

# Proyecto Final

Instituto Tecnológico de Costa Rica  
Área de Ingeniería en Computadores  
CE3104 - Lenguajes, compiladores e intérpretes  
I Semestre 2023 – Grupo 2



## Objetivo general

- Implementación de un intérprete, el cual permite la traducción de un lenguaje a otro.

## Objetivos específicos

- Permitir implementar la etapa de análisis léxico, análisis sintáctico, y traducción.
- Recibir una serie de datos, procesarlos, clasificarlos y generar la lógica necesaria para generar su equivalencia a otro lenguaje.
- Se debe contar con mecanismos especializados que puedan tomar datos tanto a nivel digital como físico para realizar su transformación al alfabeto braille.
- Se debe mostrar en pantalla la "traducción"
- Enviar la información necesaria para generar la traducción de forma sonora.

## Datos Generales

- El valor del proyecto: 25%
- Nombre código: **BrailleRead**
- El proyecto debe ser implementado por grupos de máximo de 4 personas, de tal manera que puedan balancear la carga de trabajo, y pueda presentar un proyecto lo más completo posible.
- La fecha de entrega del proyecto es de **06/06/2023**
- Cualquier indicio de copia de proyectos será calificado con una nota de 0 y será procesado de acuerdo con reglamento.

## Descripción del problema

**braille** es un sistema de lectura y escritura táctil pensado para personas algún tipo de problema en los ojos. Se conoce también como cecografía. Fue ideado por el francés Louis Braille a mediados del siglo XIX, que se quedó ciego debido a un accidente durante su niñez mientras jugaba en el taller de su padre. Cuando tenía 13 años, el director de la Escuela de Ciegos y Sordos de París –donde estudiaba el joven Braille– le pidió que probara un sistema de lecto-escritura táctil inventado por un militar llamado Charles Barbier de la Serre para transmitir órdenes a puestos de avanzada sin tener necesidad de delatar la posición durante las noches. Louis Braille descubrió al cabo de un tiempo que el sistema era válido y lo reinventó utilizando un sistema de 8 puntos. Al cabo de unos años lo simplificó dejándolo en el sistema universalmente conocido y adoptado de 6 puntos. *Tomado de Wikipedia*

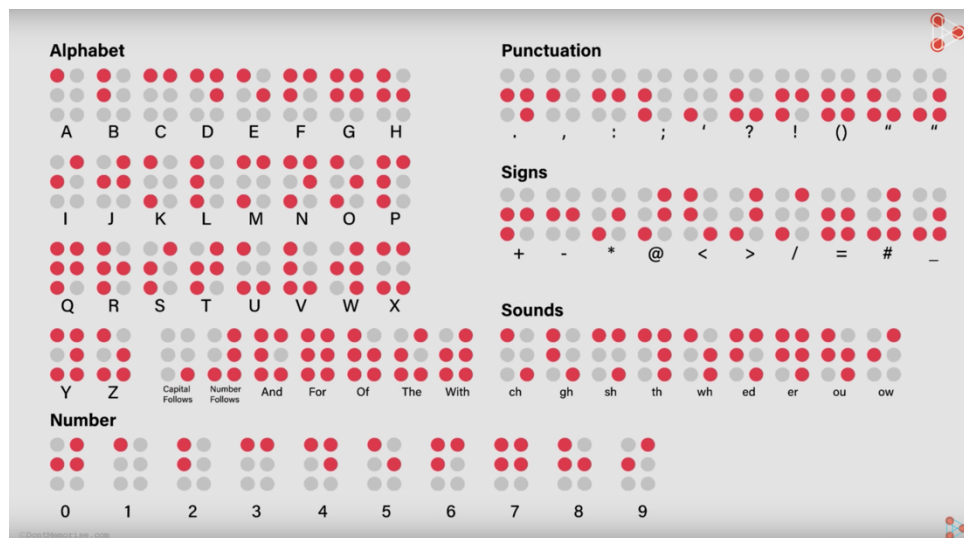
El sistema Braille se basa en **seis** puntos que se distribuyen de diferentes formas, cayendo dentro de lo que se considera un sistema binario. No se trata de un idioma, sino que de un **alfabeto** reconocido de forma internacional, capaz de exponer letras, números y hasta signos, lo que le hace realmente completo.

Cada carácter está basado en seis puntos que se ordenan en dos hileras paralelas de tres. Según lo que quiera representar, ciertos puntos están en relieve y, al tocarlos, quien sabe interpretar Braille detecta a qué letra, número o signo corresponde.

## ALFABETO

Este tipo de escritura está basada en la combinación de seis puntos por cada celda. Los puntos se ubican en dos columnas verticales de tres puntos cada una (o, dependiendo de cómo se vea, tres filas horizontales de dos puntos cada una). Una sola letra puede estar representada por un mínimo de un punto y un máximo de cinco. Existe un patrón en el alfabeto braille que corresponde a la posición de la letra en el alfabeto.

- Las primeras diez letras del alfabeto (de la A a la J) se forman exclusivamente por combinaciones de los cuatro puntos superiores.
- Las siguientes diez letras (de la K a la T) se forman añadiendo el punto inferior izquierdo a las diez letras previas. Por ejemplo, cuando el punto superior izquierdo (que representa a la letra A) está acompañado del punto inferior izquierdo, se convierte en la letra K. Luego viene la letra L, que se forma añadiendo el mismo punto inferior izquierdo a la forma que representa la letra B, y así sucesivamente hasta llegar a la letra T.
- Las siguientes cinco letras, excluyendo la letra W, se forman añadiendo dos puntos inferiores a las primeras diez letras. La letra W es anómala porque no existía en francés, idioma en el que el braille se diseñó inicialmente



## PUNTUACION

La puntuación también está compuesta por una combinación de algunos de los seis puntos en una sola celda. Así pues, un punto en la parte inferior derecha indica que la siguiente letra es mayúscula. El símbolo del punto se compone de punto en la parte inferior derecha, acompañado de dos puntos en la segunda fila. Es decir, es igual que la letra D pero una fila más abajo. Asimismo, el punto de exclamación es similar a la letra F pero una fila más abajo.

- Para indicar que toda la palabra está escrita en mayúsculas, la palabra estará precedida de dos símbolos que representan la mayúscula, es decir, dos celdas con solo un punto en la parte inferior derecha.
- Para los números, utiliza el símbolo correspondiente, compuesto por tres puntos en la columna derecha acompañado del punto inferior de la columna izquierda (haciendo una L invertida). Este símbolo para número debe estar acompañado de los símbolos que denotan las letras de A a J. Por ejemplo, la letra A acompañada del símbolo de número significa "1", la letra B se convierte en "2", y así sucesivamente hasta la letra J, que representa en número "0".

## CONTRACCIONES

Ya que el braille ocupa mucho más espacio que el alfabeto español, la escritura puede simplificarse a través del uso de contracciones. Existen 189 combinaciones adicionales de una celda de palabras comunes como "para", "y" o "el/la". Asimismo, ciertos componentes comunes de las palabras, como las terminaciones de conjugaciones en presente continuo y pasado, también tienen símbolos particulares. También es común utilizar abreviaciones. Por ejemplo, las letras "tm" son la abreviación de "mañana" en el alfabeto braille inglés.

Tomado de

<https://es.wikihow.com/escribir-en-braille>

ABECEDARIO

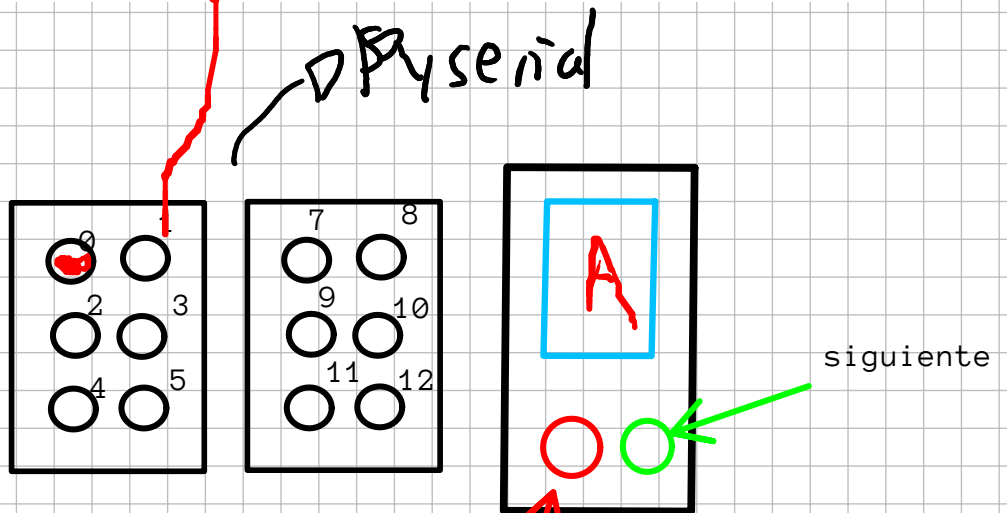
a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
u	v	w	x	y	z	ñ	ü		
á	é	í	ó	ú					

SIGNOS

-	,	:	-	"	!	¿
(	)	+	x	=	÷	-

NÚMEROS

1	2	3	4	5
6	7	8	9	0



2 bits A

`["100000000A"]`

`"000010000"`

restar/apagar/encender

12 A

Hola como estas?

C++

vector

Arduino lo recibe como un vector

`def("string")`  
`return []`

`def(["1000",])`

## 1. Comunicacion con Arduino

Hacer una funcion en Python que recibe un string y manda ese string a Arduino(Se usa aqui el Pyserial preguntar a Chatpt como usarlo y videos)

```
def send_to_hardware("010001101100H,000000100110o,000000101010l,000000100000a")  
    manda la info
```

Hacer que **Arduino** imprima este mensaje por consola

```
>>"010001101100H,000000100110o,000000101010l,000000100000a"
```

## 2. Arduino tiene un vector un los movimientos

Ahora con este string en **Arduino** toma ese string y lo pasa aun vector de strings como en cpp

este vector veria algo asi

```
vector<string> information = [  
"010001101100H","000000100110o","000000101010l","000000100000a" ]
```

ahora si yo hago

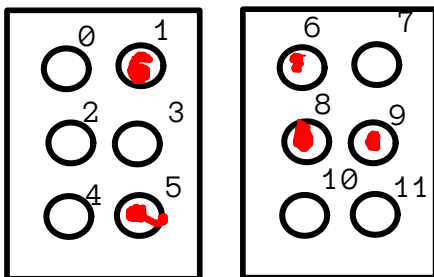
```
print(informacion[1])  
>>"000000100110o"
```

### 3. Generador de Braille

Una funcion que toma un string y genera un string con la traduccion de braille

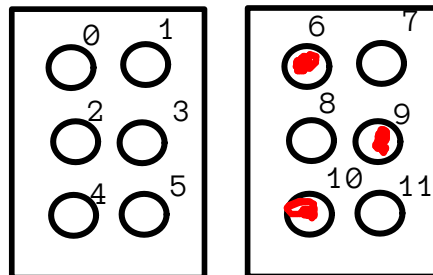


H en braille



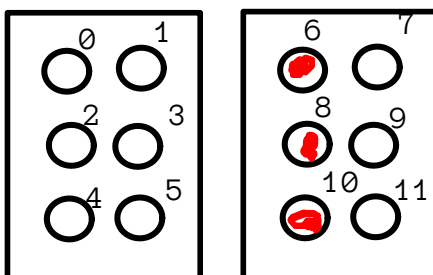
"010001101100H"

o en braille



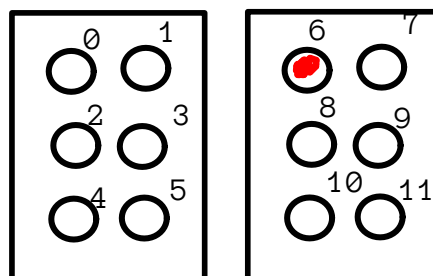
"000000100110o"

l en braille



"000000101010l"

a en braille



"000000100000a"

```
def braille_generator("Hola")
    return "010001101100H,000000100110o,000000101010l,000000100000a"
```

IMP un espacio es todos apagados y seria asi

" " = "000000000000 "

Vea que el espacio tambien se agrega





Links importantes

<https://youtu.be/2Gi3iBMTIbl>  
<https://youtu.be/685pnyaNpFU>  
<http://www.brailletranslator.org/es.html>  
<http://www.brl.org/intro/session02/abcs.html>  
[http://braillebug.afb.org/braille\\_deciphering.asp](http://braillebug.afb.org/braille_deciphering.asp)  
<https://youtu.be/R0cbckyD9Q0>

## Requerimientos del proyecto

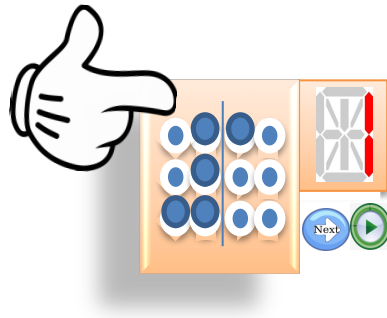
El alfabeto Braille ha sido implementado para permitir una mayor accesibilidad de personas con algún tipo de discapacidad, principalmente discapacidad visual, para que tengan la posibilidad de interactuar con su medio externo. Hoy en día se tienen medios rudimentarios para tal posibilidad, pero uno de los problemas que se les presenta a estas personas es no contar con la “traducción” a braille de manera inmediata y oportuna. La idea del proyecto es intentar satisfacer esa necesidad, permitiendo la “traducción” online de documentos digitales, así como también permitir realizar la “traducción” de una manera más eficiente y fuera de elementos informáticos.

A continuación, se presentan los requerimientos técnicos del proyecto. Se expresa el requerimiento y se presenta una imagen que ejemplifica dicho requerimiento. La idea es el cumplimiento del requerimiento usando el criterio del grupo en lo que respecta a elementos de hardware a utilizar, por tanto, la imagen solamente pretende dar una idea más amplia del requerimiento.

1. Se debe construir un lenguaje de programación en el cual se puedan implementar una texto o serie de estos para enviar al dispositivo.
2. Se debe tener un dispositivo hardware que debe tener el impulso (algo tactible) que simule el impulso de cada símbolo.
  - a. Este dispositivo debe tener como mínimo 6 elementos que generen un impulso a tacto que simulen el código braille.
  - b. Además de lo anterior, debe tener uno de los dos siguientes probables elementos:



- i. Una bocina que vaya indicando cada símbolo colocado.
- ii. Una pantalla o leds que simulen los 6 puntos del impulso actual dado por el dispositivo.



El objetivo primordial el proyecto es suplir a la persona con necesidad de reconocer alfabetos braille, un mecanismo en donde pueda ser capaz de “sentir” un texto, la letra de una canción, el ritmo de una melodía, etc.

Para efectos del proyecto se debe contemplar todo al alfabeto braille en español, dentro del cual se tomará en consideración lo siguiente:

- El alfabeto completo en español
- Todos los números
- Tildes en las vocales, la letra “ñ”
- Símbolos especiales (suma, resta, división, multiplicación, igualdad.
- Puntuación (punto, coma, mayúscula, punto y coma, comillas, guion, paréntesis, signos de interrogación – debidamente colocados, signos de exclamación – debidamente colocados, etc.)

Pueden utilizar **Lex y Yacc** para efectos de los analizadores léxicos y sintácticos en el proyecto.

Tomar como base el documento de braille en español accediendo al siguiente link:

<http://sid.usal.es/ids/F8/FD012069/signografiabasica.pdf>

## Entorno de Programación (IDE)

Se debe crear una interfaz gráfica la cual será el **único medio de revisión del proyecto**. En esta interfaz se deben ver claramente los errores y warnings del compilador. Esta interfaz debe contener los siguientes elementos:

- Poder ser capaces de “cargar” programas predefinidos para **modificarlos, compilarlos y ejecutarlos**.
- Debe existir un **botón de “compilación”** el cual no ejecute el código sino simplemente realiza “pasadas” de validación según cada etapa de los compiladores.
- Debe existir un **botón de “ejecución”** el cual ejecutará primero la “compilación” y posteriormente de no existir ningún error, ejecutará el código.
- Debe haber una **ventana** en donde se pueda cargar o editar los programas y sobre los cuales se ejecutará la compilación y ejecución del código.
- Debe existir una ventana en donde se retornen los valores del comando “PrintValues”.
- Debe haber una ventana en donde **aparecerán de forma “decente” los errores** que la “compilación” generará. No se permitirá el uso del IDE del lenguaje de programación usado para construir el Compilador ni ningún otro medio. Solamente se hará uso de la interfaz solicitada.
- Se debe tener **control sobre el número de línea del código fuente**, de tal manera que en caso de presentarse un error a la hora de la compilación, se muestre **exactamente** la línea sobre la cual se presenta el error, además el error presentado debe ser **amigable**.

## Sintaxis

- Se debe respetar la sintaxis que más adelante se indica, pero el grupo puede “enriquecer” dicha gramática añadiendo las sentencias o comportamiento que consideren necesario, pero sin eliminar o cambiar nada de la sintaxis colocada en este documento.

- Se deben tomar en cuenta las siguientes instrucciones:

- Las **variables o identificadores** deben tener las siguientes características:

- Un máximo de 12 posiciones
- Un mínimo de 2 posiciones
- Siempre debe iniciar con una @ (arroa)
- Debe permitir letras, números y los símbolos “?” y “\_” (underscore)
- La @ (arroa) solamente puede estar en la primer posición
- Si no se cumple algunos de los puntos anteriores, **se debe generar un error.**

- Los comentarios en el código fuente es por línea, y siempre debe iniciar con (//) y se comenta toda la línea, ejemplo:

// Esto es un comentario

Utilizar con clases y funciones

Clase de tipo Error para Lenguaje

```
from arhcivo.py import funcion1()
from archivo.py import clase1
```

```
def funcion1():
```

```
class clase1:
```

```
    def _int_():
        jdsakljfs
```

GUI es una clase  
para comomidad

Hablar con que van hacer el lenguaje

Repo de GitHub

cada uno en un branch\_diferente ejemplo:

```
carlos_compilador
david_gui
jose_arduino ....
```

hardware

```
archivos_python
arduino
```

lenguaje

```
main_compilador.py
```

IDE

```
main_gui.py
```

.gitignore

README.md

La GUI tiene un metodo que hace esto

```
gui.outputConsole("ajdlksfjjlfs")
```

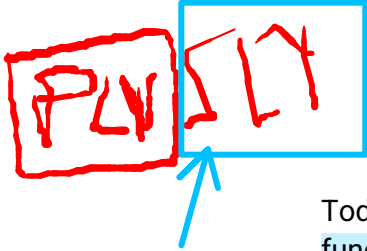
```
gui.showError("Errora sdfjakjgklajhfskl")
```

```
gui.openFile("kads1j1fsj")
```

funciones o clases

master

- Se abra un txt
- Haya un funcion @Master
- Imprima mensaje de compilado
- Sino imprime el error



comentar el código por cada función es obligatorio

Todo código fuente debe tener al menos un comentario indicando el nombre y la funcionalidad del código, y esta debe encontrarse en la primera línea de todo programa. De lo contrario **debe generar un error**. Los demás comentarios pueden estar presentes en cualquier lugar dentro del código del programa, pero siempre debe existir al menos uno en la primera línea de este.

Esto primero a hacer

- Deben realizar todas las validaciones pertinentes a nivel léxico, sintáctico, semántico, etc.
- El lenguaje de programación permite y debe utilizar procedimientos.

De las primeras cosas que hay que hacer en el lenguaje

- El cuerpo básico de un procedimiento debe cumplir con el siguiente formato:

**Proc** nombre\_del\_procedimiento

```
(  
... // Instrucciones.  
);
```

si se usa como var genera error

Donde **Proc** es una palabra reservada, los paréntesis agrupan las sentencias o cuerpo del procedimiento, y el nombre del procedimiento debe respetar las mismas reglas de las variables e identificadores.

otro error

- No puede existir más de un procedimiento con el mismo nombre, pues generaría un error.

una función puede llamar a otra

- También como en otros entornos de programación un procedimiento puede llamar a otro procedimiento, lo que hará esto es realizar las instrucciones que tiene definidas el procedimiento al que se llama.

aquí 3 error

- Debe existir un procedimiento llamado "**@Master**" que es el principal y el punto de entrada al programa. Este procedimiento no recibe parámetros. Se debe generar un error si este procedimiento no se encuentra, o bien, si hay más de uno en un programa en particular. Además, se debe generar un error si se le coloca parámetros.

- El programa "**@Master**" puede estar colocado en cualquier parte del código.
- Las variables definidas dentro de "**@Master**" se consideran variables globales y las variables definidas en cualquier procedimiento se consideran variables locales.
- Los procedimientos se ejecutan de la siguiente manera  
**CALL**( nombre\_del\_procedimiento );


No puede existir código fuera de un procedimiento, y el punto de entrada es el procedimiento "**@Master**".

- **Sintaxis mínima solicitada:**

Sentencia en Lenguaje	Acción de la sentencia
<pre><b>New</b> nombre_variable,       (tipodatos, valor);</pre>	<p>A una variable se le puede asignar un valor, este valor puede ir cambiando a lo largo de la ejecución de un programa, el tipo de datos es "fijo" en el sentido de que no puede cambiar. El valor debe ser congruente con el tipo de dato que se quiera guardar en la variable. Ésta puede guardar los siguientes tipos de datos:</p> <ul style="list-style-type: none"> <li>- Numero (Num)</li> <li>- Lógico (Bool)</li> </ul> <p>Ej.</p> <pre><b>New</b> @variable1, (Num, 5);</pre> <pre><b>New</b> @variable2, (Bool, True);</pre> <p>Tanto los tipos de datos (Num, Bool) como los valores booleanos (True, False) siempre inician con mayúscula. Caso contrario debe generar un error.</p> <p>Si el dato de inicialización de la variable no corresponde con el tipo dado, se debe generar un error.</p> <p>Si posteriormente se le asigna otro valor de distinto tipo de datos, se deberá generar un error "semántico".</p> <p>Si no se escribe la sentencia correctamente, o se obvia algún valor, se debe generar un error de sintaxis.</p> <p>Puede haber definiciones de variables en cualquier lado del código.</p> <p>Si la variable ya existe, entonces no se debe permitir volver a crearla, por tanto, debe enviar un mensaje de error.</p>
<pre><b>Values</b> (nombre_variable,         valor);</pre>	<p>Altera el valor actual de la variable por el dato colocado en la instrucción. El valor usado puede ser una constante (numero o valor booleano) o el resultado de una función.</p>

	<p>Ej.</p> <pre>Values (@variable1, 51);</pre> <pre>Values (@variable2, False);</pre> <pre>Values (@variable1,         Alter (@variable1, SUB, 3));</pre> <p>Acepta no solamente un valor sino también una operación.</p>
<pre>Alter (nombre_variable,         operador, valor);</pre>	<p>Modifica el valor de una variable, y retorna el resultado.</p> <p>Operador puede tener uno de los siguientes valores:</p> <ul style="list-style-type: none"> <li>• ADD (para sumar)</li> <li>• SUB (para restar)</li> <li>• MUL (para multiplicar)</li> <li>• DIV (para dividir)</li> </ul> <p>Se tiene que usar solamente esos valores y de la forma presentada, sino debe devolver un error de sintaxis.</p> <p>El valor puede ser una constante o el resultado de una función.</p> <p>Solo aplica para variables numéricas, caso contrario debe devolver un error.</p> <p>Ej.</p> <p>Estas instrucciones retornan el resultado de la operación.</p> <pre>Alter (@variable1, ADD, 5)</pre> <pre>Alter (@variable1, SUB, 3;</pre> <pre>Alter (@variable1, MUL, 2)</pre>

	<pre>Alter (@variable1, DIV, 1;</pre> <p>La instrucción <i>Alter</i> se utiliza en otras instrucciones para utilizar el valor retornado, por ejemplo:</p> <pre>Values (@variable1, Alter         (@variable1, SUB, 3));</pre>
<b>AlterB</b> (nombre_variable);	<p>Modifica el valor de una variable booleana, colocando el valor contrario.</p> <p>-- @variable2 es True</p> <pre>AlterB (@variable2);</pre> <p>-- @variable2 ahora es False</p>
<b>Signal</b> (position, estado);	<p>Activa una posición del sensor.</p> <p>El "sensor" tendrá por lo menos 6 elementos, los cuales generarán el movimiento táctil.</p> <p>Signal activa o desactiva alguno de estos en un momento determinado.</p> <p>Position puede ser uno de los valores del sensor, por ejemplo, del 1-5</p> <p>En el campo de posición puede tener un numero (constante), una variable, o una operación matemática.</p> <p>Estado tendrá dos posibles valores:</p> <ul style="list-style-type: none"> <li>• 1 de Activado, el dispositivo en ese punto se puede "sentir"</li> <li>• 0 de Desactivado, el dispositivo en ese punto NO se puede "sentir"</li> </ul> <p>Ej.</p>

	<p>Por ejemplo. <b>Signal</b>(1, 1);</p> <p>Contextualmente representa la A, si los demás valores están deshabilitados.</p>  <p>El valor se mantendrá hasta que se deshabilite esa posición, de tal manera que se podrán colocar varias posiciones activas a la vez.</p> <p>Cualquier otro valor fuera de los rangos mencionados, se debe interpretar como un error por el intérprete.</p>
<div style="border: 2px solid red; padding: 2px; display: inline-block;"><b>ViewSignal</b></div> (position);	<p>Retorna el estado de la posición en el dispositivo. Los valores retornados sería 0 ó 1</p> <p>Ej.</p> <p>@Var1 = <b>ViewSignal</b> (5);</p>
<p>Condicionales Numéricas</p> <p>Operando1 operando Operando2</p>	<p>Las condicionales usadas en la sintaxis son las siguientes:</p> <ul style="list-style-type: none"> <li>• &gt; (mayor qué)</li> <li>• &lt; (menor qué)</li> <li>• == (igual)</li> <li>• &lt;&gt; (distinto qué)</li> <li>• &lt;= (menor o igual qué)</li> <li>• &gt;= (mayor o igual qué)</li> </ul> <p>Ej.</p> <p>@variabl1 &gt; 10</p> <p>@variabl1 &lt;&gt; 5</p> <p>Devuelve un valor booleano de acuerdo con el resultado de la condición.</p> <p>El operando2 podría ser el resultado de una función.</p>



	<p>Si se utiliza como operando una variable booleana, se debe retornar error.</p> <p>Se puede utilizar <code>True</code> o <code>False</code> como valor de comparación en las condicionales.</p>
<p>Condicionales booleanas</p> <p><b>IsTrue</b> (nombre_variable);</p>	<p>Devuelve True si el valor que contiene la variable es True de lo contrario devuelve False.</p> <p>Ej. <code>IsTrue (@variabl2)</code></p> <p>Si se utiliza una variable numérica se debe retornar error.</p>
<p><b>Repeat</b> (instrucciones);</p>	<p>Ejecuta todas las instrucciones que se encuentran en el cuerpo hasta que se encuentra una instrucción <b>"break"</b>.</p> <p>Si no hay en el cuerpo de instrucciones, una instrucción <b>"break"</b> <u>se debe generar un error</u>.</p> <p>En el cuerpo de instrucciones puede existir cualquier otra instrucción mencionada en este documento. El cuerpo de instrucciones debe estar entre paréntesis.</p> <p>Ej. <b>Repeat</b> (   <b>New</b> @variable1, (Num, 5);   <b>Signal</b>(1, 1);    Break; );</p>
<p><b>Until</b>   (instrucciones])   Condición;</p>	<p>Repite la lista de instrucciones tantas veces hasta que se cumpla la condición.</p> <p>Primero ejecuta el conjunto de instrucciones, de esta forma se asegura que las instrucciones se ejecutan al menos una vez antes de comprobar la condición.</p> <p>En caso de no cumplirse la condición, se vuelve a ejecutar el conjunto de instrucciones.</p>

	<p>Ej.</p> <pre>-- inicializa la variable en 1 <b>Values</b> (@variable1, 1);  <b>Until</b> (  MoverLeft;    MoverRight; -- Aumenta en 1 a la variable <b>Values</b> (@variable1,         <b>Alter</b> (@variable1, ADD, 1)); ) @variable1 &gt; 10;</pre> <p>Observe que el cuerpo de la instrucción se encuentra entre paréntesis, y la condición viene inmediatamente después del cuerpo de la instrucción.</p>
<p><b>While</b> condición     (instrucciones);</p>	<p>Repite la lista de instrucciones tanta veces hasta que se cumpla la condición. Si la condición expresada NO se cumple no se ejecutan las instrucciones ni una sola vez.</p> <p>Observe que el cuerpo de la instrucción se encuentra entre paréntesis, y la condición viene inmediatamente después del cuerpo de la instrucción.</p> <p>Ej.</p> <pre>-- Inicializa la variable en True <b>Values</b> (@variable2, True);  <b>While</b>  <b>IsTrue</b>(@variable2) (  <b>Signal</b>(@variable2, 1);  -- Cambia variable a False    <b>AlterB</b>  (@variable2); );  -- Inicializa la variable en 100 <b>Values</b> (@variable1, 100);  <b>While</b>  @variable1 &gt; 10</pre>

	<pre>( Signal(@variable1, 1);  -- Resta 10 de variable Values (@variable1,         Alter (@variable1, SUB, 10)); );</pre>
<pre>Case nombre_variable   When Valor Then     (instrucciones)   When Valor Then     (instrucciones) [When Valor Then   (instrucciones) ] Else   (instrucciones);</pre>	<p>Consulta el valor de la variable, y dependiendo de ese valor ejecuta el cuerpo de instrucciones asignado.</p> <p>La instrucción ELSE es opcional.</p> <p>Ej.</p> <pre>Case @variable1   When 1 Then     ( Signal(@variable1, 1);)   When 2 Then     ( Signal(@variable2, 1))   When 3 Then     ( Signal(@variable3, 1));  Case @variable2   When True Then     ( Signal(@variable2, 1);)   Else     ( Signal(@variable2, 1));</pre>
<pre>PrintValues(valor)</pre>	<p>Imprime en pantalla el valor dado, donde el valor dado puede ser un string, una variable, o una o más combinación de ambos.</p> <pre>PrintValues ("Hola Mundo");</pre> <p>Esta operación genera un cuadro de dialogo a nivel del software presentando en la pantalla el texto colocado.</p> <p>Puede recibir como parámetro un texto, una variable o una constante.</p> <p>Puede recibir más de un parámetro a la vez.</p>

	<pre> <b>Values</b> (@variable1, 1);  <b>PrintValues</b> ("Este es el Proyecto número ", @variable1, " del Compi 2023");  Imprime  "Este es el proyecto número 1 de Compi 2023"  Este comando <b>PrintValues</b> es sumamente importante pues será la manera de revisar a nivel de software la lógica del código.  <b>Si no se implementa no será posible revisar el código del compilador, y se obtendrá una nota de 0 en dicho rubro.</b> </pre>
--	--

Para la solución del problema del proyecto presentado se espera del estudiante:

- Fuerte investigación sobre las posibles soluciones en las distintas etapas del problema
  - Búsqueda de distintas fuentes que servirán de insumo técnico-práctico para las soluciones, así como la ayuda de otras áreas para cumplir con el objetivo del proyecto.
  - Selección de criterios acordes al nivel de nuestro ambiente **Tec** en donde se seleccionen las mejores soluciones correspondientes.
  - Una solución solvente de acuerdo con lo esperado.
-

# Aspectos de Revisión del Proyecto

---

Para efectos de la revisión del proyecto, se solicita lo siguiente:

1. **El grupo deber presentar un programa completo de complejidad intermedia alta** que demuestre la programación de acuerdo con la sintaxis mostrada y que se evidencie e dispositivo “traduciendo” un texto programado en el lenguaje de programación.
2. Algún miembro del grupo debe contar con la posibilidad de **codificar durante la revisión un programa con un texto que el profesor lleve** a la revisión, y que será propio para el grupo de revisión.
3. El profesor usará **un programa en el cual se codificará una serie de “errores”** que el compilador deberá reconocer.
4. El profesor usará un **programa en el cual se codificará todos y cada uno de los elementos de la sintaxis previamente presentado**, el cual, el compilador deberá reconocer y el hardware deberá reconocer.

## Documentación y aspectos operativos

- Se deberá entregar un documento que contenga:
- ✓ Diagrama de arquitectura de la solución que refleje un nivel de detalle suficiente para entender a términos generales el funcionamiento del proyecto.
  - ✓ Problemas conocidos: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
  - ✓ Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.
  - ✓ Conclusiones y Recomendaciones del proyecto. Conclusiones y recomendaciones con sentido y que confirmen una experiencia profunda en el proyecto.
  - ✓ Bibliografía consultada en todo el proyecto

# Atributos

---

Se debe **entregar un documento aparte** que contenga una portada y el siguiente detalle.

Debe comentar de qué manera se aplicaron los atributos que posee el curso y su impacto, esto por medio de una tabla en donde se detalle para cada atributo lo siguiente:

- Aplicación del atributo en la solución del proyecto
- Impacto del proyecto en la sociedad
- Retroalimentación obtenida gracias al proyecto

Favor colocar la información antes solicitada para cada uno de los atributos siguientes:

- **Conocimiento base de ingeniería**
- **Impacto de la ingeniería en la sociedad y el medio ambiente**
- **Aprendizaje continuo**

Conocimiento de ingeniería
Capacidad para aplicar los conocimientos a nivel universitario de matemáticas, ciencias naturales, fundamentos de ingeniería y conocimientos especializados de ingeniería para la solución de problemas complejos de ingeniería.
Ambiente y Sostenibilidad
Capacidad para comprender y evaluar la sostenibilidad y el impacto del trabajo profesional de ingeniería, en la solución de problemas complejos de ingeniería en los contextos sociales y ambientales.
Aprendizaje Continuo
Capacidad para reconocer las necesidades propias de aprendizaje y la habilidad de vincularse en un proceso de aprendizaje independiente durante toda la vida, en un contexto de amplio cambio tecnológico.

# Evaluación

---

1. La implementación del dispositivo corresponde a un 40% de la nota final del proyecto.
2. El Intérprete representa un 50% de la nota final del proyecto y comunicación con la interfaz.
3. La documentación tendrá un valor de un 10% de la nota final del proyecto, cumplir con los puntos especificados en la documentación no significa que se tienen todos los puntos.
4. Cada grupo recibirá una nota en cada uno de los siguientes apartados:
  - a. Funcionalidad
  - b. Compilador
  - c. Documentación
5. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
6. No debe imprimirse la documentación, y esta debe encontrarse en formato PDF.
7. La entrega debe hacerse usando el **TECDigital**
8. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
9. La evaluación utilizada en la sesión de "revisión" del proyecto se basa en el cumplimiento de los elementos indicados en la sección "**Aspectos de Revisión del Proyecto**" del presente documento.
10. El grupo debe contar con un grupo de procedimientos definidos previamente que demuestre los avances en la ejecución del proyecto. Estos procedimientos deben cumplir con las reglas indicadas previamente en este documento.
11. Dentro de la revisión del proyecto se deberá presentar la gramática creada y dar una explicación detallada de la manera en que se implementó el compilador mostrando todos los elementos necesarios para entender su funcionalidad.
12. Todos los integrantes del grupo deberán estar presente en la defensa del proyecto, y todos deben estar preparados para contestar las preguntas que se les puede realizar en base a la implementación realizada. Solamente en casos justificados y vistos previamente con por lo menos un día de anticipación se podrá ausentar algún miembro del equipo.
13. Aun cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
  - a. Si no se entrega documentación, se penalizará fuertemente la evaluación del proyecto.
  - b. Sí la documentación no se entrega en la fecha indicada, se penalizará fuertemente la evaluación del proyecto.
  - c. Sí no hay integración entre el compilador y el dispositivo (hardware), se penalizará fuertemente la evaluación del proyecto.
14. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de arquitectura, la Gramática y su implementación, y cualquier otra documentación que el profesor considere necesaria.
15. Cada grupo tendrá como máximo 25 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa, y será responsabilidad

del grupo administrar el tiempo dispuesto para su revisión de tal manera que el profesor pueda observar todo el producto terminado.

16. Cada grupo es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberán avisar al menos 2 días antes de la revisión a el profesor para coordinar el préstamo de estos.
17. No se revisarán funcionalidades incompletas o no integradas.
18. Durante la revisión únicamente podrán participar los miembros del grupo, asistentes, otros profesores y el coordinador del área.