Condicionales en Kotlin

Kotlin admite las condiciones lógicas habituales de las matemáticas:

```
Menos que:        a < b
Menor o igual a:  a <= b
Mayor que:        a > b
Mayor o igual a:  a >= b
Igual a           a == b
No es igual a:    a != b
```

Puede utilizar estas condiciones para realizar diferentes acciones para diferentes decisiones.

Kotlin tiene los siguientes condicionales:

- **if** para especificar un bloque de código que se ejecutará, si una condición especificada es verdadera
- **else** para especificar un bloque de código que se ejecutará, si la misma condición es falsa
- **else if** para especificar una nueva condición para probar, si la primera condición es falsa
- **when** para especificar muchos bloques alternativos de código que se ejecutarán.

Kotlin si

Use if para especificar un bloque de código que se ejecutará si una condición es true.

Sintaxis

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

Tenga en cuenta que if está en letras minúsculas. Las letras mayúsculas (If o IF) generarán un error.

En el siguiente ejemplo, probamos dos valores para averiguar si 20 es mayor que 18. Si la condición es true, imprima algo de texto:

Ejemplo

```
if (20 > 18) {
  println("20 is greater than 18")
}
```

También podemos probar variables:

Ejemplo

```
val x = 20
val y = 18
if (x > y) {
  println("x is greater than y")
}
```

Condicionales en Kotlin

Kotlin else

Use else para especificar un bloque de código que se ejecutará si la condición es false.

Sintaxis

```
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

Ejemplo

```
val time = 20
if (time < 18) {
  println("Good day.")
} else {
  println("Good evening.")
}
// Outputs "Good evening."
```

Kotlin else if

Use else if para especificar una nueva condición si la primera condición es false.

Sintaxis

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and
condition2 is true
} else {
  // block of code to be executed if the condition1 is false and
condition2 is false
}
```

Ejemplo

```
val time = 22
if (time < 10) {
  println("Good morning.")
} else if (time < 20) {
  println("Good day.")
} else {
  println("Good evening.")
}
// Outputs "Good evening."
```

Condicionales en Kotlin

Kotlin if .. else

En Kotlin, también puedes usar if..else como expresiones (asigna un valor a una variable y devuélvelo):

Ejemplo
```
val time = 20
val greeting = if (time < 18) {
  "Good day."
} else {
  "Good evening."
}
println(greeting)
```

Cuando se utiliza if como expresión, también se debe incluir else (obligatorio).

Nota: Puede omitir las llaves {} cuando if solo tiene una declaración:

Ejemplo
```
fun main() {
  val time = 20
  val greeting = if (time < 18) "Good day." else "Good evening."
  println(greeting)
}
```

Kotlin when

En lugar de escribir muchas expresiones if..else, puede utilizar la expresión when, que es mucho más fácil de leer.

Se utiliza para seleccionar uno de los muchos bloques de código que se ejecutarán:

Ejemplo
Utilice el número del día de la semana para calcular el nombre del día de la semana:

```
val day = 4

val result = when (day) {
  1 -> "Monday"
  2 -> "Tuesday"
  3 -> "Wednesday"
  4 -> "Thursday"
  5 -> "Friday"
  6 -> "Saturday"
  7 -> "Sunday"
  else -> "Invalid day."
}

println(result)
```

```
// Outputs "Thursday" (day 4)
```

Así es como funciona:

- La variable when (día) se evalúa una vez
- El valor de la variable día se compara con los valores de cada rama
- Cada rama comienza con un valor, seguido de una flecha (->) y un resultado.
- Si hay una coincidencia, se ejecuta el bloque de código asociado.
- else se utiliza para especificar algún código para ejecutar si no hay coincidencia
- En el ejemplo anterior, el valor de **day** es **4**, que significa que se imprimirá "jueves"

Condicionales en Kotlin

Ejercicios

E701: Ingresar el sueldo de una persona, si supera los 3000 pesos mostrar un mensaje en pantalla indicando que debe abonar impuestos.

```
fun main() {
    print("Ingrese el sueldo del empleado:")
    val sueldo = readln().toDouble()
    if (sueldo > 3000) {
        println("Debe pagar impuestos")
    }
}
```

E702: Se ingresan por teclado 2 valores enteros. Si el primero es menor al segundo calcular la suma y la resta, luego mostrarlos, sino calcular el producto y la división.

```
fun main() {
    print("Ingrese el primer valor:")
    val valor1 = readln().toInt()
    print("Ingrese el segundo valor:")
    val valor2 = readln().toInt()
    if (valor1 < valor2) {
        val suma = valor1 + valor2
        val resta = valor1 - valor2
        println("La suma de los dos valores es: $suma")
        println("La resta de los dos valores es: $resta")
    } else {
        val producto = valor1 * valor2
        val division = valor1 / valor2
        println("El producto de los dos valores es: $producto")
        println("La división de los dos valores es: $division")
    }
}
```

E703: Se ingresan tres notas de un alumno, si el promedio es mayor o igual a siete mostrar un mensaje "Promocionado".

E704: Se ingresa por teclado un número entero comprendido entre 1 y 99, mostrar un mensaje indicando si el número tiene uno o dos dígitos.

E705: Diseñar un programa que solicite al usuario un numero e indique si es par o impar.

E706: Implementar un programa que pida por teclado un numero decimal e indique si es un numero casi-cero, que son aquellos, positivos o negativos, que se acercan a 0 por menos de 1 unidad, aunque curiosamente el 0 no se considera un numero casi-cero.
Ejemplos de números casi-cero son el 0,3, el –0,99 o el 0,123; algunos números que no se consideran casi-ceros son: el 12,3, el 0 o el –1.

E707: Pedir tres números y mostrarlos ordenados de mayor a menor.

Condicionales en Kotlin

E708: Escribir una aplicación que indique cuantas cifras tiene un numero entero introducido por teclado, que estará comprendido entre 0 y 99999.

E709: Pedir una nota entera de 0 a 10 y mostrarla de la siguiente forma: insuficiente (de 0 a 4), suficiente (5), bien (6), notable (7 y 8) y sobresaliente (9 y 10).

E710: Pedir el dia, mes y año de una fecha e indicar si la fecha es correcta. Hay que tener en cuenta que existen meses con 28, 30 y 31 días (no se considerara los años bisiestos).

E711: Escribir un programa que pida una hora de la siguiente forma: hora, minutos y segundos. El programa debe mostrar quo hora sera un segundo mas tarde. Por ejemplo:

hora actual [10:41:59] hora actual + 1 segundo: [10:42:0]

E712: Crear una aplicación que solicite al usuario una fecha (dia, mes y año) y muestre la fecha correspondiente al día siguiente.