**Flávio Linguanotto**

**Juliana Maciel Manso**

2.1. Efetue uma definição, através de enumeração, de todos os membros da sua família, considerando apenas parentesco direto de pais, irmãos e filhos.

Enumeração Família\_Linguanotto:

Pai = 1

Mãe = 1

Filho1 = 1

Filho2 = 2

Fim-enumeração

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

Enumeração Cursos\_FATEC\_Jacareí:

Meio Ambiente e Recursos Hídricos = 1

Geoprocessamento = 1

Desenvolvimento de Software Multiplataforma = 1

Fim-enumeração

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.

Enumeração Números\_Primos:

Número natural = N

Número primo = N se, e somente se, N tem exatamente dois divisores: 1 e ele mesmo.

Fim-enumeração.

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

Ígor Fonseca ≝ 

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.

Primeiro termo da sequência numérica ≝ 2

Termos da sequência numérica ≝ 3 \* [n-1] para n > 1

Sequência numérica ≝ {6, 18, 54, 162, ...}

2.6. Utilizando linguagem de programação defina:

a) Uma enumeração para os meses do ano.

enum Months {

  January = 1,

  February,

  March,

  April,

  May,

  June,

  July,

  August,

  September,

  October,

  November,

  December

}

console.log("January:", Months.January);

console.log("February:", Months.February);

console.log("March:", Months.March);

console.log("April:", Months.April);

console.log("May:", Months.May);

console.log("June:", Months.June);

console.log("July:", Months.July);

console.log("August:", Months.August);

console.log("September:", Months.September);

console.log("October:", Months.October);

console.log("November:", Months.November);

console.log("December:", Months.December);

b) Uma enumeração para os dias da semana.

enum Days {

    Sunday = 1,

    Monday,

    Tuesday,

    Wednesday,

    Thursday,

    Friday,

    Saturday

  }

  console.log("Domingo:", Days.Sunday);

  console.log("Segunda-feira:", Days.Monday);

  console.log("Terça-feira:", Days.Tuesday);

  console.log("Quarta-feira:", Days.Wednesday);

  console.log("Quinta-feira:", Days.Thursday);

  console.log("Sexta-feira:", Days.Friday);

  console.log("Sábado:", Days.Saturday);

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

function factorial(n: number): number {

  if (n == 1) {

      return 1;

  } else {

      return n \* factorial(n - 1);

  }

}

console.log(factorial(3)); // Exemplo com fatorial de 3

console.log(factorial(5)); // Exemplo com fatorial de 5

d) Uma definição que corresponda a definição do tipo gênero-diferença para uma pessoa que estude em uma faculdade. Utilize uma linguagem que dê suporte a herança.

class Pessoa {

  nome: string;

  idade: number;

  genero: string;

  constructor(nome: string, idade: number, genero: string) {

      this.nome = nome;

      this.idade = idade;

      this.genero = genero;

  }

  estudar() {

      console.log(`${this.nome} está estudando.`);

  }

}

class Estudante extends Pessoa {

  matricula: number;

  curso: string;

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

      super(nome, idade, genero);

      this.matricula = matricula;

      this.curso = curso;

  }

  apresentar() {

      console.log(`Nome: ${this.nome}, Idade: ${this.idade}, Gênero: ${this.genero}, Matrícula: ${this.matricula}, Curso: ${this.curso}`);

  }

}

const estudante1 = new Estudante("Juliana", 35, "Feminino", 12345, "Desenvolvimento de Software Multiplataforma");

estudante1.estudar();

estudante1.apresentar();

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.

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?

Quando você executa uma função recursiva que chama a si mesma um grande número de vezes, pode ocorrer um estouro de pilha, conhecido como "stack overflow". Isso acontece porque cada chamada recursiva empilha um novo quadro de pilha na memória, e quando você tem um grande número de chamadas recursivas, a pilha de chamadas fica muito profunda, consumindo todo o espaço disponível na pilha. Isso resulta em um erro e encerra a execução do programa. Conclui-se que o código iterativo é mais eficiente em termos de uso de memória e é capaz de lidar com números maiores sem causar estouros de pilha.

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.

class ProgressaoGeometrica {

    primeiroTermo: number;

    razao: number;

    constructor(primeiroTermo: number, razao: number) {

        this.primeiroTermo = primeiroTermo;

        this.razao = razao;

    }

    gerarTermos(qtdTermos: number): number[] {

        const termos: number[] = [this.primeiroTermo];

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

            const termoAnterior = termos[i - 1];

            const proximoTermo = termoAnterior \* this.razao;

            termos.push(proximoTermo);

        }

        return termos;

    }

}

const pg = new ProgressaoGeometrica(1, 2);

const primeiros50Termos = pg.gerarTermos(50);

console.log(primeiros50Termos);

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

class FibonacciSequence {

    private memo: Map<number, number>;

    constructor() {

        this.memo = new Map<number, number>();

        this.memo.set(1, 1);

        this.memo.set(2, 1);

    }

    calculateTerm(n: number): number {

        if (this.memo.has(n)) {

            // Se o valor já foi calculado antes, retorná-lo diretamente

            console.log(`Valor de Fibonacci(${n}) já calculado.`);

            return this.memo.get(n)!;

        }

        const term = this.calculateTerm(n - 1) + this.calculateTerm(n - 2);

        this.memo.set(n, term);

        return term;

    }

}

const fibonacci = new FibonacciSequence();

console.log("Termo 5:", fibonacci.calculateTerm(5));

console.log("Termo 6:", fibonacci.calculateTerm(6));

console.log("Termo 7:", fibonacci.calculateTerm(7));

console.log("Termo 8:", fibonacci.calculateTerm(8));

console.log("Termo 9:", fibonacci.calculateTerm(9));