# Estructuras que debemos conocer

## 0. Contadores

Esta sencilla estructura nos va a permitir **contar** eventos o sucesos. Esto nos permitirá, con ayuda de los condicionales e iteradores, tomar decisiones en función de cuántas veces haya sucedido un evento.

> Importante: en el mundo de la informática normalmente se **empieza a contar por el numero 0**. Luego en una lista de 8 elementos, el primer elemento ocupara la posición 0 y el octavo elemento la posición 7.

El ejemplo más sencillo y básico de contador consiste en crear una **variable** dentro del proceso a contar y **sumarle** a esta variable 1 cada vez que se ejecute el proceso. Asignaremos como valor de partida el número 0. Y cada vez que el proceso se ejecute, se sumará 1 a esa variable.

> Una **variable** es un objeto que almacenará un valor, ese valor podrá modificarse a lo largo del programa. Por ejemplo, **int i = 0** traducido a nuestro idioma sería: 'la variable "i" pertenece al tipo número entero y tiene el valor 0'.

Por lo general en un contador, la variable que creemos será del tipo integer, abreviado **int** –ya habíamos hablado de tipos de datos en este apartado–, ya que para contar nos conviene utilizar números enteros. En los lenguajes de programación textual, tradicionalmente, se utiliza la letra "**i**" para nombrar las variables contador, pero se puede utilizar la letra o nombre que queráis y si tenéis varias variables tendréis que utilizar nombres diferentes para cada una de ellas.

**int i=0;**
**int i = i+1;**

Que significa: "El entero i va a tomar el valor de si mismo más uno."
Eso se traduce en que i pasará a valer **0 + 1 = 1**.

Vamos a ver ahora un ejemplo de que pasaría en cada repetición de un programa en el que la variable contador se incrementa 1 unidad:

Antes del Arranque del programa: Variable contador = 0

Arranque del programa:

Inicio del proceso: (Incrementar en 1 unidad el valor de la variable contador: Variable contador = Variable contador + 1)

Variable contador = 0 + 1

Fin del proceso.

Si lo repetimos una segunda vez, tenemos:

Inicio del proceso: (Incrementar en 1 unidad el valor de la variable contador: Variable contador = Variable contador + 1)

Variable contador = 1 + 1

Fin del proceso.

Si lo repetimos una tercera vez, tenemos:

Inicio del proceso: (Incrementar en 1 unidad el valor de la variable contador: Variable contador = Variable contador + 1)

Variable contador = 2 + 1

Fin del proceso.

Cuarta repetición:

Inicio del proceso: (Incrementar en 1 unidad el valor de la variable contador: Variable contador = Variable contador + 1)

Variable contador = 3 + 1

Fin del proceso.

...

Cada vez que se ejecuta el proceso se le suma 1 al resultado del proceso anterior. Si el proceso se ejecuta 8 veces el valor de la Variable contador tra esas repeticiones sera 8.

> 

> Ejercicio 5: ¿Qué valor tendría la variable contador después de que el proceso se ejecute 8 veces si en cada proceso sumamos 2 en lugar de uno. Variable contador = Variable contador +2?

> Ejercicio 6: Y, ¿si el proceso se realiza 15 veces y la cantidad sumada en cada proceso es 3? Variable contador= Variable contador + 3?

# 1. Condicionales (if-else-else if)...

Estas estructuras se encargan de controlar qué acciones de nuestro algoritmo van a ejecutarse y cuáles no. Solo hay dos opciones, o bien se cumplirán las condiciones, o no.

En ejemplo del control del nivel del agua en Zaragoza tenemos un claro condicional:

## Condición, acción si la condición se cumple, acción si la condición no se cumple

En Zaragoza el nivel del agua del rio se mide una vez (Proceso 0: contar) todos los días (Proceso 1: comparar) y se apunta en una libreta (Proceso 2: almacenar). Este nivel, ha de ser siempre superior a 50 (Proceso 3: comparar). Cuando el nivel de agua en Zaragoza desciende por debajo de 50, se avisará al embalse (Proceso 4.1.1: enviar información) para que deje salir más agua (Proceso 4.1.2: cambiar), si el nivel de agua no ha bajado por debajo de 50, no se avisara al embalse y el agua que deja salir este, seguirá siendo la misma (Proceso 4.2: no cambiar).

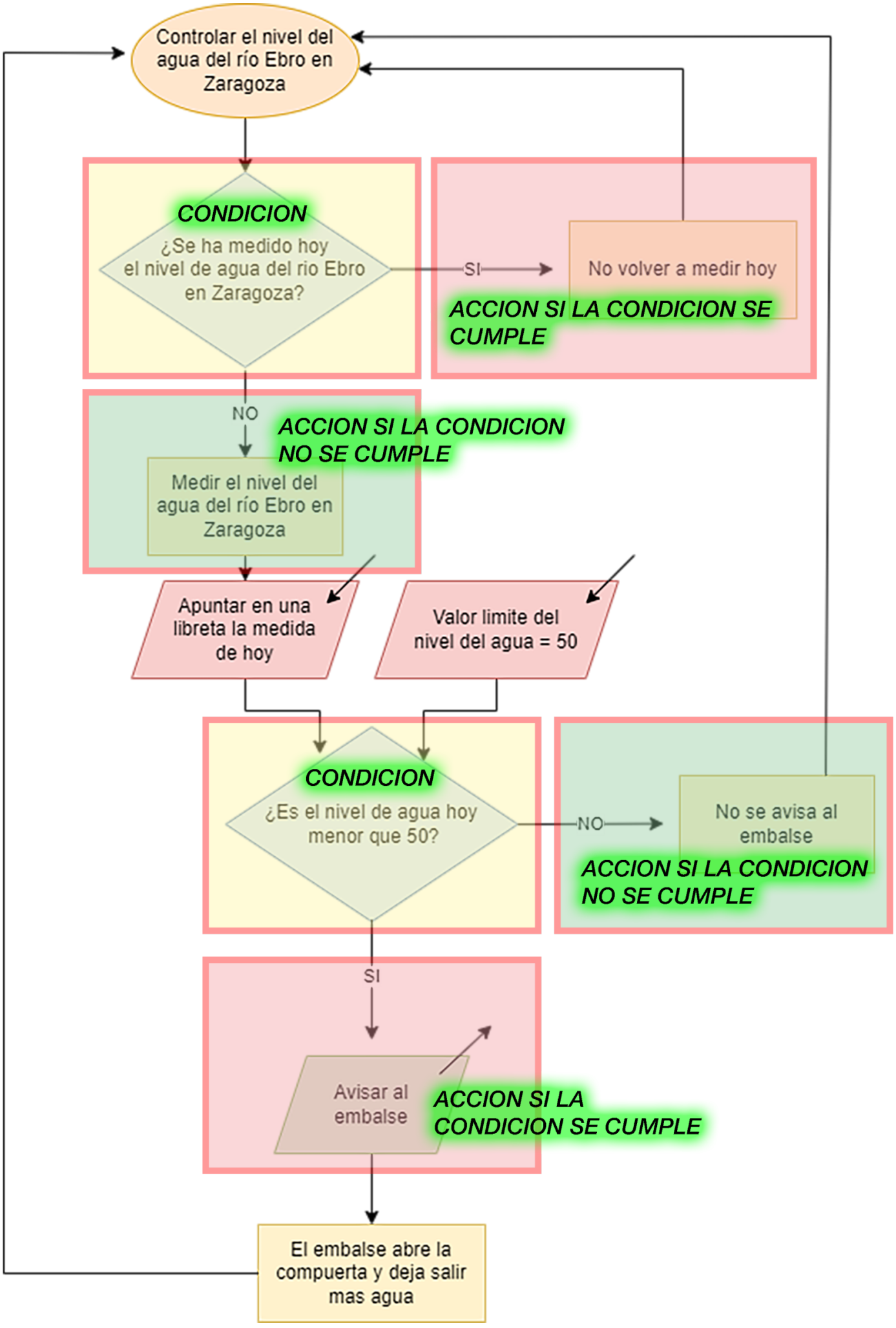
Controlar el nivel del agua del río Ebro en Zaragoza
CONDICION
¿Se ha medido hoy el nivel de agua del rio Ebro en Zaragoza?
SI
No volver a medir hoy
ACCION SI LA CONDICION SE CUMPLE
NO
ACCION SI LA CONDICION NO SE CUMPLE
Medir el nivel del agua del río Ebro en Zaragoza
Apuntar en una libreta la medida de hoy
Valor limite del nivel del agua = 50
CONDICION
¿Es el nivel de agua hoy menor que 50?
NO
No se avisa al embalse
ACCION SI LA CONDICION NO SE CUMPLE
SI
Avisar al embalse
ACCION SI LA CONDICION SE CUMPLE
El embalse abre la compuerta y deja salir mas agua

Figura 1. Condicionales marcados en el diagrama de flujo del control del nivel del agua del río Ebro en Zaragoza.

En los lenguajes de **programación visual** esta estructura se va a construir de **manera diferente** a en los lenguajes de **programación textual**.

### *En Arduino ...*

Las palabras clave de esta estructura son:

**IF - ELSE**

**IF:** A esta palabra le seguirá la condición que deberá cumplirse para ejecutar una serie de acciones

.

**ELSE:** irá seguida de las acciones que se ejecutarán en caso de que la condición no se cumpla.

## Un ejemplo para verlo en acción

No hay mejor manera para entender algo que ver una demostración de su uso. Para ello, vamos a recurrir a uno de los ejemplos que nos brinda la IDE (Integrated Development Environment, el lugar en el que vamos a escribir nuestros programas) de Arduino. Este ejemplo podemos encontrarlo dentro de:

**Archivo > Ejemplos > 01. Basics > Fade**

Aunque, como de momento no hemos instalado Arduino en nuestro ordenador, simplemente échale un vistazo al código de la siguiente imagen:


Figura 2. Imagen de la IDE de Arduino

En este ejemplo, lo que hacemos es aumentar y disminuir la luminosidad de un LED de manera progresiva.

De este ejemplo, por el momento, vamos a ver concretamente la parte relacionada con el condicional. En este algoritmo, el condicional aparece en las siguientes líneas.

```
if (brightness <= 0 || brightness >= 255) {
  fadeAmount = -fadeAmount;
}
```

En estas líneas encontramos la palabra reservada **if** seguida de dos condiciones encerradas entre paréntesis. Lo que traducido a español significaría: "Si el brillo es menor o igual que cero **o** el brillo es mayor o igual que 255". La disyunción nos viene dada por el operador lógico **OR** que para Arduino se traduce como **||.** Otros operadores lógicos muy comunes son **AND,** que se traduce por **&&** y **NOT,** que se escribe **!**.

Entre corchetes **{ }** encontramos la acción a realizar. En este caso, revertir la cantidad de brillo que emitirá nuestro LED: si lo hemos apagado, lo comenzaremos a encender y al revés, si lo hemos encendido completamente, comenzaremos a apagarlo.
En este caso, no existe una acción que deba ejecutarse en caso de que no se de alguna de las dos circunstancias encerradas en el condicional, pero si la hubiese aparecería detrás de la palabra **else** .

Por tanto, un esquema de la estructura de los condicionales sería:

```
if (Condición) {
   Acción A
} else {
   Acción B
}
```

Siendo posible también:

```
if (Condición) {
   Acción
}
```

## Una última palabra sobre los condicionales: **else if**

Existe también la posibilidad de que tengamos que realizar selecciones que den lugar a más de dos posibilidades, como por ejemplo en el caso que estemos leyendo ciertos valores de un sensor y queramos que un actuador realice diferentes acciones dependiendo de ellos.

Si el sensor recibe valores entre 0 y 255 --> Acción A
Si el sensor recibe valores entre 256 y 511 --> Acción B
Si el sensor recibe valores entre 512 y 1024 --> Acción C

Para ello necesitaremos las palabras **else if,** las cuales indican una segunda condición, o sucesivas.

La estructura sería la siguiente:

```
if (Condición 1) {
   Acción A
} else if (Condición 2) {
   Acción B
} else{
   Acción C
}
```

> Puede parecer algo lioso en un primer momento, pero conforme avancemos y veamos ejemplos, te familiarizarás con estas estructuras.

*En Pure Data ...*

En Pure Data que es el lenguaje visual que vamos a aprender en este curso se pueden crear estructuras condicionales utilizando diferentes objetos, por ejemplo algunos objetos **operadores** o el objeto "**select**" o el objeto "**spigot**" que es una puerta que se abre y deja pasar datos o se cierra y no los deja pasar. Veremos más adelante como construir estas estructuras de momento es importante que comprendáis la lógica del condicional.

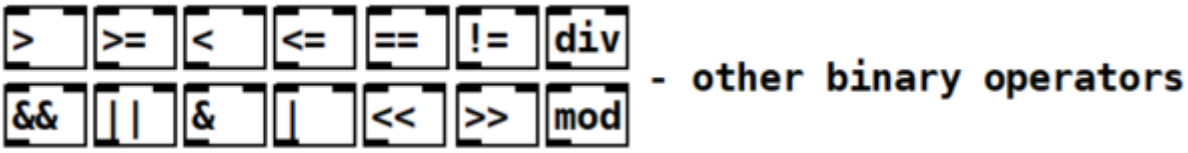


Figura 3. operadores de pure data que nos van a permitir comparar valores o caracteres. Cuando la comparación se cumpla enviaran un 1 por su salida, si no se cumple, enviaran un 0.

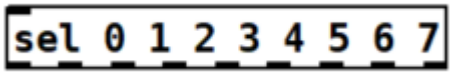


Figura 4. objeto "select" de pure data que nos va a permitir clasificar en base a una comparación. En la imagen se comparan si el valor que llegue al objeto es igual que 1, igual que 2, igual que 3, igual que 4, igual que 5, igual que 6, igual que 7 o igual que 8 y lo envía por la salida correspondiente a uno de los números anteriores.

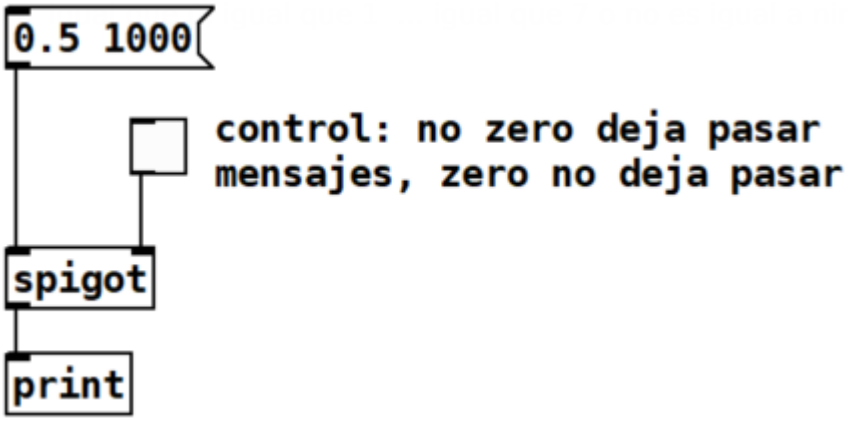


Figura 5. Cuando la cajita de control envíe un 1 la puerta se abrirá y el mensaje "0.5 1000" llegara al objeto "print". En la imagen el control está cerrado y ha enviado un 0, por lo que la puerta esta cerrada y el mensaje no llega al objeto "print".

# 2. Bucles o Iteraciones (for/while)...

Definiremos como iteración a la ejecución sistemática, repetida, de una serie de acciones mientras se dé una condición. Existe una **condición** que se analiza, normalmente, cada vez que se repite dicha iteración.

Si esta condición se sigue cumpliendo, volveremos a repetir el proceso; si, por el contrario, ya no se cumple, pasaremos al siguiente proceso y ejecutaremos la acción correspondiente.

## *En Arduino ...*

Las dos iteraciones más comunes son **for** y **while**. En la sentencia for, existe un índice cuyo valor va aumentando o disminuyendo conforme se ejecuta el algoritmo y la condición de la iteración va comprobando si se ha alcanzado el límite o no.

Un esquema de esta estructura sería:

```
for (int indice= inicial; indice<= final; indice ++){
   Acción
}
```

Para entenderlo con un ejemplo, imagina que tienes una caja en la que caben solamente 10 cubos. Al principio, la caja estaría vacía, por lo que el valor de **indice = 0.** La caja se llena cuando hay 10 cubos, por lo que **indice<=10.** Y lo que queremos hacer es añadir cajas **de una en una,** por lo que tenemos que escribir que el valor de indice aumente utilizando la expresión **indice ++.** En acción es donde iríamos añadiendo una caja de bombones cada vez.

Así que, nuestro **for** añadirá bombones hasta que nuestro índice sea menor o igual al valor de final, que en este caso es 10.

> Es importante no olvidar colocar los paréntesis, puntos y comas y los corchetes donde corresponda, o Arduino se quejará y no nos hará caso...

A parte del bucle **for** existe el bucle **while**, cuyo esquema general sería:

```
while (condición){
   Acción
}
```

Un ejemplo, para verlo en funcionamiento aunque no tengo mucha utilidad real, sería:

```
int var = 0;
while (var < 200) {
  // haz algo 200 veces
  var=var+1; //también podría escribirse como var++.
}
```

En él, hemos creado una **variable** con el valor 0, la cuál es aumentada dentro del bucle while con la operación **var = var+1.** Antes de aumentar este valor, en la línea superior deberíamos escribir la acción que querríamos ejecutar 200 veces.


REFERENCIAS:
Sentencia *for* en Arduino y ejemplo:

https://www.arduino.cc/reference/en/language/structure/control-structure/for/
Sentencia *while* en Arduino y ejemplo:

https://www.arduino.cc/reference/en/language/structure/control-structure/while/Cerrada Somolinos, J. A., & Collado Machuca, M. E. (2015). *Fundamentos de programación*. Editorial Universitaria Ramón Areces : UNED.

Puerto serie: https://www.luisllamas.es/arduino-puerto-serie/


Figuras:

Figura 1. Condicionales marcados en el diagrama de flujo del control del nivel del agua del río Ebro en Zaragoza.

Figura 2. Captura de pantalla de la IDE de Arduino

Figura 3. operadores de pure data que nos van a permitir comparar valores o caracteres. Cuando la comparación se cumpla enviaran un 1 por su salida, si no se cumple, enviaran un 0.

Figura 4. objeto "select" de pure data que nos va a permitir clasificar en base a una comparación. En la imagen se comparan si el valor que llegue al objeto es igual que 0, igual que 1  ... igual que 7 o no es igual a ninguno de los números anteriores.

Figura 5. Cuando la cajita de control envíe un 1 la puerta se abrirá y el mensaje "0.5 1000" llegara al objeto "print". En la imagen el control está cerrado y ha enviado un 0, por lo que la puerta esta cerrada y el mensaje no llega al objeto "print".

Revision #1
Created 17 February 2023 11:30:15 by Julia del Río
Updated 17 February 2023 11:30:15 by Julia del Río