



# **Tecnológico Nacional de México**

**Instituto Tecnológico de Veracruz**

EJ24 7J1A - Lenguajes y Autómatas 2

Docente: Ofelia Gutiérrez Giraldi

Proyecto: Fase Semántica del Compilador

Equipo 1 - Integrantes:

Honorio Acosta Ruiz

Laura Espejo Alvarado

Rodrigo Ernesto Loyo García

H. Veracruz, Ver., 19 de febrero del 2024

## Tabla de contenido

ELL (EasyLearningLanguage) _____	4
Propósito _____	4
Origen del nombre ELL: EasyLearningLanguage _____	4
Manual de Usuario _____	5
Conceptos Básicos _____	6
Tipo de datos _____	6
Palabras Clave _____	6
Operadores _____	9
Nombres de las variables en ELL _____	10
Literales en ELL _____	12
Sintaxis _____	13
Indicar el comienzo y fin del programa _____	13
Declaración de variables _____	13
Asignación de variables _____	13
Lectura de variables _____	13
Imprimir _____	14
Matrices en ELL _____	14
Arreglos en ELL _____	14
Funciones en ELL _____	14
Sentencias decisión en ELL _____	15
Sentencias Bucle en ELL _____	17
Tabla de errores _____	19
Manual del Sistema _____	21
Requisitos del sistema _____	21
Descripción del sistema _____	21
Tabla de valores _____	21
Expresiones regulares _____	23
Nombres de las variables en ELL _____	24
Tipo de Datos _____	24
Operadores _____	25

Sintaxis _____	25
Programa principal en ELL _____	25
Declaración de variables en ELL _____	25
Asignación de variables en ELL _____	26
Leer del teclado en ELL _____	26
Imprimir en pantalla en ELL _____	26
Matrices en ELL _____	26
Arreglos en ELL _____	26
Funciones en ELL _____	27
Sentencia Si en ELL _____	27
Sentencia Conforme en ELL _____	27
Sentencia Mientras en ELL _____	28
Sentencia Repetir en ELL _____	28
Sentencia Para en ELL _____	28
Gramáticas del Lenguaje _____	29
Tabla de errores. _____	32

# ELL (EasyLearningLanguage)

## Propósito

EasyLearningLanguage (ELL) se propone a ser un lenguaje de programación tipado y de propósito general que será diseñado para permitir el desarrollo de aplicaciones básicas.

Se centrará en la lógica de programación básica, lo que lo hará adecuado para la enseñanza de conceptos fundamentales como variables, operaciones, condicionales y ciclos. Aunque ELL no tendrá una amplia gama de características avanzadas, será diseñado para ser fácilmente escalable a medida que los usuarios adquieren más habilidades en programación.

ELL será creado con el propósito de ser un lenguaje de programación en español de nivel básico.

## Origen del nombre ELL: EasyLearningLanguage



*Figura 1 - Icono del lenguaje*

El nombre de nuestro lenguaje es ELL, debido a que es un lenguaje de programación orientado a facilitar el aprendizaje de la lógica básica de programación para programadores novatos.

El nombre ELL proviene de las siglas EasyLearningLanguage, lo que en español significa Lenguaje Fácil de Aprender.

## Manual de Usuario

El respectivo manual busca proporcionar una guía clara y completa para nuestro lenguaje que permita a los programadores utilizar el lenguaje de manera efectiva, donde como objetivos tiene los siguientes:

- Facilitar el aprendizaje del lenguaje:

El manual proporciona una introducción clara al lenguaje de programación, explicando los conceptos básicos, sintaxis y características únicas del lenguaje.

- Ayudar a los programadores a detectar y corregir errores:

Incluye información sobre los errores que pueden ocurrir durante la programación con el lenguaje, incluyendo como corregir estos errores.

- Proporcionar información de referencia:

Es una fuente completa de información de referencia para el lenguaje, que incluye una lista detallada de las palabras clave, operadores y tipos de datos; permitiendo a los programadores buscar rápidamente información específica cuando la necesita.

# Conceptos Básicos

## Tipo de datos

Cuando escribimos programas en el lenguaje ELL, necesitamos utilizar diferentes tipos de datos para almacenar información. Los que podemos usar en ELL son los siguientes:

**Entero:** Este tipo de dato representa un número entero.

**Flotante:** El tipo de dato "flotante" se utiliza para representar números con decimales.

**Cadena:** La cadena es un tipo de dato que se utiliza para representar texto. Podemos almacenar palabras, frases o cualquier combinación de caracteres en una cadena.

**Carácter:** El tipo de dato "carácter" se utiliza para representar símbolos individuales, como letras, números o símbolos especiales.

**Booleano:** El tipo de dato "booleano" se utiliza para representar valores de verdadero o falso. Es útil cuando necesitamos evaluar condiciones o tomar decisiones en nuestro programa.

## Palabras Clave

En el lenguaje ELL, existen palabras clave especiales que tienen un significado especial y no pueden ser usadas como nombre de variables u otros identificadores en el código. A continuación, se muestra una lista de ellas con su respectiva descripción:

**Inicio:** Esta palabra clave se utiliza para indicar el comienzo del programa. Marca el punto de partida desde donde se ejecutarán las instrucciones.

**Fin:** Esta palabra clave indica el final del programa. Marca el punto en el que el programa termina su ejecución.

**Establecer:** Con la palabra clave "Establecer", podemos definir una variable en el programa. Una variable es como una caja donde podemos guardar diferentes tipos de información, como números o texto.

**Interpretar:** La palabra clave "Interpretar" se utiliza para leer el valor de una variable. Podemos obtener información del usuario o de otra parte del programa mediante esta instrucción.

**Escribir:** La palabra clave "Escribir" se utiliza para mostrar en la pantalla el valor de una variable o un texto específico. Con esta instrucción, podemos imprimir información para que el usuario la vea.

**Falso y Verdadero:** Estas palabras clave representan valores lógicos. "Falso" se utiliza para indicar algo que es incorrecto o no se cumple, mientras que "Verdadero" indica algo que es correcto o se cumple.

**Para:** La palabra clave "Para" marca el inicio de un ciclo repetitivo llamado bucle "Para". Nos permite ejecutar un bloque de código varias veces, siguiendo una condición y un incremento específicos.

**Hasta que:** La palabra clave "Hasta que" se utiliza en conjunto con el bucle "Para" y establece la condición para terminar el ciclo. Indica la condición que se evalúa antes de cada iteración.

**Con incremento:** La palabra clave "Con incremento" se utiliza en el bucle "Para" para indicar el incremento que se aplicará en cada iteración del ciclo. Por defecto, el incremento es 1, lo que significa que la variable de control del bucle aumentará en 1 después de cada iteración. Sin embargo, con la palabra clave "Con incremento", puedes especificar un valor diferente para el incremento, lo que te permite controlar cómo cambia la variable de control en cada repetición del bucle.

**FinPara:** Esta palabra clave marca el final del bucle "Para". Especifica el punto donde termina la repetición del bloque de código.

**Mientras:** La palabra clave "Mientras" indica el inicio de un bucle repetitivo llamado bucle "Mientras". Nos permite ejecutar un bloque de código siempre que se cumpla una condición específica.

**FinMientras:** La palabra clave "FinMientras" marca el final del bucle "Mientras". Indica el punto donde termina la repetición del bloque de código.

**Repetir:** La palabra clave "Repetir" inicia un bucle repetitivo llamado bucle "Repetir". Nos permite ejecutar un bloque de código al menos una vez y luego repetirlo mientras se cumpla una condición específica.

**Finaliza cuando:** La palabra clave "Finaliza cuando" se utiliza en conjunto con el bucle "Repetir" para especificar la condición de salida del bucle. Indica la condición que se evalúa después de cada iteración.

**Si:** La palabra clave "Si" marca el inicio de una estructura condicional llamada "Si". Permite evaluar una condición y ejecutar un bloque de código si la condición es verdadera.

**Entonces:** La palabra clave "Entonces" se utiliza después de una condición en una estructura condicional "Si" para indicar el bloque de código que se ejecutará si la condición es verdadera.

**Sino:** La palabra clave "Sino" se utiliza en una estructura condicional "Si" para indicar un bloque de código alternativo que se ejecutará si la condición es falsa.

**FinSi:** La palabra clave "FinSi" marca el final de la estructura condicional "Si". Indica el punto donde termina el bloque de código condicional.

**Conforme:** La palabra clave "Conforme" marca el inicio de una estructura condicional llamada "Conforme". Permite evaluar múltiples casos y ejecutar un bloque de código según el caso correspondiente.

**Hacer:** La palabra clave "Hacer" se utiliza después de un caso en una estructura condicional "Conforme" para indicar el bloque de código que se ejecutará si se cumple ese caso.



**Caso:** La palabra clave "Caso" se utiliza en una estructura condicional "Conforme" para indicar un caso específico que se evalúa.

**En otro caso:** La palabra clave "En otro caso" se utiliza en una estructura condicional "Conforme" como un caso por defecto. Indica el bloque de código que se ejecutará si ninguno de los otros casos se cumple.

**FinConforme:** La palabra clave "FinConforme" marca el final de la estructura condicional "Conforme". Indica el punto donde termina el bloque de código condicional.

**Funcion:** La palabra clave "Funcion" se utiliza para declarar una función.

**FinFuncion:** La palabra clave "FinFuncion" indica el término de una función.

**Retornar:** La palabra clave "Retornar" es para que una función retorne un valor.

## Operadores

Existen operadores que nos permiten realizar diferentes tipos de operaciones en el código. Los operadores son símbolos especiales que se utilizan para realizar cálculos, comparaciones y manipulaciones de datos.

**Operadores de asignación:** Estos operadores se utilizan para asignar valores a variables.

=            Operador que indica asignación.

**Operadores aritméticos:** Estos operadores se utilizan para realizar cálculos matemáticos.

+	Operador que indica una suma.
-	Operador que indica una resta.
*	Operador que indica una multiplicación.
/	Operador que indica una división.
%	Operador que indica el módulo.

**Operadores de comparación:** Estos operadores se utilizan para comparar valores y evaluar si una condición es verdadera o falsa.

==	Operador de comparación Igual que.
!=	Operador de comparación diferente.
>	Operador de comparación mayor que.
<	Operador de comparación menor que.
>=	Operador de comparación mayor o igual que.
<=	Operador de comparación menor o igual que.

**Operadores lógicos:** Estos operadores se utilizan para realizar operaciones lógicas y combinar condiciones.

&&	Operador lógico AND.
	Operador lógico OR.
!	Operador lógico NOT.

## **Nombres de las variables en ELL**

Cuando necesitemos asignar un nombre a una variable en ELL, debemos seguir ciertas normas. Estas normas nos ayudarán a escribir nombres de variables que sean claros, legibles y no entren en conflicto con las palabras reservadas del lenguaje. Aquí están las reglas para nombrar variables en ELL:

- El primer carácter del nombre de la variable debe ser una letra (mayúscula o minúscula) o un guion bajo (\_).
- Después del primer carácter, se pueden utilizar números, letras o guiones bajos en el nombre de la variable.
- Es recomendable que los nombres de las variables sean legibles y descriptivos, de manera que podamos entender su significado al leerlos. Por ejemplo, en lugar de utilizar acrónimos o abreviaturas que no sean claros, es mejor utilizar nombres que se auto-documenten.

- Es importante tener en cuenta que los nombres de las variables no pueden coincidir con las palabras reservadas del lenguaje. Las palabras reservadas son palabras que tienen un significado especial en el lenguaje y se utilizan para funciones específicas. Algunos ejemplos de palabras reservadas en ELL son "Inicio", "Fin", "Para", "Si", entre otras.

## Literales en ELL

Los valores literales son aquellos que podemos asignar a las variables. Dependiendo del tipo de variables podremos asignar valores u otros.

### Literales de enteros

El entero que podemos utilizar será *Entero*. Los literales que le asignemos siempre será un numero entero.

```
Establecer Entero variableEnt = 10;
```

### Literales de decimales

El tipo de dato decimal que podemos manejar son el *Flotante*. Para este caso la representación del literal de decimales será con separación de un punto entre la parte entera y la parte decimal.

```
Establecer Flotante variableFlo = 10.5;
```

### Literales de caracteres y cadenas

Tanto los caracteres del tipo de dato *Caracter*, como las cadenas del tipo de datos *Cadena* contienen caracteres Unicode UTF-16. Los caracteres UTF-16 solo se pueden escribir directamente en la cadena.

Para los caracteres utilizaremos comillas simples para delimitarlos, mientras que para las cadenas utilizaremos comillas dobles.

```
Establecer Caracter variableCar = 'a';
```

```
Establecer Cadena variableCad = "cadena";
```

### Literales de booleanos

Para los booleanos se maneja el tipo de dato *Booleano*. Las literales que se le asigna siempre será Verdadero o Falso.

```
Establecer Booleano variableBolV = Verdadero;
```

# Sintaxis

## Indicar el comienzo y fin del programa

Con la siguiente estructura indicamos el inicio y el final de nuestro código.

Inicio

#Bloque de sentencias

Fin

## Declaración de variables

Las variables son como “cajas de memoria” donde podemos guardar información.

Para declarar una variable seguimos la siguiente estructura, toma en cuenta los tipos de datos (*Tipo de datos*) que tiene el lenguaje ELL y el cómo deben nombrarse (*Nombres de las variables en ELL*). Es importante no olvidar nuestro delimitador al final de la sentencia (;).

```
Establecer [Tipo_Dato] [Nombre_Variable];
```

## Asignación de variables

Una vez que hemos declarado una variable, podemos asignarle un valor dependiendo del tipo de dato que sea.

```
[Variable] [Operador Asignación] [ Valor ];
```

Al igual lo podemos hacer directamente al declarar nuestra variable.

```
Establecer [ Tipo_Dato ] [ Nombre_Variable ] [Operador Asignación]  
[ Valor ];
```

## Lectura de variables

Se usa para leer el valor de una variable ingresada por el usuario.

```
Interpretar [ Variable ];
```

## Imprimir

Imprimir se refiere a mostrar información; es una forma fácil de visualizar resultados, mensajes o datos. Para imprimir definimos la siguiente estructura:

```
Escribir "Cadena";
```

```
Escribir [ Variable ];
```

Solo se puede hacer una acción a la vez, es decir, si vas a imprimir una cadena solamente será la cadena, pero no puedes imprimir una cadena y una variable juntas. Las cadenas de texto van entre comillas dobles.

## Matrices en ELL

Una matriz es como una cuadrícula donde puedes organizar datos en filas y columnas. Para declarar una matriz se sigue la siguiente estructura, donde la longitud es un numero entero y puedes tener 2 o más longitudes:

```
Establecer [ Tipo_Dato ] [ LONGITUD ] [ LONGITUD2 ] [ Variable ];
```

## Arreglos en ELL

Un arreglo es como una lista ordenada de elementos del mismo tipo, donde cada elemento tiene una posición única. Lo podemos ver como una fila de casillas donde puede guardar información y cada casilla tiene su propio número para identificarla. Para declarar un arreglo se sigue la siguiente estructura:

```
Establecer [ Tipo_Dato ] [ LONGITUD ] [ Variable ];
```

## Funciones en ELL

Una función es un bloque de código reutilizable que realiza una acción específica cuando se llama desde otra parte del programa, al igual podemos conocerlo como un método.

Es importante aclarar que las funciones solo pueden ser declaradas antes del Inicio. Para declarar una función se sigue la siguiente estructura:

```
Funcion [ Tipo_Dato ] [ Nombre_Método ] ( [ Argumentos ] )
```

```
#Bloque de sentencias

Retornar [ Expresión ];

FinFuncion [ Nombre_Método ]
```

Un método vacío, es una función que no devuelve ningún valor cuando es invocado, comúnmente usados para imprimir mensajes en consola o cualquier tipo de acción sin necesidad de devolver un valor. Para declarar un método vacío solo no hay que agregar el tipo de dato del método y excluir el retornar.

## **Sentencias decisión en ELL**

Las sentencias de decisión son sentencias que nos permiten tomar una decisión para poder ejecutar un bloque de sentencias u otro. Las sentencias de decisión que contiene el lenguaje son:

### **Si – Entonces – Sino**

La estructura de la sentencia Si-Entonces-Sino es:

```
Si [ condición ] Entonces

    #Bloque de sentencias

Sino

    #Bloque de sentencias

FinSi
```

La parte del Sino no tiene por qué existir. En este caso tendríamos una sentencia Si-Entonces.

```
Si [condición] Entonces

    #Bloque de sentencias

FinSi
```

La sentencia Si-Entonces-Sino pueden estar anidadas y así nos encontraríamos con una sentencia Si-Entonces-SinoSi, la cual tendría la siguiente estructura:

```
Si [condición] Entonces
    #Bloque de sentencias
Sino Si [condición] Entonces
    #Bloque de sentencias
Sino
    #Bloque de sentencias
FinSi
FinSi
```

### **Conforme**

Para los casos en los que se tienen muchas ramas o caminos de ejecución en una sentencia Si tenemos la sentencia Conforme. La sentencia Conforme evalúa una expresión y ejecutará el bloque de sentencias que coincida con el valor de la expresión.

El valor de la expresión puede ser numérico o al igual se pueden utilizar expresiones cuya evaluación sean cadenas.

La estructura de la sentencia Conforme es:

```
Conforme [condición] Hacer
    Caso [valor1]:
        #Bloque de sentencias
    Caso [valor2]:
        #Bloque de sentencias
    En Otro Caso:
        #Bloque de sentencias
FinConforme
```



## Sentencias Bucle en ELL

Las sentencias de bucle nos van a permitir ejecutar un bloque de sentencias tantas veces como queramos, o tantas veces como se cumpla una condición. Las sentencias de bucle que contiene el lenguaje son:

### Para

La estructura del bucle Para es:

```
Para [sentencia_inicio] Hasta que [condición] Con incremento  
[valor_entero_ó_decimal]  
    #Bloque de sentencias  
FinPara
```

Las funcionalidades en las que podemos utilizar la sentencia Para puede ser como un contador.

### Mientras

La estructura repetitiva Mientras realiza una primera evaluación antes de ejecutar el bloque. Si la expresión es verdadera pasa a ejecutar de forma repetitiva el bloque de sentencias.

La estructura de la sentencia Mientras es la siguiente:

```
Mientras [condición] Hacer  
    #Bloque de sentencias  
FinMientras
```

### Repetir

La estructura de la sentencia Repetir es la siguiente:

```
Repetir  
    #Bloque de sentencias
```

Finalizar cuando [condición];

## Tabla de errores

Número de error	Tipo	Ubicación	Token	Descripción	Solución.
1	Error léxico	Línea 3, columna 6	,	Símbolo ‘,’ desconocido	Remove símbolo ‘,’
2	Error léxico	Línea 5, columna 6	17.16.3	Se encontró un numero invalido	Remove o revisar numero
2	Error sintáctico	Línea 3, Columna 24	;	Se encontró símbolo “;”	Se esperaba uno de los siguientes: <ul style="list-style-type: none"> <li>- &lt;CADENA_TEXTO&gt;</li> <li>- &lt;CARÁCTER_TEXTO&gt;</li> <li>- &lt;NUMERO_ENTERO&gt;</li> <li>- &lt;NUMERO_DECIMAL&gt;</li> <li>- “Falso”</li> <li>- “Verdadero”</li> <li>- “!”</li> <li>- “(“</li> <li>- &lt;VARIABLE&gt;</li> </ul>
3	Error sintáctico	Línea 13, columna 32	Mientras	Se encontró el símbolo “Mientras”	Se esperaba uno de los siguientes: <ul style="list-style-type: none"> <li>- “,”</li> <li>- “+”</li> <li>- “_”</li> <li>- “*”</li> <li>- “/”</li> <li>- “%”</li> <li>- “ ”</li> <li>- “==”</li> <li>- “!=”</li> <li>- “&gt;”</li> <li>- “&lt;”</li> <li>- “&gt;=”</li> <li>- “&lt;=”</li> </ul>
4	Error sintáctico	Línea 19, columna 9	<VARIABLE>	Se encontró el símbolo <VARIABLE>	Se esperaba uno de los siguientes: <ul style="list-style-type: none"> <li>- &lt;CADENA_TEXTO&gt;</li> <li>- &lt;CARÁCTER_TEXTO&gt;</li> <li>- &lt;NUMERO_ENTERO&gt;</li> <li>- &lt;NUMERO_DECIMAL&gt;</li> <li>- “Falso”</li> <li>- “Verdadero”</li> </ul>

5	Error sintáctico	Línea 1, columna 1	Fin	Se encontró el símbolo "Fin"	Se esperaba uno de los siguientes: <ul style="list-style-type: none"> <li>- &lt;COMENTARIO&gt;</li> <li>- "Interpretar"</li> <li>- "Escribir"</li> <li>- "Establecer"</li> <li>- "Para"</li> <li>- "Mientras"</li> <li>- "Repetir"</li> <li>- "Si"</li> <li>- "Conforme"</li> <li>- &lt;VARIABLE&gt;</li> </ul>
6	Error semántico	Línea 15, columna 4	b	El símbolo "b" ya está declarado en este ámbito	Quitar declaración doble
7	Error semántico	Línea 12, columna 3	mostrar	La función "mostrar" no puede retornar ningún valor	Quitar retornar
8	Error semántico	Línea 4, columna 9	salario	La función "salario" no puede retornar un tipo de dato "Entero"	Retornar tipo de dato "Flotante"
9	Error semántico	Línea 8, columna 4		La función debe terminar con el mismo nombre	Colocar el nombre la función después del "FinFunsion"
10	Error semántico	Línea 4, columna 2	cal1	El símbolo "cal1" no está declarado en este ámbito	Declarar el símbolo
11	Error semántico	Lina 1, columna 7		El símbolo no es una variable	No se le puede asignar un valor a una llamada de función

## Manual del Sistema

El respectivo manual busca proporcionar una guía clara y completa para nuestro lenguaje que permita a los programadores utilizar el lenguaje de manera efectiva y dar información clara del lenguaje, donde como objetivos tiene los siguientes:

- Describir el sistema: El manual de sistema tiene como objetivo principal proporcionar una descripción completa y detallada del sistema.
- Orientar en la solución de problemas: El manual debe proporcionar información sobre la solución de problemas comunes que los usuarios pueden enfrentar al utilizar el sistema.
- Servir como referencia: El manual de sistema debe ser una referencia útil para los usuarios, brindando información detallada sobre todas las funciones, configuraciones y aspectos técnicos del sistema.

### Requisitos del sistema

- Tener Java instalado.
- Tener JavaCC instalado.

### Descripción del sistema

EC (EasyCompiler) es un compilador de código abierto que admite el lenguaje de programación ELL (EasyLearningLanguage). El compilador se ejecuta en cualquier plataforma que tenga instalada Java.

### Tabla de valores

Valor	Tipo
Inicio	PR_Arranque_Programa
Fin	PR_Cierre_Programa
#	Simbolo_Comentario
edad	Variable
Establecer	PR_Definicion_Variable
Entero	PR_Tipo_De_Dato
Flotante	PR_Tipo_De_Dato

Cadena	PR_Tipo_De_Dato
Caracter	PR_Tipo_De_Dato
Booleano	PR_Tipo_De_Dato
;	Delimitador
Interpretar	PR_Lectura
Escribir	PR_Escritura
123	Numero_Entero
123.123	Numero_Flotante
"Cadena"	Cadena_Texto
'C'	Caracter_Texto
=	Operador_Asignacion
+	Operador_Suma
-	Operador_Resta
*	Operador_Mult
/	Operador_Div
%	Operador_Mod
Falso	PR_Booleano_Falso
Verdadero	PR_Booleano_Verdadero
&&	Operador_Logico_AND
	Operador_Logico_OR
!	Operador_Logico_NOT
==	Operador_igualQue
!=	Operador_diferente
>	Operador_mayorQue
<	Operador_menorQue
>=	Operador_mayorIgualQue
<=	Operador_menorIgualQue
Para	PR_Ciclo_Para
Hasta que	PR_Condicion_Ciclo_Para
Con incremento	PR_Incremento_Ciclo_Para

FinPara	PR_Fin_Ciclo_Para
Mientras	PR_Ciclo_Mientras
FinMientras	PR_Fin_Ciclo_Mientras
Repetir	PR_Ciclo_Repetir
Finaliza cuando	PR_Condicion_Ciclo_Repetir
Si	PR_Condicional_Si
Entonces	PR_Entonces
Sino	PR_Conficional_Sino
FinSi	PR_Fin_Condicional_Si
Conforme	PR_Condicional_Conforme
Hacer	PR_Hacer
Caso	PR_Conforme_Caso
:	Operador_Dos_Puntos
En otro caso	PR_Conforme_Caso_Predeterminado
FinConforme	PR_Fin_Condicional_Conforme
(	Parentesis_Abierto
)	Parentesis_Cerrado
[	Indice_Abierto
]	Indice_Cerrado
,	Separador_Parametros
Funcion	PR_Declarar_Funcion
FinFuncion	PR_Fin_Funcion
Retornar	PR_Retornar

Tabla 3 – Tabla de valores

## Expresiones regulares

Número entero: (número)<sup>+</sup>

Número flotante: (número)<sup>+</sup>. (número)<sup>+</sup>

Comentario: #(letra| símbolo | número)\*

Variable: (letra | \_)(número | letra | \_)\*

Una cadena: "(letra | número | símbolo)\* "

Un carácter: '(letra | número | símbolo | €) '

## Nombres de las variables en ELL

Cuando necesitemos asignar un nombre a una variable en ELL, debemos seguir ciertas normas. Estas normas nos ayudarán a escribir nombres de variables que sean claros, legibles y no entren en conflicto con las palabras reservadas del lenguaje. Aquí están las reglas para nombrar variables en ELL:

- El primer carácter del nombre de la variable debe ser una letra (mayúscula o minúscula) o un guion bajo (\_).
- Después del primer carácter, se pueden utilizar números, letras o guiones bajos en el nombre de la variable.
- Es recomendable que los nombres de las variables sean legibles y descriptivos, de manera que podamos entender su significado al leerlos. Por ejemplo, en lugar de utilizar acrónimos o abreviaturas que no sean claros, es mejor utilizar nombres que se auto-documenten.
- Es importante tener en cuenta que los nombres de las variables no pueden coincidir con las palabras reservadas del lenguaje. Las palabras reservadas son palabras que tienen un significado especial en el lenguaje y se utilizan para funciones específicas. Algunos ejemplos de palabras reservadas en ELL son "Inicio", "Fin", "Para", "Si", entre otras.

## Tipo de Datos

El lenguaje ELL tiene los siguientes tipos de datos básicos:

- **Entero:** Tipo de dato que representa un numero entero.
- **Flotante:** Tipo de dato que representa un numero en decimal.
- **Cadena:** Tipo de dato que representa un texto.
- **Carácter:** Tipo de datos que representa un símbolo.
- **Booleano:** Tipo de dato que representa aquellos que tienen un valor de verdadero o falso.



## Operadores

Los operadores con los que cuenta el lenguaje ELL son:

Operador	Descripción
=	Operador que indica asignación.
+	Operador que indica una suma.
-	Operador que indica una resta.
*	Operador que indica una multiplicación.
/	Operador que indica una división.
%	Operador que indica al módulo.
&&	Operador lógico AND.
	Operador lógico OR.
!	Operador lógico NOT.
==	Operador de comparación Igual que.
!=	Operador de comparación diferente.
>	Operador de comparación mayor que.
<	Operador de comparación menor que.
>=	Operador de comparación mayor o igual que.
<=	Operador de comparación menor o igual que.

*Tabla 1 – Operadores*

## Sintaxis

### Programa principal en ELL

Un programa en ELL siempre tiene que iniciar con la palabra reservada “Inicio” y terminar con la palabra reservada “Fin”, además entre estas dos palabras debe de ir al menos una sentencia.

**PROGRAMA** → **Inicio** ( **SENTENCIA** )<sup>+</sup> **Fin**

### Declaración de variables en ELL

La declaración de una variable se considera una sentencia, la cual siempre tiene que iniciar con la palabra reservada “Establecer” seguida de un tipo de dato y luego

un nombre para la variable, opcionalmente puede estar el operador de asignación y el valor el cual tomara la variable, y finalmente un punto y coma.

**DECLARACION\_VARIABLE → Establecer TIPO\_DATO VARIABLE (= VALOR)?;**

### **Asignación de variables en ELL**

La asignación de variables es también una sentencia, la cual tiene que iniciar con un nombre de variable, seguido de un operador de asignación, luego un valor y finalmente un punto y coma.

**ASIGNACION\_VARIABLE → VARIABLE OPERADOR\_ASIGNACION VALOR ;**

### **Leer del teclado en ELL**

La sentencia de leer un dato empieza por la palabra reservada “Interpretar” seguida de un nombre de variable y finalmente un punto y coma.

**LEER\_DATO → Interpretar VARIABLE ;**

### **Imprimir en pantalla en ELL**

La sentencia de imprimir en pantalla debe iniciar con la palabra reservada “Escribir” seguida por algún valor y luego un punto y coma.

**IMPRIMIR\_DATO → Escribir VALOR ;**

### **Matrices en ELL**

Para declarar una matriz, debe iniciar con la palabra reservada “Establecer” seguida del tipo de dato y la longitud entera por cada dimensión de la matriz, al final ira el nombre que se le asignara a la matriz.

**MATRIZ → Establecer TIPO\_DATO [ENTERO] ([ ENTERO ])+ VARIABLE ;**

### **Arreglos en ELL**

Para declarar un arreglo, debe iniciar con la palabra reservada “Establecer” seguida del tipo de dato y la longitud entera del arreglo, al final ira el nombre que se le asignara al arreglo.

**ARREGLO → Establecer TIPO\_DATO [ENTERO] VARIABLE ;**

## **Funciones en ELL**

Para declarar una función, debe iniciar con la palabra reservada “Funcion”, seguida del tipo de dato que devolverá la función, es importante aclarar que las funciones solo pueden ser declaradas antes del Inicio. Si no se coloca el tipo de dato se tomará como una función vacía. Seguido del tipo de dato irá el nombre del método y luego, entre paréntesis, los parámetros. Después, puede o no seguir una sentencia y, finalmente la palabra reservada “FinFuncion” junto al nombre del método. Si la función no es vacía, entonces al final de la sentencia debe estar la palabra reservada “Retornar” para devolver.

**DECLARAR\_FUNCION → Funcion (TIPO\_DATO)? IDENT. ARGUMENTOS**

**( SENTENCIA ) \* ( Retornar EXPRESION ; ) ? FinFuncion IDENT.**

## **Sentencia Si en ELL**

Una sentencia “Si” debe de iniciar con la palabra reservada “Si” seguida de una condición, luego de la palabra reservada “Entonces”, luego debe de haber por lo menos una sentencia, opcionalmente puede estar la palabra reservada “Sino” seguida de al menos una sentencia, y finalmente debe terminar con la palabra reservada “FinSi”.

**SENTENCIA\_SI → Si CONDICION Entonces ( SENTENCIA )<sup>+</sup>**

**( Sino ( SENTENCIA )<sup>+</sup> ) ? FinSi**

## **Sentencia Conforme en ELL**

Una sentencia “Conforme” debe de iniciar con la palabra reservada “Conforme” seguida de un nombre de variable, luego de la palabra reservada “Hacer”, después debe de haber por lo menos una palabra reservada “Caso” seguida de una constante, luego dos puntos y al menos una sentencia, opcionalmente puede estar la palabra reservada “En otro caso” seguida de dos puntos y por lo menos una sentencia, finalmente la palabra reservada “FinConforme”.

**SENTENCIA\_CONFORME → Conforme VARIABLE Hacer**

**( Caso CONSTANTES : ( SENTENCIA )<sup>+</sup> )<sup>+</sup>**

**( En otro caso : ( SENTENCIA )<sup>+</sup> )? FinConforme**

## **Sentencia Mientras en ELL**

Una sentencia “Mientras” debe iniciar con la palabra reservada “Mientras” seguida de una condición y luego la palabra reservada “Hacer”, debe seguir por lo menos una sentencia y finalmente la palabra reservada “FinMientras”.

**SENTENCIA\_MIENTRAS → Mientras CONDICION Hacer ( SENTENCIA )<sup>+</sup> FinMientras**

## **Sentencia Repetir en ELL**

Una sentencia “Repetir” inicia con la palabra reservada “Repetir” seguida de por lo menos una sentencia, luego debe de ir la palabra reservada “Finaliza cuando” seguida de una condición y finalmente un punto y coma.

**SENTENCIA\_REPETIR → Repetir ( SENTENCIA )<sup>+</sup> Finaliza cuando CONDICION ;**

## **Sentencia Para en ELL**

Una sentencia “Para” inicia por la palabra reservada “Para” seguida de una asignación de variable o de una declaración de variable, luego de la palabra reservada “Hasta que” seguida de una condición, después la palabra reservada “Con incremento” seguida de un número entero o decimal, luego debe haber al menos una sentencia y finalmente la palabra reservada “FinPara”.

**SENTENCIA\_PARA → Para (ASIGNACION\_VARIABLE | DECLARACION\_VARIABLE)**

**Hasta que CONDICION Con incremento ( NUMERO\_ENTERO | NUMERO\_DECIMAL )**

**( SENTENCIA )<sup>+</sup> FinPara**

## Gramáticas del Lenguaje

$\langle \text{PROGRAMA} \rangle \rightarrow ( \langle \text{DECLARACION\_FUNCION} \rangle )^* \text{Inicio} ( \langle \text{SENTENCIAS} \rangle )^+ \text{Fin}$

$\langle \text{CONSTANTES} \rangle \rightarrow \text{NUMERO\_ENTERO} \mid \text{NUMERO\_DECIMAL} \mid \text{CADENA\_TEXTO} \mid$   
 $\text{CARACTER\_TEXTO} \mid \text{BOOLEANO\_FALSO} \mid \text{BOOLEANO\_VERDADERO}$

$\langle \text{TIPO\_DATO} \rangle \rightarrow \text{ENTERO} \mid \text{FLOTANTE} \mid \text{CADENA} \mid \text{CARACTER} \mid \text{BOOLEANO}$

$\langle \text{EXPRESION} \rangle \rightarrow \langle \text{LOGICO\_OR} \rangle$

$\langle \text{LOGICO\_OR} \rangle \rightarrow \langle \text{LOGICO\_AND} \rangle ( \text{LOGICO\_OR} \langle \text{LOGICO\_AND} \rangle )^*$

$\langle \text{LOGICO\_AND} \rangle \rightarrow \langle \text{COMPARACION} \rangle ( \text{LOGICO\_AND} \langle \text{COMPARACION} \rangle )^*$

$\langle \text{COMPARACION} \rangle \rightarrow \langle \text{OPERANDO} \rangle ( ( \text{OPERADOR\_IGUAL} \mid \text{OPERADOR\_DIFERENTE} \mid$   
 $\text{OPERADOR\_MENOR\_IGUAL} \mid \text{OPERADOR\_MENOR} \mid \text{OPERADOR\_MAYOR} ) \langle \text{OPERANDO} \rangle )^?$

$\langle \text{OPERANDO} \rangle \rightarrow \langle \text{TERMINO} \rangle ( ( \text{SUMA} \mid \text{RESTA} ) \langle \text{TERMINO} \rangle )^*$

$\langle \text{TERMINO} \rangle \rightarrow \langle \text{FACTOR} \rangle ( ( \text{MULTIPLICACION} \mid \text{DIVISION} \mid \text{MODULO} ) \langle \text{FACTOR} \rangle )^*$

$\langle \text{FACTOR} \rangle \rightarrow \text{VARIABLE} ( ( \text{INDICE\_ARREGLO} \mid \text{PAREN\_ABIERTO} ( \langle \text{ARGUMENTOS} \rangle )^?$   
 $\text{PAREN\_CERRADO} ) )^?$

$\langle \text{FACTOR} \rangle \rightarrow \langle \text{CONSTANTES} \rangle$

$\langle \text{FACTOR} \rangle \rightarrow \text{PAREN\_ABIERTO} \langle \text{EXPRESION} \rangle \text{PAREN\_CERRADO}$

$\langle \text{FACTOR} \rangle \rightarrow \text{LOGICO\_NOT} \langle \text{FACTOR} \rangle$

$\langle \text{FACTOR} \rangle \rightarrow ( \text{SUMA} \mid \text{RESTA} ) \langle \text{FACTOR} \rangle$

$\langle \text{INDICE\_ARREGLO} \rangle \rightarrow ( \text{INDICE\_ABIERTO} \langle \text{EXPRESION} \rangle \text{INDICE\_CERRADO} )^+$

$\langle \text{DECLARACION} \rangle \rightarrow \text{ESTABLECER} \langle \text{TIPO\_DATO} \rangle ( \langle \text{INDICE\_ARREGLO} \rangle )^? \text{VARIABLE}$   
 $( \langle \text{ASIGNACION} \rangle )^?$

$\langle \text{ASIGNACION} \rangle \rightarrow \text{ASIGNACION} \langle \text{EXPRESION} \rangle$

$\langle \text{DECLARACION\_ARGUMENTOS} \rangle \rightarrow \langle \text{TIPO\_DATO} \rangle \text{VARIABLE}$

$( \text{COMA} \langle \text{TIPO\_DATO} \rangle \text{VARIABLE} )^*$

⟨SENTENCIAS⟩ → ⟨SENTENCIA\_ASIGNACION⟩

⟨SENTENCIAS⟩ → ⟨SENTENCIA\_DECLARACION⟩

⟨SENTENCIAS⟩ → ⟨LEER\_DATO⟩

⟨SENTENCIAS⟩ → ⟨IMPRIMIR\_DATO⟩

⟨SENTENCIAS⟩ → ⟨SENTENCIA\_SI⟩

⟨SENTENCIAS⟩ → ⟨SENTENCIA\_CONFORME⟩

⟨SENTENCIAS⟩ → ⟨SENTENCIA\_PARA⟩

⟨SENTENCIAS⟩ → ⟨SENTENCIA\_REPETIR⟩

⟨SENTENCIAS⟩ → ⟨SENTENCIA\_MIENTRAS⟩

⟨SENTENCIA\_DECLARACION⟩ → ⟨DECLARACION⟩ DELIMITADOR

⟨SENTENCIA\_ASIGNACION⟩ → VARIABLE ⟨ASIGNACION⟩ DELIMITADOR

⟨LEER\_DATO⟩ → INTERPRETAR VARIABLE DELIMITADOR

⟨IMPRIMIR\_DATO⟩ → ESCRIBIR ⟨EXPRESION⟩ DELIMITADOR

⟨SENTENCIA\_SI⟩ → INICIO\_CONDICIONAL\_SI ⟨EXPRESION⟩ ENTONCES

( ⟨SENTENCIAS⟩ )<sup>\*</sup> ( CONDICIONAL\_SINO ( ⟨SENTENCIAS⟩ )<sup>\*</sup> )<sup>?</sup> FIN\_CONDICIONAL\_SI

⟨SENTENCIA\_CONFORME⟩ → INICIO\_CONDICIONAL\_CONFORME VARIABLE HACER

( CASO ⟨CONSTANTES⟩ OPERADOR\_DOS\_PUNTOS ( ⟨SENTENCIAS⟩ )<sup>\*</sup> )<sup>+</sup>

( CASO\_PREDETERMINADO OPERADOR\_DOS\_PUNTOS ( ⟨SENTENCIAS⟩ )<sup>\*</sup> )<sup>?</sup>

FIN\_CONFORME

⟨SENTENCIA\_PARA⟩ → INICIO\_CICLO\_PARA (VARIABLE ⟨ASIGNACION⟩ |

⟨DECLARACION⟩) CONDICION\_CICLO\_PARA ⟨EXPRESION⟩ INCREMENTO\_CICLO\_PARA

( NUMERO\_ENTERO | NUMERO\_DECIMAL ) ( <SENTENCIAS> ) \* FIN\_CICLO\_PARA

<SENTENCIA\_REPETIR> → INICIO\_CICLO\_REPETIR ( <SENTENCIAS> ) \*

CONDICION\_CICLO\_REPETIR <EXPRESION> DELIMITADOR

<SENTENCIA\_MIENTRAS> → INICIO\_CICLO\_MIENTRAS <EXPRESION> HACER

( <SENTENCIAS> ) \* FIN\_CICLO\_MIENTRAS

<DECLARACION\_FUNCION> → FUNCION ( <TIPO<sub>DATO</sub>> ) ? VARIABLE PAREN\_ABIERTO

( <DECLARACION\_ARGUMENTOS> ) ? PAREN\_CERRADO ( <SENTENCIAS> ) \* (RETORNAR

<EXPRESION> DELIMITADOR) ? FIN\_FUNCION

<ARGUMENTOS> → <EXPRESION> (COMA <EXPRESION> ) \*

### Tabla de errores.

Número de error	Tipo	Ubicación	Token	Descripción	Solución.
1	Error léxico	Línea 3, columna 6	,	Símbolo ‘,’ desconocido	Remove símbolo ‘,’
2	Error léxico	Línea 5, columna 6	17.16.3	Se encontró un numero invalido	Remove o revisar numero
2	Error sintáctico	Línea 3, Columna 24	;	Se encontró símbolo “;”	Se esperaba uno de los siguientes: <ul style="list-style-type: none"> <li>- &lt;CADENA_TEXTO&gt;</li> <li>- &lt;CARÁCTER_TEXTO&gt;</li> <li>- &lt;NUMERO_ENTERO&gt;</li> <li>- &lt;NUMERO_DECIMAL&gt;</li> <li>- “Falso”</li> <li>- “Verdadero”</li> <li>- “!”</li> <li>- “(“</li> <li>- &lt;VARIABLE&gt;</li> </ul>
3	Error sintáctico	Línea 13, columna 32	Mientras	Se encontró el símbolo “Mientras”	Se esperaba uno de los siguientes: <ul style="list-style-type: none"> <li>- “,”</li> <li>- “+”</li> <li>- “_”</li> <li>- “*”</li> <li>- “/”</li> <li>- “%”</li> <li>- “ ”</li> <li>- “==”</li> <li>- “!=”</li> <li>- “&gt;”</li> <li>- “&lt;”</li> <li>- “&gt;=”</li> <li>- “&lt;=”</li> </ul>
4	Error sintáctico	Línea 19, columna 9	<VARIABLE>	Se encontró el símbolo <VARIABLE>	Se esperaba uno de los siguientes: <ul style="list-style-type: none"> <li>- &lt;CADENA_TEXTO&gt;</li> <li>- &lt;CARÁCTER_TEXTO&gt;</li> <li>- &lt;NUMERO_ENTERO&gt;</li> <li>- &lt;NUMERO_DECIMAL&gt;</li> <li>- “Falso”</li> <li>- “Verdadero”</li> </ul>



5	Error sintáctico	Línea 1, columna 1	Fin	Se encontró el símbolo "Fin"	Se esperaba uno de los siguientes: <ul style="list-style-type: none"> <li>- &lt;COMENTARIO&gt;</li> <li>- "Interpretar"</li> <li>- "Escribir"</li> <li>- "Establecer"</li> <li>- "Para"</li> <li>- "Mientras"</li> <li>- "Repetir"</li> <li>- "Si"</li> <li>- "Conforme"</li> <li>- &lt;VARIABLE&gt;</li> </ul>
6	Error semántico	Línea 15, columna 4	b	El símbolo "b" ya está declarado en este ámbito	Quitar declaración doble
7	Error semántico	Línea 12, columna 3	mostrar	La función "mostrar" no puede retornar ningún valor	Quitar retornar
8	Error semántico	Línea 4, columna 9	salario	La función "salario" no puede retornar un tipo de dato "Entero"	Retornar tipo de dato "Flotante"
9	Error semántico	Línea 8, columna 4		La función debe terminar con el mismo nombre	Colocar el nombre la función después del "FinFunsion"
10	Error semántico	Línea 4, columna 2	cal1	El símbolo "cal1" no está declarado en este ámbito	Declarar el símbolo
11	Error semántico	Lina 1, columna 7		El símbolo no es una variable	No se le puede asignar un valor a una llamada de función