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 un modo de mostrar la información en una computadora y tener buenas prácticas ya que tiene un comportamiento interno con una serie de reglas. Son aquellas que nos permiten organizar toda la información que entra de manera eficiente y luego diseñar una solución

b. Ventajas de utilizar estructuras de datos en Kotlin

R/El tiempo de acceso a los datos es muy corto ya que es eficiente y nos crea un software optimizado, nos brinda muchas herramientas para solucionar problemas.Al igual que en Java, Kotlin tiene arreglos, mapas, listas, etc que nos ayudan a mejorar nuestro código aunque tienen pequeñas diferencias.

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

R/ Java tiene un proceso más largo que Kotlin pero se quiere menos tiempo para analizarlo, en cambio kotlin tiene un código más reducido, pero tarda más tiempo en solucionarlo.

2. Arreglos en Kotlin

a. ¿Qué es un arreglo?

R/Un arreglo es una matriz que captura datos (de cualquier tipo) en una sola variable y los ordena

b. Creación de arreglos en Kotlin

R/ val array = arrayOf <Double>(1.5,2.7,2.5)

c. Accediendo a los elementos de un arreglo

R/array.get(2)

d. Modificando los elementos de un arreglo

R/ array.set(3,2.4)

e. Recorriendo un arreglo

R/ val array = arrayOf<Int>(3,5,6)

for (i in 0 until array.size){

print("[$i,${array[i]}]")

}

f. Funciones útiles para trabajar con arreglos en Kotlin

set : Sirve para asignar un valor a un índice

get: Se utiliza para llamar un índice

3. Listas en Kotlin

a. ¿Qué es una lista?

R/ Es un conjunto de elementos con un orden en específico y puede tener cualquier tamaño

b. Creación de listas en Kotlin

R/ val list= listOf<Int>(1,2,3) //Lista de solo lectura

val list = mutableListOf <Int>(1,2,3)// que nos permite agregar,remover o actualizar sus elementos

c. Accediendo a los elementos de una lista

R/ println(list[0])

print(list.get[2])

d. Modificando los elementos de una lista

R/ list.add(3)

e. Recorriendo una lista

R/ for (i in list){

print(i)

}

f. Funciones útiles para trabajar con listas en Kotlin

- R/ add:Para agregar algo a una lista
- clear :Para eliminar todo lo de la lista
- size: Para saber el tamaño de una lista
- addAll:para agregar una lista entera de elementos a nuestra lista
- contains: Para saber si hay un valor en específico en la lista
- isEmpty: revisar si la lista está vacía
- reversed: Para mostrar los elementos de forma inversa
- remove: Para quitar un elemento de una lista
- removeAt: Quitamos un elemento a través de un índice
- first: para saber el primer elemento
- las: Para ver el último elemento

4. Conjuntos en Kotlin

a. ¿Qué es un conjunto?

R/es una colección de información sin un orden y no puede haber ningún duplicado, hay conjuntos de solo lectura y mutables.

b. Creación de conjuntos en Kotlin

R/ val n = setOf (1,2,3,4,5) // PRIMERA FORMA

Val n = mutableSetOf <Int>(1,2,3,4,5) // SEGUNDA FORMA

c. Accediendo a los elementos de un conjunto

R/ Solo podemos acceder de manera grupal a los elementos, ya que se utilizan en conjunto

print(n)

d. Modificando los elementos de un conjunto

R/ Para agregar :

n.add(3)

n+=1

Para Remover:

n.remove(2)

n-=3

e. Recorriendo un conjunto

R/

for(i in n) {

Print(i)

}

f. Funciones útiles para trabajar con conjuntos en Kotlin

- R/ add: Para agregar un elemento
- remove: Para remover un elemento
- contains: para buscar un elemento
- intersect: Aislar dos elementos que estén dentro de dos colecciones
- unión: toma dos conjuntos y retorna un conjunto con los elementos que sean de ambos
- subtract:Diferenciar dos conjuntos

5. Mapas en Kotlin

a. ¿Qué es un mapa?

R/Es una herramienta del sistema que sirve para almacenar objetos de clave y valor

b. Creación de mapas en Kotlin

R/ val map : Map <String, String> = mapOf(“juan” to “armenia”, “mari” to “calarca”)

val map = mutableMapOf("Sinsajo" to 2010”)

c. Accediendo a los elementos de un mapa
R/print(map.get(1))
print(map.containskey(1))
print(map.containsvalue(“armenia”))// para saber si tienen un valor o clave en específico

d. Modificando los elementos de un mapa

R/ map.put(“hola ”, 1990)

e. Recorriendo un mapa

R/ for ((key,value) in map){

print(“la clave $key pertenece a $value”)

}

f. Funciones útiles para trabajar con mapas en Kotlin

R/

- get(K) :obtiene el valor de la clave
- isEmpty() : Revisa si el mapa está vacío
- containsKey(K) : Revisa si existe la clave en el mapa (devuelve un boolean)
- containsValue(V) : Revisa si el valor está en el mapa (devuelve un boolean)
- keys : devuelve un Set inmutable con todas las claves en el mapa.
- values :Collection inmutable de todos los valores en el mapa.
- remove(K) : borra la clave y su valor enlazado. (Solo en mutables)
- clear() : elimina todos los elementos del mapa.
- put():agregar un nuevo elemento al mapa

Pares en Kotlin

a. ¿Qué es un par?

R/ Es una estructura que permite guardar dos valores

b. Creación de pares en Kotlin

R/ var pair = Pair("Kotlin Pair",2)

c. Accediendo a los elementos de un par

R/print (pair.first)
print(pair.second)
//Otra manera
val (name,id)= pair(“mari”,123)

d. Modificando los elementos de un par

R/ name =”juan”
id=”234”
print(name)
print(id)

e. Recorriendo un par

R/ nn

f. Funciones útiles para trabajar con pares en Kotlin

R/ nn

7. Prácticas de estructuras de datos en Kotlin

a. Ejercicios prácticos para aplicar los conceptos aprendidos
b. Solución a los ejercicios prácticos

R/

1. Realice un algoritmo que pida n cantidad de materias y las guarde en un arreglo
2. Realice un algoritmo con una lista de herramientas en donde se puede agregar y remover herramientas e imprimirlas
3. Realice un ejercicio que saque los números repetidos de dos conjuntos e imprima si es igual a 4.4
4. Realice un ejercicio donde hayan 5 diferentes nombres y se puedan conocer las llaves para acceder a los nombres y se impriman
5. Realice un ejercicio que guarde en nombre y id de una persona y los imprima