**Comprender y Utilizar las Principales Estructuras de Datos en Kotlin, Incluyendo arreglos, listas, conjuntos, mapas y pares**

Camilo A. Forero Mira

Tecnología en Análisis y Desarrollo de Sistemas de Información, Servicio Nacional de Aprendizaje

2469285: Taller Estructuras de Datos en Kotlin

Inst. Cristian David Henao Hoyos

20 de febrero de 2023

# Estructuras de Datos

El aprendiz deberá realizar un informe donde se evidencien los siguientes puntos:

**1. Introducción a las estructuras de datos en Kotlin**

**a. ¿Qué son las estructuras de datos y para qué se utilizan?**

R/= En Kotlin las estructuras de datos se utilizan para organizar y almacenar datos en la memoria de un programa de software. Pueden realizar operaciones específicas como buscar, insertar, eliminar, ordenar. Son parte fundamental de la programación y se utilizan en una amplia variedad de aplicaciones.

**b. Ventajas de utilizar estructuras de datos en Kotlin**

R/=

* **Mayor eficiencia:** Las estructuras de datos optimizan el acceso a los datos y las operaciones, por lo que puede mejorar el rendimiento y la eficiencia del programa.
* **Mejor organización:** Las estructuras de datos permiten organizar los datos de una manera más lógica y coherente, facilitar el mantenimiento y la modificación del código.
* **Mayor seguridad:** Al utilizar estructuras de datos, se puede evitar el riesgo de errores y problemas comunes que pueden surgir al trabajar con datos de forma manual.

**c. Diferencias entre las estructuras de datos en Kotlin y Java**

R/=

* Kotlin ofrece una estructura de datos adicionales que no están disponibles en Java, como variables nulas seguras y las listas inmutables.
* Kotlin tiene una sintaxis más clara y concisa para trabajar con estructuras de datos, por lo que el código se hace más fácil de leer y de mantener.
* En Kotlin no es necesario declarar explícitamente el tipo de variable, en Java se debe especificar explícitamente el tipo (cadena, flotante, etc.) al declarar una variable.
* Java cuenta con una gran comunidad de desarrolladores debido a su presencia en el campo del desarrollo hace mucho tiempo, por lo que se encontrara una solución fácil a un problema, Kotlin tiene comparativamente menos desarrolladores en su comunidad, la adaptación ha aumentado desde el anuncio de Google.

2. Arreglos en Kotlin

**a. ¿Qué es un arreglo?**

R/= Un arreglo es una estructura que nos permitirá almacenar datos en diferentes posiciones a los que se les conoce como índices y estos van desde 0 hasta el tamaño -1.

**b. Creación de arreglos en Kotlin**

Texto

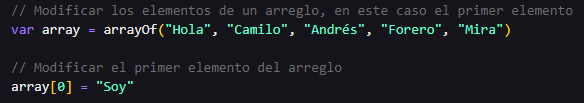
Descripción generada automáticamenteR/= En Kotlin, se pueden crear arreglos utilizando la función arrayOf() o el constructor Array().

**c. Accediendo a los elementos de un arreglo**

Texto

Descripción generada automáticamenteR/= Para acceder a los elementos de un arreglo en Kotlin, se utilizan los corchetes [] y se indica dentro de ellos el índice del elemento al cual se quiere acceder.

**d. Modificando los elementos de un arreglo**

R/= Para modificar los elementos de un arreglo en Kotlin, al igual que para acceder a ellos se utilizan los corchetes, para indicar el elemento al que le queremos asignar un nuevo valor.

**e. Recorriendo un arreglo**

Texto

Descripción generada automáticamenteR/= Para recorrer un arreglo en Kotlin, se utiliza un bucle for o forEach.

**f. Funciones útiles para trabajar con arreglos en Kotlin**

R/=

* **size:** Devuelve el número de elementos del arreglo.
* **get:** Esta función se utiliza para acceder a un elemento específico dentro de un arreglo.
* **set:** Asigna un nuevo valor al elemento del arreglo en el índice especificado.
* **slice:** Esta función se utiliza para obtener un subconjunto de elementos de un arreglo.
* **indexOf():** Devuelve el índice del primer elemento en el arreglo que es igual al elemento especificado.
* **sort()**: Ordena los elementos del arreglo en orden ascendente.
* **filter:** Esta función se utiliza para filtrar elementos de un arreglo que cumplan cierta condición.
* **map:** Esta función se utiliza para aplicar una transformación a cada elemento de un arreglo.

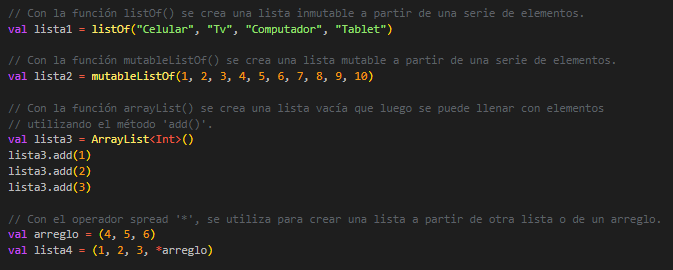
**3. Listas en Kotlin**

**a. ¿Qué es una lista?**

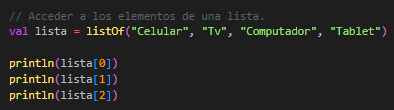
R/= Una lista en Kotlin es una colección ordenada de elementos del mismo tipo, similar a un arreglo, pero con la diferencia de que se pueden agregar, eliminar o modificar elementos de ella.

**b. Creación de listas en Kotlin**

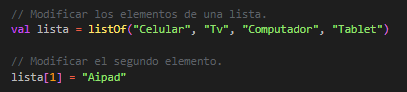
R/= Una lista en Kotlin se puede crear de varias maneras:



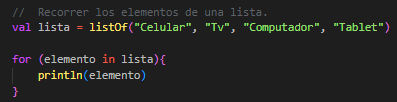
**c. Accediendo a los elementos de una lista**

R/= Para acceder a los elementos de una lista en Kotlin, empieza por el índice 0 para el primer elemento.

**d. Modificando los elementos de una lista**

R/=

**e. Recorriendo una lista**

R/=

**f. Funciones útiles para trabajar con listas en Kotlin**

R/=

* **size:** Devuelve el número de elementos en la lista.
* **contains:** Devuelve ‘true’ si la lista contiene un elemento determinado o ‘false’ en caso contrario.
* **indexOf:** Devuelve el índice de la primera ocurrencia del elemento en la lista, o -1 si el elemento no está presente.
* **subList:** Devuelve una sublista de la lista original, que va desde un índice inicial hasta un índice final.
* **sorted:** Devuelve una lista ordenada en orden ascendente según el orden natural de los elementos de la lista.
* **filter:** Devuelve una lista que contiene solo los elementos de la lista original que cumplen cierta condición.
* **map:** Devuelve una lista que contiene los resultados de aplicar una función determinada a cada elemento de la lista original.

**4. Conjuntos en Kotlin**

**a. ¿Qué es un conjunto?**

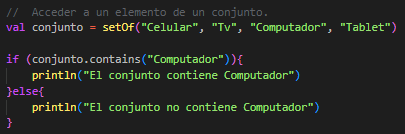
R/= Un conjunto en Kotlin set es una colección de elementos únicos e inmutables, los elementos no se pueden repetir y se almacenan en orden no definido.

**b. Creación de conjuntos en Kotlin**

R/= En Kotlin se puede crear un conjunto usando la función ‘setOf()’.

**c. Accediendo a los elementos de un conjunto**

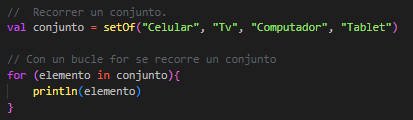
R/= Para acceder a un conjunto se utiliza la función ‘contains()’.



**d. Modificando los elementos de un conjunto**

R/= Los conjuntos en Kotlin son inmutables, por lo que no se pueden modificar una vez se hayan creado. Para modificar el contenido del conjunto, se puede crear un nuevo conjunto con los datos actualizados.

**e. Recorriendo un conjunto**

R/= Para recorrer un conjunto se utiliza un bucle for.

**f. Funciones útiles para trabajar con conjuntos en Kotlin**

R/=

* **union():** Devuelve un conjunto que contiene todos los elementos de dos conjuntos.
* **intersect():** Devuelve un conjunto que contiene solo los elementos comunes de dos conjuntos.
* **subtract():** Devuelve un conjunto que contiene todos los elementos del primer conjunto que no están en el segundo conjunto.
* **isEmpty():** Devuelve ‘true’ si el conjunto está vacío o ‘false’ en caso contrario.

**5. Mapas en Kotlin**

**a. ¿Qué es un mapa?**

R/= Un mapa en Kotlin es una colección de partes clave-valor, donde cada clave esta asociada con un valor. Se utiliza para almacenar y acceder a datos de manera eficiente utilizando una clave única como identificador.

**b. Creación de mapas en Kotlin**

R/= Para crear un mapa en Kotlin se utiliza la clase ‘HashMap’, que implementa la interfaz ‘MutableMap’.



También se puede inicializar el mapa con valores predeterminados:



**c. Accediendo a los elementos de un mapa**

R/= Para acceder a los elementos de un mapa se utiliza la clave correspondiente



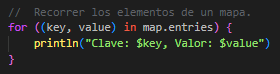
Esto devolverá el valor ‘1’, ya que la clave es ‘a’. Si la clave no está presente devolverá null.

**d. Modificando los elementos de un mapa**

R/= Para modificar un elemento en un mapa, se utiliza la misma sintaxis que se usa para acceder a él, pero se le asigna un nuevo valor.

**e. Recorriendo un mapa**

R/= Para recorrer un mapa, se utiliza el bucle for junto con la condición ‘entries’.



**f. Funciones útiles para trabajar con mapas en Kotlin**  
R/=

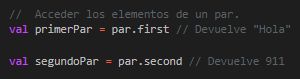
* **isEmpty():** Devuelve ‘true’ si el mapa está vacío.
* **size:** Devuelve el número de pares clave-valor en el mapa.
* **containsKey(key: K):** Devuelve ‘true’ si el mapa contiene la clave especificada.
* **containsValue(value: V):** Devuelve ‘true’ si el mapa contiene el valor especificado.
* **put(key: K, value: V):** Agrega un nuevo par clave-valor al mapa, o actualiza el valor si la clave ya está presente.
* **remove(key: K):** Elimina el par clave-valor con la clave especificada del mapa.
* **clear():** Elimina todos los pares clave-valor del mapa.

**6. Pares en Kotlin**

**a. ¿Qué es un par?**  
R/= Un par es una estructura de datos que contiene dos elementos, que pueden ser de cualquier tipo y que se almacenan juntos.

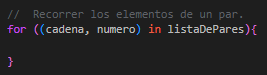
**b. Creación de pares en Kotlin**  
R/= para crear pares en Kotlin se utiliza la función ‘pair()’.

**c. Accediendo a los elementos de un par**  
R/= Para acceder a los elementos de un par, se utilizan las propiedades ‘first’ y ‘second’.



**d. Modificando los elementos de un par**R/= Los elementos de un par son inmutables, por lo que no se pueden modificar directamente. Para modificar algún elemento, se debe crear un nuevo par con los nuevos valores.

**e. Recorriendo un par**  
R/= Ya que un par es una estructura de datos de dos elementos, no se puede recorrer, pero se puede utilizar restructuración para extraer los elementos del par y utilizarlos en un ciclo.



**f. Funciones útiles para trabajar con pares en Kotlin**  
R/=

* **pair:** Esta función permite crear un nuevo par.
* **to:** Esta función es una forma más legible de crear un nuevo par.
* **component1 y component2:** Estas 2 funciones permiten acceder al primer y segundo elemento de un par, respectivamente.
* **first y second:** Estas propiedades permiten acceder al primer y segundo elemento de un par, respectivamente.
* **copy:** Esta función permite crear una copia del par, con la posibilidad de modificar uno o ambos elementos.
* **swap:** Esta función permite intercambiar los elementos de un par.
* **toList():** Esta función se utiliza para convertir un par en una lista que contenga ambos elementos.

**7. Prácticas de estructuras de datos en Kotlin**

a. Ejercicios prácticos para aplicar los conceptos aprendidos  
R/=

**Ejercicio de arreglo:** Cree un arreglo de 9 posiciones y luego imprímalo.

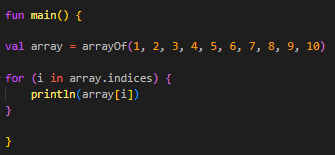
**Ejercicio de lista:** Cree una lista, sume todos sus elementos e imprima el resultado.

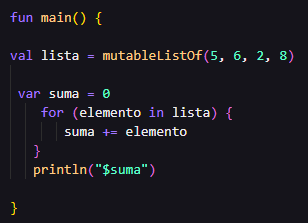
**Ejercicio de conjunto:** Cree un conjunto, utilice una condición para acceder a un elemento y luego imprima la lista.

**Ejercicio de mapa:** Cree un mapa, asígnele a cada clave un valor y luego imprima cada clave con su respectivo valor.

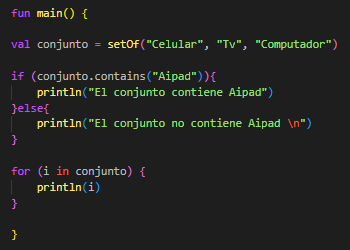
**Ejercicio de par:** Cree un par, accede a uno de los 2 elementos del conjunto e imprímela.

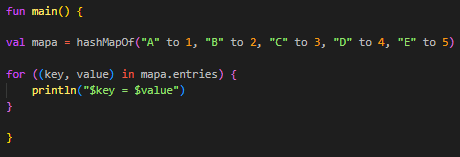
**b. Solución a los ejercicios prácticos**

**Ejercicio de arreglo:** Cree un arreglo de 9 posiciones y luego imprímalo usando un bucle for.

 **Ejercicio de lista:** Cree una lista, sume todos sus elementos e imprima el resultado.

**Ejercicio de conjunto:** Cree un conjunto, utilice una condición para acceder a un elemento y luego imprima la lista.



**Ejercicio de mapa:** Cree un mapa, asígnele a cada clave un valor y luego imprima cada clave con su respectivo valor.

**Ejercicio de par:** Cree un par, accede a uno de los 2 elementos del conjunto e imprímela.

