# Documentación del Lenguaje

Este es un lenguaje de programación diseñado para ser intuitivo y fácil de usar. Proporciona estructuras básicas de control de flujo, expresiones, funciones y asignaciones, permitiendo escribir código legible y organizado.

## Gramática General

## Estructura de un programa

Un programa en este lenguaje está compuesto por una serie de líneas (“line”) que pueden ser declaraciones, bloques condicionales, o bucles. El programa finaliza con `EOF` (fin del archivo).

```antlr

grammar Simple;

program: line\* EOF;

```

## Declaraciones permitidas

1. Declaración o ejecución de funciones

**- Sintaxis**:`IDENTIFIER '(' [expression (',' expression)\*] ')' ';'`

- **Ejemplo**:

escribir("Hola Mundo");

2. Asignación

- **Sintaxis:** `IDENTIFIER 'asigno' expression;`

- **Ejemplo**:

x asigno 5;

## Bloques condicionales

1. **Condicionales (`si` y `sino`)**

- Sintaxis:

si (expression) {

// Bloque de código

} sino {

// Bloque alternativo

} - Ejemplo:

si (x igualito 5) {

escribir("Es cinco");

} sino {

escribir("No es cinco");

}

Bucles

1. \*\*Bucle (`bucle` o `hasta`)\*\*

- \*\*Sintaxis:\*\*

bucle (expression) {

// Bloque de código

}

- \*\*Ejemplo:\*\*

bucle (x pequenito 10) {

escribir(x);

x asigno x sumita 1;

}

# Operadores

## Operadores Aritméticos

- **Multiplicación**: `estrella`

- **División**: `solecito`

- **Suma**: `sumita`

- **Resta**: `restita`

**Ejemplo**:

y asigno 3 sumita 4 estrella 2;

## Operadores de Comparación

- **Igualdad**: `igualito`

- **Diferencia**: `diferente`

- **Menor que**: `pequenito`

- **Mayor que**: `grandecito`

- **Menor o igual**: `pequeigual`

- **Mayor o igual**: `granigual`

**Ejemplo**:

si (x igualito y) {

escribir("Son iguales");

}

### Operadores Booleanos

- AND: `y`

- OR: `o`

- XOR: `xor`

**Ejemplo**:

si (a y b) {

escribir("Ambos verdaderos");

}

## Operadores Avanzados

- Exponenciación: `superstar`

- Raíz cuadrada: `supersol`

**Ejemplo**:

x asigno 3 superstar 2;

resultado asigno x supersol x;

## Constantes y Tipos de Datos

- **Números enteros:**

INTEGER: [0-9]+;

**Ejemplo:** `123`

- **Números flotantes:**

FLOAT: [0-9]+ '.' [0-9]\*;

**Ejemplo:** `3.14`

-**Cadenas de texto**:

STRING: ('"' ~'"'\* '"') | ('\'' ~'\''\* '\'');

**Ejemplo:** `'Hola'`, `"Mundo"`

- **Booleanos:**

BOOL: 'verdadero' | 'falso';

**Ejemplo:** `verdadero`, `falso`

- **Nulo:**

NULL: 'null'

Ejemplo: `null`

## Bloques de Código

Los bloques de código se definen entre llaves `{}` y pueden contener varias líneas.

**Ejemplo**:

{

x asigno 10;

escribir(x);

}

## Espacios en blanco

El lenguaje ignora los espacios en blanco y saltos de línea.

antlr

WS: [ \t\r\n]+ -> skip;

## Identificadores

Los identificadores pueden ser letras, números y guiones bajos, pero no pueden comenzar con un número.

antlr

IDENTIFIER: [a-zA-Z\_][a-zA-Z0-9\_]\*;

## Implementación

El lenguaje soporta una implementación que permite manejar variables predefinidas, como `PI` y `E`, y funciones predefinidas como `escribir` para imprimir mensajes:

### Variables Predefinidas

- `PI`: Representa el valor de pi.

- `E`: Representa el valor de la constante de Euler.

Funciones Predefinidas

- `escribir`: Imprime un mensaje en la salida.

- Sintaxis: `escribir(expression);`

- Ejemplo:

escribir("Hola Mundo");

## Ejemplo Completo

simple

x asigno 5;

y asigno 10;

si (x pequenito y) {

escribir("x es menor que y");

} sino {

escribir("x no es menor que y");

}

bucle (x pequenito 20) {

escribir(x);

x asigno x sumita 1;

}