
Universidad
Tecmilenio®

Universidad Tecmilenio

Introducción al Desarrollo de Aplicaciones Móviles

Modelo de negocio de aplicación móvil

Introducción

Los operadores son los símbolos que se utilizan para indicar a la computadora que realice operaciones aritmèticas, cambie o combine valores. En este tema conocerás algunos de los operadores que se utilizan en Swift.

También conocerás la forma de crear código para tomar decisiones dentro de un programa, es decir, valorar una condición, y de acuerdo a su valor, elegir uno u otro camino en el flujo del programa.

Explicación

Universidad Tecmilenio

Operadores

Asignar un valor

Usa el operador = para asignar un valor

```
var personaFavorita = "Luke"
```

usa el operador = para modificar o reasignar un valor

```
var tamanioPie = 28
tamanioPie = 29
```

Aritmética básica

Puedes usar los operadores +, -, *, / para realizar operaciones básicas

```
var marcadorOponente= 3*8
var miMarcador =  100/4
```

También puedes usar el valor de otras variables

```
var marcadorTotal = macadorOponente + miMarcador
```

O puedes usar la variable actual que estás utilizando

```
miMarcador = miMarcador + 2
```

Asignar un valor

Usa el operador = para asignar un valor

```
var personaFavorita = "Luke"
```

usa el operador = para modificar o reasignar un valor

```
var tamanioPie = 28
tamanioPie = 29
```

Explicación

Orden de asignación

1. ( )
2. * /
3. + -

```
var x = 2
var y = 3
var z = 5

print(x + y * z)
print((x + y) * z)
```

Asignación compuesta

```
var miMarcador = 10
miMarcador =  miMarcador + 3
miMarcador += 3
miMarcador -= 5
miMarcador *= 2
miMarcador /= 2
```

Conversión de tipo de numérico

```
let x = 3
let y = 0.1415927
let pi = x + y
```

Aritmética básica

```
let x = 51
let y = 4
let z = x / y
print(z)
```

Explicación

Flujo condicional

Operadores lógicos

Operador	Descripción
==	Dos elementos deben ser iguales.
!=	Los valores no deben ser iguales entre sí.
>	El valor de la izquierda debe ser mayor que el valor de la derecha.
>=	El valor de la izquierda debe ser mayor o igual que el valor de la derecha.
<	El valor de la izquierda debe ser menor que el valor de la derecha.
<=	El valor de la izquierda debe ser menor o igual que el valor de la derecha.
&&	And (Y): las instrucciones condicionales a la izquierda y a la derecha deben ser verdaderas (true).
||	OR (O): la instrucción condicional a la izquierda o la instrucción condicional a la derecha debe ser verdadera (true).
!	Devuelve el opuesto de la instrucción condicional que sigue inmediatamente al operador.

Explicación

Universidad Tecmilenio

Condicionales

Instrucciones "if"

```
if condición {
    Código
}
let  temperatura = 100
if temperatura >= 100 {
      print("El agua está hirviendo")
}
```

Swith

```
switch valor {
case n:
   Código
case n:
   Código
case n:
   Código
default:
   Código
}
```

Instrucciones "if-else"

```
if condición {
    código
} else  {
    Código
}

let  temperatura = 100
if temperatura >= 100 {
      print("El agua está hirviendo")
} else  {
      print("El agua no está hirviendo")
}
```

Swith

```
let numeroLlantas = 2
switch numeroLlantas {
case 0:
   print("¿Falta Algo?")
case 1:
    print("Monociclo")
case 2:
    print("Bicicleta")
case 3:
    print("Triciclo")
default:
     print("Automovil")
}
```

Explicación

Universidad Tecmilenio

Valores Booleanos

```
let  numero = 1000
let  esNumeroChico = numero< 10

let  limiteVelocidad= 65
let  velocidadActual = 1000
let  esVeloz = velocidadActual > limiteVelocidad
```

NOT

```
var  estaNevando = false
if  !estaNevando {
     print("No está nevando")
}
```

AND

```
let  temperatura = 25
if  temperatura >= 20 && temperatura <= 30 {
     print("la temperatura esta adecuada")
} else if  temperatura < 20 {
     print("Hace demasiado frio")
} else {
     print("Hace demasiado calor")
}
```

OR

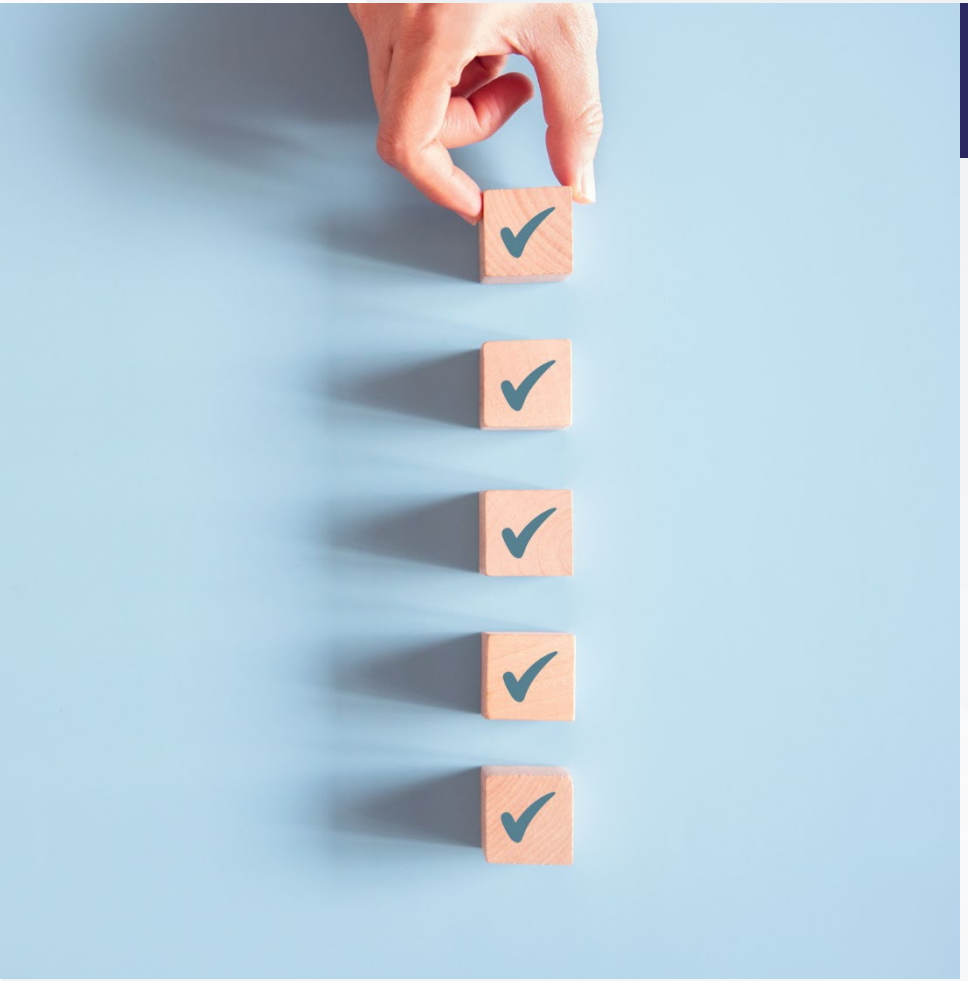
```
var  estaConectado = false
var  tieneBateria = true

if  estaConectado ||  tieneBateria {
     print("Puedes usar tu laptop")
} else {
     print("Ups!")
}
```

Actividad

- Reescribe el siguiente código usando una instrucción switch:

```
let  temperatura = 25
if  temperatura >= 20 && temperatura <= 30 {
    print("la temperatura esta adecuada")
} else if  temperatura < 20 {
    print("Hace demasiado frio")
} else {
    print("Hace demasiado calor")
}
```

Cierre



En este tema conociste la forma de incluir operadores para asignar valores a una variable o para ejecutar operaciones matemáticas. También revisaste la forma de evaluar condiciones sencillas y complejas para definir el flujo de un programa de acuerdo a las necesidades del usuario.

Universidad Tecmilenio

Introducción al Desarrollo de Aplicaciones Móviles

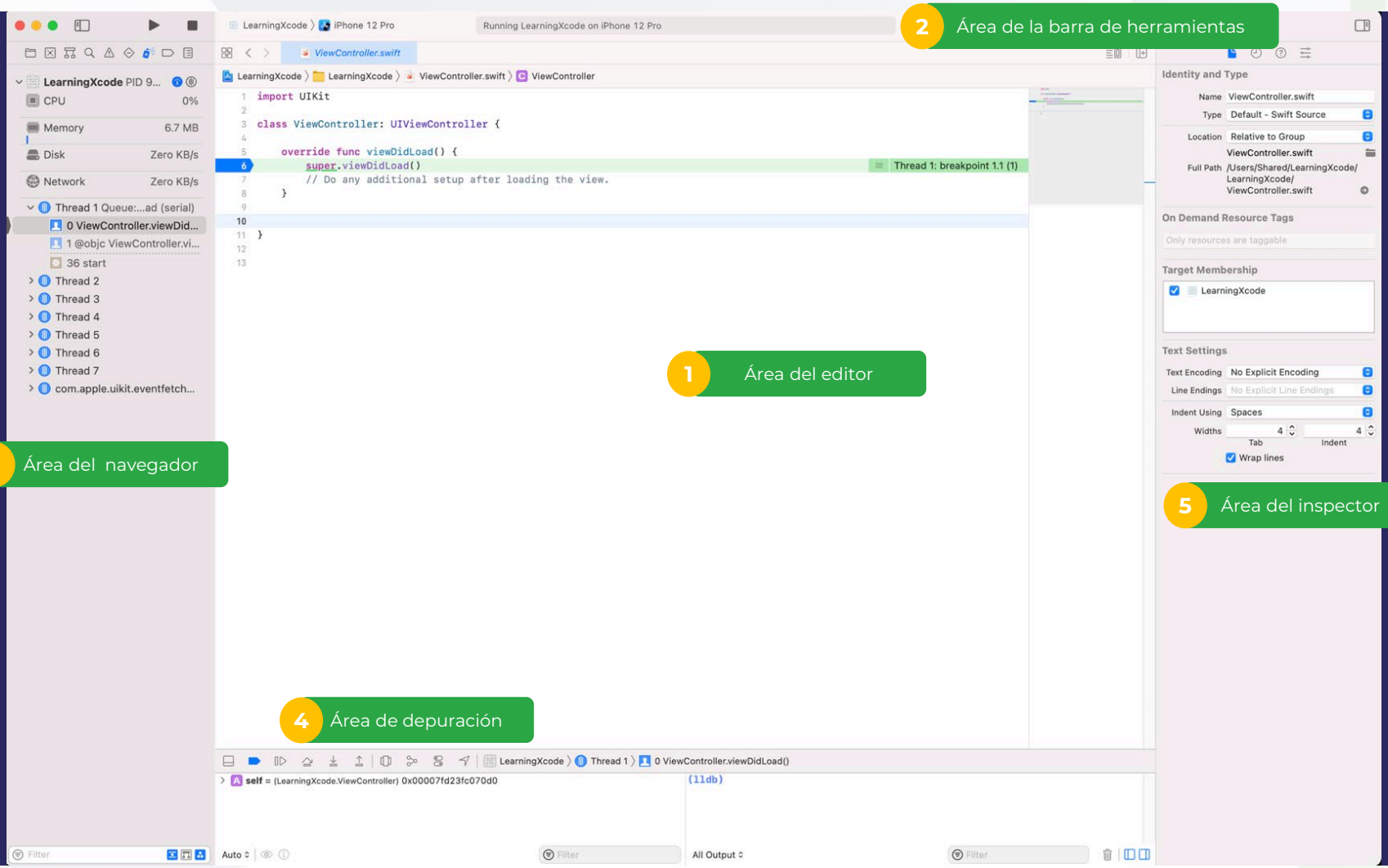
Xcode

Introducción

Xcode es el ambiente que utilizan los desarrolladores de aplicaciones iOS para escribir, depurar, compilar código, crear documentación y enviar aplicaciones a la AppStore.

En este tema aprenderás a crear aplicaciones básicas para que te familiarices con la interfaz de Xcode y sus principales características.

Explicación



Explicación

Funciones rápidas

- **Comando + B**: compilar el proyecto.
- **Comando + R**: compilar y ejecutar el proyecto.
- **Comando-.**: dejar de compilar o ejecutar.
- **Comando + /**: activar/desactivar los comentarios en las filas de código seleccionadas.
- **Comando + [:** desplazar el código seleccionado hacia la izquierda.
- **Comando +]**: desplazar el código seleccionado hacia la derecha.
- **Control + I**: volver a poner sangría al código seleccionado.
- **Comando + 0**: mostrar y ocultar el navegador.
- **Opción + Comando + 0**: mostrar y ocultar el inspector.
- **Comando + ↑:** ir a la parte de arriba de un archivo.
- **Comando + ←:** ir al archivo anterior.

Explicación

Para desarrollar en Xcode necesitas aprender a hacer lo siguiente:

Compilar y ejecutar una aplicación

Debes utilizar el Menu Scheme para elegir el dispositivo en el simulador.

También puedes utilizar tu dispositivo físico, solo debes informar a Xcode y puedes utilizar el cable o hacerlo de manera inalámbrica .

Depurar una aplicación

La depuración es el proceso de identificar y eliminar problemas que pueden surgir en tu app. Xcode proporciona herramientas para ayudarte en este proceso.

En Xcode encontrarás tres tipos de problemas: advertencias, errores del compilador y bugs.

Actividad

- Explora las propiedades de Xcode, modifica las preferencias, practica las funciones rápidas del teclado y familiarízate con el entorno de trabajo con el que vas a trabajar en las próximas semanas; haz pruebas y documéntate sobre funcionamiento sobre IDE.

Cierre

En este tema conociste el ambiente donde se desarrollan las aplicaciones que funcionan en el sistema operativo iOS. También aprendiste a utilizar el simulador para realizar pruebas de las aplicaciones que vas desarrollando y a correr aplicaciones en un dispositivo físico.

Universidad Tecmilenio

Introducción al Desarrollo de Aplicaciones Móviles

Interfaces y cadenas de caracteres

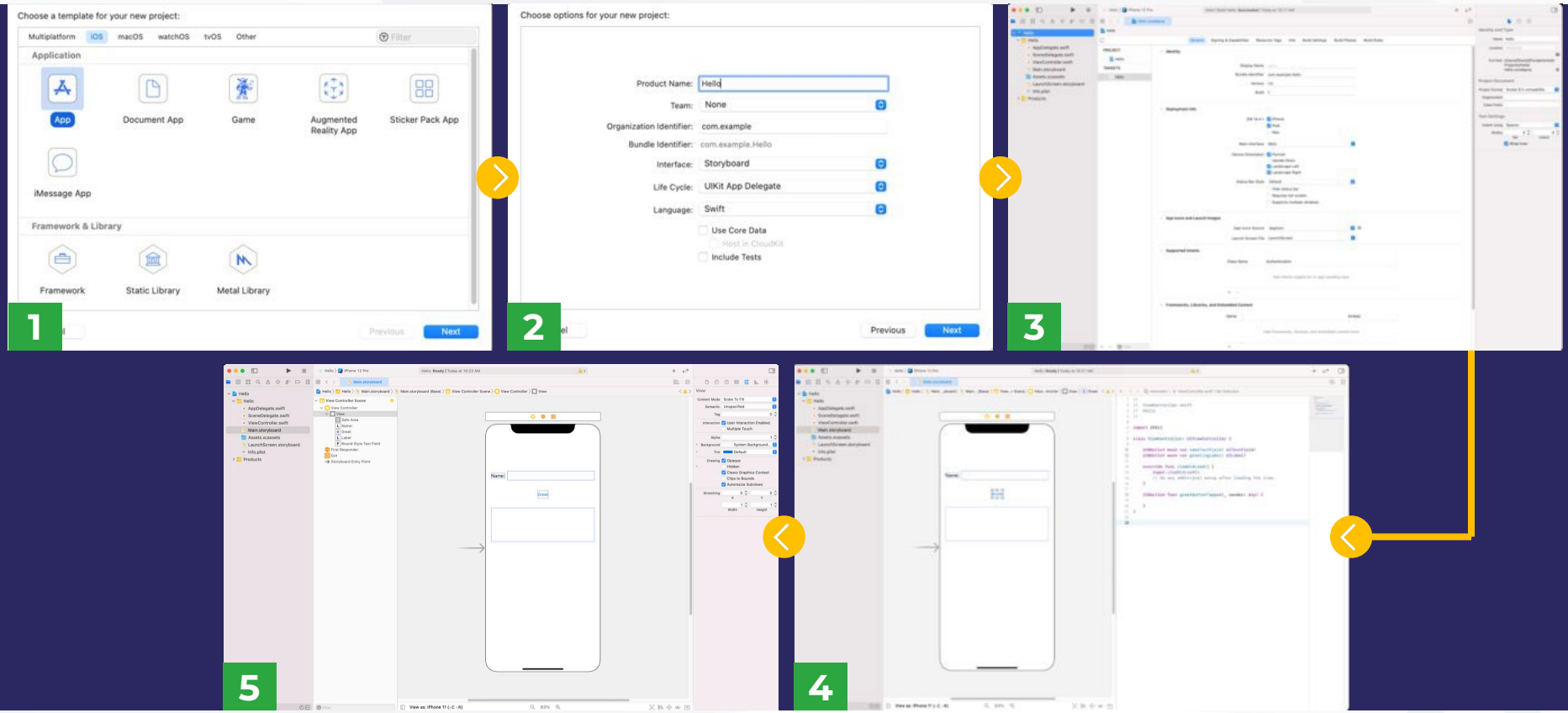
Introducción

La mejor manera de aprender los conceptos básicos de Interface Builder es sumergirse en Xcode y explorar algunas de sus características. Por ejemplo:

- Storyboards
- Interface Builder

Explicación

Proceso para crear tu primera aplicación



Explicación

Conceptos de cadenas

Cadenas

```
let saludo = "Hola"
var otroSaludo = "Saludos"

let chiste = """
P: ¿Por qué cruzó el pollo el camino?
R: Para ir al otro lado
"""
print(chiste)
```

Cadenas vacías

```
var miCadena = ""

if miCadena.isEmpty = {
   print("La cadena esta vacia.")
}
```

Concatenación

```
let string1 = "¡Hola"
let string1 = ", mundo!"
var miCadena: string1 +
string2  // "¡Hola, mundo!"

miCadena += " ¡Hola!"  // "¡Hola,
mundo! ¡Hola!"
```

Escape

```
let saludo = "El tradicional en programación imprimir \"¡Hola, mundo! \""
```

Escape	Descripción
\”	Comilla doble
****	Barra invertida
\t	Tabulador
\r	Retorno de carro (retorno al comienzo de la línea siguiente)

Caracteres

```
let a= "a" // 'a' es una cadena
let b: Character = "b" // 'b' es un
carácter
```

Explicación

Universidad Tecmilenio

Conceptos de cadenas

Interpolación

```
let nombre = "Juan"
let edad = 30
print("\(nombre) tiene \(edad) años")
```

```
Rick tiene 30 años
```

Interpolación Expresiones

```
let a = 4
let b = 5
print("Si a es \(a) y b es \(b), entonces a + b es igual a \(a + b)")
```

```
Si a es 4 y b es 5, entonces a + b es igual a 9.
```

Igualdad y comparación de cadenas

```
let mes = "Enero"
let otroMes = "Enero"
let mesMinuscula = "enero"

if mes == otroMes{
    print("Son iguales.")
}

if mes != mesMinuscula{
    print("No son iguales.")
}
```

```
Son iguales.
No son iguales.
```

Igualdad y comparación de cadenas Ignorar las mayúsculas

```
let nombre = "Juan Hernandez"
if mes.lowercased() == "juAn HErnanDez".lowercased(){
    print("Los dos nombres son iguales.")
}
```

```
Los dos nombres son iguales.
```

Actividad

Lee la sección Cadenas en el siguiente link para explorar conceptos avanzados de cadenas: https://docs.swift.org/swift-book/LanguageGuide/StringsAndCharacters.html

- Realiza un resumen con lo que consideres mas relevante.

Cierre

En este tema conociste el ambiente donde se diseña la interfaz gráfica y la forma en que se conectan cada uno de los elementos al código de programación. Es tiempo de que te pongas a practicar, replicando las pantallas de las aplicaciones que conoces o generando nuevas ideas.

También aprendiste a trabajar con las cadenas de caracteres de Swift, pudiste definir, manipular y realizar comparaciones con este tipo de datos para posteriormente desplegarlos.

Universidad Tecmilenio

Introducción al Desarrollo de Aplicaciones Móviles

Funciones y estructuras en Swift

Explicación

Universidad Tecmilenio

Funciones

```
miZapato ()

hacerDesayuno(food: "huevos revueltos", drink: "jugo de naranja")
```

Definir una función

```
func nombrefuncion(parámetros) -> TipoRetorno {
        //Cuerpo de la función
}

func mostrarPi()  {
  print("3.1415926535")
}

displayPi()
```

3.1415926535

Definir una función

```
func triple(value: Int) {
    let result = value * 3
    print("Si multiplicas \(value) por 3, obtendrás \(result).")
}

triple(value: 10)
```

Si multiplicas 10 por 3, obtendrás 30.

Multiplicación de parámetros

```
func multiplicacion(primerNumero: Int, segundoNumero: Int) {
    let result = primerNumero * segundoNumero
    print("El resultado es \(result).")
}

multiplicacion(primerNumero: 10, segundoNumero: 5)
```

El resultado es 50.

Explicación

Universidad Tecmilenio

Funciones

Valores de devolución

```
func multiplicacion(primerNumero: Int, segundoNumero: Int) -> Int {
    let result = primerNumero * segundoNumero
    return result
}
```

Valores de devolución

```
func multiplicacion(primerNumero: Int, segundoNumero: Int) -> Int {
    result  primerNumero * segundoNumero
}

let miResultado = multiplicacion(primerNumero: 10, segundoNumero: 5)
print("10 * 5 es \(miResultado)")

print("10 * 5 es \(multiplicacion(primerNumero: 10, segundoNumero: 5))")

func multiplicacion(primerNumero: Int, segundoNumero: Int) -> Int {
    primerNumero * segundoNumero
}
```

Etiquetas de argumento

```
func diHola(nombre: String){
    print("¡Hola, \(nombre)!")
}

diHola(nombre; "Amy")
```

Explicación

Universidad Tecmilenio

Estructuras

Usar mayúsculas para los nombres de los tipos
Usar minúsculas para los nombres de las propiedades

```
struct Persona {
   var nombre: String
}
```

Inicializadores

```
let cadena = String.init() //""
let entero = Int.init() //0
let cadena = Bool.init() //false

let cadena = String() //""
let entero = Int() //0
let cadena = Bool() //false
```

Acceso a los valores de una propiedad

```
struct Persona {
   var nombre: String
}

let persona = Persona(nombre: "Yazmin")
print(persona.nombre)
```

Yazmin

Explicación

Estructuras

Usar Mayúsculas para los nombres de los tipos
Usar minúsculas para los nombres de las propiedades

Agregar funcionalidad

```
struct Persona {
   var nombre: String

   func diHola() {
      print("¡Hola! ¡Mi nombre es \(nombre)")
    }
}

let persona = Persona(nombre: "Yazmin")
persona.diHola()
```

¡Hola! ¡Mi nombre es Yazmin!

Instancias

```
struct Camisa {
   var talla: String
   var color: String
}

let miCamisa = Camisa(talla: "XL", color: "azul")

let tuCamisa = Camisa(talla: "M", color: "rojo")
```

Actividad

- Elige una app, la que más te guste, y escribe algunas de las estructuras que podrías haber utilizado en el desarrollo de esta aplicación.

Cierre

En este tema aprendiste a crear tus propias funciones para generar códigos más fáciles de leer y de mantener.

Las funciones se pueden definir de una manera simple con diferentes argumentos y tipos de retorno.

Además, creaste tus propias estructuras de datos, haciendo más fácil el trabajo con datos complejos y secuencias lógicas.