Condicionales y operadores

Operadores de comparación

Swift también tiene operadores de `comparación`, que funcionan muy parecido a como los conocemos.

Declaremos dos variables para demostrarlo:

```
let firstNumber = 6
let secondNumber = 4
```

Para `comparar la igualdad` de dos variables, tenemos el operador `==` **,** que verifica si dos variables son iguales y `!=` que verifica que dos variables son distintas.

Por ejemplo:

```
firstNumber == secondNumber //Es falso, son distintos
firstNumber != secondNumber //Es verdad, son distintos
```

Luego, tenemos operadores para `comparar si un valor el mayor, menor o igual a otro.`

Igual que en matemáticas, utilizamos `<` para indicar menor, `>` para indicar mayor y los agregamos un `=` para indicar el valor puede ser igual.

Por ejemplo:

```
firstNumber > secondNumber // Es mayor
secondNumber < firstNumber // Es menor
firstNumber >= secondNumber // Es mayor o igual
```

Condicionales

Ahora que ya vimos los operadores, podemos introducir el concepto de las `condiciones if`.

Esto funciona así: Escribimos una condición y si (if) esta es verdad, se corre cierta parte del código y si es falsa, ejecutamos algo distinto.

Por ejemplo:

```
edadUno = 19

if edadUno > 18 {
  print("Es mayor de edad")
}
```

Estructura:

Primero, escribimos la palabra `if`, luego declaramos la condición que se debe cumplir y finalmente, entonces llaves escribimos el código que se ejecuta si esta es verdadera.

```
if condicion {
  //codigo que se ejecuta si es verdadera la condicion
}
```

Pero supongamos que queremos que se ejecute una porción de código si la condición es falsa. Para esos casos, agregamos la cláusula `else`.

```
edadUno = 19

if edadUno > 18 {
  print("Es mayor de edad")
} else {
  print("No es mayor de edad")
}
```

Al agregar la `clausula else`, estamos aclarando que si no se cumple la condición, entonces se va a ejecutar otra porción de código.

Finalmente, si queremos declarar más de dos condiciones, podemos utilizar la cláusula `else if`.

La estructura es la siguiente:

```
if edadUno < 18 {
    print("Es menor de edad")
} else if edadUno == 18 {
    print("Tiene 18 años")
} else {
    print("Es mayor de edad")
}
```

¿Por qué `else if` ?

Supongamos que tenemos una situación en la que tenemos tres o más caminos diferentes. Bueno, en ese caso podemos hacer 3 if diferentes, indicando las tres condiciones, así:

```
let edadUno = 18

if edadUno < 18 {
    print("Es menor de edad")
}

if edadUno > 18 {
    print("Es mayor de edad")
}

if edadUno == 18 {
    print("Tiene 18")
}
```

Y aunque funciona perfectamente, pero aca Swift tiene que verificar el valor de edadUno tres veces distintas. Y aunque con un Int es rápido, hace la diferencia cuando nuestra data es mas compleja!

Busquemos otra solución. Otra cosa que podemos hacer es “a`nidar condicionales`”. Esto se hace de la siguiente manera:

```
if edadUno > 18 {
  print("Es mayor de edad")
} else {
  if edadUno == 18{
      print("Tiene 18")
  } else{
      print("Es mayor de edad")
  }
}
```

Y si bien esta solución es completamente factible, también es complicada de leer y puede hacer que nuestro código se vea enroscado y desprolijo.

> 💡 Es por eso que si tenemos mas de dos condiciones, else if nos permite combinar el else con el if de manera mas prolija y podemos tener un flujo mucho mas sencillo de leer!