**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);

  }

}

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:

class Veiculo {

  public longitude: number = 0;

  public latitude: number = 0;

  public mover(coordenadaX: number, coordenadaY: number) {

    this.longitude += coordenadaX;

    this.latitude += coordenadaY;

  }

}

class VeiculoEletrico extends Veiculo {

  public bateria: number = 100;

class VeiculoVoador extends VeiculoEletrico {

  public altitude: number = 0;

  public decolar() {

    this.altitude += 1000;

  }

  public pousar() {

    this.altitude -= 1000;

  }

}

import { VeiculoVoador } from "./modelo";

const meuVeiculo = new VeiculoVoador();

meuVeiculo.longitude = -40;

meuVeiculo.latitude = -20;

meuVeiculo.bateria = 100;

meuVeiculo.altitude = 0;

meuVeiculo.mover(10, 20);

meuVeiculo.acelerar();

meuVeiculo.decolar();

meuVeiculo.pousar();

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.