

Estructura de datos

Las estructuras de datos son una forma de almacenar información para luego acceder a ella, modificarla y manipularla más fácilmente, tienen un comportamiento interno y se rige por reglas dadas por la forma en que está construida internamente.

- Ventajas de utilizar las estructuras de datos

1. Permite modificar globalmente las variables sin tener que recorrer el código buscando cada aparición
2. Define las variables y evita que cambien entre rutinas
3. Separa desde el inicio del programa el espacio en memoria
4. Se trabaja más fácil el paso de datos entre aplicaciones.

¿Qué es un arreglo?

Los arreglos son una estructura que nos permitirá almacenar diferentes datos de un mismo tipo, los datos se almacenan en diferentes posiciones, pueden ser datos numéricos, texto, objetos o también pueden ser combinados dependiendo de la necesidad.

- Creación de arreglos en Kotlin

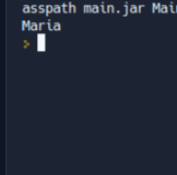
```
fun main() {  
    var arreglo:Array<String> = arrayOf("Pepe" , "Juan" , "Carlos")  
}
```

- Accediendo a los elementos de un arreglo

```
fun main() {  
    var arreglo:Array<String> = arrayOf("Pepe" , "Juan" , "Carlos")  
  
    println(arreglo[0])  
    println(arreglo[1])  
    println(arreglo[2])  
}
```

- Modificando los elementos de un arreglo

```
fun main() {  
    var arreglo:Array<String> = arrayOf("Pepe" , "Juan" , "Carlos")  
    arreglo[1] = "Maria"  
    println(arreglo[1])  
}
```



- Recorriendo un arreglo

```
fun main() {  
    var arreglo:Array<String> = arrayOf("Pepe" , "Juan" , "Carlos")  
    for (i in arreglo)  
        println(i)  
}
```

- Funciones útiles para trabajar con arreglos en Kotlin

- ArrayOf = Función arrayOf toma los elementos del arreglo como parámetros y muestra un arreglo del tipo que coinciden con los parámetros que se pasaron

```
fun main() {  
    val Planetas = arrayOf<String>("Mercury", "Venus", "Earth", "Mars")  
}
```

Listas en kotlin

- ¿Qué es una lista?

Una lista es la colección de los elementos con un orden específico sin embargo hay dos tipos de listas :

List: es una interfaz la cual no se pueden modificar después de su creación.

MutableList: Nos permite modificar después de su creación, lo que permitirá modificar una lista, agregar elementos o quitarlo.

- Creación de listas en Kotlin

Elementos que no se podrán modificar.

```
fun main() {  
  
    val list = listOf(1,2,3,4,5)  
    println(list)  
  
}
```

Elementos que se pueden modificar.

```
fun main() {  
  
    val list = mutableListOf(1,2,3,4,5)  
    println(list)  
  
}
```

- Accediendo a los elementos de una lista

```
fun main() {  
  
    val list = mutableListOf(1,2,3,4,5)  
    println(list[2])  
    println(list)  
  
}
```

- Modificando los elementos de una lista

```
fun main() {  
  
    val list = mutableListOf(1,2,3,4,5)  
    list.add(6)  
    println(list)  
  
}
```

- Recorriendo una lista

```
fun main() {  
  
    val list = listOf(1,2,3,4,5)  
    println(list)  
  
    for (i in list){  
        println(i)  
    }  
  
}
```

```
fun main() {  
  
    val list = mutableListOf(1,2,3,4,5)  
    println(list)  
  
    for (i in list){  
        println(i)  
    }  
  
}
```

- Funciones útiles para trabajar con listas en Kotlin

ListOf : Toma los elementos como parámetros, pero muestra un elemento.

```
fun main() {  
  
    val list = mutableListOf(1,2,3,4,5)  
    |  
  
}
```

SetOf : Nos devuelve los valores ingresados, pero nos muestra los elementos repetidos

```
fun main() {
    val list = setOf(1,2,3,4,4,5,6,2,9)
    println(list)
}
```

```
asspath main.jar: main()
[1, 2, 3, 4, 5, 6, 9]
>
```

Conjuntos en Kotlin

- ¿Qué es un conjunto?

Un conjunto en kotlin es una colección que no tiene orden específico y no permite los valores duplicados

- Creación de conjuntos en Kotlin

```
fun main() {
    val conjunto: Set<Int> = setOf(1, 3, 4)
    val conjuntoMezclado = setOf(2, 4.454, "casa", 'c')
}
```

- Accediendo a los elementos de un conjunto

```
fun main() {
    val conjunto: Set<Int> = setOf(1, 3, 4)
    val conjuntoMezclado = setOf(2, 4.454, "casa", 'c')

    println(conjunto)
    println(conjuntoMezclado)
}
```

- Modificando los elementos de un conjunto
- Recorriendo un conjunto
- Funciones útiles para trabajar con conjuntos en Kotlin

Mapas en Kotlin

- ¿Qué es un mapa?

Estructura de datos que consta de claves y valores, El subíndice puede ser cualquier tipo de clase igual que si valor.

- Creación de mapas en Kotlin

```
fun main() {  
  
    val mapNum = mapOf("element" to 1, "element1" to 7)  
    println("mapa : $mapNum")  
}
```

- Accediendo a los elementos de un mapa

```
fun main() {  
  
    val mapNum = mapOf("element" to 1, "element1" to 7)  
    println("mapa : ${mapNum.keys}")  
}
```

```
asspath main.jar Matrukt  
mapa : [element, element1]  
>
```

```
fun main() {  
  
    val mapNum = mapOf("element" to 1, "element1" to 7)  
    println("mapa : ${mapNum.values}")  
}
```

```
asspath main.jar Matrukt  
mapa : [1, 7]  
>
```

- Modificando los elementos de un mapa

```
fun main() {  
  
    val mapNum = mutableMapOf("element" to 1, "element1" to 7)  
  
    mapNum["Element3"] = 78  
    println(mapNum)  
}
```

```
asspath main.jar Matrukt  
{element=1, element1=7, Element3=78}  
>
```

```
fun main() {  
    val mapNum = mutableMapOf("element" to 1, "element1" to 7)  
  
    mapNum["element"] = 78  
    println(mapNum)  
}
```

```
asspath main.jar MainKt  
{element=78, element1=7}  
>
```

- Recorriendo un mapa

```
fun main() {  
    val mapNum = mutableMapOf("element" to 1, "element1" to 7)  
  
    for ((clave, valor) in mapNum)  
        println("Para la clave $clave tenemos almacenado $valor")  
}
```

```
asspath main.jar MainKt  
Para la clave element tenemos almacenado 1  
Para la clave element1 tenemos almacenado 7  
>
```

Pares en Kotlin

- ¿Qué es un par?

Los data class Pair son una estructura que permite guardar dos valores.

- Creación de pares en Kotlin

```
fun main() {  
  
    var pair = Pair("Kotlin Pair", 2)  
}
```

- Accediendo a los elementos de un par

```
fun main() {  
    var pair = Pair("Kotlin Pair", 2)  
    println(pair.first)  
}
```

```
asspath main.jar MainKt  
Kotlin Pair  
>
```

- Modificando los elementos de un par
- Recorriendo un par
- Funciones útiles para trabajar con pares en Kotlin

7. Prácticas de estructuras de datos en Kotlin

a. Ejercicios prácticos para aplicar los conceptos aprendidos

b. Solución a los ejercicios prácticos

```
//Desarrollar un programa que permita ingresar un arreglo de 8 elementos enteros, e informe:
//El valor acumulado de todos los elementos.
//El valor acumulado de los elementos que sean mayores a 36.
//Cantidad de valores mayores a 50.
//Definir dos for, en el primero realizar la carga y en el segundo proceder a analizar cada elemento)

fun main(parametro: Array<String>) {
    val arreglo = IntArray(8)
    for(i in arreglo.indices) {
        print("Ingrese elemento:")
        arreglo[i] = readln().toInt()
    }
    var suma = 0
    var sumaMayor36 = 0
    var cantMayor50 = 0
    for(elemento in arreglo) {
        suma += elemento
        if (elemento > 36)
            sumaMayor36 += elemento
        if (elemento > 50)
            cantMayor50++
    }
    println("Valor acumulado del arreglo: $suma")
    println("Valor acumulado de los elementos mayores a 36: $sumaMayor36")
    println("Cantidad de elementos mayores a 50: $cantMayor50")
}
```

```
atn.jar Kotlin
main.kt:7:10: warning: parameter 'parametro' is never used
r used
fun main(parametro: Array<String>) {
    ^
Ingrese elemento:44
Ingrese elemento:50
Ingrese elemento:87
Ingrese elemento:6
Ingrese elemento:65
Ingrese elemento:77
Ingrese elemento:67
Ingrese elemento:3
Valor acumulado del arreglo: 390
Valor acumulado de los elementos mayores a 36: 390
Cantidad de elementos mayores a 50: 4
> 4
bash: 4: command not found
>
```

```
//Crear una lista inmutable de 100 elementos enteros con valores aleatorios comprendidos entre 0 y 20.
//Contar cuantos hay comprendidos entre 1 y 4, 5 y 8 y cuantos entre 10 y 13.
//Imprimir si el valor 20 está presente en la lista.

fun main(args: Array<String>) {
    val lista1 = List(100, { (Math.random() * 21).toInt() })
    println(lista1)
    var cant1 = 0
    var cant2 = 0
    var cant3 = 0
    lista1.forEach { when(it) {
        in 1..4 -> cant1++
        in 5..8 -> cant2++
        in 10..13 -> cant3++
    } }
    println("Cantidad de valores comprendidos entre 1..4: $cant1")
    println("Cantidad de valores comprendidos entre 5..8: $cant2")
    println("Cantidad de valores comprendidos entre 10..13: $cant3")
    if (lista1.contains(20))
        println("La lista contiene el 20")
    else
        println("La lista no contiene el 20")
}
```

```
atn.jar Kotlin
main.kt:1:10: warning: parameter 'args' is never used
d
fun main(args: Array<String>) {
    ^
[12, 15, 18, 8, 18, 20, 19, 1, 15, 12, 13, 16, 16, 3, 6, 13, 3, 13, 6, 3, 6, 4, 1, 18, 17, 20, 20, 1, 20, 5, 13, 2, 0, 20, 3, 12, 12, 14, 2, 20, 13, 0, 18, 2, 15, 8, 14, 6, 6, 5, 7, 16, 18, 4, 7, 9, 7, 1, 20, 19, 17, 3, 4, 11, 8, 17, 1, 9, 11, 18, 16, 11, 3, 2, 19, 7, 19, 18, 18, 14, 5, 5, 7, 5, 14, 2, 15, 15, 5, 10, 12, 4, 20, 6, 0, 3, 15, 17, 9]
Cantidad de valores comprendidos entre 1..4: 19
Cantidad de valores comprendidos entre 5..8: 21
Cantidad de valores comprendidos entre 10..13: 19
La lista contiene el 20
> []
```

- Confeccionar una agenda. Utilizar un MutableMap cuya clave sea de la clase Fecha:
data class Fecha(val dia: Int, val mes: Int, val año: Int)
Como valor en el mapa almacenar un String.
Implementar las siguientes funciones:
1) Carga de datos en la agenda.
2) Listado completo de la agenda.
3) Consulta de una fecha.


```

1 data class Fecha(val dia: Int, val mes: Int, val año: Int)
2
3 fun cargar(agenda: MutableMap<Fecha, String>) {
4     do {
5         println("Ingrese fecha")
6         print("Ingrese el día:")
7         val dia = readln().toInt()
8         print("Ingrese el mes:")
9         val mes = readln().toInt()
10        print("Ingrese el año:")
11        val año = readln().toInt()
12        print("Ingrese todas las actividades para ese día:")
13        val actividades = readln()
14        agenda[Fecha(dia, mes, año)] = actividades
15        print("Ingrese otra fecha (si/no):")
16        val opcion = readln()
17    } while (opcion == "si")
18 }
19
20 fun imprimir(agenda: MutableMap<Fecha, String>) {
21     for((fecha, actividad) in agenda) {
22         println("Fecha ${fecha.dia}/${fecha.mes}/${fecha.año}")
23         println("Actividades: $actividad")
24         println()
25     }
26 }
27
28 fun consultaFecha(agenda: MutableMap<Fecha, String>) {
29     println("Ingrese una fecha a consultar")
30     print("Ingrese el día:")
31     val dia = readln().toInt()
32     print("Ingrese el mes:")
33     val mes = readln().toInt()
34     print("Ingrese el año:")
35     val año = readln().toInt()
36     val fecha = Fecha(dia, mes, año)
37 }

```

```

fun main(args: Array<String>) {
    //
    Ingrese fecha
    Ingrese el día:20
    Ingrese el mes:09
    Ingrese el año:2022
    Ingrese todas las actividades para ese día:jugar
    Ingrese otra fecha (si/no):si
    Ingrese fecha
    Ingrese el día:5
    Ingrese el mes:11
    Ingrese el año:2003
    Ingrese todas las actividades para ese día:pelear
    Ingrese otra fecha (si/no):no
    Fecha 20/9/2022
    Actividades: jugar

    Fecha 5/11/2003
    Actividades: pelear

    Ingrese una fecha a consultar
    Ingrese el día:

```

```

9
10
11 fun consultaFecha(agenda: MutableMap<Fecha, String>) {
12     println("Ingrese una fecha a consultar")
13     print("Ingrese el día:")
14     val dia = readln().toInt()
15     print("Ingrese el mes:")
16     val mes = readln().toInt()
17     print("Ingrese el año:")
18     val año = readln().toInt()
19     val fecha = Fecha(dia, mes, año)
20     if (fecha in agenda)
21         println("Actividades: ${agenda[fecha]}")
22     else
23         println("No existen actividades registradas para dicha fecha")
24 }
25
26 fun main(args: Array<String>) {
27     val agenda: MutableMap<Fecha, String> = mutableMapOf()
28     cargar(agenda)
29     imprimir(agenda)
30     consultaFecha(agenda)
31 }

```