TALLER KOTLIN

1. Introducción a las estructuras de datos en Kotlin
a. ¿Qué son las estructuras de datos y para qué se utilizan?
R/ Las estructuras de datos son una forma o una manera de almacenar y gestionar de una manera más eficiente la información. Se emplea para almacenar múltiples valores en una variable y manejar la información. Podemos almacenar vectores donde son usados los arreglos unidimensionales y las matrices para arreglos bidimensionales.

b. Ventajas de utilizar estructuras de datos en Kotlin
Las estructuras de datos en Kotlin son eficientes, fáciles de usar, seguras en términos de tipos, ofrecen funciones de orden superior y pueden ser inmutables. Todo esto hace que trabajar con estructuras de datos sea más fácil y seguro en Kotlin.

c. Diferencias entre las estructuras de datos en Kotlin y Java
Kotlin y Java comparten muchas estructuras de datos comunes, pero Kotlin ofrece características adicionales como soporte para tipos de datos no nulos, colecciones inmutables, funciones de orden superior y una sintaxis más concisa que hace que trabajar con estructuras de datos sea más fácil y seguro en Kotlin.

2. Arreglos en Kotlin
a. ¿Qué es un arreglo?
Un arreglo en Kotlin es una estructura de datos que almacena una colección de elementos del mismo tipo en una sola variable.

b. Creación de arreglos en Kotlin
Creación de arreglos en kotlin

```kotlin
fun main(){
    //Declaracion de arreglos ->
    // Arreglo vacío con un tamaño específico
    val arreglo = arrayOfNulls<Int>(5)

    //Arreglo con valores iniciales
    val arreglo2 = arrayOf(1, 2, 3)

    // arreglo con elementos de diferentes tipos
    val arreglo3 = arrayOf(1, "dos", true)
}
```

c. Accediendo a los elementos de un arreglo

Para acceder a los elementos de un arreglo en kotlin debemos tener un arreglo definido, luego en los corchetes ([ ]) y mandamos el índice al que queremos acceder. Otra manera de acceder a ellos puede ser mediante el método get .

```
val array = arrayOfNulls<Int>( size: 3)
array[0] = 1
array[1] = 6
array[2] = 9
val accederElemento = array[1]
print(accederElemento)
```

```
fun main(){
    val names = listOf<String>("Andres", "Jaramillo", "Celada")
    // Accedemos con get a un indice de la lista
    names.get(1)
}
```

d. Modificando los elementos de un arreglo

Para modificar un arreglo debemos llamar al arreglo, pasarle el índice a modificar y le pasamos el nuevo valor.

```
fun main(){
    val names = listOf<String>("Andres", "Jaramillo", "Celada")
    names.set(2, "Santigo")
}
```

e. Recorriendo un arreglo

```
fun main(){
    val names = listOf<String>("Andres", "Jaramillo", "Celada")

    for(i in names.indices){
        println("$names")
    }
}
```

f. Funciones útiles para trabajar con arreglos en Kotlin

a. size -> Devuelve la cantidad de elementos de un arreglo
b. get -> devuelve el elemento en el índice indicado

c. slice -> Devuelve un arreglo que contiene los elementos en los índices especificados.

3. Listas en Kotlin

a. ¿Qué es una lista?

En Kotlin, una lista es una colección ordenada de elementos, en la que cada elemento puede tener un índice único que lo identifica.

b. Creación de listas en Kotlin

```
fun main(){
    var meses = listOf("Enero", "Febrero", "Marzo")

    var meses2 = mutableListOf("Enero", "Febrero", "Marzo")

    var numeros = arrayList(1, 9, 2, 8)
}
```

c. Accediendo a los elementos de una lista

```
fun main(){
    var numeros = arrayList(1, 9, 2, 8)

    val num1 = numeros.get[1]
    val num2 = numeros.get[0]

    print(numeros[0])
}
```

d. Modificando los elementos de una lista

```
fun main(){
    var lastNames = arrayList("Henao", "Celada", "Orozco", "Ospina")

    lastNames.set(2, "Jaramillo")
    lastNames[0] = "Arias"
    lastNames.remove(3)
}
```

e. Recorriendo una lista

```
fun main(){
    var lastNames = arrayList("Henao", "Celada", "Orozco", "Ospina")

    for (i in lastNames){
        print("$lastNames")
    }

}
```

f. Funciones útiles para trabajar con listas en Kotlin
add -> agrega un elemento a la lista
remove -> elimina el elemento especificado de la lista
indexOf -> devuelve el índice del primer elemento que coincide con el valor especificado.
sort -> ordena la lista de manera ascendente
filter -> devuelve una lista de elementos que cumplen con un cierto criterio
map -> devuelve una lista de elementos que resultan de aplicar una función a cada elemento de la lista original.

4. Conjuntos en Kotlin

A) R/ Una lista es una colección ordenada de elementos. Para trabajar con listas inmutables se utiliza la interface List. Extiende de la interfaz Collection, y define una serie de funciones propias, entre las que podemos destacar:

B) Creación de conjuntos en Kotlin

```kotlin
fun main(){

    val primos: List<Int> = listOf(2, 3, 5, 7)
    val nombres: List<String> = listOf("Esteban", "cristian", "juan")
    val listaMezclada = listOf("Esteban", 1, 2.445, 's')

    println(primos)
    println(nombres)
    println(listaMezclada)


}
```

C) Accediendo a los elementos de un conjunto

```kotlin
println(primos.get(index = 2))
println(nombres.get(1))
println(listaMezclada.get(0))
println(listaMezclada.get(2))
```

D) Modificando los elementos de un conjunto

```kotlin
// Modificar elementos
val listaModificada = listaMezclada.toMutableList()
listaModificada.add("nuevo elemento")
listaModificada.remove("Estevan")
listaModificada[0] = "nuevo elemento"
val primosModificados = primos.toMutableList()
primosModificados.add(11)
primosModificados.remove(2)
primosModificados[0] = 1
val nombresModificados = nombres.toMutableList()
nombresModificados.add("nuevo elemento")
nombresModificados.remove("Estevan")
nombresModificados[0] = "nuevo elemento"

println(nombresModificados)
println(primosModificados)
println(listaModificada)
```

E) Recorriendo un conjunto

```kotlin
// Modificar elementos
for (i in 0..primos.size-1){
    println(primos[i])
}

for (i in 0..nombres.size-1){
    println(nombres[i])
}
for (i in 0..listaMezclada.size-1){
    println(listaMezclada[i])
}
listaMezclada.forEach { println(it) }

println(listaMezclada)
```

F) Funciones útiles para trabajar con conjuntos en Kotlin

```kotlin
fun main(args: Array<String>) {
    val array = arrayOf(1, 2, 3, 4, 5)
    println(array.sum())
}
```

Mapas en Kotlin

A) Un mapa es una **colección que almacena sus elementos (entradas) en forma de pares clave-valor**.

B) Creación de mapas en Kotlin

```kotlin
fun main(){

    val mapaNumeros = mapOf("cod1" to 1, "cod2" to 2, "cod3" to 3, "cod4" to 4, "cod5" to 5)

    println("Mapa de numeros: $mapaNumeros")
}
```

C) Accediendo a los elementos de un mapa

```kotlin
otlin taller > ijercice.kt > main
1
2  fun main(){
3
4      val mapaNumeros = mapOf("cod1" to 1, "cod2" to 2, "cod3" to 3, "cod4" to 4, "cod5" to 5)
5
6      println("Mapa de numeros por key: ${mapaNumeros.keys}")
7      println("Mapa de numeros por value: ${mapaNumeros.values}")
8
9  }
10
```

D) Modificando los elementos de un mapa

```kotlin
val mapaNumeros = mapOf("cod1" to 1, "cod2" to 2, "cod3" to 3, "cod4" to 4, "cod5" to 5)


val mapaNumerosModificado = mapaNumeros.toMutableMap()

mapaNumerosModificado.put("cod1", 10)
```

E) Recorriendo un mapa

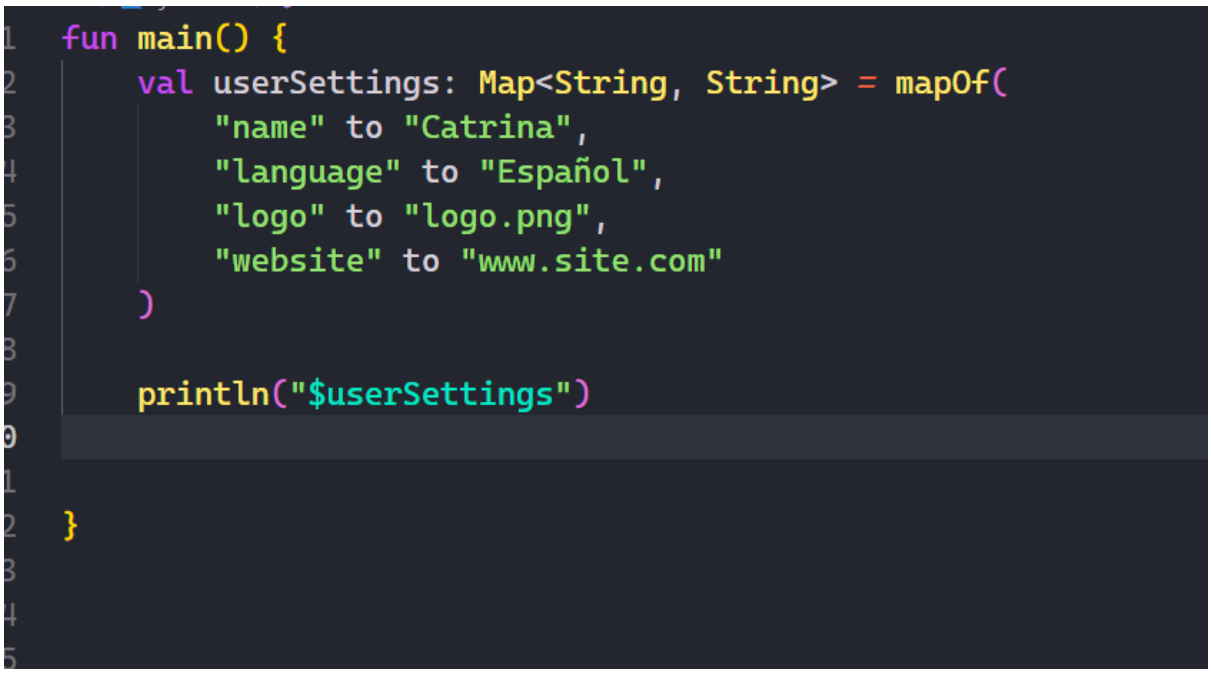
```kotlin
fun main(){

    val mapaNumeros = mapOf("cod1" to 1, "cod2" to 2, "cod3" to 3, "cod4" to 4, "cod5" to 5)


    for ((key, value) in mapaNumeros) {
        println("key: $key, value: $value")
    }

}
```

F) Funciones útiles para trabajar con mapas en Kotlin



```kotlin
fun main() {
    val userSettings: Map<String, String> = mapOf(
        "name" to "Catrina",
        "language" to "Español",
        "logo" to "logo.png",
        "website" to "www.site.com"
    )

    println("$userSettings")


}
```

Pares en Kotlin

A) son **una representación genérica (cualquier tipo de datos o clases) de dos valores** (pares). Los data class Pair son una estructura que permite guardar dos

B) Creación de pares en Kotlin

```kotlin
fun main() {
    var pair = Pair("Kotlin Pair",2)
    var pair1 = "Kotlin Pair" to 2
    val (user, password) = Pair("usuario", "contrasena")

    println(pair)
    println(pair1)
    println(user)
    print(password)



}

```

C) Accediendo a los elementos de un par

```kotlin
fun main() {
    val pair = Pair("Kotlin", "Java")

    println(pair.first)
    println(pair.second)
}
```

D) Modificando los elementos de un par

ijercice.kt

ijercice.kt > main

```kotlin
fun main() {
    val pair = Pair("Kotlin", "Java")

    val (first, second) = pair
    val newPair = first to second
    newPair.first
    newPair.second
    println(newPair)

}
```

E) Recorriendo un par

ijercice.kt > ...

```kotlin
fun main() {

    val par = 1..10 step 2
    for (i in par) {
        println(i)
    }

}
```

F) Funciones útiles para trabajar con pares en Kotlin

ijercice.kt

ijercice.kt > main

```kotlin
fun main() {
    val pair = Pair("Kotlin", "Java")

    println(pair.first)
    pair.hashCode()
    pair.toString()
    pair.component1()
    pair.component2()
    pair.toList()
    println(pair)

}
```

A) Ejercicios prácticos para aplicar los conceptos aprendidos
permiten almacenar varios valores, generalmente del mismo tipo de datos, de manera organizada

ijercice.kt > main

```kotlin
fun main() {

val lista: List<String> = listOf("uno", "dos", "tres")
println(lista.last())
println(lista.last{ it.length == 3 })
println(lista.count()) // 3
println(lista.count{ it.length == 3 })
val conjunto: Set<Int> = setOf(3, 6, 5, 5, 5, 3)
println(conjunto.last())
println(conjunto.count())

val par = Pair("uno", 1)
println(par.first)
println(par.second)
val triple = Triple("uno", 1, 1.0)
println(triple.first)

}
```