Programación en Swift: Funciones

✅ ***Diploma:*** https://platzi.com/p/rubseg/curso/1790-swift-5-func/diploma/detalle/

Funciones

Las funciones son fragmentos de código autónomos que realizan una tarea específica. Usted le da a una función un nombre que identifica lo que hace, y este nombre se usa para "llamar" a la función para que realice su tarea cuando sea necesario.

La sintaxis de funciones unificadas de Swift es lo suficientemente flexible como para expresar cualquier cosa, desde una función simple estilo C sin nombres de parámetros hasta un método complejo estilo Objective-C con nombres y etiquetas de argumentos para cada parámetro.

Los parámetros pueden proporcionar valores predeterminados para simplificar las llamadas a funciones y pueden pasarse como parámetros de entrada y salida, que modifican una variable pasada una vez que la función ha completado su ejecución.

Cada función en Swift tiene un tipo, que consta de los tipos de parámetros de la función y el tipo de retorno.

Las funciones también se pueden escribir dentro de otras funciones para encapsular funcionalidades útiles dentro del alcance de una función anidada.

Una función simple:

```
func greeting(person: String) -> String {
    return "Greeting, \(person)"
}

print(greeting(person: "Rubén"))
```

Con parámetros de entrada:

```
func greeting(person: String, isMale: Bool) -> String {
    var greet = ""
    if isMale {
        greet = "Hola caballero \(person)"
    } else {
        greet = "Hola señorita \(person)"
    }
```

```
    return greet
}

print(greeting(person: "Rubén", isMale: true))
```

Con parámetros de salida:

```
func greet2(name: String) {
    print("Hola \(name)")
}

func greet3(name: String) -> String{
    return "Hola \(name)"
}

func printAndCount(cad: String) -> Int {
    print(cad)
    return cad.count
}
printAndCount(cad: "Toyota C-HR")

func minMax(array:[Int]) -> (min: Int, max: Int) {

    var currentMin = array[0]
    var currentMax = array[0]

    for value in array[1..<array.count] {
        if value < currentMin {
            currentMin = value
        } else if value > currentMax {
            currentMax = value
        }
    }

    return (currentMin, currentMax)
}

let bounds = minMax(array: [6, 4, 3, 1, 6, 5, 4, 7, 9])
print("El valor minimo es \(bounds.min) y el valor maximo es
\(bounds.max)")
```

Etiquetas de params

Podemos poner etiqueta antes del nombre. Es buena práctica cuando hay 2 o más parámetros.

```
func someFunction(f1 firstParamName: Int, f2
```

```
secondParamName: Int) {
   print(firstParamName + secondParamName)
}
someFunction(f1: 5, f2: 12)

func newGreeting(_ person: String, from homeTown: String) ->
String {
    return "Hola \(person) un placer que nos visites desde \
(homeTown)"
}
newGreeting("Rubén", from: "Jaén")
```

Valores por defecto

```
func someFunction(f1 firstParamName: Int = 18, f2
secondParamName: Int = 7) {
   print(firstParamName + secondParamName)
}
someFunction()
```

Valores variádicos

Podría haber 1, 2, o mas parámetros.

```
func mean(_ numbers: Double...) -> Double {
    var total: Double = 0
    for number in numbers {
        total += number
    }
    return total / Double(numbers.count)
}

mean(1,8,5,6,3,10)
mean(5,7,2)
mean(9,6,5.5,8.7,9.5,1)
```

Parámetros tipo inout

Son parámetros que sirven cuando queremos que una función modifique el valor de un parámetro y desea que esos cambios persistan después de que finalice la llamada a la función.

```
var x = 5
func addOne(number: inout Int) {
    number += 1
    print("El número vale \(number)")
```

```
}
addOne(number: &x)
```

Function types

Son las funciones como tipo de dato.

Funciones anidadas

Se pueden definir funciones dentro del cuerpo de otras funciones, conocidas como funciones anidadas.

```
func resultadoNotaFina(_ n: Double) -> String {
    return "Tu nota final es \( mean(n) )"
}
resultadoNotaFina(9)
```

Closures

Es un tipo especial de función, similares a las Lambdas.

Pueden capturar y almacenar referencias a cualquier constante y variable del contexto en el que están definidas. Esto se conoce como closing sobre esas constantes y variables.

Swift maneja toda la gestión de la memoria de la captura por usted.

... ‼️ ✏️

https://www.youtube.com/watch?v=j3E9DXine1U&ab_channel=SwiftBeta

Enumeradores

Los enumerados son un tipo especial de datos, como una “clase vaga”.

Tiene un valor posible de entre un conjunto disponible.

Por ejemplo:

```
enum CompassPoint {
    case north
    case sout
    case east
    case west
}

var directionToGo = CompassPoint.east
```

```
directionToGo = .north
directionToGo = .west

switch directionToGo {
    case .north:
        print("Dirigete al norte")
    case .sout:
        print("Dirigete al sur")
    case .east:
        print("Dirigete al este")
    case .west:
        print("Dirigete al oeste")
}

enum Planet {
    case mercury, venus, earth, mars, jupiter, saturn,
uranus, neptune
}

print(Planet.jupiter)
```

También podemos indicar diferentes tipo dentro del enumerado.

```
enum Barcode {
    case upc(Int, Int, Int, Int)
    case qrCode(String)
}

var productBarcode = Barcode.upc(2, 43, 9, 102)
productBarcode = .qrCode("cmikdslle")
```