Unidad 3

Datos: Variables

Elementos que se introducen en esta Unidad:
`int,float, boolean, true, false, = (asignación), width, height`

¿Qué son los datos? Los datos, con frecuencia son características físicas asociadas a un algo. Por ejemplo, cuando decimos que una persona llamada Juan Pérez, en su registro de conducir figura una M (por género masculino) sabemos que ese valor M esta asociado a la persona Juan Pérez.
En los sistemas informáticos, los datos son guardados como números y caracteres. Por ejemplo, los ordenadores están constantemente recibiendo datos del mouse y el teclado. Cuando se crea un programa, pueden guardarse datos de, por ejemplo, una forma, un color, o el cambio constante de la posición del mouse.

-Tipos de datos

Processing puede **almacenar y modificar** muchos tipos de datos, como números, letras, palabras, imágenes, colores, fuentes y valores booleanos (`true` y `false`). El hecho de guardar datos implica un mayor o menor uso de la memoria del ordenador donde estemos trabajando. No es lo mismo guardar la palabra “Andalucía” que guardar simplemente la “A”. Cada dato es representado como una serie de bits (0 y 1). Por ejemplo, 010000100 puede ser interpretado como una letra.
Como seres humanos, no hará falta aprender el lenguaje binario para programar, Processing se presta para que el trabajo nos sea mucho más sencillo. Sin embargo, 01000001 puede ser interpretado como al letra “A” o como el número 65. Por lo tanto, es importante definir que tipo de dato estamos trabajando.
El primer tipo de datos que enunciaremos, serán los datos numéricos. Existen dos clases de datos numéricos: enteros y decimales (flotantes). Cuando hablamos de enteros, nos referimos a números completos, como 5, -120, 8 y 985. Processing representa variables de tipo entero con `int`. En cambio, los número decimales, también llamados de *punto-flotante*, crean fracciones de los números enteros como 12.8, -120.75, 8.333 y 985.8676543. Processing representa los datos tipo decimales con `float`.
La variable más simple en Processing es de tipo `boolean`. Solo admite dos valores `true` o `false` */verdadero o falso/*. Suele utilizarse cuando es necesario que el programa tome una decisión ignorando el resto de las posibilidades. A continuación, una tabla con los tipos de variable y el tamaño que ocupa cada una:

Nombre	Tamaño	Rango de Valores
`boolean`	`1 bit`	`true / false`
`byte`	`8 bits`	`-128 a 127`
`char`	`16 bits`	`0 a 65535`
`int`	`32 bits`	`-2147483648 a 2147483647`
`float`	`32 bits`	`3.40282347E+38 a - 3.40282347E+38`
`color`	`32 bits`	`16777216 colores`

-Variables

Una variable es un **contenedor** para guardar datos. Las variables permiten que cada datos sea reutilizado **muchas veces** en un programa. Cada variable tiene dos partes un *nombre* y un *valor*. Si una variable almacena el valor 21 y es llamada *edad*, el nombre *edad* puede aparecer muchas veces en el programa. Cuando el programa se ejecute, la palabra *edad* cambiará por el valor de 21. En adición a este nombre y valor, hay que declarar que tipo de datos soporta esa variable.
Una variable debe ser, siempre, declarada antes de ser usada. Una declaración de variable consta del tipo de datos que aceptará esa variable, seguida de un nombre creado por nosotros. En el siguiente ejemplo se declaran varios tipos de variables y se les asigna un valor:

```
int x;           //Declaración de variable x del tipo entero
```

```
float y;          //Declaración de variable y del tipo flotante
boolean b;        //Declaración de variable b del tipo booleana

x = 50;           //Asignar el valor 50 a la variable x
y = 12.6;         //Asignar el valor 12.6 a la variable y
b = true;         //Asignar el valor true a la variable b
```

Hay una forma mas sintetizada de hacer lo mismo. Podemos, entonces, escribir lo mismo en una sola línea:

```
int x = 50;
float y = 12.6;
boolean b = true;
```

Más de una variable del mismo tipo pueden ser declaradas en la misma línea y luego asignarse un valor por separado a cada una:

```
float x, y, b;
x = -5.56;
y = 12.6;
b = 76.789;
```

Cuando una variable es declarada, es importante ver que clase de dato se le va a asignar para elegir correctamente el tipo de dato. Ni el nombre, ni el tipo de dato puede ser cambiado una vez declarado. Si es posible reasignar otro valor:

```
int x = 69;       //Declara variable x y le asigna el valor de 69
x = 70;           //Cambiar el valor de x por 70
int x = 71;       //ERROR – La variable x ya existe
```

El símbolo de igual (=) se utiliza para asignar valores, únicamente. Le asigna el valor que se encuentra en el lado derecho, a la variable del lado izquierdo. Por lo tanto, es importante que lo que se encuentre del lado izquierdo sea una variable:

```
5 = 25;     //ERROR – Lo que se encuentre del lado izquierdo debe ser una
            //variable
```

Hay que tener en cuenta el tipo de datos que estemos manejando. No es posible asignar un tipo de datos a una variable que solo acepte otra clase. Por ejemplo, no podemos asignar valores decimales a una variable tipo entero:

```
int x = 12.89;    //ERROR – La variable es tipo entero y se le está asignando un
                  //valor decimal

float f = 12.89;
int y = f;        //ERROR – La variable es tipo entero y se le está asignando un
                  //valor decimal
```

Las variables pueden tener nombres que describan su contenido. Eso simplifica mucho la tarea a la hora de programar. Además, esto podría ayudar a reducir la cantidad de comentarios. Aún así, queda en el programador elegir que clase de nombres utilizar. Por ejemplo, para variables de temperatura, se podrían utilizar los siguientes nombres:

```
t
temp
temperatura
```

```
tempCuarto
temperaturaCuarto
```

La primer letra tiene que ser un carácter en minúsculas, las siguientes pueden ser prácticamente cualquier cosa. Aún así, no se admiten acentos en ninguna parte del código.

-Variables de Programa

Processing, como lenguaje, ha construido variables muy útiles que ya vienen pre-programadas, a fin de facilitarle el trabajo al usuario. Se las denomina *variables del programa* o *variables del sistema*. El ancho y el alto de la ventana de representación están almacenadas como variables denominadas `width` */ancho/* y `height` */alto/*, respectivamente. Si el programa no incluye `size()`, por defecto `width` y `height` adquieren el valor de 100.

```
println(width +” , “+ height);      //Imprime “100, 100” en la consola

size(300, 200);
println(width +” , “+ height);      //Imprime “300, 200” en la consola
```

Frecuentemente son usadas cuando se requiere mantener la escritura del programa en diferentes escalas y que estas sean proporcionales.

```
size(100, 100);
ellipse(width * 0.5, height * 0.5, width * 0.66, height * 0.66);
line(width * 0.5, 0, width * 0.5, height);
```