MiniCurso - Kotlin

Breno Klyver Olegario de Moura

Ciência da Computação – Universidade Federal Rural do Semi Árido (UFERSA)
Rio Grande do Norte – RN – Brazil

breno.moura@alunos.ufersa.edu.br

Entrada e Saída

`readln()` e `readlnOrNull()`

- `readln()`: Lê uma linha de entrada do usuário e a retorna como uma **String**. Caso o valor fornecido seja inválido ou nulo, ele lança uma exceção.

- `readlnOrNull()`: Semelhante ao `readln()`, mas se a entrada for nula, ele retorna null ao invés de lançar uma exceção.

Básico:

- Crie um programa que solicite ao usuário o nome e a idade, utilizando a função readln() para capturar a entrada como texto. O programa deve exibir uma mensagem personalizada de saudação utilizando println, mostrando os dados informados.

```
Unset
fun main() {
	print("Digite seu nome: ")
	val nome = readln()
```

```
        print("Digite sua idade: ")
        val idade = readln().toInt()

        println("Olá, $nome! Você tem $idade anos.")
}
```

- Crie um programa que solicite ao usuário o nome da cidade onde ele mora e o número da casa. Depois, exiba uma frase do tipo: "Você mora na cidade X, no número 777".

```
Unset
fun main() {
        print("Digite sua cidade: ")
        val cidade = readlnOrNull()

        print("Digite o número da sua casa: ")
        val numero = readlnOrNull()?.toIntOrNull()

        // O operador ? evita NullPointerException
        // Verificar automaticamente se a variável é nula antes de acessar algo
dela.

        // Só acessará o length se cidade não for nula
        println(cidade?.length)

        println("Você mora na cidade $cidade, no número $numero")
}
```

Completo:

- Reescreva o programa anterior utilizando boas práticas e recursos mais modernos da linguagem Kotlin. Crie uma função chamada prompt para simplificar a solicitação de entrada do usuário, e use readlnOrNull() com operador de coalescência nula (?:) para tratar possíveis valores inválidos ou nulos. Ao final, o programa deve exibir uma mensagem de boas-vindas formatada com os dados do usuário.

```
Unset
fun main() {
        val nome = prompt("Digite seu nome")
        val idade = prompt("Digite sua idade").toIntOrNull() ?: 0
```

```
        println("Seja bem-vindo(a), $nome! Sua idade é $idade anos.")
}

fun prompt(mensagem: String): String {
        print("$mensagem: ")
        return readlnOrNull().orEmpty()
}
```

Tipos e Operadores

Básico:

- Crie um programa que leia dois números do tipo Int, e exiba na tela o resultado das seguintes operações:
    - Soma
    - Subtração
    - Multiplicação
    - Divisão
    - Módulo (resto da divisão)

Unset

```
fun main() {
        print("Digite o primeiro número: ")
        val num1 = readlnOrNull()?.toIntOrNull() ?: 0

        print("Digite o segundo número: ")
        val num2 = readlnOrNull()?.toIntOrNull() ?: 0

        println("Soma: ${num1 + num2}")
        println("Subtração: ${num1 - num2}")
        println("Multiplicação: ${num1 * num2}")
        println("Divisão: ${num1 / num2}")
        println("Módulo: ${num1 % num2}")
}
```

- Faça um programa que leia um número inteiro e mostre:
    - O valor após o incremento (++),
    - O valor após o decremento (--),
    - O valor dobrado usando *=,
    - O valor dividido pela metade usando /=.

```
Unset
fun main() {
	print("Digite um número inteiro: ")
	var numero = readlnOrNull()?.toIntOrNull() ?: 0

	println("Valor original: $numero")

	numero++
	println("Após incremento (++): $numero")

	numero--
	println("Após decremento (--): $numero")

	numero *= 2
	println("Dobrando o valor (*=): $numero")

	numero /= 2
	println("Dividindo pela metade (/=): $numero")

	// A posição dos operadores de incremento e decremento é importante

	println("Valor Antes dos Testes: $numero")

	println("Valor Teste1: ${numero++}")

	println("Valor Pós-Teste: $numero")

	println("Valor Teste2: ${++numero}")
}
```

Completo:

- Crie um programa que:
    - Leia o nome (String), a idade (Int), a altura (Double) e se está matriculado (Boolean).
    - Mostre o tipo de cada dado usando o operador **is**.
    - Compare a idade com 18 usando operadores relacionais (<, >, ==, !=).
    - Mostre o resultado das comparações.

Unset

```
fun main() {
       print("Digite seu nome: ")
       val nome = readlnOrNull() ?: "Desconhecido"

       print("Digite sua idade: ")
       val idade = readlnOrNull()?.toIntOrNull() ?: 0

       print("Digite sua altura: ")
       val altura = readlnOrNull()?.toDoubleOrNull() ?: 0.0

       println("Altura: $altura") // Padrão é número ponto flutuante com "." e
não com ","

       print("Está matriculado? (true/false): ")
       val matriculado = readlnOrNull()?.lowercase()?.toBooleanStrictOrNull() ?:
false

       println("\nTipos:")
       println("Nome é String? ${nome is String}")
       println("Idade é Int? ${idade is Int}")
       println("Altura é Double? ${altura is Double}")
       println("Matriculado é Boolean? ${matriculado is Boolean}")

       println("\nComparações com idade:")
       println("Idade > 18: ${idade > 18}")
       println("Idade < 18: ${idade < 18}")
       println("Idade == 18: ${idade == 18}")
       println("Idade != 18: ${idade != 18}")
}
```

- Crie um programa que:
    - Leia um número e verifique se ele está no intervalo de 1 a 10 usando in.
    - Crie uma variável nula e use o operador de chamada segura (?.) para acessar sua propriedade.
    - Use o operador Elvis (?:) para fornecer um valor padrão caso a variável seja nula.
    - Combine expressões booleanas usando operadores lógicos (&&, ||, !).

Unset

```
fun main() {
       print("Digite um número entre 1 e 10: ")
```

```
        val numero = readlnOrNull()?.toIntOrNull() ?: 0

        // O operador in verifica se o número está dentro do intervalo
        // E o operador de intervalo (..) cria um intervalo de números
        println("Número está entre 1 e 10? ${numero in 1..10}")

        val texto: String? = null // Pode ser nulo ou não

        // O operador de chamada segura (?.) verifica se algo é nulo antes de
acessar sua propriedade
        println("Tamanho do texto nulo (com ?.): ${texto?.length}")

        // O operador Elvis (?:) fornece um valor padrão caso o texto seja nulo
        println("Texto com valor padrão (Elvis ?:): ${texto ?: "Texto padrão"}")

        val condicao1 = numero > 5
        val condicao2 = numero < 8

        println("\ncondicao1 (numero > 5): $condicao1")
        println("condicao2 (numero < 8): $condicao2")
        println("condicao1 && condicao2: ${condicao1 && condicao2}")
        println("condicao1 || condicao2: ${condicao1 || condicao2}")
        println("!condicao1: ${!condicao1}")
}
```

Estrutura Sequencial

Um programa com **estrutura sequencial** é aquele em que as instruções são executadas **uma após a outra**, na ordem em que foram escritas, do início ao fim, sem desvios ou repetições automáticas.

Básicos:

- Crie um programa simples que declare variáveis com informações básicas (nome e idade), e as exiba na tela usando comandos de saída. O programa deve seguir um fluxo sequencial, onde cada linha é executada diretamente, sem desvios ou repetições.

Unset

```
fun main() {
```

```
        val nome = "Lucas"
        val idade = 20

        println("Nome: $nome")
        println("Nome: $idade")
        println("Cadastro Completo")
}
```

- Crie um programa que calcule o valor total de uma compra simples, pedindo a quantidade e o preço unitário, e exibindo o total.

```
Unset
fun main() {
        print("Quantidade: ")
        val qtd = readlnOrNull()?.toIntOrNull() ?: 0

        print("Preço unitário: ")
        val preco = readlnOrNull()?.toDoubleOrNull() ?: 0.0

        val total = qtd * preco

        println("Total a pagar: R$ %.2f".format(total))
}
```

Completo:

- Desenvolva um programa em Kotlin com foco em estrutura sequencial moderna e interativa. O programa deve:
    - Exibir uma saudação inicial ao usuário;
    - Coletar o nome e a idade via entrada de dados (readlnOrNull);
    - Armazenar os dados em uma estrutura (mapOf);
    - Exibir os dados formatados;
    - Realizar um cálculo simples (idade em meses) e exibir o resultado;
    - Finalizar com uma mensagem de agradecimento.

```
Unset
fun main() {
        println("Bem-vindo ao sistema de cadastro!")
```

```
        // Coletando dados do usuário
        print("Por favor, insira seu nome: ")
        val nome = readlnOrNull() ?: "Desconhecido" // Lê o nome ou atribui
"Desconhecido" se nulo

        print("Agora, insira sua idade: ")
        val idade = readlnOrNull()?.toIntOrNull() ?: 0  // Tenta converter para
inteiro

        // Criando o mapa de dados do usuário
        val usuario = mapOf("nome" to nome, "idade" to idade) // O tipo desse
mapa é mapOf<String, Any>

        // Exibindo os dados do usuário
        println("\nCadastro finalizado com sucesso!\n")
        println("Nome: ${usuario["nome"]}")
        println("Idade: ${usuario["idade"]} anos")

        // Adicionando algo mais interativo e divertido
        println("\nAgora, vamos calcular sua idade em meses! 🧮")
        val idadeEmMeses = idade * 12
        println("Você tem aproximadamente $idadeEmMeses meses de vida!")

        println("\nObrigado por usar o sistema. Até logo, $nome!")
}
```

- Faça um programa que receba nome, idade, salário e cargo. Armazene em objeto Funcionario. Calcule o salário anual e exiba tudo formatado com texto personalizado.

- Um **objeto** é uma **estrutura que agrupa dados e comportamentos relacionados** em um único lugar. Ele é uma **instância de uma classe** (ou seja, um "exemplar real" de um modelo) e é usado para **representar coisas do mundo real ou conceitos** na programação.

- Um **objeto** tem:
    - **Atributos (ou propriedades):** dados que ele carrega.
    - **Métodos (ou funções):** ações que ele pode realizar.

Unset

```
data class Funcionario(
        val nome: String,
```

```
        val idade: Int,
        val salarioMensal: Double,
        val cargo: String,
        val salarioAnual: Double = salarioMensal * 12
) {
        constructor (nome: String, idade: Int, salarioMensal: Double, cargo:
String) : this(
        nome,
        idade,
        salarioMensal,
        cargo,
        salarioMensal * 12
        )
}

fun main() {
        println("Bem-vindo ao sistema de cadastro!")

        println("Por favor, insira os seguintes dados:")

        print("Nome: ")
        val nome = readlnOrNull() ?: ""

        print("Idade: ")
        val idade = readlnOrNull()?.toIntOrNull() ?: 0

        print("Salário Mensal: ")
        val salarioMensal = readlnOrNull()?.toDoubleOrNull() ?: 0.0

        print("Cargo: ")
        val cargo = readlnOrNull() ?: ""

        val dados = Funcionario(nome, idade, salarioMensal, cargo)

        // Exibe os dados formatados sem usar estrutura de repetição
        println("Dados cadastrados:")
        println("Nome: ${dados.nome}")
        println("Idade: ${dados.idade} anos")
        println("Cargo: ${dados.cargo}")
        println("Salário Mensal: ${dados.salarioMensal}")
        println("Salário Anual: ${dados.salarioAnual}")
}
```

Estruturas de Repetição

Básico:

- Crie um programa em Kotlin que receba nome, idade e período de um aluno. Armazene os dados em uma Lista de alunos e exiba as informações formatadas.

```
Unset
fun main() {
	// Programa que recebe nome, idade e período. E adiciona em uma lista.
	val alunos = mutableListOf<Aluno>()

	print("Digite seu nome: ")
	val nome = readlnOrNull() ?: "Desconhecido"

	print("Digite sua idade: ")
	val idade = readlnOrNull()?.toIntOrNull() ?: 0

	print("Digite seu período: ")
	val periodo = readlnOrNull()?.toIntOrNull() ?: 0

	// Adiciona o aluno à lista
	alunos.add(Aluno(nome, idade, periodo.toString()))

	// Exibe os dados formatados
	println("Dados cadastrados:")
	println("Nome: ${alunos[0].nome}")
	println("Idade: ${alunos[0].idade} anos")
	println("Período: ${alunos[0].periodo}")
}
```

- Agora, vamos adicionar estrutura de repetições para nos auxiliar a adicionar mais alunos.

```
Unset
fun main() {
	// Programa que recebe nome, idade e período. E adiciona em uma lista.
	val alunos = mutableListOf<Aluno>()

	print("Quantos alunos deseja cadastrar? ")
	val quantidade = readlnOrNull()?.toIntOrNull() ?: 0

	for (i in 0 until quantidade){
	print("Digite seu nome: ")
```

```
        val nome = readlnOrNull() ?: "Desconhecido"

        print("Digite sua idade: ")
        val idade = readlnOrNull()?.toIntOrNull() ?: 0

        print("Digite seu período: ")
        val periodo = readlnOrNull() ?: "Desconhecido"

        println()

        // Adiciona o aluno à lista
        alunos.add(Aluno(nome, idade, periodo))
        }

        // Exibe os dados formatados
        println("Dados cadastrados:\n")
        for (aluno in alunos) {
                println("Nome: ${aluno.nome}")
                println("Idade: ${aluno.idade} anos")
                println("Período: ${aluno.periodo}")
                println()
        }
}
```

Completo:

- Crie um programa que:
    - Use um laço for para exibir todos os números de 1 a 10 (inclusive), e faça a soma desses números.
    - Use um laço while para contar de 10 até 1, e exibir os números na tela.
    - Utilize o do..while para mostrar os números de 2 em 2, de 0 até 10.
    - Por fim, use o operador step para exibir todos os múltiplos de 3 entre 1 e 30.

Unset

```
fun main() {
        // Laço for com range de 1 a 10 e soma
        var soma = 0
        for (i in 1..10) {
                soma += i
        println("Número: $i")
        }
        println("Soma de 1 a 10: $soma")
```

```
	// Laço while de 10 a 1
	var contador = 10
	while (contador >= 1) {
		println("Contando de 10 até 1: $contador")
		contador--
	}

	// Laço do..while de 0 até 10, pulando de 2 em 2
	var i = 0
	do {
		println("Número de 2 em 2 (do..while): $i")
		i += 2
	} while (i <= 10)

	// Usando step para múltiplos de 3
	println("\nMúltiplos de 3 entre 1 e 30:")
	for (l in 1..30 step 3) {
		println(l)
	}
}
```

- Crie um programa que:
  - Use forEach para percorrer uma lista de números e exibir cada número multiplicado por 2.
  - Use repeat() para imprimir uma mensagem de saudação 5 vezes.
  - Utilize fold() para somar todos os valores de uma lista de números e strings e mostrar o resultado final.
  - Use forEachIndexed para exibir os índices e valores de uma lista.

Unset

```
fun main() {
	// Lista de números
	val numeros = listOf(1, 2, 3, 4, 5)

	// forEach - Percorrer a lista e multiplicar cada número por 2
	println("Multiplicando cada número por 2:")
	numeros.forEach { numero ->
	println("$numero multiplicado por 2: ${numero * 2}")
	}

	// repeat - Imprimir saudação 5 vezes
	println("\nSaudação usando repeat():")
```

```
        repeat(5) {
        println("Olá, aluno: $it")
        }

        // fold - Somar todos os valores da lista
        val soma = numeros.fold(0) { acc, num -> acc + num }
        println("\nSoma dos números usando fold(): $soma")

        /// Exemplo de fold com strings
        val words = listOf("Kotlin", "é", "uma", "linguagem", "moderna")

        val sentence = words.fold("") { acc, word ->
        "$acc $word"  // Operação de concatenação
        }

        println(sentence)

        // forEachIndexed - Exibir índice e valor de cada elemento
        println("\nÍndices e valores com forEachIndexed():")
        numeros.forEachIndexed { index, value ->
        println("Índice $index tem o valor $value")
        }
}
```

Desvio Condicional

Básico:

- Crie um programa que leia a idade de uma pessoa e determine se ela é maior ou menor de idade. Utilize a estrutura condicional if/else para realizar a verificação e exibir a mensagem adequada.

Unset

```
fun main() {
        print("Informe sua idade: ")
        val idade = readlnOrNull()?.toIntOrNull() ?: 0

        if (idade >= 18) {
        println("Você é maior de idade.")
        } else {
```

```
        println("Você é menor de idade.")
        }
}
```

- Crie um programa que leia um número inteiro do usuário e verifique se ele é par ou ímpar. Use a estrutura if/else e o operador % (módulo) para realizar a verificação.

```
Unset
fun main() {
        print("Digite um número inteiro: ")
        val numero = readlnOrNull()?.toIntOrNull() ?: 0

        if (numero % 2 == 0) {
        println("O número $numero é par.")
        } else {
        println("O número $numero é ímpar.")
        }
}
```

Completo:

- Crie um programa que leia a nota de um aluno (entre 0 e 10) e classifique seu desempenho com base nos seguintes critérios:
    - Excelente: nota ≥ 9
    - Bom: nota ≥ 7
    - Regular: nota ≥ 5
    - Reprovado: nota < 5
- Use a estrutura when para tornar o código mais claro e organizado.

```
Unset
fun main() {
        print("Digite a nota do aluno (0 a 10): ")
        val nota = readln().toDouble()

        val resultado = when {
        nota >= 9 -> "Excelente"
        nota >= 7 -> "Bom"
        nota >= 5 -> "Regular"
        else -> "Reprovado"
        }
```

```
        println("Desempenho: $resultado")
}
```

- Crie um programa que leia a idade de uma pessoa e identifique sua faixa etária:
    - Criança: até 12 anos
    - Adolescente: de 13 a 17 anos
    - Adulto: de 18 a 59 anos
    - Idoso: 60 anos ou mais
- Use a estrutura when com ranges (in ..) para facilitar a verificação de faixas.

Unset
```
fun main() {
        print("Digite a sua idade: ")
        val idade = readlnOrNull()?.toIntOrNull() ?: 0


        val faixa = when (idade) {
        in 0..12 -> "Criança"
        in 13..17 -> "Adolescente"
        in 18..59 -> "Adulto"
        else -> "Idoso"
        }

        println("Faixa etária: $faixa")
}
```

Funções

O que são funções?

Funções são **blocos de código que realizam uma tarefa** específica.
Elas servem para **organizar**, **reaproveitar** e **facilitar** o código.

Pense numa função como uma **caixinha que faz algo**. Você coloca uma entrada (se quiser), ela faz um trabalho e pode te dar uma saída (resultado).

Por que usar funções?

1. **Organização –** O código fica mais claro e dividido em partes.
2. **Reutilização** – Você escreve uma vez e usa várias.
3. **Facilidade de manutenção** – Se algo der errado, você conserta só no lugar certo.
4. **Facilita o entendimento** – Quem lê o código sabe o que cada parte faz

Básico:

- Crie uma função chamada soma que receba dois números inteiros como parâmetros e retorne o resultado da soma. No main, solicite que o usuário informe dois números e mostre o resultado da soma. Esse exemplo demonstra o uso básico de uma função com parâmetros e retorno.

```
Unset
fun soma(a: Int, b: Int): Int {
	return a + b
}

fun main() {
	print("Digite o primeiro número: ")
	val num1 = readlnOrNull()?.toIntOrNull() ?: 0

	print("Digite o segundo número: ")
	val num2 = readlnOrNull()?.toIntOrNull() ?: 0

	val resultado = soma(num1, num2)
	println("A soma dos números é: $resultado")
}
```

- Crie uma função chamada mostrarMensagem que receba um nome como parâmetro e apenas exiba uma saudação personalizada. Essa função não deve retornar valor (Unit). Esse exemplo mostra o uso de funções que executam ações, sem necessidade de retornar algo.

```
Unset
fun mostrarMensagem(nome: String) {
	println("Olá, $nome! Bem-vindo(a) ao curso de Kotlin!")
}

fun main() {
	print("Digite seu nome: ")
	val nome = readlnOrNull() ?: "Visitante"

	mostrarMensagem(nome)
}
```

Completo:

- Crie um programa que utilize:
    - Uma função com parâmetro com valor padrão;
    - Uma função com parâmetros nomeados;
    - Uma função que receba quantidade variável de números (vararg).
- Essas abordagens mostram como as funções podem ser mais flexíveis em Kotlin.

Unset

```
// Parâmetro com valor padrão
fun saudacao(nome: String, saudacao: String = "Olá") {
	println("$saudacao, $nome!")
}

// Parâmetros nomeados
fun apresentarPessoa(nome: String, idade: Int, cidade: String) {
	println("Nome: $nome, Idade: $idade, Cidade: $cidade")
}

// Função com vararg (quantidade variável de argumentos)
fun somarVarios(vararg numeros: Int): Int {
	var soma = 0
	for (numero in numeros) {
	soma += numero
	}
	return soma
}

fun main() {
	saudacao("Lucas")
	saudacao("Ana", "Bem-vinda")

	apresentarPessoa(nome = "Carlos", cidade = "São Paulo", idade = 30)

	val resultado = somarVarios(10, 20, 30, 40)
	println("Soma total: $resultado")
}
```

- Desenvolva um programa com o máximo de variações modernas e idiomáticas de funções em Kotlin:
    - Função de extensão;
    - Função de ordem superior;
    - Lambda atribuída a variável;
    - Função aninhada (local);

    - Todas as funções anteriores sendo utilizadas no main.
- Esse programa deve servir como um guia de estudo e prática.

Unset

```
// Função de extensão
fun Int.quadrado(): Int = this * this

// Função que retorna outra função (ordem superior)
fun operacao(tipo: String): (Int, Int) -> Int {
	return when (tipo) {
	"soma" -> { a, b -> a + b }
	"multiplicacao" -> { a, b -> a * b }
	else -> { a, b -> 0 }
	}
}

// Função lambda atribuída a uma variável
val saudacao2: (String) -> Unit = { nome -> println("Olá, $nome!") }

fun main() {
	// Função de extensão
	val numero = 5
	println("O quadrado de $numero é ${numero.quadrado()}")

	// Função de ordem superior
	val somar = operacao("soma")
	val multiplicar = operacao("multiplicacao")
	println("Soma: ${somar(10, 5)}")
	println("Multiplicação: ${multiplicar(10, 5)}")

	// Lambda
	saudacao2("Maria")

	// Função aninhada (local)
	fun mensagemFinal() {
	println("Todas as funções foram demonstradas com sucesso!")
	}

	mensagemFinal()
}
```

Arrays

Básico:

- Crie um programa que declare um array de inteiros com cinco posições. O programa deve:
    - Atribuir valores manualmente ao array;
    - Exibir todos os elementos;
    - Mostrar o primeiro e o último elemento.
- Esse exemplo ensina a criação de arrays, acesso por índice e iteração simples com for.

```
Unset
fun main() {
	val numeros = IntArray(5) // Cria um array com 5 posições (todos com
valor 0)

	// Atribuindo valores manualmente
	numeros[0] = 10
	numeros[1] = 20
	numeros[2] = 30
	numeros[3] = 40
	numeros[4] = 50

	// Exibindo todos os valores com for tradicional
	for (i in 0 until numeros.size) { // Poderia usar também numeros.indices
	println("Elemento na posição $i: ${numeros[i]}")
	}

	println("Primeiro elemento: ${numeros.first()}")
	println("Último elemento: ${numeros.last()}")
}
```

- Crie um programa que leia 5 números inteiros do usuário e armazene em um array. Em seguida:
    - Calcule a soma total;
    - Calcule a média dos valores;
    - Exiba os elementos usando for, forEach e indices.
- Esse exercício consolida a entrada de dados, uso de arrays e iteração com diferentes estruturas.

```
Unset
fun main() {
	val numeros = IntArray(5)

	// Leitura dos números
	for (i in numeros.indices) {
	print("Digite o ${i + 1}º número: ")
	numeros[i] = readln().toInt()
	}

	// Soma e média
	var soma = 0
	for (numero in numeros) {
	soma += numero
	}

	val media = soma / numeros.size

	println("Elementos digitados com forEach:")
	numeros.forEach { println(it) }

	println("Soma: $soma")
	println("Média: $media")
}
```

Completo:

- Crie um programa que leia 5 nomes e armazene em um array. Em seguida:
    - Capitalize todos os nomes (primeira letra maiúscula);
    - Filtre apenas os que começam com a letra "A";
    - Ordene em ordem alfabética;
    - Junte os nomes filtrados em uma string separada por vírgulas.
- Utilize funções map, filter, sorted e joinToString.

```
Unset
fun main() {
	// Declaração do array de nomes com 5 posições
	val nomes = Array(5) { "" }

	// Leitura dos nomes
	for (i in nomes.indices) {
	print("Digite o ${i + 1}º nome: ")
```

```
        nomes[i] = readlnOrNull() ?: "Erro ao ler o nome"
        }

        val resultado = nomes
        .map { it.replaceFirstChar { c -> c.uppercase() } }
        .filter { it.startsWith("A") }
        .sorted()
        .joinToString(", ")

        println("Nomes que começam com A (ordenados): $resultado")
}
```

- Crie um programa que:
    - Leia 10 números inteiros e armazene em um array;
    - Calcule o maior, menor e a média;
    - Conte quantos números são pares e ímpares;
    - Mostre os números acima da média.
- Esse exemplo mostra como extrair estatísticas simples de arrays, aplicando conhecimento de laços, operadores e condicionais.

Unset

```
fun main() {
        val numeros = IntArray(10)

        for (i in numeros.indices) {
        print("Digite o ${i + 1}º número: ")
        numeros[i] = readlnOrNull()?.toIntOrNull() ?: 0
        }

        val maior = numeros.maxOrNull()
        val menor = numeros.minOrNull()
        val media = numeros.average()

        var pares = 0
        var impares = 0
        for (n in numeros) {
        if (n % 2 == 0) pares++ else impares++
        }

        val acimaDaMedia = numeros.filter { it > media }

        println("Maior número: $maior")
        println("Menor número: $menor")
```

```
        println("Média: $media")
        println("Pares: $pares | Ímpares: $impares")
        println("Números acima da média: $acimaDaMedia")
}
```