco y los comentarios.