Lista de Exercícios de Programação

Lista de Abreviações:

Abreviação	Significado
EX1IN	Exercício 1 – Inicial
EX2COND	Exercício 2 – Condicional
EX3WH	Exercício 3 – Laço While
EX4FOR	Exercício 4 – Laço For
EX5FU	Exercício 5 – Função
EX6LI	Exercício 6 – Lista
EX7LI	Exercício 7 – Lista
EX8CONJ	Exercício 8 – Conjunto
EX9DIC	Exercício 9 – Dicionário
EX10REC	Exercício 10 – Recursividade

Exercício EX1IN - Operações Aritméticas Básicas

Escreva um programa que leia do usuário dois números inteiros *a* e *b*. Seu programa deve calcular e exibir:

- A soma de *a* e *b*
- A diferença quando *b* é subtraído de *a*
- O produto de *a* por *b*
- O quociente quando *a* é dividido por *b*
- O resto quando *a* é dividido por *b*
- O resultado de $\log_{10}(a)$
- O resultado de *a* elevado à *b* (*a*^*b*)

Exercício EX2COND - Ordenação de três números

Escreva um programa que leia três números inteiros e os exiba em ordem crescente.

Exercício EX3LRWH - Média Aritmética

Escreva um programa que calcule a média aritmética de um conjunto de valores fornecidos pelo usuário. O programa deve solicitar números repetidamente até que o valor 0 seja informado, o qual indica que não serão inseridos novos valores.

Durante a execução, o programa deve somar todos os valores fornecidos e contar quantos números foram digitados. Caso o primeiro valor informado seja 0, o programa deve exibir uma mensagem de erro informando que é necessário inserir pelo menos um valor válido para realizar o cálculo da média. Ao final, o programa deve exibir a média aritmética dos valores válidos informados

Exercício EX4LRFO - Tabela de Conversão de Temperaturas

Escreva um programa que exiba uma tabela de conversão de temperaturas de graus Celsius para graus Fahrenheit. A tabela deve listar todas as temperaturas de 0 a 100 graus Celsius, incluindo apenas os valores que sejam múltiplos de 10. O programa deve apresentar cabeçalhos apropriados e tabulações para suas colunas.

Exercício EX5FU - Conversão de Decimal para Binário

Implemente duas versões de um programa que solicita ao usuário três números inteiros positivos em base decimal e exibe suas representações equivalentes em binário. Na primeira versão, a conversão deve ser realizada diretamente no corpo principal do programa, sem o uso de função auxiliar. Na segunda versão, a conversão deve ser realizada por meio de uma função específica, responsável por realizar a conversão de decimal para binário.

Exercício EX6LI - Lista de Divisores

Um divisor de um número inteiro positivo *n* é todo número inteiro positivo menor ou igual a *n* que o divide exatamente, ou seja, com resto zero. O programa deve conter uma função responsável por calcular e retornar, em uma lista, todos os divisores de um número inteiro positivo. Além disso, implemente uma função principal que leia um número inteiro informado pelo usuário e exiba todos os seus divisores.

Exercício EX7LI - Tokenização de Strings

Tokenizar uma `string` significa dividi-la em partes menores chamadas *tokens*. Em expressões matemáticas, os *tokens* são elementos como números inteiros (com sinal opcional), operadores (+, -, *, /, ^) e parênteses ("(" e ")"). *Tokens* como *, /, ^, "(" e ")" são sempre isolados, mas os símbolos + e - podem ser operadores ou parte de um número, dependendo do contexto.

Dica: Um + ou - é considerado operador se for precedido (ignorando espaços) por um número ou por um parêntese fechado. Caso contrário, faz parte de um número.

Implemente uma função que receba uma `string` contendo uma expressão matemática válida e retorne uma lista com os *tokens* da expressão. A função deve ignorar espaços em branco e considerar apenas números inteiros. Além disso, implemente um programa principal que leia uma expressão fornecida pelo usuário e exiba a lista de *tokens* resultante.

Exercício EX8CONJ - Caracteres Únicos

Escreva uma função para verificar se uma `string` contém apenas caracteres únicos. Por exemplo, a `string` "azul" não possui letras repetidas, enquanto a `string` "ferramenta" contém letras repetidas. A função deve receber uma `string` como entrada e retornar `true` se todas as letras forem únicas ou `false` caso contrário. A implementação da função deve utilizar conjuntos, quando possível, para verificar a unicidade dos caracteres.

Exercício EX9DIC - Busca Reversa

Escreva uma função chamada `buscaReversa`, que encontra todas as chaves de um dicionário mapeadas para um determinado valor. A função deve receber como parâmetros um dicionário, quando possível, e um valor a ser buscado no dicionário. Ela deve retornar uma lista (possivelmente vazia) com as chaves encontradas. Além disso, implemente uma função principal para demonstrar o funcionamento da função `buscaReversa`. A função deve ser capaz de retornar múltiplas chaves, uma única chave ou nenhuma chave, dependendo do caso.

Exercício EX10REC - Ordem Reversa

Implemente duas versões de um programa que solicite ao usuário quantos números inteiros serão informados e, em seguida, leia esses números um por um. Após a leitura, o programa deve exibir os números na ordem inversa em que foram digitados. Na primeira versão, a inversão deve ser feita diretamente no corpo principal do programa, sem o uso de recursividade. Na segunda versão, a inversão deve ser realizada utilizando recursividade, sem o uso de listas ou estruturas auxiliares para armazenar os números.