



# Kotlin

Rafael Vieira Coelho  
[rafaelvc2@gmail.com](mailto:rafaelvc2@gmail.com)

<https://kotlinlang.org>

# Laço de Repetição do-while

---

- Diferentemente de Java, não terminamos o while com ponto-e-vírgula (;)

```
do {  
    /*...*/  
} while (condition)
```

# Laço de Repetição for

- Estamos usando uma lista de Strings e o operador in para percorrer cada um.

```
val list = listOf("a", "b", "c")
for (s in list) {
    print(s)
}
```

*abc*

# Iteração em um Mapa

---

```
val map = mapOf(1 to "one",
                2 to "two",
                3 to "three")

for ((key, value) in map) {
    println("$key = $value")
}

1 = one
2 = two
3 = three
```

# Acessando os Índices da Lista

- Para acessar os índices de uma lista, usamos a função `withIndex()` que retorna uma lista de Pair (índice, elemento)

```
val list = listOf("a", "b", "c")
for ((index, element) in list.withIndex()) {
    println("$index: $element")
}
```

0: a  
1: b  
2: c

# Iterando um Intervalo (range)

---

```
including upper bound
for (i in 1..9) {
    print(i)           123456789
}

excluding upper bound
for (i in 1 until 9) {
    print(i)           12345678
}
```

# Iterando Passo a Passo

```
for (i in 9 downTo 1 step 2) {  
    print(i)  
}
```

97531

# Pergunta: O que será mostrado na tela?

---

RESPOSTA: "bcd"

```
for (ch in "abc") {  
    print(ch + 1)  
}
```

# Pergunta: Qual seria a melhor forma de reescrever este código em Kotlin?

```
for (char c = '0'; c < '9'; c++) {  
    System.out.print(c);  
}
```

012345678

1. `for (c in '0' to '9') { print(c) }`
2. `for (c in '0'..'9') { print(c) }`
3. `for (c in '0' until '9') { print(c) }`

# while x do-while

---

```
fun eatACake() = println("Eat a Cake")
fun bakeACake() = println("Bake a Cake")

fun main(args: Array<String>) {
    var cakesEaten = 0
    var cakesBaked = 0

    while (cakesEaten < 5) {                                // 1
        eatACake()
        cakesEaten ++
    }

    do {                                                 // 2
        bakeACake()
        cakesBaked++
    } while (cakesBaked < cakesEaten)

}
```

# Iteradores

1. Define um Iterador

2. Retorna o iterador com os métodos `next(): Animal` e `hasNext(): Boolean`

3. Percorre a lista de objetos `Animal`

```
class Animal(val name: String)

class Zoo(val animals: List<Animal>) {

    operator fun iterator(): Iterator<Animal> {           // 1
        return animals.iterator()                           // 2
    }
}

fun main() {

    val zoo = Zoo(listOf(Animal("zebra"), Animal("lion")))

    for (animal in zoo) {                                // 3
        println("Watch out, it's a ${animal.name}")
    }
}
```



[https://github.com/rafael-vieira-coelho/kotlin/tree/Aula-06--  
Repetição](https://github.com/rafael-vieira-coelho/kotlin/tree/Aula-06--Repetição)

---

[rafaelvc2@gmail.com](mailto:rafaelvc2@gmail.com)