Universidad Simón Bolivar
Lenguajes de Programacion
CI-3641

Jesus Gutierrez
20-10332
Prof. Ricardo Monascal
Sep - Dic 2024

Swift - Tarea 3

Swift es un lenguaje de programación desarrollado por Apple, diseñado para ser seguro, rápido y expresivo. Es utilizado principalmente para el desarrollo de aplicaciones para iOS, macOS, watchOS y tvOS. Swift combina características de programación moderna como la gestión automática de memoria y la seguridad de tipos, con un rendimiento cercano al de C. Además, ofrece una sintaxis concisa y clara, lo que facilita la lectura y el mantenimiento del código.

1. Diga qué tipos de datos posee y qué mecanismos ofrece para la creación de nuevos tipos (incluyendo tipos polimórficos de en caso de haberlos).

   Los tipos de dato que posee Swift son:

   - **Tipos básicos**: `Int, Float, Double, Bool, String, Character, UInt`
   - **Colecciones**: `Array, Dictionary, Set`
   - **Enumeraciones**: `enum`
   - **Estructuras**: `struct`
   - **Clases**: `class`
   - **Tuplas**: `(Int, String)`
   - **Opcionales**: `Int?, String?`

   Para la creación de nuevos tipos, Swift ofrece los siguientes mecanismos:

   - **Estructuras(`struct`)**: Permiten definir nuevos tipos de datos que pueden tener propiedades y métodos.
   - **Clases(class)**: Similares a las estructuras pero con capacidades adicionales como herencia y referencia.
   - **Enumeraciones(`enum`)**: Permiten definir un tipo con un conjunto limitado de valores relacionados.
   - **Protocolos(`protocol`)**: Definen un conjunto de métodos y propiedades que una clase, estructura o enumeración debe adoptar.

   Para los tipos poliformicos se tiene:

   - **Generics**: Permiten definir funciones y tipos que pueden trabajar con cualquier tipo, especificado en tiempo de ejecución.
   - **Protocolos con asociaciones (`associatedtype`)**: Permiten definir métodos y propiedades en protocolos que dependen de un tipo asociado, permitiendo el polimorfismo en las implementaciones de dichos protocolos.

2. Describa el funcionamiento del sistema de tipos del lenguaje, incluyendo el tipo de equivalencia para sus tipos, reglas de compatibilidad y capacidades de inferencia de tipos.

   El sistema de tipos de Swift es fuerte y estáticamente tipado, lo que significa que los tipos de todas las variables deben conocerse en tiempo de compilación. Su funcionamiento es el siguiente:

   - **Equivalencia de Tipos**: Swift utiliza equivalencia por nombre, donde dos tipos son equivalentes si tienen el mismo nombre o declaración. Por ejemplo, dos estructuras con los mismos atributos pero con nombres diferentes no se consideran equivalentes. Cabe descatar que en el caso de tuplas se comparan sus elementos y etiquetas, haciendo una equivalencia estructural.
   - **Reglas de Compatibilidad**: Swift impone reglas estrictas para la compatibilidad de tipos. No permite la conversión implícita entre tipos diferentes, por lo que se debe usar conversión explícita (casting) para convertir entre tipos compatibles. Por ejemplo, `Int` no se convierte automáticamente a Double. En el caso de la herencia, una variable de tipo base puede referenciar instacias de clases derivadas.
   - **Inferencia de Tipos**: Swift tiene un sistema de inferencia de tipos que permite al compilador deducir el tipo de una variable a partir de su valor inicial o del contexto en que se usa, reduciendo la necesidad de anotaciones de tipos explícitas. Por ejemplo:

```
let number = 42  // El compilador infiere que `number` es de tipo `Int`
let message = "Hello, World!"  // El compilador infiere que `message`
                               // es de tipo `String`
```

   - **Tipos Opcionales**: Swift maneja la ausencia de valor con opcionales (`Optional`), que pueden contener un valor o `nil`. Esto ayuda a evitar errores comunes relacionados con valores nulos.
   - **Tipos Genéricos**: Swift permite la creación de funciones y tipos genéricos que pueden trabajar con cualquier tipo, proporcionando flexibilidad y reutilización de código. Por ejemplo:

```
func swapTwoValues<T>(_ a: inout T, _ b: inout T) {
    let temporaryA = a
    a = b
    b = temporaryA
}
```

3. Diga si la totalidad de los tipos de datos del lenguaje conforman un álgebra. Diga si posse construcciones que corresponden a: el tipo producto, el tipo suma, el tipo cero (neutro de la suma) y el tipo uno (neutro del producto).

   En Swift, no todos los tipos de datos conforman un álgebra en el sentido matemático estricto, pero se pueden relacionar algunos conceptos con construcciones algebraicas de tipos. Por ejemplo:

   - **Tipo Producto**: Las tuplas y las estructuras (`struct`) en Swift corresponden al tipo producto. Permiten agrupar varios valores en un solo tipo.

```swift
struct Point {
    var x: Int
    var y: Int
}
let point = Point(x: 1, y: 2)
```

   - **Tipo Suma**: Las enumeraciones (`enum`) representan el tipo suma, donde un valor puede ser uno de varios casos posibles.

```swift
enum Shape {
    case circle(radius: Double)
    case rectangle(width: Double, height: Double)
}
let shape = Shape.circle(radius: 5.0)
```

   - **Tipo Cero (Neutro de la Suma)**: En Swift, no existe un tipo que represente directamente el tipo cero. Sin embargo, a veces se considera que el tipo `Never` puede aproximarse a este concepto, ya que no tiene valores posibles y se utiliza típicamente en situaciones que nunca deberían ocurrir.

```swift
func fatalErrorExample() -> Never {
    fatalError("Esta funcion no retorna nunca")
}
```

   - **Tipo Uno (Neutro del Producto)**: El tipo uno se puede representar con la tupla vacía `()`, también conocido como `Void`. Es un tipo con un único valor.

```swift
let value: Void = ()
```