**Atividades - Estruturas de Dados e Tratamento de Exceções**

**1. O que é uma lista (ou array)? Dê um exemplo de criação de lista com 5 frutas.**

Uma **lista** (chamada de **array**) é uma estrutura de dados que armazena uma coleção ordenada de elementos, podendo conter valores de qualquer tipo (como números, strings, objetos etc.). Os elementos dentro do array são acessados por índices, que começam do zero.

const frutas = ["maçã", "banana", "bergamota", "uva", "morango"];

* ***frutas*** é um array que contém 5 elementos, cada um representando uma fruta.

**2. Como acessar o terceiro item de uma lista? Escreva um exemplo.**

***JavaScript***

let frutas = ["maçã", "banana", "laranja", "uva", "abacaxi"];

// Para acessar o terceiro item (que é "laranja"), usamos o índice 2

let terceiraFruta = frutas[2];

console.log(terceiraFruta); // Isso vai imprimir "laranja" no console

* A lógica é a mesma: a lista é criada com colchetes [] e os itens são acessados usando o índice, que começa em 0.

**3. Escreva um código que adicione um novo item ao final de uma lista.**

// Nossa lista inicial de frutas

let frutas = ["maçã", "banana", "laranja"];

console.log("Lista antes de adicionar:", frutas);

// Adicionando 'morango' ao final da lista

frutas.push("morango");

console.log("Lista depois de adicionar:", frutas);

Lista antes de adicionar: [ 'maçã', 'banana', 'laranja' ]

Lista depois de adicionar: [ 'maçã', 'banana', 'laranja', 'morango' ]

**4. Como remover um item específico de uma lista? Explique e exemplifique.**

* Remover pelo valor: o programa procura o item que tem aquele valor e remove a primeira ocorrência.
* Remover pelo índice: você informa a posição do item na lista, e o item nessa posição é removido.

const frutas = ["maçã", "banana", "laranja", "uva", "manga"];

// Remover "laranja" pelo valor

const index = frutas.indexOf("laranja");

if (index !== -1) {

frutas.splice(index, 1); // remove 1 item na posição index

}

console.log(frutas); // ["maçã", "banana", "uva", "manga"]

**5. Crie uma lista com 5 números e mostre a soma de todos os elementos.**

const numeros = [10, 20, 30, 40, 50];

let soma = 0;

for (let i = 0; i < numeros.length; i++) {

soma += numeros[i];

}

console.log("A soma dos números é:", soma);

Saída:

*A soma dos números é*: ***150***

**6. Escreva um código que ordene uma lista de números em ordem crescente.**

const numeros = [40, 10, 100, 5, 25];

numeros.sort((a, b) => a - b);

console.log("Números ordenados:", numeros);

Saída:

Números ordenados: [ 5, 10, 25, 40, 100]

**7. Crie uma lista com os nomes dos dias da semana e exiba apenas os dias úteis.**

const diasDaSemana = ["domingo", "segunda", "terça", "quarta", "quinta", "sexta", "sábado"];

// Filtrando os dias úteis (segunda a sexta)

const diasUteis = diasDaSemana.filter(dia =>

dia !== "domingo" && dia !== "sábado"

);

console.log("Dias úteis:", diasUteis);

Saída:

Dias úteis: [ 'segunda', 'terça', 'quarta', 'quinta', 'sexta' ]

**8. Escreva um código que verifique se um item existe dentro de uma lista.**

const frutas = ["maçã", "banana", "laranja", "uva", "manga"];

const itemProcurado = "laranja";

if (frutas.includes(itemProcurado)) {

console.log(itemProcurado + " está na lista.");

} else {

console.log(itemProcurado + " não está na lista.");

}

Saída:

laranja está na lista.

**9. O que é uma tupla? Em que ela se diferencia de uma lista?**

Uma tupla é uma estrutura de dados que armazena um conjunto fixo de valores, geralmente de tipos diferentes, e em uma ordem específica. O conceito de tupla existe em várias linguagens (como Python), mas JavaScript não possui tuplas nativas.

No entanto, em JavaScript podemos simular uma tupla usando um array com estrutura fixa ou congelando o array com Object.freeze( ) para impedir alterações.

A **principal diferença** entre uma **tupla** e uma **lista** está na **mutabilidade** — ou seja, se os dados podem ou não ser modificados após a criação.

#### Lista:

* É uma estrutura de dados **mutável**, ou seja, você pode **adicionar, remover ou alterar** elementos depois de criar.
* É usada quando os dados podem mudar com o tempo.
* Em JavaScript, listas são representadas por **arrays** ([]), que são muito flexíveis.

#### Tupla:

* É uma estrutura de dados **imutável**, ou seja, os valores **não podem ser alterados** depois de criados.
* Usada para armazenar **dados fixos ou constantes**, como pares de valores, datas, coordenadas, etc.
* Em JavaScript não existe tupla como tipo nativo, mas podemos **simular** uma usando Object.freeze( ) para "congelar" um array e impedir alterações.

**10. Crie uma tupla com 3 cores favoritas e tente alterar uma delas. O que acontece?**

### *JavaScript (tupla simulada com Object.freeze)*

const cores = Object.freeze(["roxo", "preto", "branco"]);

// Tentando alterar o primeiro elemento

cores[0] = "azul";

console.log(cores);

**O que acontece?** A tentativa de alterar cores[0] **não tem efeito** (em modo estrito pode lançar erro, mas geralmente é ignorada). O array permanece com os valores originais, porque está congelado.

**11. Escreva um código que percorra uma tupla e imprima cada item.**

### *JavaScript (tupla simulada com Object.freeze)*

**javascript**

const cores = Object.freeze(["roxo", "preto", "branco"]);

for (const cor of cores) {

console.log(cor);

}

Saída:

roxo

preto

branco

**12. O que é um dicionário em programação? Dê um exemplo com nome, idade e cidade de uma pessoa.**

Em programação, um **dicionário** é uma estrutura de dados que armazena pares **chave-valor**. Ou seja, cada valor é associado a uma chave única, e você pode acessar o valor rapidamente usando essa chave.

Em **JavaScript**, o dicionário é representado por um **objeto**.

const pessoa = {

nome: "João",

idade: 30,

cidade: "São Paulo"

};

console.log(pessoa.nome); // João

console.log(pessoa.idade); // 30

console.log(pessoa.cidade); // São Paulo

* "nome", "idade" e "cidade" são as chaves.
* "João", 30 e "São Paulo" são os valores associados a essas chaves.
* Você pode acessar os valores usando a notação de ponto (pessoa.nome) ou colchetes (pessoa["nome"]).

**13. Como acessar o valor de uma chave em um dicionário? Escreva um exemplo.**

Em JavaScript, para acessar o valor de uma chave em um dicionário (objeto), você pode usar:

***notação de ponto***

***objeto.chave***

***ou***

***colchetes***

***objeto["chave"]***

const pessoa = {

nome: "Ana",

idade: 25,

cidade: "Rio de Janeiro"

};

// Usando notação de ponto

console.log(pessoa.nome); // Ana

// Usando notação de colchetes

console.log(pessoa["cidade"]); // Rio de Janeiro

**14. Escreva um código que adicione uma nova chave e valor em um dicionário já criado.**

const pessoa = {

nome: "Carlos",

idade: 28,

cidade: "Curitiba"

};

// Adicionando uma nova chave "profissao" com valor "Engenheiro"

// Usando notação de ponto

pessoa.profissao = "Engenheiro";

// Ou usando notação de colchetes

// pessoa["profissao"] = "Engenheiro";

console.log(pessoa);

Saída:

{

nome: "Carlos",

idade: 28,

cidade: "Curitiba",

profissao: "Engenheiro"

}

**15. Crie um dicionário com 3 pessoas e suas idades. Exiba apenas os nomes.**

const pessoas = {

Ana: 25,

Bruno: 30,

Carla: 22

};

// Exibindo apenas os nomes (as chaves do objeto)

const nomes = Object.keys(pessoas);

console.log(nomes); // ["Ana", "Bruno", "Carla"]

* pessoas é o dicionário (objeto) com nomes como chaves e idades como valores.
* Object.keys(pessoas) retorna uma lista com todas as chaves do objeto, ou seja, os nomes das pessoas.

**16. O que é um conjunto (set)? Em que ele é diferente de uma lista?**

Um **conjunto** (ou **Set**) é uma estrutura de dados usada para armazenar **valores únicos**, ou seja, **não permite elementos repetidos**. Ele é útil quando você quer garantir que não haja duplicatas entre os dados.

Já uma **lista** (ou array) é uma estrutura que **permite elementos repetidos** e **mantém a ordem** dos itens. Com listas, você pode ter o mesmo valor várias vezes e acessar os elementos pela posição (índice).

**17. Crie dois conjuntos com números e exiba a união e a interseção entre eles.**

const conjuntoA = new Set([1, 2, 3, 4]);

const conjuntoB = new Set([3, 4, 5, 6]);

// União: elementos de A e B (sem repetição)

const uniao = new Set([...conjuntoA, ...conjuntoB]);

console.log("União:", [...uniao]); // [1, 2, 3, 4, 5, 6]

// Interseção: apenas elementos que existem nos dois conjuntos

const intersecao = new Set([...conjuntoA].filter(item => conjuntoB.has(item)));

console.log("Interseção:", [...intersecao]); // [3, 4]

**18. Crie um conjunto com 5 números e remova um deles.**

const numeros = new Set([10, 20, 30, 40, 50]);

// Removendo o número 30

numeros.delete(30);

console.log(numeros); // Set(4) { 10, 20, 40, 50 }

**19. O que é uma exceção em programação? Dê um exemplo simples com try/except.**

Uma **exceção** é um **erro** que acontece **durante a execução** do programa. Quando isso ocorre, o programa **para imediatamente**, a menos que o erro seja **tratado**.

Para evitar que o programa quebre, usamos **blocos try...catch** (em JavaScript) para **capturar e tratar** esses erros de forma controlada.

try {

// Código que pode gerar erro

let resultado = 10 / 0;

console.log("Resultado:", resultado);

// Forçando um erro (acessando variável que não existe)

console.log(naoExiste);

} catch (erro) {

// Código que trata o erro

console.log("Ocorreu um erro:", erro.message);

}

* O código dentro do try é executado normalmente.
* Se um erro acontecer (como naoExiste não estar definido), o programa **não trava**.
* Em vez disso, ele vai direto para o catch, onde o erro é tratado.

***Resultado esperado no console:***

Resultado: Infinity

Ocorreu um erro: naoExiste is not defined

**20. Escreva um código que tente dividir um número por zero e trate esse erro.**

try {

const numerador = 10;

const denominador = 0;

if (denominador === 0) {

throw new Error("Divisão por zero não é permitida!");

}

const resultado = numerador / denominador;

console.log("Resultado:", resultado);

} catch (erro) {

console.log("Erro capturado:", erro.message);

}

* Antes de fazer a divisão, o código verifica se o denominador é zero.
* Se for, lança um erro com throw new Error(...).
* Esse erro é capturado pelo bloco catch, que exibe a mensagem.

***Saída esperada:***

Erro capturado: Divisão por zero não é permitida!