LISTA DE EXERCICIOS KOTLIN

ALUNO: Gabriel Felipe Jess Meira.

EX 1) Crie uma função que receba um número inteiro e retorne verdadeiro se o número for par e falso caso contrário.

fun main() {  
 var num : Int? = null;  
  
 *print*("Insira o numero desejado: ");  
 num = *readLine*()!!.*toInt*();  
  
 if (*verificaPar*(num)){  
 *println*("Eh par!");  
 }else {  
 *println*("Eh impar!");  
 }  
}  
  
fun verificaPar(num : Int) : Boolean{  
 if (num % 2 == 0){  
 return true;  
 }  
 return false;  
}

EX 2) Crie uma função que receba um array de inteiros e retorne o maior número.

fun main(){  
 var numeros : Array<Int> = *arrayOf*(34, 57, 32, 37, 91);  
  
 *println*("O maior numero eh " + *retornaMaior*(numeros));  
}  
  
fun retornaMaior(lista : Array<Int>) : Int{  
 var maior : Int = 0;  
 for (i in lista){  
 if(i > maior){  
 maior = i;  
 }  
 }  
 return maior;  
}

EX 3) Crie uma classe chamada "Pessoa" com os atributos "nome" e "idade". Em seguida, crie uma lista de objetos "Pessoa" e ordene a lista em ordem alfabética pelo atributo "nome".

fun main() {  
 var pessoa1 = Pessoa("Gabriel", 19);  
 var pessoa2 = Pessoa("Gustavo", 16);  
 var pessoa3 = Pessoa("Luciane", 49);  
 var pessoa4 = Pessoa("Eliseu", 62);  
  
 var pessoas : List<Pessoa> = mutableListOf(pessoa1, pessoa2, pessoa3, pessoa4);  
  
 pessoas = pessoas.sortedBy {it.nome};  
  
 for(Pessoa in pessoas){  
 println(Pessoa.nome + " - " + Pessoa.idade);  
 }  
}  
  
class Pessoa(var nome: String, var idade: Int);

EX 4) Crie uma função que receba uma string e retorne verdadeiro se a string for um palíndromo (ou seja, pode ser lida da mesma forma de trás para frente).

fun main() {  
 *println*("Insira a String: ");  
 var string : String = *readLine*()!!;  
  
 string = string.*toUpperCase*();  
  
 if(string == *inverterString*(string)){  
 *println*("Eh palíndrono");  
 }else{  
 *println*("Não eh palíndrono");  
 }  
}  
  
fun inverterString(string: String) : String{  
 var stringInvertida = "";  
 for(i in string.length - 1 *downTo* 0){  
 stringInvertida += string[i];  
 }  
 return stringInvertida;  
}

EX 5) Implemente uma função lambda que retorne o maior valor entre dois números.

fun main() {  
 *println*("Insira o primeiro número: ")  
 var num1 : Int = *readLine*()!!.*toInt*();  
  
 *println*("Insira o segundo número: ")  
 var num2 : Int = *readLine*()!!.*toInt*();  
  
 *println*("O maior número entre $num1 e $num2 é " + *maiorNumero*(num1, num2));  
}  
  
val *maiorNumero* : (Int, Int) -> Int = **{**a, b **->** if (a > b) a else b**}**;

EX 6) Crie uma classe "ContaBancaria" com os atributos "saldo" e "limite". Adicione um método chamado "saque" que recebe um valor como parâmetro e subtrai do saldo da conta. Se o valor do saque for maior que o saldo da conta, o método deve lançar uma exceção com a mensagem "Saldo insuficiente".

fun main() {  
 try{  
 var conta1 : ContaBancaria = ContaBancaria(1250.50f,500f);  
  
 *println*("Insira o valor a ser sacado: ");  
 var valorSacado : Float = *readLine*()!!.*toFloat*();  
  
 conta1.saque(valorSacado);  
 }catch (e : exception){  
 *println*(e.message);  
 }  
}  
  
class ContaBancaria(var saldo : Float, var limite : Float){  
  
 fun saque(valor : Float){  
 if (this.saldo >= valor){  
 this.saldo -= valor;  
 *println*("R$$valor sacado, saldo atualizado para R$${this.saldo}");  
 }else{  
 throw exception("Saque insuficiente!");  
 }  
 }  
}  
  
class exception(message: String) : Exception(message);

EX 7) Crie uma função que receba uma lista de strings e retorne a string mais longa da lista.

fun main() {  
 *println*("Insira as strings abaixo: ")  
 *println*("Para parar escreva 'parar'")  
  
 var strings : MutableList<String> = *mutableListOf*();  
 var string : String? = null;  
 var flag : Boolean = true;  
 while(flag){  
 string = *readLine*()!!;  
 if (string == "parar"){  
 flag = false;  
 }else{  
 strings.add(string);  
 }  
 }  
  
 var maiorString : String = "";  
 for (String in strings){  
 if (String.length > maiorString.length){  
 maiorString = String;  
 }  
 }  
  
 *println*("A maior string é $maiorString com ${maiorString.length} caracteres");  
}

EX 8) Crie uma classe "Funcionario" com os atributos "nome", "idade" e "salario". Crie uma função que receba uma lista de funcionários e retorne o funcionário com o maior salário.

fun main() {  
 var func1 : Funcionario = Funcionario("Gabriel", 19, 10000f);  
 var func2 : Funcionario = Funcionario("Antônio", 23, 6780f);  
 var func3 : Funcionario = Funcionario("José", 25, 9865f);  
  
 var funcionarios : List<Funcionario> = *listOf*(func1, func2, func3);  
  
 var funcMaiorSalario : Funcionario? = *maiorSalario*(funcionarios);  
  
 *println*("O maior salário dos funcionarios é R$" + funcMaiorSalario?.salario + ", do funcionário " + funcMaiorSalario?.nome);  
}  
  
class Funcionario(var nome : String, var idade : Int, var salario : Float);  
  
fun maiorSalario(funcionarios : List<Funcionario>) : Funcionario?{  
 var funcionario : Funcionario? = null;  
  
 for(Funcionario in funcionarios){  
 if(funcionario == null){  
 funcionario = Funcionario;  
 }  
 if (Funcionario.salario > funcionario.salario){  
 funcionario = Funcionario;  
 }  
 }  
  
 return funcionario;  
}

EX 9) Crie uma função que receba uma lista de números inteiros e retorne uma lista com os números em ordem crescente, sem usar o método de ordenação da linguagem.

fun main() {  
 var numeros : MutableList<Int> = *mutableListOf*(0, 45, 32, -1, 67, 5, 42, 90, 142);  
  
 *print*("Números desordenados:")  
 for(Int in numeros){  
 *print*(" $Int ");  
 }  
 *println*();  
  
 var aux : Int = 0;  
 for (i in 0..numeros.size - 1){  
 for (j in 0..numeros.size - 1){  
 if (numeros[j] > numeros[i]){  
 aux = numeros[i];  
 numeros[i] = numeros[j];  
 numeros[j] = aux;  
 }  
 }  
 }  
  
 *print*("Números ordenados:")  
 for(Int in numeros){  
 *print*(" $Int ");  
 }  
 *println*();  
}

EX 10) Crie uma classe "Triangulo" com os atributos "base" e "altura". Adicione um método chamado "area" que calcula e retorna a área do triângulo.

fun main() {  
 var triangulo : Triangulo = Triangulo(10f, 15f);  
  
 *println*("A área do triângulo com base ${triangulo.base} e altura ${triangulo.altura} é: ${triangulo.area()}");  
}  
  
class Triangulo(var base : Float, var altura : Float){  
  
 fun area() : Float{  
 return (this.base \* this.altura) / 2;  
 }  
  
}

EX 11) Crie uma função que receba uma lista de strings e retorne uma lista com todas as strings que começam com a letra "A", em ordem alfabética.

fun main(){  
 var lista : List<String> = *listOf*("Banana", "Abacaxi", "Arara", "Curupira", "Bagre", "Amendoim");  
  
 *println*("Lista completa desordenada:");  
 for (String in lista){  
 *println*(String);  
 }  
  
 *println*();  
  
 *println*("Lista com palavras que começam com 'A' ordenada:");  
 for (String in *retornaAOrdenada*(lista)){  
 *println*(String);  
 }  
}  
  
fun retornaAOrdenada(lista : List<String>) : List<String>{  
 var listaAOrdenada : MutableList<String> = *mutableListOf*();  
  
 for(String in lista){  
 if (String[0] == 'A' || String[0] == 'a'){  
 listaAOrdenada.add(String);  
 }  
 }  
  
 listaAOrdenada.*sort*();  
  
 return listaAOrdenada;  
}

EX 12) Utilize um mapa para representar um dicionário de palavras e suas traduções.

fun main(){  
 val dicionario : Map<String, String> = *mapOf*(  
 "car" *to* "carro",  
 "truck" *to* "caminhão",  
 "bus" *to* "ônibus",  
 "street" *to* "rua",  
 "semaphore" *to* "semáforo"  
 )  
  
 *println*("Tradução de 'car': '" + dicionario["car"] + "'");  
 *println*("Tradução de 'truck': '" + dicionario["truck"] + "'");  
 *println*("Tradução de 'bus': '" + dicionario["bus"] + "'");  
 *println*("Tradução de 'street': '" + dicionario["street"] + "'");  
 *println*("Tradução de 'semaphore': '" + dicionario["semaphore"] + "'");  
}

EX 13) Crie uma função de ordem superior chamada operacaoMatematica que aceita dois números e uma função lambda. A função operacaoMatematica deve aplicar a função lambda aos dois números e retornar o resultado. Em seguida, crie algumas funções lambda para realizar operações matemáticas, como soma, subtração, multiplicação e divisão. Use a função de ordem superior para realizar essas operações com diferentes pares de números.

fun main(){  
 *println*("Soma de 10 e 5 = " + *operacaoMatematica*(10, 5, *soma*));  
 *println*("Subtração de 10 e 5 = " + *operacaoMatematica*(10, 5, *subtracao*));  
 *println*("Multiplicação de 10 e 5 = " + *operacaoMatematica*(10, 5, *multiplicacao*));  
 *println*("Divisão de 10 e 5 = " + *operacaoMatematica*(10, 5, *divisao*));  
}  
  
val *soma* : (Int, Int) -> Int = **{**a, b **->** a + b**}**;  
val *subtracao* : (Int, Int) -> Int = **{**a, b **->** a - b**}**;  
val *multiplicacao* : (Int, Int) -> Int = **{**a, b **->** a \* b**}**;  
val *divisao* : (Int, Int) -> Int = **{**a, b **->** a / b**}**;  
  
fun operacaoMatematica(a: Int, b: Int, operacao: (Int, Int) -> Int) : Int{  
 return operacao(a,b);  
}

EX 14) Crie uma função de extensão chamada isPalindromo para a classe String que verifica se a string é um palíndromo. A função deve ignorar espaços em branco e ser case-insensitive (não distinguir maiúsculas de minúsculas). Em seguida, use essa função para verificar se algumas palavras são palíndromos.

fun main() {  
 var palavras : List<String> = *listOf*("O vo", "ovo", "Net uno", "ne tuno", "Na T aN", "Na tan");  
  
 var palavra : String = "";  
  
 for(String in palavras){  
 palavra = String.*trim*();  
 if(palavra == *inverterString*(palavra)){  
 *println*("$String eh palíndrono");  
 }else{  
 *println*("$String não eh palíndrono");  
 }  
 }  
}  
  
fun inverterString(string: String) : String{  
 var stringInvertida = "";  
 for(i in string.length - 1 *downTo* 0){  
 stringInvertida += string[i];  
 }  
 return stringInvertida;  
}

EX 15) Vamos criar um programa que utilize funções de alta ordem para operar em um array de números inteiros. Você deve criar funções de alta ordem para realizar as seguintes operações: filtrarPares: Uma função que filtra e retorna apenas os números pares do array; dobrarValores: Uma função que dobra o valor de cada número no array; somarValores: Uma função que calcula a soma de todos os valores no array. Em seguida, crie um array de números inteiros e utilize as funções de alta ordem para realizar essas operações.

fun main() {  
 val numeros: IntArray = *intArrayOf*(10, 20, 35, 45, 50, 60, 75, 85, 90, 105)  
  
 *execucao*(numeros, *filtrarPares*, *dobrarValores*, *somarValores*);  
}  
  
fun execucao(numeros: IntArray, filtroPares: (IntArray)-> IntArray, dobrarValor: (IntArray)-> IntArray, somarValor: (IntArray) -> Int){  
 *println*("Números Pares: ${filtroPares(numeros).*joinToString*()}")  
 *println*("Números Dobrados: ${dobrarValor(numeros).*joinToString*()}")  
 *println*("Soma dos Valores: ${somarValor(numeros)}")  
}  
  
val *filtrarPares*: (IntArray) -> IntArray = **{** inteiros **->** inteiros.*filter* **{ it** % 2 == 0 **}**.*toIntArray*() **}**val *dobrarValores*: (IntArray) -> IntArray = **{** inteiros **->** inteiros.*map* **{ it** \* 2 **}**.*toIntArray*() **}**val *somarValores*: (IntArray) -> Int = **{** inteiros **->** inteiros.*sum*() **}**