



Kotlin

Professor Emerson Alencar

emerson@imd.ufrn.br

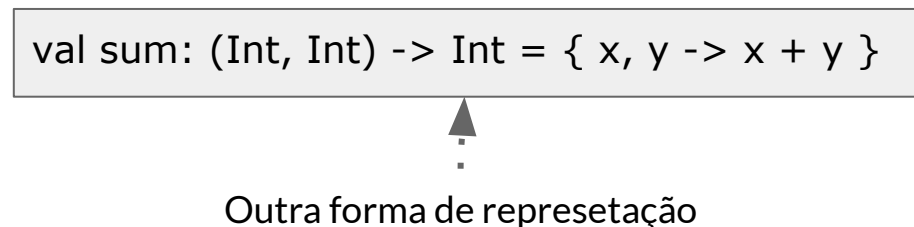
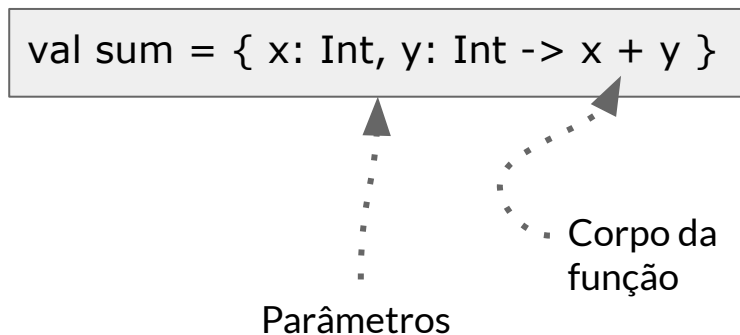
Passando função como parâmetro

- ▶ Kotlin é uma linguagem de paradigma funcional que permite que funções recebam outras funções como parâmetro, além de permitir que uma função retorne outra função, na documentação oficial isso é chamado de Higher-Order Functions

```
class Operacoes{  
    fun somar(a: Int, b: Int): Int{  
        return a + b  
    }  
}  
  
fun somar(a: Int, b: Int): Int{  
    return a + b  
}  
  
fun calc(a: Int, b: Int, funcao:(Int, Int) -> Int): Int{  
    return funcao(a,b)  
}  
  
fun main(args: Array<String>) {  
    println(calc(2,3,Operacoes()::somar))  
    println(calc(5,3,::somar))  
}
```

Expressões Lambda

- ▶ São funções tratadas como dados
 - Podem ser passadas como parâmetro
 - Podem ser retornadas
- ▶ São a base de funcionamento do paradigma de programação funcional
- ▶ Expressões lambda e funções anônimas são funções que não são declaradas, mas transmitidas imediatamente como uma expressão.
- ▶ Sintaxe de uma expressão lambda:



Expressões Lambda

```
fun main(args: Array<String>) {  
    var soma = {x: Int, y: Int -> x + y}  
    println(soma(4,6))  
}
```

```
fun main(args: Array<String>) {  
  
    val myList = listOf(1,3,6,7)  
  
    val greaterThan5 = myList.count({ x -> x > 5 })  
    println(greaterThan5) // -> Imprime 2  
}
```

Expressões Lambda na API do Kotlin

- ▶ A API do Kotlin faz uso intenso de expressões lambda

```
var s1 = "abbcbbaababbacca"  
var s2 = s1.filter { c -> c != 'a' }
```

.....
"bbcbbbbcc"

```
var i = 30  
var s = i.apply { toString() }
```

.....
"30"


```
(1..10)  
    .filter { v -> v % 2 == 0 }  
    .map { v -> v * 2 }  
    .forEach { v -> print("$v ") }
```

.....
"4 8 12 16 20"

O parâmetro *it*


- ▶ Se existe apenas 1 parâmetro na definição da expressão lambda, ele pode ser omitido e referenciado como *it*

```
var s = "ADSJHVKALSHAKLJWJKHLA"  
var c = s.count { c -> c == 'A' }
```



```
var c = s.count { it == 'A' }
```

```
var array = arrayOf(1, 2, 3, 4)  
array.forEach { e -> println(e) }
```



```
array.forEach { println(it) }
```

Função Inline

- ▶ Quando passamos um função lambda como parâmetro para outra função, é criado um objeto que implementa aquela função, e este objeto é passado como parâmetro. Se existirem muitas funções lambdas no sistema, a quantidade de objetos em memória pode crescer e afetar a performance.
- ▶ As funções inline indica ao compilador que o corpo desta função deve ser copiado para dentro do código que está fazendo a chamada. Apesar de deixar o código-fonte final maior, ganha em termos de performance.

Função Inline

```
fun main(args: Array<String>) {  
    var a = 2  
    println(someMethod(a, {println("Just some dummy function")}))  
}  
  
fun someMethod(a: Int, func: () -> Unit):Int {  
    func()  
    return 2*a  
}
```

```
inline fun someMethod(a: Int, func: () -> Unit):Int {  
    func()  
    return 2*a  
}
```


Obrigado!
Dúvidas?

Professor Emerson Alencar
emerson@imd.ufrn.br