## Funções - [Kotlin]

Hybrid Mobile App Development

1. Funções de Membro (Member Functions):

```
//São funções definidas dentro de uma classe e podem acessar as propriedades e métodos dessa classe. Elas são chamadas em instâncias da classe.

class Pessoa(val nome: String) {
    fun saudacao() {
        println("Olá, meu nome é $nome.")
    }
}

fun main() {
    val pessoa = Pessoa("Alice")
    pessoa.saudacao()
}
```

## 2. Funções de Extensão (Extension Functions):

```
//Permitem adicionar novos métodos a classes existentes sem precisar modificar o código fonte original da classe. Elas são especialmente úteis para estender classes de bibliotecas ou APIs.
fun String.inverter(): String {
    return this.reversed()
}
fun main() {
    val palavra = "Kotlin"
    val palavraInvertida = palavra.inverter()
    println("Palavra original: $palavra invertida: $palavraInvertida")
}
```

3. Funções de Nível Superior (Top-Level Functions):

```
//São funções definidas fora de qualquer classe. Elas podem ser acessadas diretamente do escopo do arquivo ou módulo em que estão definidas.
fun somar(a: Int, b: Int): Int {
    return a + b
}

fun main() {
    val resultado = somar(5, 3)
    println("Resultado da soma: $resultado")
}
```

4. Funções Anônimas (Anonymous Functions):

```
//São funções que não possuem um nome associado. Geralmente são usadas como expressões lambda ou passadas como argumentos para outras funções.
fun main() {
   val numeros = listOf(1, 2, 3, 4, 5)
   val numerosDobrados = numeros.map(fun(numero): Int {
        return numero * 2
   })
   println("Números dobrados: $numerosDobrados")
}
```

5. Funções de Ordem Superior (Higher-Order Functions):

```
//São funções que recebem outras funções como argumentos e/ou retornam funções como resultados. Elas são fundamentais para a programação funcional em Kotlin.
fun operacaoMatematica(a: Int, b: Int, operacao: (Int, Int) -> Int): Int {
    return operacao(a, b)
}

fun main() {
    val soma = operacaoMatematica(5, 3) { x, y -> x + y }
    val multiplicacao = operacaoMatematica(5, 3) { x, y -> x * y }
    println("Soma: $soma, Multiplicação: $multiplicacao")
}
```

6. Funções Lambda (Lambda Functions):

```
// São funções anônimas que podem ser usadas como argumentos de outras funções. Elas são frequentemente usadas com funções de ordem superior.
fun main() {
   val numeros = listOf(1, 2, 3, 4, 5)
   val numerosPares = numeros.filter { it % 2 == 0 }
   println("Números pares: $numerosPares")
}
```

7. Funções Inline:

```
//São funções que são substituídas pelo seu código de corpo no local onde são chamadas, otimizando o desempenho ao evitar a alocação de objetos
inline fun calcularDobro(numero: Int): Int {
    return numero * 2
}

fun main() {
    val resultado = calcularDobro(5)
    println("Resultado: $resultado")
}
```

8. Funções com Parâmetros Default (Default Parameter Functions):

```
//Permitem definir valores padrão para os parâmetros da função, tornando alguns parâmetros opcionais ao serem chamados.
fun saudacao(nome: String = "Visitante") {
    println("Olá, $nome!")
}

fun main() {
    saudacao()
    saudacao("Alice")
}
```

9. Funções com Parâmetros Nomeados (Named Parameter Functions):

```
//Permitem passar argumentos para a função especificando seus nomes, o que torna o código mais legível e evita confusões na ordem dos argumentos.
fun calcularTotal(preco: Double, quantidade: Int): Double {
    return preco * quantidade
}
fun main() {
    val total = calcularTotal(preco = 10.0, quantidade = 3)
    println("Total: $total")
}
```

+ •

10. Funções de Recursão (Recursive Functions):

```
//São funções que chamam a si mesmas para resolver problemas divididos em subproblemas menores.
fun fatorial(n: Int): Int {
    if (n <= 1) {
        return 1
    }
    return n * fatorial(n - 1)
}

fun main() {
    val numero = 5
    val resultado = fatorial(numero)
    println("Fatorial de $numero é $resultado")
}</pre>
```

## 11. Funções com Retorno de Múltiplos Valores:

```
//Kotlin permite retornar múltiplos valores de uma função por meio da declaração de um tipo especial chamado Pair ou Triple, ou usando estruturas de dados personalizadas.
fun obterNomeEIdade(): Pair<String, Int> {
    return Pair("Alice", 25)
}

fun main() {
    val (nome, idade) = obterNomeEIdade()
    println("Nome: $nome, Idade: $idade")
}
```

12. Funções de Manipulação de Coleções:

```
//Kotlin fornece várias funções úteis para trabalhar com coleções, como map, filter, reduce, entre outras, que simplificam a manipulação de elementos em listas, mapas, e outros tipos de coleções.

fun main() {
    val numeros = listOf(1, 2, 3, 4, 5)
    val soma = numeros.reduce { acc, numero -> acc + numero }
    val numerosDobrados = numeros.map { it * 2 }
    println("Soma: $soma, Números dobrados: $numerosDobrados")
}
```

## Dúvidas

- Divirta-se!
- Profº Vinny Albuquerque
- profvinny.albuquerque@fiap.com.br
- © @mvalbuquerque