Unidad 5

Control: Decisiones

Elementos que se introducen en esta Unidad:
`> (mayor a), < (menor a), >= (mayor o igual a), <= (menor o igual a), == (igual), != (distinto)`
`if, else, { } (corchetes), || (lógica O), && (lógica Y), ! (lógica NO)`

Cuando un programa corre, lo hace en el orden en que ubicamos la líneas. Primero la primer línea, luego la segunda, luego la tercera, y así. El programa finaliza cuando lee la última línea. Muchas veces, y para agregar interés, es necesario que esta lectura se quiebre. A ese orden suele llamárselo *flujo*. Existen, entonces, estructuras de control para rompe el flujo del programa.

-Expresiones Relacionales

¿Qué es la verdad? Esta gran cuestión filosófica es muy sencilla de responder en la programación. Sencillamente, si algo es `true` (verdadero) es una **verdad**, y si algo es `false` (falso) es todo lo **contrario**. Se trata tan solo de una noción lógica, no es necesario que se trate realmente de una "verdad". Sin embargo, si en la lógica del programa, eso es una verdad, entonces el programa devolverá un valor `true`, de lo contrario devolverá un valor `false`. Tan sencillo como un 1 y un 0.

Expresión	Evaluación
`3 < 5`	`true`
`3 > 5`	`false`
`5 < 3`	`false`
`5 > 3`	`true`

Cualquiera de estas expresiones puede leerse en español. ¿El número tres es menor al número cinco? Si la respuesta es "si" expresa el valor `true` (verdadero). Al compararse dos valores con una expresión relacional, solo pueden dar dos resultados posibles: `true` o `false`. A continuación, una tabla con los valores relacionales y su significado:

Operador	Significado
`>`	`Mayor que`
`<`	`Menor que`
`>=`	`Mayor o igual a`
`<=`	`Menor o igual a`
`==`	`Igual a`
`!=`	`Distinto de`

Las siguientes líneas de código muestran el resultado de comprar dos valores con una expresión relacional:

```
println(3 >  5);        //Imprime false
println(5 >  3);        //Imprime true
println(5 >  5);        //imprime false

println(3 <  5);        //Imprime true
println(5 <  3);        //Imprime false
println(5 <  5);        //imprime false

println(3 >=  5);       //Imprime false
println(5 >=  3);       //Imprime true
println(5 >=  5);       //imprime true

println(3 <=  5);       //Imprime true
println(5 <=  3);       //Imprime false
println(5 <=  5);       //imprime true
```

El operador de igualdad (==) determina si los dos valores que se evalúan son equivalentes. Devuelve `true` si la igualdad se cumple. En cambio, el operador de diferenciación (`!=`) evalúa si dos valores no son equivalentes. Devuelve `true` si la desigualdad se cumple.

```
println(3 ==  5);       //Imprime false
println(5 ==  3);       //Imprime false
println(5 ==  5);       //imprime true

println(3 !=  5);       //Imprime true
println(5 !=  3);       //Imprime true
println(5 !=  5);       //imprime false
```

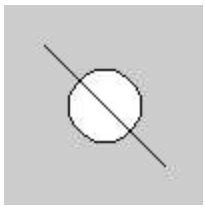
-Estructuras Condicionales

Las estructuras condicionales le permiten a un programa saber que línea de código ejecutar y cuales no. Las líneas de código solo serán "visibles" para el programa si se cumple una condición. Permiten al programa diferenciar acciones dependiendo el valor de variables. Por ejemplo, el programa dibuja una línea o una elipse dependiendo el valor de una variable. Las estructura `IF` es usada en Processing para tomar esas decisiones:

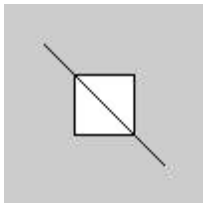
```
if(condición){
acciones;
}
```

La condición debe ser una expresión que se resuelve con `true` o `false`. Si la condición es `true`, el código que este desde la apertura del corchete (`{`) hasta el cierre del mismo (`}`), se ejecuta. En caso de ser `false`, el programa directamente no "lee" las acciones.

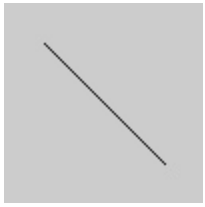
Los siguientes tres ejemplos pretenden mostrar el funcionamiento de la estructura `IF`. Se recurre al mismo código, solo que con diferentes valores la variable `x`:



```
//Las condiciones son "x > 100" y "x < 100"
//Como el valor de x es 150
//se ejecutará el primer bloque IF
//y se "eliminará" lo que ocurre en el segundo
int x = 150;
if (x > 100){                       //si es mayor que 100
      ellipse(50, 50, 36, 36);      //dibuja una elipse
}
if(x < 100){                  //si es menor que 100
      rect(35, 35, 30, 30);   //dibuja un rectángulo
}
line(20, 20, 80, 80);
```

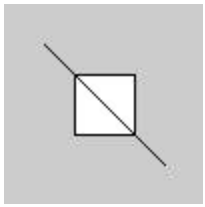


```
//Las condiciones son "x > 100" y "x < 100"
//Como el valor de x es 50
//se ejecutará el segundo bloque IF
//y se "eliminará" lo que ocurre en el primero
int x = 50;
if (x > 100){                       //si es mayor que 100
      ellipse(50, 50, 36, 36);      //dibuja una elipse
}
if(x < 100){                  //si es menor que 100
      rect(35, 35, 30, 30);   //dibuja un rectángulo
}
line(20, 20, 80, 80);
```

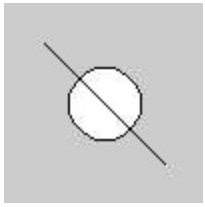


```
//Las condiciones son “x > 100” y “x < 100”
//Como el valor de x es 100
//no se ejecutará lo que ocurre en ninguna estructura IF
int x = 100;
if (x > 100){                        //si es mayor que 100
      ellipse(50, 50, 36, 36);       //dibuja una elipse
            }
if(x < 100){                  //si es menor que 100
      rect(35, 35, 30, 30);   //dibuja un rectángulo
}
line(20, 20, 80, 80);
```

En el caso específico de que la condición de como resultado `false`, y estemos deseando que aún así, ocurra algún evento, se utilizará, como agregado a la estructura `IF`, la estructura `ELSE`. La estructura `ELSE` extiende a la estructura `IF`, permitiendo agregar acciones cuando la condición devuelve un resultado `false`.



```
int x = 90;       //Como x vale 90, dibujará un rectángulo
if (x > 100){                        //si es mayor que 100
      ellipse(50, 50, 36, 36);       //dibuja una elipse
} else {                             //sino,
      rect(35, 35, 30, 30);          //dibuja un rectángulo
}
line(20, 20, 80, 80);   //siempre dibuja una línea
```

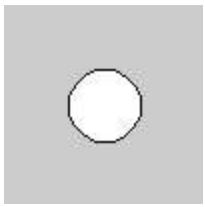


```
int x = 290;            //Como x vale 290, dibujará una elipse
if (x > 100){           //si es mayor que 100
      ellipse(50, 50, 36, 36);       //dibuja una elipse
} else {                             //sino,
      rect(35, 35, 30, 30);          //dibuja un rectángulo
}
line(20, 20, 80, 80);         //siempre dibuja una línea
```

Las condicionales pueden ser “encadenadas” una dentro de otra para tener completo control de las líneas de código. En el siguiente ejemplo se evalúa si el valor es mayor a 100, y luego de eso si es mayor a 300.


```
int x = 420;
if (x > 100){     //Condición para dibujar elipse o línea
      if (x < 300){           //Condición, lo determina
            ellipse(50, 50, 36, 36);
      } else {
            line(50, 0, 50, 100);
                  }
} else {
      rect(33, 33, 34, 34);
}
```

Podemos, también, ganar incluso un mayor control en las decisiones al combinar la estructura `IF` con la estructura `ELSE`, consiguiendo la estructura `ELSE IF`.



```
int x = 420;
if (x < 100){                 //Si es menor a 100
      rect(33, 33, 34, 34);   //dibuja un rectángulo
}else if (x > 300){                  //Sino, si es mayor a 300
      ellipse(50, 50, 36, 36);       //dibuja una ellipse
} else {                      //Sino,
      line(50, 0, 50, 100);   //dibuja una línea
}
```

-Operadores Lógicos

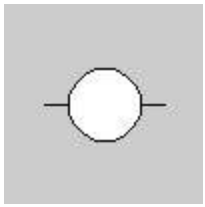
Los operadores lógicos se utilizan al combinar dos o mas expresiones relacionales y para invertir los valores lógicos. Los símbolos de los operadores lógicos corresponden a los conceptos de Y, O y NO.

Operador	Significado
&&	Y
||	O
!	NO

La siguiente tabla muestra todas las operaciones posibles y los resultados:

Expresión	Evaluación
`true && true`	`true`
`true && false`	`false`
`false && false`	`false`
`true || true`	`true`
`true || false`	`true`
`false || false`	`false`
`!true`	`false`
`!false`	`true`

El operador lógico O (`||`), hace que el valor sea un `true` solo si una parte de la expresión es `true` (verdadera).



```
int a = 10;
int b = 20;
if((a > 5 ) || (b < 30)){     //Si a es mayor a 5 o b es menos
                              //a 30 dibuja una
      line(20, 50, 80, 50);   //línea. Como ambas condiciones
                              //se cumples, se
}                             //dibuja la línea
if((a > 15) || (b < 30)){     //Si a es mayor a 15 o b es menor
                              //a 30, dibujar una
      ellipse(50, 50, 36, 36);//elipse. Solo una de las
                              //condiciones se cumplen, se
}                             //dibuja la elipse
```

Los procesos de lógica se **descomponen en pasos**. Los paréntesis son utilizados para delimitar las componentes y así **simplificar** el trabajo. En las siguientes líneas se muestra un breve paso a paso de como debe interpretarse:

```
Paso 1          (a > 5) || (b < 30)
Paso 2          (10 > 5) || (20 < 30)
Paso 3          true || true
Paso 4          true
```

El operador lógico Y (`&&`), hace que el valor sea `true` solo si ambas expresiones son `true`.


```
int a = 10;
int b = 20;
if((a > 5 ) && (b < 30)){     //Si a es mayor a 5 y b es menos
                              //a 30 dibujar una
      line(20, 50, 80, 50);   //línea. Como ambas condiciones
```

```
                                    //se cumples, se
                }                   //dibuja la línea
                if((a > 15) && (b < 30)){     //Si a es mayor a 15 y b es menor
                                    //a 30 dibujar una
                      ellipse(50, 50, 36, 36);//elipse. Solo una de las
                                    //condiciones se cumplen,
                }                   //NO se dibuja la elipse
```

El operador lógico NO (`!`) es una marca. Simplemente **invierte** el valor lógico. Por ejemplo, si se trata de un valor `true`, al escribir `!true`, estaríamos convirtiendo su valor en `false`. Solo es posible aplicarlo a una variable del tipo `boolean`.

```
boolean b = true;
println(b);             //Imprime true
b = !b;
println(b);             //Imprime false
println(!b);            //Imprime true
println(5 > 3);         //Imprime true
println(!(5 > 3));      //Imprime false
int x = 20;
println(!x);            //ERROR – Solo puede aplicarse a variables boolean
```

Apéndice de Estructura IF, ELSE y ELSE-IF