Taller De Estructuras De Datos

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

       R/ son aquellas que nos permiten, como desarrolladores, organizar la información de manera eficiente, y se utilizan para diseñar la solución correcta para un determinado problema.

    b. Ventajas de utilizar estructuras de datos en Kotlin

       R/

       - permite modificar globalmente las variables sin tener que recorrer el código buscando cada aparición.
       - define variables y evita que cambien entre rutinas.
       - separa desde el inicio del programa el espacio de memoria.

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

       R/

       - Ambos tienen casi la misma velocidad de codificación, pero Java requiere escribir más código. Sin embargo, se dedica menos tiempo a pensar la solución con Java.
       - Kotlin tiene construcciones más concisas, por lo que requiere escribir menos código. Sin embargo, se tarda más tiempo en encontrar la solución a una tarea.

2. Arreglos en Kotlin

    a. ¿Qué es un arreglo?

    R/ es una colección ordenada de datos que se emplean para almacenar múltiples valores en una sola variable, frente a las variables que sólo pueden almacenar un valor.

    b. Creación de arreglos en Kotlin

    R/ fun main(args: Array<String>){

    val arreglo = arrayOf(elemento1, elemento2, ...)

    }

    c. Accediendo a los elementos de un arreglo

    R/ println(”nombre del arreglo”(posición a la cuan se quiera acceder))

    d. Modificando los elementos de un arreglo

    R/ “nombre del arreglo". Set(”posición del contenido a modificar, “contenido nuevo” ”)

    e. Recorriendo un arreglo

    R/ for(”desde la posición en la cual quiere iniciar” in “el nombre del arreglo en el cual quiere acceder”){

        println(”que imprima desde la posición inicial”)

    }

    f. Funciones útiles para trabajar con arreglos en Kotlin

    R/ “nombre del arreglo".joinToString()//permite crear un String legible para imprimir.

    “nombre del arreglo".size //Muestra el tamaño de la lista

3. Listas en Kotlin

    a. ¿Qué es una lista?

    R/ Una lista es una colección genérica de elementos que se caracteriza por almacenarlos de forma ordenada, donde pueden existir duplicados (incluso un ítem null) y se indexan los elementos con base 0

b. Creación de listas en Kotlin

R/ fun main(){

//lista inmutable

val “nombre de la variable”: List<String> = listOF(”elemento 1”,”elemento 2”, …)

//lista mutable

var“nombre de la variable”: MutableList<String> = mutableListOf()

}

c. Accediendo a los elementos de una lista

R/ println(”nombre de la lista”.get(”posición a la que se quiera acceder”)

d. Modificando los elementos de una lista

R/ // solo en mutables

“nombre del arreglo". Set(”posición del contenido a modificar, “contenido nuevo” ”)

e. Recorriendo una lista

R/ for(”desde la posición en la cual quiere iniciar” in “el nombre de la lista en el cual quiere acceder”){

println(”que imprima desde la posición inicial”)

}

f. Funciones útiles para trabajar con listas en Kotlin

R/

“nombre de la lista”.size //Muestra el tamaño de la lista

“nombre de la lista”.get**()** //Devuelve el valor

“nombre de la lista”.first**()** //Devuelve el primer valor

“nombre de la lista”.last**()** //Devuelve el último valor

“nombre de la lista”.none**()** //Nos devuelve un true si está vacía la lista

“nombre de la lista”.firstOrNull**()** //Nos devolverá el primer campo, y si no hay, un null.

“nombre de la lista”.elementAtOrNull**(**2**)** //El elemento del índice 2, si no hay, devolverá un null

“nombre de la lista”.lastOrNull**()** //Último valor de la lista o null

“nombre de la lista”.average()//saca un promedio de los datos en la lista

“nombre de la lista”.removeAt()//elimina un elemento de una determinada posición

println**(**“nombre de la lista”**)**//Devuelve la lista completa

4. Conjuntos en Kotlin
    a. ¿Qué es un conjunto?

        R/ Un conjunto es una colección que no tiene un orden específico y no permite valores duplicados.

    b. Creación de conjuntos en Kotlin

        R/ fun main(){

            // conjunto de valores de un solo tipo

            val “nombre de la variable”: Set<Tipo de valor> = setOf(elemento1,elemento2,…)

            // conjunto de valores mezclados

            val “nombre de la variable” = setOf(elemento1,elemento2,…)

            // conjunto de valores mutables

            val “nombre de la variable” = mutableSetOf(elemento1,elemento2,…)

        }

    c. Accediendo a los elementos de un conjunto

        R/ println(”nombre de la variable”.contains(”valor a consultar”)

    d. Modificando los elementos de un conjunto

        R/ ”nombre de la variable”.remove(”valor a eliminar”)

        ”nombre de la variable”.add(”valor a agregar”)

    e. Recorriendo un conjunto

R/ for(”desde la posición en la cual quiere iniciar” in “el nombre de la lista en el cual quiere acceder”){

println(”que imprima desde la posición inicial”)

}

f. Funciones útiles para trabajar con conjuntos en Kotlin

R/contains()// Deja validar si un dato se encuentra en una variable

remove()// Deja eliminar valores de una variable

add()// Deja añadir valores a una variable

union()//toma como argumentos dos colecciones y retorna en un conjunto con todos los elementos que pertenezcan a ambas

intersect()//El resultado será en conjunto intermedio de coincidencias

subtract()// el conjunto que resulta de eliminar de A cualquier elemento que esté en B

5. Mapas en Kotlin

a. ¿Qué es un mapa?

R/ es una colección que almacena sus elementos (entradas) en forma de pares clave-valor

b. Creación de mapas en Kotlin

R/ fun main() {

// variable inmutable
val “nombre de la variable”: Map<”propiedades”, “valores”> =
mapOf(”propiedades” to “valores”,…)

// variable mutable

val “nombre de la variable” = mutableMapOf(”propiedades” to “valores”,…)

}

c. Accediendo a los elementos de un mapa

R/
println(”Nombre de la variable”.entries)
println(”Nombre de la variable”.values)

d. Modificando los elementos de un mapa

   ”Nombre de la variable”[”nombre de la propiedad a cambiar”] = “valor del cambio”

e. Recorriendo un mapa

   R/ for(”desde la posición en la cual quiere iniciar” in “el nombre de la lista en el cual quiere acceder”){

   println(”que imprima desde la posición inicial”)

f. Funciones útiles para trabajar con mapas en Kotlin

   R/ ”Nombre de la variable”.remove(”propiedad”)

   ”Nombre de la variable”[”clave”]//sintaxis permite obtener el valor a partir de la clave en el corchete

   ”Nombre de la variable”.getOrDefault(”propiedad”, “valor”)//Obtiene el valor correspondiente a la clave, de lo contrario retorna

   ”Nombre de la variable”.isEmpty()// Retorna true si el mapa no contiene entradas y false en caso contrario

   ”Nombre de la variable”.containsKey(”propiedad”) Retorna true si key existe en el mapa

   ”Nombre de la variable”.containsValue(”valores”)// Retorna true si una o varias claves se relacionan con value

6. Pares en Kotlin

a. ¿Qué es un par?

   R/Es una clase genérica simple que puede almacenar dos valores de tipos de datos iguales o diferentes, y puede haber o no una relación entre los dos valores.

b. Creación de pares en Kotlin

   R/fun main(){

   val(”nombre del valor 1”,”nombre del valor 2”) = Pair(”valor 1”,valor2”)

   }

c. Accediendo a los elementos de un par

println(pair.first)

println(pair.second)

d. Modificando los elementos de un par

   R/ no se pueden modificar porque es una variable inmutable

e. Recorriendo un par

   R/ no se puede recorrer porque es una variable inmutable

f. Funciones útiles para trabajar con pares en Kotlin

   R/pair.first//trae el primer valor

   pair.second//trae el segundo valor

7. Prácticas de estructuras de datos en Kotlin

   a. Ejercicios prácticos para aplicar los conceptos aprendidos

      R/

      - Se desea guardar los sueldos de 5 operarios. Según lo conocido deberíamos definir 5 variables si queremos tener en un cierto momento los 5 sueldos almacenados en memoria.
      - Crear una lista mutable con las edades de varias personas. Probar las propiedades y métodos principales para administrar la lista mutable.0
      - cree 2 variables de números de un tamaño de 5 y luego pruebe las diferentes propiedades y un ciclo.
      - cree un mapa de libros mutable y luego pruebe las diferentes propiedades y un ciclo.
      - variable que imprima el nombre y el apellido de una persona

   b. Solución a los ejercicios prácticos Recursos adicionales:

      R/ https://github.com/ELMASTER72/Kotlin_ejercicios.git

> Documentación oficial de Kotlin:
> https://kotlinlang.org/docs/reference/
>
> **Entrega.**

> Se deberá realizar la entrega de un informe con la solución de los puntos anteriores, el aprendiz acompañará la investigación con ejemplos practicos de cada estructura y deberá publicar el código fuente en un repositorio en GitHub.