

Taller Estructuras de Datos en Kotlin

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

R// Las estructuras de datos son aquellas que nos permiten organizar la información de manera más fácil y eficiente para diseñar la solución correcta para un problema

* 1. **Ventajas de utilizar estructuras de datos en Kotlin**

R// Las ventajas de utilizar las estructuras de datos en kotlin son útiles porque nos permiten tener una batería de herramientas para solucionar un tipo de problema, también nos permite hacer que el Software este mas organizado.

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

R// La diferencia entre Kotlin y Java es que proporcionan estructuras de datos similares, pero Kotlin tiene una sintaxis más concisa y segura en cuanto a tipos, y algunas estructuras de datos más eficientes en términos de rendimiento.

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

R// Un arreglo es una estructura con valores de datos, que están almacenados de forma contigua en memoria. Todos los elementos son referenciados por un mismo nombre y tienen el mismo tipo de dato.

* 1. **Creación de arreglos en Kotlin**

R// Usar Constructor Array (), si deseas crear un array con un tamaño especifico y calcular todos sus elementos a partir de una función, entonces usa el constructor Array (size, init).

arrayOf (var arg elements: T): recibe un argumento variable con elementos de tipo de dato que se le asigne (T) y retorna el arreglo que los contiene.

arrayOfNulls(size:Int): crea un arreglo de tamaño size con elementos de tipo de dato que se le asigne (T) e inicializa los valores con null

emptyArray(): crear un arreglo vacío con el tipo de dato que se le asigne (T)

* 1. **Accediendo a los elementos de un arreglo**

La clase Array te provee operadores set() y get() para asignar y obtener valores en los índices de tus arreglos. Sin embargo, es preferible usar el operador de acceso indexado [ ] para referirte a los elementos.

* 1. **Modificando los elementos de un arreglo**

R// Se puede modificar tomando el arreglo y colocarle el índice o posición que quiera cambiarle

* 1. **Recorriendo un arreglo**

Una de las formas para recorrer arreglos es a través del bucle **For** a través del operador **in** junto a la propiedad de extensión índices. Esta contiene el rango válido de los índices del array.

* 1. **Funciones útiles para trabajar con arreglos en Kotlin**

R// Kotlin proporciona una serie de funciones útiles para trabajar con arreglos que facilitan su manipulación y procesamiento. Algunas de las funciones más comunes son:

arrayOf() - crea un arreglo de elementos dados.

size - devuelve el tamaño del arreglo.

get() - devuelve el elemento en el índice especificado.

set() - establece el elemento en el índice especificado.

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

Una lista es una secuencia ordenada de valores, donde el valor puede ocurrir más de una vez

* 1. **Creación de listas en Kotlin**

Se crea a variable n (Depende del tipo de dato que vaya a ser la lista) de tipo List<TipoDeVariable> y se le agrega la función llamada listOf() y pasa los elementos de la lista como argumentos separados por comas.

* 1. **Accediendo a los elementos de una lista**

La funcionalidad específica de las listas es que puedes acceder a cada elemento de ellas mediante su índice, que es un número entero que representa la posición.

* 1. **Modificando los elementos de una lista**

Se puede modificar con algunos métodos como el es add(). AddAll(), remove() y muchos mas, pero esto solo funciona con las listas mutables la cual deja modificar las listas

* 1. **Recorriendo una lista**

Dado que List es una colección ordenada, puede usar for basado en índice para acceder a los elementos por su índice en la lista. El bucle for también tiene otra variación llamada bucle forEach(), función que realiza la acción dada en cada elemento de la lista.

* 1. **Funciones útiles para trabajar con listas en Kotlin**

R// listOf: esta función se utiliza para crear una lista inmutable. Recibe una lista de elementos y devuelve una lista que no se puede modificar.

mutableListOf: esta función se utiliza para crear una lista mutable. Recibe una lista de elementos y devuelve una lista que se puede modificar.

add: este método se utiliza para agregar un elemento al final de la lista mutable.

remove: este método se utiliza para eliminar un elemento de la lista mutable.

get: este método se utiliza para obtener el elemento en una posición específica de la lista.

contains: este método se utiliza para verificar si la lista contiene un elemento específico.

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

R// Un conjunto es una colección de ciertos valores, sin ningún orden concreto ni valores repetidos.

* 1. **Creación de conjuntos en Kotlin**

R//Un conjunto (Set) es una colección sin orden de elementos únicos, esto es, no puede tener ningún duplicado. Set extiende de la interfaz Collection, por lo que es inmutable, y no agrega nuevos métodos. Para crear conjuntos inmutables utiliza:

setOf() crea un conjunto inmutable y devuelve un objeto de tipo Set.

* 1. **Accediendo a los elementos de un conjunto**

R// La funcionalidad específica de los conjuntos es que puedes acceder a cada elemento de ellas mediante su índice, que es un número entero que representa la posición.

* 1. **Modificando los elementos de un conjunto**

Al igual que List, Set tiene un método add(). Agrega al conjunto otro elemento con el método add(). Solo se necesita un parámetro para el elemento que se agrega. Los elementos de los conjuntos no tienen necesariamente un orden, por lo que no hay índice.

* 1. **Recorriendo un conjunto**

Usando forEach() función

En Kotlin, puede recorrer fácilmente un conjunto con la ayuda del forEach() función

puede usar un bucle for o foreach para recorrer Set.

* 1. **Funciones útiles para trabajar con conjuntos en Kotlin**

R// setOf: esta función se utiliza para crear un conjunto inmutable. Recibe una lista de elementos y devuelve un conjunto que no se puede modificar.

mutableSetOf: esta función se utiliza para crear un conjunto mutable. Recibe una lista de elementos y devuelve un conjunto que se puede modificar.

add: este método se utiliza para agregar un elemento al conjunto mutable.

remove: este método se utiliza para eliminar un elemento del conjunto mutable.

contains: este método se utiliza para verificar si el conjunto contiene un elemento específico.

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

R// Un Map está definido por un conjunto de claves, una colección de valores y un conjunto de pares clave-valor (también llamadas entradas), que son realmente los elementos de los que está compuesto. informática: representar diccionarios o propiedades, almacenar en memoria tablas de bases de datos, cachés, etc.

* 1. **Creación de mapas en Kotlin**

R// La interfaz que representa a los mapas en Kotlin es Map<K,V>. Donde los parámetros de tipo K y V representan a los tipos para claves (propiedad keys) y valores (propiedad values).

Los mapas mutables te otorgan el poder de usar comandos de operaciones sobre los elementos como agregar, actualizar y remover entradas. El diseño de esta figura la encuentras en la interfaz MutableMap<K,V>, la cual extiende de Map<K,V>.

* 1. **Accediendo a los elementos de un mapa**

R// Como las listas la clase Map dispone de una propiedad size que nos retorna la cantidad de elementos del mapa

* 1. **Modificando los elementos de un mapa**

R// Para modificar el valor de un elemento específico del mapa, simplemente se debe asignar un nuevo valor a ese índice.

* 1. **Recorriendo un mapa**

R// Los mapas mutables te otorgan el poder de usar comandos de operaciones sobre los elementos como agregar, actualizar y remover entradas. El diseño de esta figura la encuentras en la interfaz MutableMap<K,V>, la cual extiende de Map<K,V>.

* 1. **Funciones útiles para trabajar con mapas en Kotlin**

R// mapOf: esta función se utiliza para crear un mapa inmutable. Recibe una lista de pares clave-valor y devuelve un mapa que no se puede modificar.

mutableMapOf: esta función se utiliza para crear un mapa mutable. Recibe una lista de pares clave-valor y devuelve un mapa que se puede modificar.

put: este método se utiliza para agregar un elemento al mapa mutable.

get: este método se utiliza para obtener el valor asociado con una clave.

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

R// La operación módulo obtiene el resto de la división de un número entre otro (a veces llamado residuo o resto).Cuando un número entero se divide entre 2 y su residuo es 0, quiere decir que es divisible entre 2 y además es par.

* 1. **Creación de pares en Kotlin**

R// Los data class Pair en Kotlin, que simplemente 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 valores.

* 1. **Accediendo a los elementos de un par**

R// Para esto se emplean las palabras reservadas first y second respectivamente

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

R// Para modificar los elementos de un par, se puede crear un nuevo par utilizando la función copy() de la clase Pair, que devuelve un nuevo par con los mismos valores pero con los elementos actualizados.

* 1. **Recorriendo un par**

R // En Kotlin, se puede recorrer un par utilizando la función component1() y component2() que están definidas por defecto para las clases de datos (data classes).

* 1. **Funciones útiles para trabajar con pares en Kotlin**

R// Existen algunas funciones útiles para trabajar con pares, que pueden hacer que el código sea más legible y conciso como, por ejemplo:

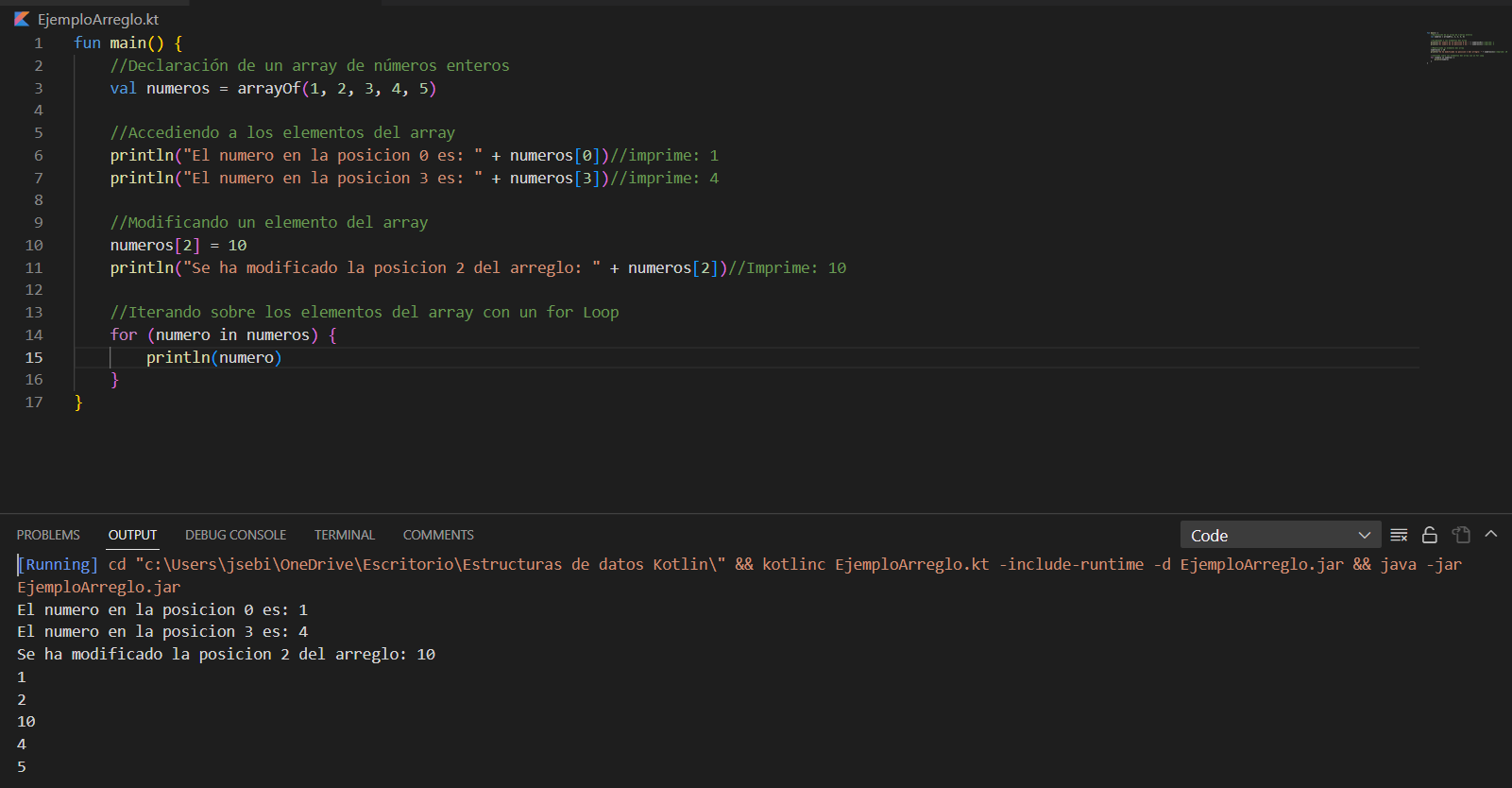
first y second: estas son propiedades de los pares que nos permiten acceder a su primer y segundo elemento, respectivamente.

to(): esta función nos permite crear un par a partir de dos valores.

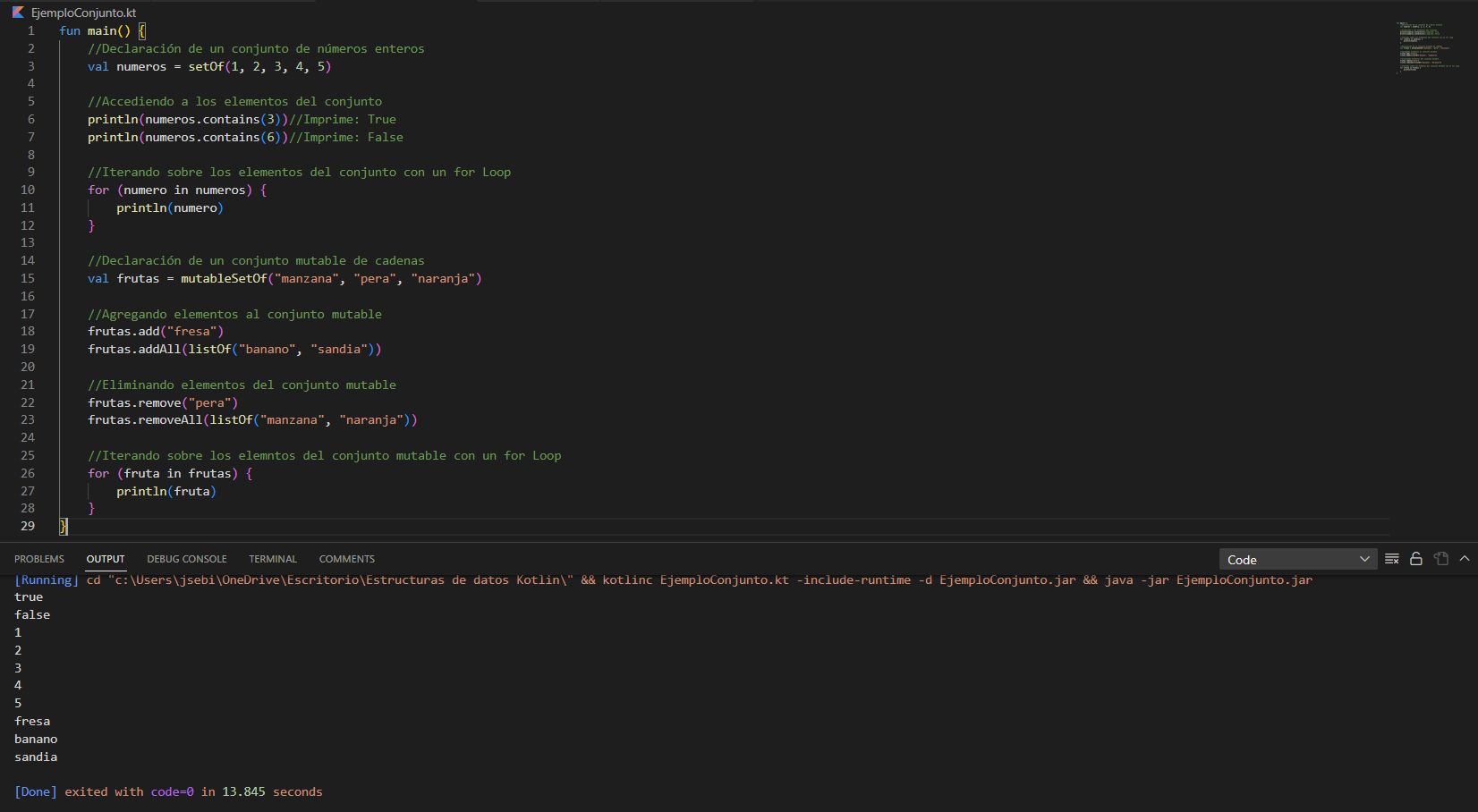
toList(): esta función nos permite convertir un par en una lista de dos elementos.

1. **Prácticas de estructuras de datos en Kotlin**
   1. **Ejercicios prácticos para aplicar los conceptos aprendidos**
   2. **Solución a los ejercicios**

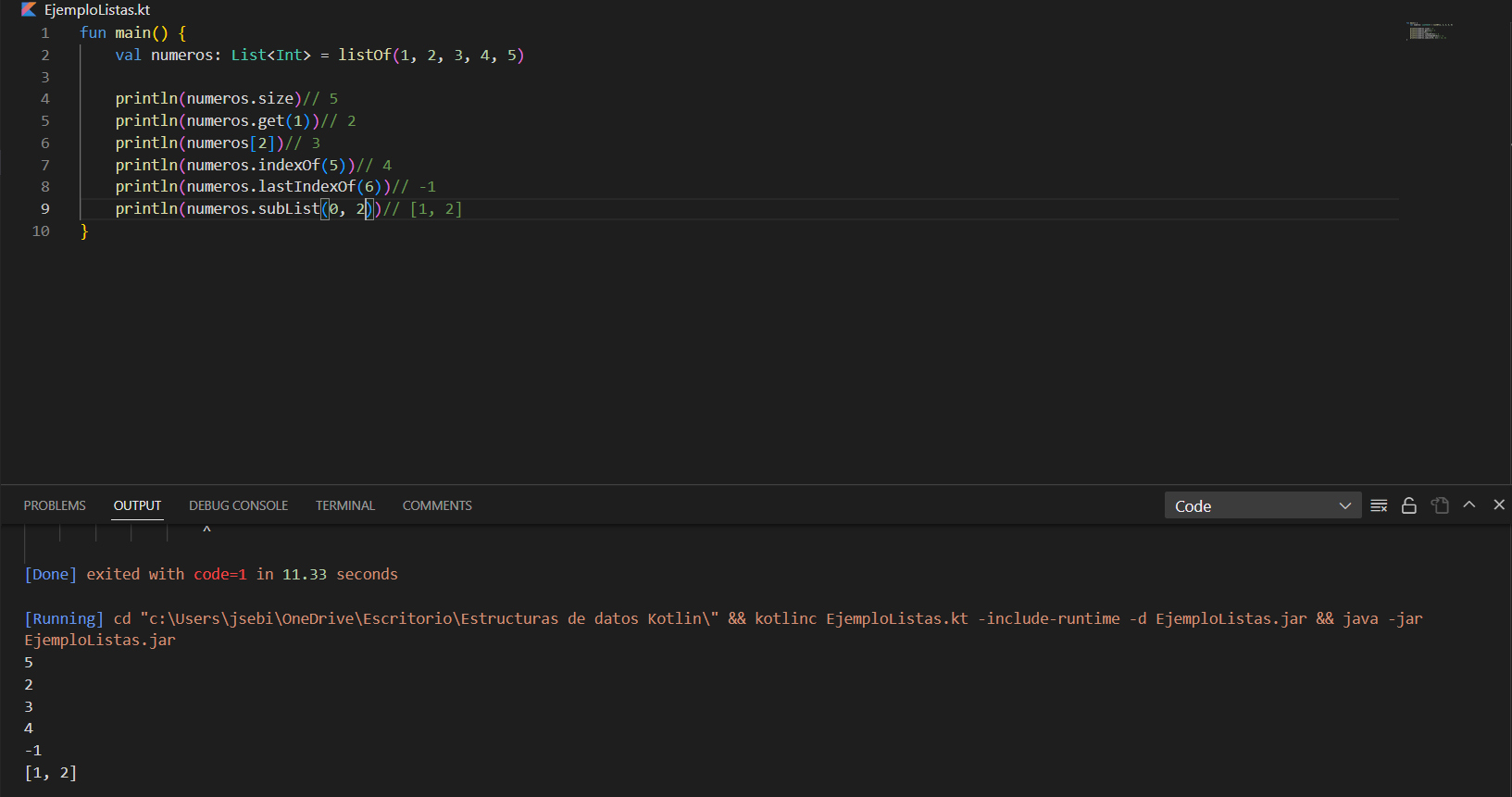
R// Arreglos:



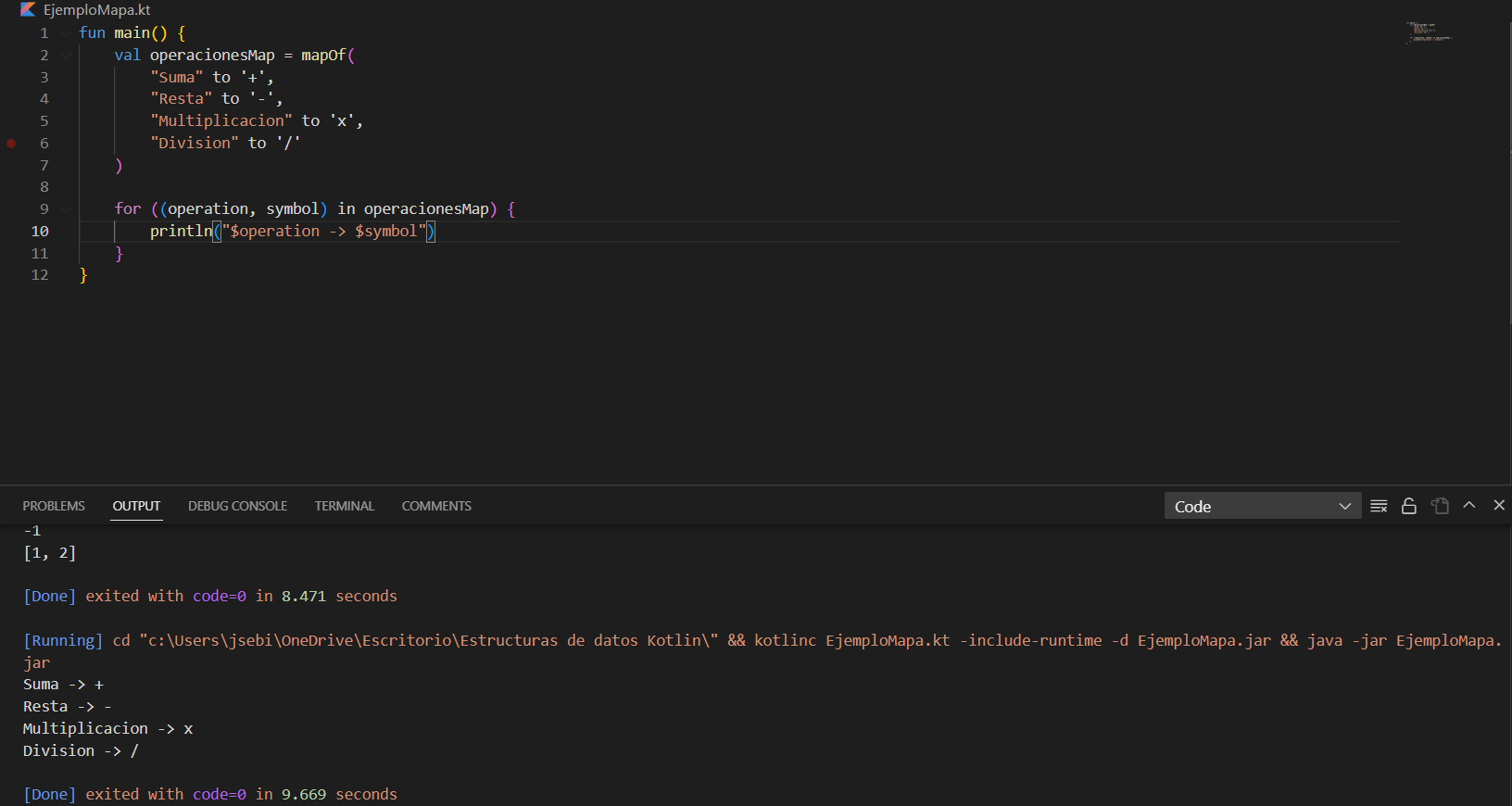
Conjuntos:



Listas:



Mapas:



Pares:



Documentación oficial de Kotlin: [**https://kotlinlang.org/docs/reference/**](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.