Bloco de Exercícios 1

Exercício 01:

```
function lerEImprimirVetor() {
  const numeros = [];
  console.log("Por favor, insira 10 números inteiros:");

  for (let i = 0; i < 10; i++) {
    let numeroValido = false;
    while (!numeroValido) {
      const input = prompt(`Digite o ${i + 1}º número:`);
      const numero = parseInt(input);

      if (!isNaN(numero)) {
        numeros.push(numero);
        numeroValido = true;
      } else {
        alert("Entrada inválida. Por favor, digite um número inteiro.");
      }
    }
  }

  console.log("\nVetor armazenado (sentido normal):");
  console.log(numeros);

  console.log("\nVetor armazenado (sentido inverso):");

  const numerosInvertidos = [...numeros].reverse();
  console.log(numerosInvertidos);
}

lerEImprimirVetor();
```

Exercício 02:

```
const numeros = [];

console.log("Por favor, insira 10 números inteiros:");
for (let i = 0; i < 10; i++) {

  numeros[i] = +prompt(`Digite o ${i + 1}º número:`);
}

const numerosInvertidos = [];

for (let i = 0; i < 10; i++) {

  numerosInvertidos[i] = numeros[9 - i];
```

```
}

console.log("\nVetor Original:");
console.log(numeros);

console.log("\nVetor Invertido:");
console.log(numerosInvertidos);
```

Exercício 03:

```
function processarVetor() {
  const vetorOriginal = [];
  const vetorProcessado = [];
  const tamanhoVetor = 10;

  for (let i = 0; i < tamanhoVetor; i++) {
    let numero;
    do {
      numero = parseInt(prompt(`Digite o ${i + 1}º número inteiro e positivo:`));
      if (isNaN(numero) || numero <= 0) {
        alert("Por favor, insira um número inteiro e positivo.");
      }
    } while (isNaN(numero) || numero <= 0);
    vetorOriginal.push(numero);
  }

  for (let i = 0; i < tamanhoVetor; i++) {
    if (i % 2 === 0) {
      vetorProcessado.push(vetorOriginal[i] / 2);
    } else {
      vetorProcessado.push(vetorOriginal[i] * 3);
    }
  }

  console.log("Vetor Original:", vetorOriginal);
  console.log("Vetor Processado:", vetorProcessado);
```

Exercício 04:

```
function readNames() {
  const names = [];
  for (let i = 0; i < 10; i++) {
    const name = prompt(`Digite o ${i + 1}º nome:`);
    if (name) {
      names.push(name.trim());
    } else {
      i--;
    }
```

```
  }
  return names;
}
function searchName() {
  const nameList = readNames();
  const searchName = prompt("Digite um nome para procurar:");

  if (nameList.includes(searchName.trim())) {
    console.log("ACHEI");
    alert("ACHEI");
  } else {
    console.log("NÃO ACHEI");
    alert("NÃO ACHEI");
  }
}
searchName();
```

Exercício 05:

```
function getRandomInt(min, max) {
  min = Math.ceil(min);
  max = Math.floor(max);
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

const vetor1 = [];
const vetor2 = [];
const arraySize = 20;

for (let i = 0; i < arraySize; i++) {
  vetor1.push(getRandomInt(1, 100));
  vetor2.push(getRandomInt(1, 100));
}

console.log("Vetor 1:", vetor1);
console.log("Vetor 2:", vetor2);


const vetorDiferenca = vetor1.filter(item => !vetor2.includes(item));

const vetorSoma = vetor1.map((value, index) => value + vetor2[index]);

const vetorMultiplicacao = vetor1.map((value, index) => value * vetor2[index]);

console.log("\nVetor Diferença (elementos em vetor1 não presentes em vetor2):",
vetorDiferenca);
console.log("Vetor Soma (elemento a elemento):", vetorSoma);
console.log("Vetor Multiplicação (elemento a elemento):", vetorMultiplicacao);
```

Exercício 06:

```
function organizarnumeros ()
const numeros = [];
let imput;

while (true) {
    imput = prompt("Insira um número inteiro):");
    if (imput === null || imput.trim() === "") {
        break;
    }
    const numero = parseInt(imput, 10);
    if (!isNaN(numero)) {
        numeros.push(numero);
    } else {
        alert("Por favor, insira um número inteiro válido.");
    }
}

const pares = numeros.filter(num => num % 2 === 0).sort((a, b) => a - b);
const impares = numeros.filter(num => num % 2 !== 0).sort((a, b) => a - b);

pares.sort((a, b) => a - b);
impares.sort((a, b) => b - a);

const organizarnumeros = [...pares, ...impares];
console.log("Números organizados:", organizarnumeros);
```

Exercício 07:

```
function possuemconteudoigualJson(arr1, arr2) {
   return JSON.stringify(arr1) === JSON.stringify(arr2);
    }

const vetorP = [1,hello,true];
const vetorQ = [1,hello,true];
const vetorR = [1,hello,false];

console.log(`Vetor P e Vetor Q possuem conteúdo igual?
${possuiConteudoIgualJson(vetorP, vetorQ)}`); // Saída: true
console.log(`Vetor P e Vetor R possuem conteúdo igual?
${possuiConteudoIgualJson(vetorP, vetorR)}`); // Saída: false
```

Bloco de Exercícios 2

Exercício 01:

```
function processarmatriz() {
    const numerolinhas = 10;
    const numerocolunas = 15;
    const matriz = [];

    for (let i = 0; i < numerolinhas; i++) {
        matriz[i] = [];
        for (let j = 0; j < numerocolunas; j++) {
            matriz[i][j] = Math.floor(Math.random() * 100) + 1;
        }
    }

    console.log("Matriz Gerada:");
    for (let i = 0; i < numerolinhas; i++) {
        console.log(matriz[i].join(" "));
    }

    console.log("/n---n/");

    console.log("soma dos elementos de cada linha:");
    for (let i = 0; i < numerolinhas; i++) {
        let somaLinha = 0;
        for (let j = 0; j < numerocolunas; j++) {
            somaLinha += matriz[i][j];
        }
        console.log(`Soma da linha ${i + 1}: ${somaLinha}`);
    }
   const parity = rowsum % 2 === 0 ? "par" : "ímpar";
   console.log(`A soma da linha ${i + 1} é ${parity}.`);
}
console.log("/n---n/");
    console.log("soma dos elementos de cada coluna:");{
    for (let j = 0; j < numerocolunas; j++) {
        let somaColuna = 0;
    }
    constparity = colsum % 2 === 0 ? "par" : "ímpar";
    console.log(`A soma da coluna ${j + 1} é ${parity}.`);
}

processarmatriz();
```

Exercício 02:

```
function processarmatriz() {
    const numerolinhas = 50;
    const numerocolunas = 50;
    const matriz = [];
}

for (let i = 0; i < numerolinhas; i++) {
    matriz[i] = [];
    for (let j = 0; j < numerocolunas; j++) {
        matriz[i][j] = Math.floor(Math.random() * 100);
    }
}

console.log(diagonalPrincipal(matriz));

function diagonalPrincipal(matriz) {
    const diagonal = [];
    for (let i = 0; i < Math.min(matriz.length, matriz[0].length); i++) {
        diagonal.push(matriz[i][i]);
    }
    return diagonal;
}
```

Exercício 03:

```
function matrizTransposta() {
    const matrizoriginal = {};
    const linhaoriginal = 15;
    const colunaariginal = 15;
}

const matriztransposta = {};
for (let i = 0; i < colunaariginal; i++) {
    matriztransposta[i] = {};
    for (let j = 0; j < linhaoriginal; j++) {
        matriztransposta[i][j] = matrizoriginal[j][i];
    }
matrizoriginal [i][j] = i + colunaariginal + j + 1;
process.stdout.write(String(matrizoriginal[i][j]).padStart(4));
}
console.log();

const matriztransposta = {}; {
for (let j = 0; j < colunaariginal; j++)
    matrizTransposta[j] = {};
```

```
for (let i = 0; i < linhaoriginal; i++) {
   matrizTransposta[j][i] = matrizoriginal[i][j];
   }
}

console.log("Matriz transposta:");
for (let i = 0; i < matrizTransposta.length; i++) {
   for (let j = 0; j < matrizTransposta[i].length; j++)
   }
console.log();

gerarmatrizTransposta();
```

Exercício 04:

```
function multiplicarMatrizes(matrizA, matrizB) {
   const linhasA = matrizA.length;
   const colunasA = matrizA[0].length;
   const linhasB = matrizB.length;
   const colunasB = matrizB[0].length;
}
const matrizresultado = Array(linhasA).fill(0).map(() => Array(colunasB).fill(0));
   for (let i = 0; i < linhasA; i++) {
      for (let j = 0; j < colunasB; j++) {
         for (let k = 0; k < colunasA; k++) {
            matrizresultado[i][j] += matrizA[i][k] * matrizB[k][j];
         }
      }
   }

   return matrizresultado;
```

Exercício 05:

```
function somaMatrizes(matrizA, matrizB) {
   const linhasA = (1,2,3,4);
   const colunasA = (5,6,7,8);
   const linhasB = (9,10,11,12);
   const colunasB = (13,14,15,16);
   const resultado = [];
}
   for (let i = 0; i < linhasA.length; i++) {
      resultado[i] = [];
      for (let j = 0; j < colunasA.length; j++) {
         resultado[i][j] = matrizA[i][j] + matrizB[i][j];
      }
   }
   return resultado;
```

Exercício 06:

```
function criarMatriz(ordem) {
    let matriz = [];
    for (let i = 0; i < ordem; i++) {
        matriz[i] = [];
        for (let j = 0; j < ordem; j++) {
        matriz[i][j] = parseFloat(prompt(`Digite o elemento [${i + 1},${j + 1}] da matriz:`));
        }
    }
    return matriz;
}
function imprimirMatriz(matriz) {
    console.log("elementos da matriz:");
    for (let i = 0; i < matriz.length; i++) {
    console.log(matriz[i].join(" "));
    }
}
function somarQuadradosPrimeiraColuna(matriz) {
    let soma = 0;
    for (let i = 0; i < matriz.length; i++) {
        soma += matriz [i] [0] **2;
    }
    return soma;
}
function somaTerceiraLinha(matriz) {
    let soma = 0;
   if (matriz.length < 3) {
    for (let i = 0; i < matriz.length; i++) {
        soma += matriz [2] [j];
    }
}
    return soma;
}
function somarDiagonalPrincipal(matriz) {
    let soma = 0;
    for (let i = 0; i < matriz.length; i++) {
        soma += matriz[i][i];
    }
    return soma;
}
function somarElementosParSegundaLinha(matriz) {
    let soma = 0;
    if (matriz.length < 2) {
    for (let j = 0; j < matriz.length; j++) {
        if (matriz[1][j] % 2 === 0) {
            soma += matriz[1][j];
```

```
        }
    }
}
    return soma;
}
function exibirMenu() {
    console.log("Menu de Opções:");
    console.log("1. Somar os quadrados dos elementos da primeira coluna");
    console.log("2. Somar os elementos da terceira linha");
    console.log("3. Somar os elementos da diagonal principal");
    console.log("4. Somar os elementos pares da segunda linha");
    console.log("5. Sair");
}
function main() {
    const ordem = 4;
    const matriz = criarMatriz(ordem);
    imprimirMatriz(matriz);
    let opcao;
    do {
        exibirMenu();
        opcao = parseInt(prompt("Escolha uma opção:"));
        switch (opcao) {
            case 1:
                const somaQuadrados = somarQuadradosPrimeiraColuna(matriz);
                console.log(`Soma dos quadrados dos elementos da primeira coluna:
${somaQuadrados}`);
                break;
            case 2:
                const somaTerceira = somaTerceiraLinha(matriz);
                console.log(`Soma dos elementos da terceira linha: ${somaTerceira}`);
                break;
            case 3:
                const somaDiagonal = somarDiagonalPrincipal(matriz);
                console.log(`Soma dos elementos da diagonal principal: ${somaDiagonal}`);
                break;
            case 4:
                const somaPares = somarElementosParSegundaLinha(matriz);
                console.log(`Soma dos elementos pares da segunda linha: ${somaPares}`);
                break;
            case 5:
                console.log("Saindo...");
                break;
            default:
                console.log("Opção inválida. Tente novamente.");
        }
    } while (opcao !== 'fim');
}
```