**Luiz Fernando dos Santos Nascimento**

**Exercícios 23/08**

2.1. Efetue uma definição, através de enumeração, de todos os membros da sua família,

considerando apenas parentesco direto de pais, irmãos e filhos.

R: 1 Pai: Carlos Roberto do Nascimento; 2 Mãe: Nilza dos Santos Nascimento; 3 Irmã: Ana Paula dos Santos Nascimento; 4 Eu: Luiz Fernando dos Santos Nascimento.

2.2. Efetue uma definição, através de enumeração, de cursos existentes na instituição de ensino em que você está estudando.

R: 1: Desenvolvimento de software multiplataforma; 2 Geoprocessamento; 3 Meio-ambiente.

2.3. Efetue uma definição, através de enumeração, de números primos. Um número é dito

Primo se só possui como divisor ele mesmo e o número um.

R: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173...

2.4. Efetue uma definição ostensiva, de um colega de classe.

R:



2.5. Efetue uma definição recursiva de uma sequência numérica cujos elementos são obtidos

multiplicando-se, a partir do segundo elemento, o elemento anterior por 3. Considere que o

primeiro elemento vale 2.

R: 2 \* 3 = 6 \* 3 = 18 \* 3 = 54 \* 3 = 162 \* 3 = 486...

2.6. Utilizando linguagem de programação defina:

a) Uma enumeração para os meses do ano.

R:

 enum MesesDoAno {

  Janeiro=1,

  Fevereiro=2,

  Março=3,

  Abril=4,

  Maio=5,

  Junho=6,

  Julho=7,

  Agosto=8,

  Setembro=9,

  Outubro=10,

  Novembro=11,

  Dezembro=12,

}

console.log(MesesDoAno.Janeiro);

console.log(MesesDoAno.Junho);

b) Uma enumeração para os dias da semana.

R:

enum DiasDaSemana {

  Domingo=1,

  Segunda=2,

  Terça=3,

  Quarta=4,

  Quinta=5,

  Sexta=6,

  Sábado=7,

}

console.log(DiasDaSemana.Domingo);

console.log(DiasDaSemana.Quarta);

c) Uma função recursiva para o cálculo do fatorial de um número

R:

function fatorial(n: number): number {

  if (n == 0 || n == 1) {

    return 1;

  } else {

    return n \* fatorial(n - 1);

  }

}

d) Uma definição que corresponda a definição do tipo gênero-diferença para um uma pessoa

que estude em uma faculdade. Utilize uma linguagem que dê suporte a herança.

R:

class Pessoa {

    constructor(public nome: string, public idade: number) {}

}

class Estudante extends Pessoa {

    constructor(nome: string, idade: number, public matricula: string, public curso: string) {

        super(nome, idade);

    }

}

const estudante = new Estudante('Alice', 20, '12345', 'Ciência da Computação');

console.log(estudante.nome);

console.log(estudante.curso);

2.7. Um veículo possui a capacidade de se mover, expressa pela alteração na sua coordenada de longitude e latitude. Um veículo elétrico é um veículo que possui como fonte de energia primária a eletricidade (armazenada em uma bateria). Um veículo elétrico e voador é um veículo que também possui a capacidade de se mover na vertical, expressa pela alteração de sua altitude em relação ao solo. Represente um veículo elétrico e voador utilizando uma cadeia de herança. Defina o código-fonte representativo do modelo em um arquivo separado daquele que faz uso desse e, adicionalmente exemplifique o acesso e a modificação desses atributos através de chamada de suas operações.

R: Código grande demais para colar aqui, mas está no arquivo que roda no VSCode

2.8. O que acontece ao se executar uma chamada a uma função recursiva que chama a si

mesma um elevado número de vezes? Dê um exemplo utilizando o código-fonte da progressão

aritmética fornecido pelo professor. Faça um comparativo escrevendo um algoritmo e código

que sejam equivalentes ao recursivo em termos de entradas e saídas mas que utilizem iteração

ao invés de recursão. Qual sua conclusão?

R:  Uma função recursiva chama a si mesma várias vezes, consumindo espaço de memória. Se a quantidade de chamadas for muito grande, isso pode levar ao “Stack Overflow”, que é quando não há espaço o suficiente para se processar a chamada, causando mal funcionamento do programa.

Conclui-se portanto, que por vezes é melhor usar algoritmos iterativos, principalmente em casos onde há a possibilidade de muitas chamadas, de forma a evitar esse tipo de problema.

2.9. Uma progressão geométrica é uma sequência numérica onde cada elemento, a partir do

segundo, é obtido multiplicando-se o anterior por uma constante. Utilizando uma linguagem

de programação que dê suporte a orientação a objetos, defina uma progressão geométrica e

dê exemplo de geração de seus primeiros 50 termos.

R:

class GeometricProgression {

    constructor(private initialTerm: number, private ratio: number) {}

    generateTerms(numTerms: number): number[] {

        const terms: number[] = [this.initialTerm];

       for (let i = 1; i < numTerms; i++) {

            const previousTerm = terms[i - 1];

            const currentTerm = previousTerm \* this.ratio;

            terms.push(currentTerm);

        }

        return terms;

    }

}

const progression = new GeometricProgression(2, 3);

const first50Terms = progression.generateTerms(50);

console.log('Primeiros 50 termos da progressão geométrica:');

console.log(first50Terms.join(', '));

2.10. A sequência de Fibonacci é definida da seguinte forma: {1, 1, 2, 3, 5, 8, 13, ...}, ou seja,

para k>2, xk = xk-1 + xk-2. Utilizando uma linguagem de programação com suporte a orientação a

objetos, defina uma classe que modele a sequência de Fibonacci e exemplifique o cálculo de

alguns de seus termos. Ilustre a chamada recursiva e identifique chamadas repetidas a um

mesmo valor.

R:

class Fibonacci {

    private memo: Map<number, number> = new Map<number, number>();

    fibonacciRecursivo(n: number): number {

        if (n <= 2) {

            return 1;

        }

        if (this.memo.has(n)) {

            return this.memo.get(n)!;

        }

        const resultado = this.fibonacciRecursivo(n - 1) + this.fibonacciRecursivo(n - 2);

        this.memo.set(n, resultado);

        return resultado;

    }

    fibonacciIterativo(n: number): number {

        if (n <= 2) {

            return 1;

        }

        let anteriorAnterior = 1;

        let anterior = 1;

        let atual = 0;

        for (let i = 3; i <= n; i++) {

            atual = anterior + anteriorAnterior;

            anteriorAnterior = anterior;

            anterior = atual;

        }

        return atual;

    }

}

const fibonacci = new Fibonacci();

console.log("Calculando termos usando abordagem recursiva:");

console.log(fibonacci.fibonacciRecursivo(6));

console.log(fibonacci.fibonacciRecursivo(10));

console.log("Calculando termos usando abordagem iterativa:");

console.log(fibonacci.fibonacciIterativo(6));

console.log(fibonacci.fibonacciIterativo(10));