

Exercícios - Curso Fundamentos de Programação

- Crie uma função que retorna a *string* "Olá, " concatenada com um argumento *text* (a ser passado para a função) e com ponto de exclamação "!" no final.

Exemplos:

```
cumprimentar("Leonardo") // retornará "Olá, Leonardo!"  
cumprimentar("Maria") // retornará "Olá, Maria!"
```

- Escreva uma função que receba a idade de uma pessoa em anos e retorne a mesma idade em dias.

Obs: considere que um ano tem 365 dias. Desconsidere anos bissextos (com 366 dias) e desconsidere também dias decorridos desde o último aniversário.

Exemplos:

```
converterIdadeEmAnosParaDias(25) // retornará 9125  
converterIdadeEmAnosParaDias(70) // retornará 25550
```

- Desenvolva uma função que recebe dois parâmetros, um é a quantidade de horas trabalhadas por um funcionário num mês, e o quanto ele recebe por hora. A função deverá calcular o salário líquido mensal do funcionário, que é da quantidade de horas trabalhadas no mês multiplicada pelo valor da sua hora. Desse valor, deve ser subtraído 30%, relativo a impostos. O retorno da função deve ser a string "Salário líquido é igual a R\$ X", em que X é o salário líquido do funcionário no mês.

Exemplos:

```
calcularSalario(150, 40.5) // retornará "Salário igual a R$ 6075"
```

- Crie uma função que recebe um número (de 1 a 12) e retorne o mês correspondente como uma *string*. Por exemplo, se a entrada for 2, a função deverá retornar "fevereiro", pois este é o 2º mês.

Exemplos:

```
nomeDoMes(1) // retornará "janeiro"  
nomeDoMes(4) // retornará "abril"
```

- Crie uma função que receba dois números e retorne se o primeiro é maior ou igual ao segundo.

Exemplos:

```
maiorOuIgual(0, 0) // retornará true  
maiorOuIgual(0, "0") // retornará false  
maiorOuIgual(5, 1) // retornará true
```

- Escreva uma função que receba um valor booleano ou numérico. Se o parâmetro fornecido for booleano, o retorno da função deverá ser o inverso. Por exemplo, se a entrada for *false*, retornará *true*. Se o parâmetro for numérico, o retorno será o número inverso. Por exemplo, se for fornecido 1, o retorno será $\neq 1$. Se o parâmetro de entrada não for de nenhum dos tipos acima, retorne "booleano ou número esperados, mas o parâmetro é do tipo ...".

Exemplos:

```

inverso(true) // retornará false
inverso("6") // retornará "booleano ou número esperados, mas o parâmetro é do tipo string"
inverso(-2000) // retornará 2000
inverso("programação") // retornará "booleano ou números esperado, mas o parâmetro é do tipo string"

```

- Crie uma função que receba três números como parâmetro (numero, minimo, maximo) e também um parâmetro booleano (inclusivo) e retorne se o parâmetro numero (o primeiro parâmetro) está entre minimo e maximo. Quando o parâmetro inclusivo for true, tenha "entre" como inclusivo, ou seja, considerando se numero é igual a minimo ou a maximo. Caso o parâmetro inclusivo não seja informado, seu valor padrão deverá ser false, portanto, a lógica será exclusiva, não considerando se numero é igual a minimo ou a maximo.

Exemplos:

```

estaEntre(10, 100, 50) // retornará true
estaEntre(16, 100, 160) // retornará false
estaEntre(3, 150, 3) // retornará false
estaEntre(3, 150, 3, true) // retornará true

```

- Desenvolva uma função que recebe dois números inteiros não negativos (maiores ou iguais a zero) e realize a multiplicação deles. Porém, não utilize o operador de multiplicação.

Exemplo:

```

multiplicar(5, 5) // retornará 25
multiplicar(0, 7) // retornará 0

```

- Escreva uma função que receba dois parâmetros. O primeiro parâmetro é o elemento que repetirá, enquanto que o segundo será o número de vezes que haverá repetição. Um array será retornado.

Exemplos:

```

repetir("código", 2) // retornará ["código", "código"]
repetir(7, 3) // retornará [7, 7, 7]

```

- Elabore uma função que receba um número como parâmetro e retorne uma string com o símbolo "+" na quantidade especificada no parâmetro.

Exemplos:

```

simboloMais(2) // retornará "++"
simboloMais(4) // retornará "++++"

```

- Crie uma função que receba uma *array* e retorne o primeiro e o último elemento desse array como um novo *array*:

Exemplos:

```

receberPrimeiroUltimoElemento([7,14,"olá"]) // retornará [7, "olá"]
receberPrimeiroUltimoElemento([-100, "aplicativo", 16]) // retornará [-100, 16]

```

- Quando temos um objeto e manipulamos seus atributos, adicionando, atualizando ou removendo-os, estamos apenas modificando-o, mas, em essência, o objeto continua o mesmo, ou seja a sua referência de memória é a mesma.

Num projeto que você está trabalhando, você foi designado a refatorar diversas funções para que façam cópias de objetos e manipulem os dados dessas cópias, com o intuito de evitar efeitos indesejáveis em algumas situações devido a referências a objetos. Abaixo, está a descrição de uma dessas funções.

Você escreverá uma função que recebe um objeto como primeiro parâmetro e, como segundo parâmetro, o nome de uma propriedade contida nesse objeto. Em seguida, retorne uma cópia desse objeto sem a propriedade especificada no segundo parâmetro.

Exemplos:

```
removerPropriedade({a: 1, b: 2}, "a") // retornará {b: 2}
removerPropriedade({
  id: 20,
  nome: "caneta",
  descricao: "Não preenchido"
}, "descricao") // retornará {id: 20, nome: "caneta"}
```

A fim de testar se o objeto retornado não é o mesmo que foi passado como parâmetro para a função *removerPropriedade*, você poderá usar a função *Object.is()*, por exemplo:

```
Object.is(removerPropriedade(objeto, "descricao"), objeto)
```

Retornará *false* se o objeto não for o mesmo.

- Crie uma função que receba um array de elementos e retorne um array somente com os números presentes no array recebido como parâmetro.

Exemplos:

```
filtrarNumeros(["Javascript", 1, "3", "Web", 20]) // retornará [1, 20]
filtrarNumeros(["a", "c"]) // retornará []
```

- Desenvolva uma função que receba como parâmetro um objeto e retorne um array de arrays, em que cada elemento é um array formado pelos pares *chave/valor* que corresponde a um atributo do objeto. Observe os exemplos abaixo para um melhor entendimento:

Exemplos:

```
objetoParaArray({
  nome: "Maria",
  profissao: "Desenvolvedora de software"
}) // irá retornar [["nome", "Maria"], ["profissao", "Desenvolvedora de Software"]]

objetoParaArray({
  codigo: 11111,
  preco: 12000
}) // irá retornar [["codigo", 11111], ["preco", 12000]]
```

- Elabore uma função que receba um *array* de números e retorne um array que tenha todos os números que são pares e que também tenham índices pares.

Lembre-se que um número é par porque é divisível por 2, ou seja, o resto da divisão da divisão dele por 2 é zero.

Exemplos:

```
receberSomenteOsParesDeIndicesPares([1, 2, 3, 4]) // retornará []
receberSomenteOsParesDeIndicesPares([10, 70, 22, 43]) // retornará [10, 22]
```

- A fim de manter o calendário anual ajustado com o movimento de translação da Terra, criou-se os anos bissextos, que têm 366 dias em vez dos 365 presentes nos anos normais.

Para determinar se um ano é bissexto, é necessário saber se ele é múltiplo de 4. Não pode ser múltiplo de 100, exceto se for também múltiplo de 400.

Com isso em mente, desenvolva uma função que receba um número correspondente a um ano e retorne se ele é bissexto ou não.

Exemplos:

```
checarAnoBissexto(2020) // retornará true
checarAnoBissexto(2100) // retornará false, pois é múltiplo de 100 e não é múltiplo de 400
```

- Escreva uma função que receba um array de números e retornará a soma de todos os números desse array.

Exemplos:

```
somarNumeros([10, 10, 10]) // retornará 30
somarNumeros([15, 15, 15, 15]) // retornará 60
```

- Você está trabalhando numa aplicação pessoal de controle de despesas. Na tela principal dessa aplicação, é possível adicionar produtos ou serviços, informando nome, categoria e preço. Uma funcionalidade que você está desenvolvendo no momento é a de somar o total das despesas.

Crie uma função que receba um *array* de produtos e retorne o total das despesas.

Exemplos:

```
despesasTotais([
  {nome: "Jornal online", categoria: "Informação", preco: 89.99},
  {nome: "Cinema", categoria: "Entretenimento", preco: 150}
]) // retornará 239.99

despesasTotais([
  {nome: "Galaxy S20", categoria: "Eletrônicos", preco: 3599.99},
  {nome: "Macbook Pro", categoria: "Eletrônicos", preco: 30999.90}
]) // retornará 34599.89
```

- Numa aplicação Web de investimento financeiro da qual você faz parte da equipe de desenvolvimento, pretende-se adicionar a funcionalidade de calcular a média de um conjunto de números informados pelo usuário.

Com o intuito de realizar esse cálculo, crie uma função que receba um array com uma quantidade indeterminada de números e retorne a média simples desses números.

A média simples é o resultado da soma de todos os números dividido pela quantidade de números.

Exemplos:

```
calcularMedia([0, 10]) // retornará 5
calcularMedia([1, 2, 3, 4, 5]) // retornará 3
```

- Faça uma função que receba a base e a altura de um triângulo e retorne a área desse triângulo. A precisão deverá ser de duas casas decimais, arredondando se necessário.

Obs: a fórmula para calcular a área de um triângulo é $(base \times altura) / 2$

Exemplos:

```
areaDoTriangulo(10, 15) // retornará 75.00
areaDoTriangulo(7, 9) // retornará 31.50
areaDoTriangulo(9.25, 13.1) // retornará 60.59
```

- Criar uma função que receba um array de números e retorne o menor número desse array.

Exemplos:

```
menorNumero([10, 5, 35, 65]) // retornará 5
menorNumero([5, -15, 50, 3]) // retornará -15
```

- Desenvolva uma função que receba como parâmetro um número de 1 a 10. Internamente, na função, será gerado um número aleatório de 1 a 10. A função deverá retornar se o parâmetro de entrada foi igual ao número sorteado internamente. Se o valor fornecido foi o sorteado, retorne "Parabéns! O número sorteado foi o X". Se não for igual, retorne "Que pena! O número sorteado foi o X". X é o número que foi sorteado.

Exemplos:

```
funcaoDaSorte(10) // retornará "Parabéns! O número sorteado foi o 10"
funcaoDaSorte(5) // retornará "Que pena! O número sorteado foi o 3"
funcaoDaSorte(5) // retornará "Que pena! O número sorteado foi o 1"
```

- Criar uma função que receba uma string como parâmetro e conte quantas palavras tem nela.

Considere que todas as palavras só são separadas por um espaço.

Exemplos:

```
contarPalavras("Sou uma frase") // retornará 3
contarPalavras("Me divirto aprendendo a programar") // retornará 5
```

- Desenvolva uma função que recebe um caractere e uma string como parâmetros e retorne a quantidade de vezes que o caractere se repete na string. A função deverá ser *case-sensitive*, ou seja, irá diferenciar maiúsculas de minúsculas.

Exemplos:

```
contarCaractere("r", "A sorte favorece os audazes") // retornará 2
contarCaractere("c", "Conhece-te a ti mesmo") // retornará 1
```

- A fim de criar um mecanismo de busca para sua aplicação, você precisa iniciar criando uma função para identificar palavras semelhantes.

Escreva uma função que recebe como primeiro parâmetro uma palavra e, como segundo parâmetro, um *array* de *strings*. A função deverá filtrar as palavras do *array* que contêm nela a *string* do primeiro parâmetro.

Exemplos:

```
buscarPalavrasSemelhantes("pro", ["programação", "mobile", "profissional"]) // retornará ["programação", "profissional"]
buscarPalavrasSemelhantes("python", ["javascript", "java", "c++"]) // retornará []
```

- Desenvolva uma função que receba uma *string* como parâmetro e retorne essa *string* somente com as consoantes, ou seja, sem as vogais.

Exemplos:

```
removerVogais("Cod3r") // retornará "Cd3r"
removerVogais("Fundamentos") // retornará "Fndmnts"
```

- Desenvolva uma função que recebe um objeto como parâmetro e retorne um outro objeto que corresponde ao objeto recebido como parâmetro, porém com as posições das chaves e valores invertidas, conforme exemplo a seguir:

Exemplo:

```
inverter({ a: 1, b: 2, c: 3 }) // retornará { 1: "a", 2: "b", 3: "c" }
```

- Elabore uma função que recebe dois parâmetros: o primeiro é um array de números e o segundo é um número que especifica uma quantidade de dígitos. Essa função deverá retornar somente aqueles números do array que têm a quantidade de dígitos indicada pelo segundo parâmetro.

Exemplos:

```
filtrarPorQuantidadeDeDigitos([38, 2, 365, 10, 125, 11], 2) // retornará [38, 10, 11]
filtrarPorQuantidadeDeDigitos([5, 9, 1, 125, 11], 1) // retornará [5, 9, 1]
```

- Crie uma função que recebe um array de números e retorna o segundo maior número presente nesse array.

Exemplos:

```
segundoMaior([12, 16, 1, 5]) // retornará 12
segundoMaior([8, 4, 5, 6]) // retornará 6
```

- Elabore uma função que recebe um objeto com estudantes e suas notas. As notas de cada estudante estarão num array, conforme exemplo abaixo. Você deverá calcular a média da nota de cada aluno e retornar um objeto com os atributos *nome* e *média*, que indica o aluno que teve o melhor desempenho nas notas.

Exemplo:

```
receberMelhorEstudante({
  Joao: [8, 7.6, 8.9, 6], // média 7.625
  Mariana: [9, 6.6, 7.9, 8], // média 7.875
  Carla: [7, 7, 8, 9] // média 7.75
}) // retornará { nome: "Mariana", media: 7.875 }
```