UTT
UNIVERSIDAD TECNOLÓGICA DE TIJUANA
GOBIERNO DE BAJA CALIFORNIA

TEMA:

Sentencias de iteración, Sentencias de Control, Funciones

PRESENTADO POR:

Hernández Miranda Rafael Francisco

GRUPO:

9B

MATERIA:

Desarrollo para Dispositivos Inteligentes

PROFESOR:

Ray Brunett Parra Galaviz

FECHA:

02/10/2024

Sentencias de Iteración:

Las sentencias de iteración son estructuras de control que permiten ejecutar repetidamente un bloque de código hasta que se cumpla una condición específica. Es útil cuando necesitas repetir tareas como recorrer una lista o contar elementos. Los bucles (for, while, do-while) son ejemplos de sentencias de iteración.for loop: Se utiliza para iterar sobre rangos, matrices, colecciones u otras estructuras de datos.

```
for (i in 1..5) {
    println(i)
}
```

También se puede usar con índices:

```
val array = arrayOf(10, 20, 30)
for (i in array.indices) {
    println("Index $i has value ${array[i]}")
}
```

- **while loop**: Ejecuta un bloque de código mientras una condición sea verdadera.

```
var x = 5
while (x > 0) {
    println(x)
    x--
}
```

- **do-while loop**: Similar a while, pero se asegura de que el bloque de código se ejecute al menos una vez, ya que la condición se evalúa después de ejecutar el bloque.

```
var x = 5

do {

    println(x)

    x--

} while (x > 0)
```

2. Sentencias de Control:

Las sentencias de control son estructuras que permiten modificar el flujo normal de ejecución de un programa en función de condiciones específicas. Estas sentencias deciden qué bloques de código ejecutar en función de una o varias condiciones. Los tipos más comunes son if-else, when, y try-catch.if-else: Evalúa una condición y ejecuta uno de los bloques de código, dependiendo del resultado (verdadero o falso).

```
Val a = 10
val b = 20
if (a > b) {
   println(“a is greater”)
} else {
   println(“b is greater”)
}
```

También puede ser utilizado como una expresión:

```
val max = if (a > b) a else b
```

- **when**: Es similar a una sentencia switch en otros lenguajes. Permite comprobar múltiples condiciones de una manera más elegante.

```
Val x = 3
when (x) {
   1 -> println(“x is 1”)
   2 -> println(“x is 2”)
   else -> println(“x is neither 1 nor 2”)
}
```

También puede devolver un valor:

```
val result = when (x) {
   in 1..10 -> “x is in the range 1 to 10”
   else -> “x is out of range”
}
```

- **try-catch**: Para el manejo de excepciones. Permite intentar ejecutar un bloque de código y capturar excepciones en caso de error.

```
Try {
    val data = arrayOf(1, 2, 3)
    println(data[5]) // Esto lanzará una excepción
} catch (e: Exception) {
    println("Exception caught: ${e.message}")
} finally {
    println("This will always execute")
}
```

3. Funciones:

Una función es un bloque de código que se puede reutilizar. Las funciones permiten organizar el código en tareas más pequeñas y definidas. Pueden recibir parámetros (entradas) y devolver un valor de salida. Las funciones promueven la reutilización y hacen que el código sea más modular y legible.

- **Declaración de una función**:

```
fun greet(name: String): String {
    return "Hello, $name"
}


println(greet("John"))
```

- **Funciones de una sola expresión**: Si el cuerpo de la función es simple, puedes definirla en una sola línea.

```
Fun square(x: Int) = x * 2
```

- **Funciones con valores predeterminados**: Se pueden definir parámetros con valores por defecto.

```
Fun greet(name: String = "Guest") {
    println("Hello, $name")
}


greet()        // Imprime "Hello, Guest"
greet("Jane")  // Imprime "Hello, Jane"
```

- **Funciones de orden superior**: Son funciones que aceptan otras funciones como parámetros o devuelven una función.

```
Fun operateOnNumbers(a: Int, b: Int, operation: (Int, Int) -> Int): Int {
    return operation(a, b)
}

val sum = operateOnNumbers(5, 10, { x, y -> x + y })
println(sum) // Imprime 15
```

References

Caules, C. Á. (2021, December 23). *Kotlin Ranges y sentencia de Control*. Arquitectura Java. https://www.arquitecturajava.com/kotlin-ranges-y-sentencia-de-control/

Cómo crear y usar funciones en Kotlin. (n.d.). Android Developers. Retrieved October 3, 2024, from https://developer.android.com/codelabs/basic-android-kotlin-compose-functions?hl=es-419

Estructuras de Programación en Kotlin - Ejemplos y Explicación. (n.d.). Codea.app. Retrieved October 3, 2024, from https://codea.app/blog/estructuras-de-programacion-en-kotlin