

Taller Estructuras de Datos en Kotlin

El objetivo de este taller es que los aprendices sean capaces de comprender y utilizar las principales estructuras de datos en Kotlin, incluyendo arreglos, listas, conjuntos, mapas y pares.

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

1. Introducción a las estructuras de datos en Kotlin
   1. ¿Qué son las estructuras de datos y para qué se utilizan?

Las estructuras de datos permiten organizar y almacenar una gran cantidad de datos, su objetivo es facilitar su uso y manipulación

* 1. Ventajas de utilizar estructuras de datos en Kotlin

Facilita la manipulación de datos

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

En kotlin nos es necesario indicar un tipo de dato en las estructuras a diferencia de java que es fuertemente tipado, por tal razón solo se permite un tipo de dato.

1. Arreglos en Kotlin
   1. ¿Qué es un arreglo?

Un arreglo en una secuencia de valores que cuentan con el mismo tipo de dato

* 1. Creación de arreglos en Kotlin

Para declarar un arreglo en kotlin, se usa la función arrayOf() la sintaxis es la siguiente:

val variable\_nombre = arrayOf<data\_type>(element,element,element)

* 1. Accediendo a los elementos de un arreglo

Para acceder a un arreglo se usan los [ ] y se usa el índice del valor al que se quiere acceder de la siguiente manera:

Nombre\_arreglo[1] = "elemento"

* 1. Modificando los elementos de un arreglo

Para modificar se hace de la misma manera para agregar pero en este caso se pone el índice del valor a modificar y se pone el nuevo valor

Nombre\_arreglo[0] = "elemento\_nuevo"

* 1. Recorriendo un arreglo

Para recorrer un arreglo en kotlin se utiliza el ciclo for

for (i in 0  ejemplo.size) {

    println("${ejemplo[i]} está en la posición ${i + 1}")

}

* 1. Funciones útiles para trabajar con arreglos en Kotlin

set() get() add()

set sirve para actualizar un valor del arreglo

get sirve para traer un valor de la arreglo

add sirve para añadir un nuevo dato al arreglo

1. Listas en Kotlin
   1. ¿Qué es una lista?

Es una colección de elementos ordenados sin un limite de datos

* 1. Creación de listas en Kotlin

Para crear una lista se utiliza listOf

val variable\_nombre = listOf(element,element,element)

* 1. Accediendo a los elementos de una lista

Para buscar un dato de una lista se pone el índice de este en [ ], otra forma es usar get y el índice entre ()

println(arreglo[2])

println(solarSystem.get(3))

* 1. Modificando los elementos de una lista

Para modificar se utiliza el índice entre [ ] y se da el nuevo valor

ejemplo[3] = "Future Moon"

* 1. Recorriendo una lista

Se crea una variable que solo sirva para interar y mostrar nuestro arreglo de esta manera:

 for (planet in solarSystem) {

    println(planet)

}

* 1. Funciones útiles para trabajar con listas en Kotlin

Add(),indexOf(),removeAt()

1. Conjuntos en Kotlin
   1. ¿Qué es un conjunto?

Es una colección de valores que no tiene un orden especifico y no permite duplicados

* 1. Creación de conjuntos en Kotlin

 val solarSystem = mutableSetOf("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus", "Neptune")

  println(solarSystem.size)

* 1. Accediendo a los elementos de un conjunto

Contains nos devuelve un valor booleano es decir verdadero o falso en este caso si contiene el planeta venus

println(solarSystem.contains("Venus"))

* 1. Modificando los elementos de un conjunto

Se utiliza remove y el elemento que se quiere borrar

solarSystem.remove("Pluto")

* 1. Recorriendo un conjunto

 for (planet in solarSystem) {

    println(planet)

}

* 1. Funciones útiles para trabajar con conjuntos en Kotlin

Add(),remove(), contains()

1. Mapas en Kotlin
   1. ¿Qué es un mapa?

Es una colección de elementos que estan organizados en pares clave valor

* 1. Creación de mapas en Kotlin

Para declarar un mapa en Kotlin, se utiliza la función "mapOf()"

 val map = mapOf("a" to 1, "b" to 2, "c" to 3)

* 1. Accediendo a los elementos de un mapa

println(map["a"])

* 1. Modificando los elementos de un mapa

val mapa = mutableMapOf("a" to 1, "b" to 2)

  mapa.put("c", 3) *// Agrega un nuevo elemento*

mapa.put("b", 3) *// Actualiza el valor asociado a la clave "b" a 3*

    mapa.remove("b") *// Elimina el elemento con la clave "b" del mapa*

* 1. Recorriendo un mapa

for ((clave, valor) in mapa) {

  println("$clave -> $valor")*//recorrer un mapa*

* 1. Funciones útiles para trabajar con mapas en Kotlin

Get(),put(),clear()

1. Pares en Kotlin
   1. ¿Qué es un par?

En kotlin un par es una tupla de dos elementos que pueden tener diferentes tipos de datos

* 1. Creación de pares en Kotlin

 val miPar = Pair("hola", 123)

* 1. Accediendo a los elementos de un par

 val miCadena = miPar.first *// acceder a pares, miCadena será "hola"*

val miEntero = miPar.second *// acceder a pares, miEntero será 123*

* 1. Modificando los elementos de un par
  2. Recorriendo un par

for ((valor1, valor2) in listOf(miPar)) {

println("Valor 1: $valor1")

println("Valor 2: $valor2")

}

* 1. Funciones útiles para trabajar con pares en Kotlin

toString(), first, second

**Videos:**

Estructuras de control: <https://youtube.com/playlist?list=PLj1Imxz3p1adyixa1ZV5pBRv-RK027QgH>

Estructura de datos: <https://youtube.com/playlist?list=PLj1Imxz3p1aeDerguT6UADndE94OJ6W0z>