EEL670 - Linguagens de Programação
Departamento de Engenharia Eletrônica e de Computação - DEL-Poli/UFRJ
Prof.: Miguel Elias Mitre Campista
== Laboratório 2

---

1) Lista de exercícios: Resolva os exercícios abaixo como se pede.

a) Escreva um programa que calcule o volume de um paralelepípedo reto. Para isso, o usuário deverá passar as dimensões do paralelepípedo para um objeto da classe `Paralelepipedo` através do construtor da classe. Esse construtor inicializa as dimensões do paralelepípedo através de métodos do tipo set para cada dimensão x, y e z. Note que os argumentos usados na inicialização das dimensões não podem ser menores ou iguais a zero. Ainda, que as dimensões x, y e z são atributos privados da classe. Caso um valor menor ou igual a zero seja passado, o método set correspondente deve inicializar o atributo com um valor unitário padrão e exibir uma mensagem de erro na tela

Como requisito adicional do programa, os métodos set devem ser públicos para que seja possível alterar as dimensões do paralelepípedo. A classe `Paralelepípedo` ainda oferece um método do tipo get para recuperar o volume calculado. Assim, é necessária a definição de um método privado `computeVolume` a ser invocado sempre que o método `getVolume` for invocado. Escreva o programa contendo um arquivo para a função principal, um arquivo .h para definição da classe `Paralelepipedo` e um .cpp para implementação dos métodos da mesma classe.

b) Escreva um programa para calcular a distância de dois pontos no espaço. Para isso, o programa deve implementar a classe `Linha` e a classe `Ponto`. A primeira classe possui como atributos privados dois objetos da classe `Ponto`, além de métodos públicos do tipo set para inicialização (e atualização) de cada um dos atributos através de atribuição do tipo `ponto = p`. A classe `Linha` possui ainda um método público chamado `getComprimento`, que calcula e retorna a distância entre os dois pontos, e um construtor para inicialização dos atributos.

A classe `Ponto`, por outro lado, possui um construtor com argumentos padrão, utilizado para inicialização das coordenadas x, y e z. Além do construtor, a classe `Ponto` possui ainda métodos do tipo set e get para cada uma das coordenadas.

Há ainda uma função global chamada `printCoordenadas` que imprime as coordenadas de um objeto da classe `Ponto` passado como argumento. Essa função escreve na tela a seguinte mensagem, assumindo que seja a primeira impressão do ponto `p` com coordenadas `(1.0, 2.0, 3.0)`:

`"[Impressao no. 1] : Coordenadas de p (1.0, 2.0, 3.0)"`

Note que o número de vezes que a função `printCoordenadas` é chamada é conhecida através de uma variável estática local à função.

== Respostas da Lista de Exercícios

1)

a)

```
/*************************************************************************/
/************************** Programa Principal ***************************/

#include <iostream>
#include "paralelepipedo.h"

/* Programa do Laboratório 2:
   Cálculo do volume de um paralelepípedo reto.
   Autor: Miguel Campista */

using namespace std;

int main () {
    double dx = 1.1, dy = 2.2, dz = 3.3;

    Paralelepipedo paralelepipedo (dx, dy, dz);

    cout << "\n\nO volume eh: " << paralelepipedo.getVolume () << endl;

    cout << "\n\nmudando os valores das dimensoes..." << endl;

    paralelepipedo.setDimX (-1);
    paralelepipedo.setDimY (3.3);
    paralelepipedo.setDimZ (4.4);

    cout << "O NOVO volume eh: " << paralelepipedo.getVolume () << endl;

    return 0;
}
/*************************************************************************/
/*********************** Arquivo paralelepipedo.h ************************/
#include <iostream>

using namespace std;

class Paralelepipedo {
    public:
            Paralelepipedo (double, double, double);

            void setDimX (double);
            void setDimY (double);
            void setDimZ (double);

            double getVolume ();

    private:
            double dimX, dimY, dimZ;

            double computeVolume ();
};
/*************************************************************************/
/*********************** Arquivo paralelepipedo.cpp **********************/

#include "paralelepipedo.h"

Paralelepipedo::Paralelepipedo (double x, double y, double z) {
    setDimX (x); setDimY (y); setDimZ (z);
}

void Paralelepipedo::setDimX (double x) {
    if (x > 0) dimX = x;
    else dimX = 1;
}

void Paralelepipedo::setDimY (double y) {
    if (y > 0) dimY = y;
```

```
    else dimY = 1;
}

void Paralelepipedo::setDimZ (double z) {
    if (z > 0) dimZ = z;
    else dimZ = 1;
}

double Paralelepipedo::getVolume () {
    return computeVolume ();
}

double Paralelepipedo::computeVolume () {
    return dimX * dimY * dimZ;
}
/*****************************************************************************/
```

b)

```
/*****************************************************************************/
/*************************** Programa Principal ******************************/

#include <iostream>

// include ponto.h já foi realizado em linha.h
//#include "ponto.h"
#include "linha.h"

/* Programa do Laboratório 2:
   Cálculo do comprimento de linhas usando composição entre classes.
   Autor: Miguel Campista */

using namespace std;

void printCoordenadas (Ponto p) {
    // Variável local estática conta o número de vezes que a função foi chamada
    static int count = 1;

    cout << "[Impressao no. " << count
            << "] : Coordenadas de p (" << p.getCoordX ()
            << ", " << p.getCoordY ()
            << ", " << p.getCoordZ ()
            << ")"  << endl;
    count++;
}

int main () {
    Ponto p1 (2, 2, 1);
    Ponto p2; // Construtor da classe Ponto com argumentos padrão (1.0, 1.0, 1.0)
    Linha linha (p1, p2);

    printCoordenadas (p1);
    printCoordenadas (p2);

    cout << "== O comprimento da linha eh: " << linha.getComprimento () << endl;

    cout << "\nNovas coordenadas para p2...\n" << endl;
    p2.setCoordX(2); // Método setCoordX atualiza a coordenada X do ponto p2
    printCoordenadas (p2);

    linha.setP2 (p2); // Método setP2 atualiza o ponto p2 da linha

    cout << "== O NOVO comprimento da linha eh: " << linha.getComprimento () << endl;

    return 0;
}
/*****************************************************************************/
/*************************** Arquivo ponto.h *********************************/

using namespace std;

class Ponto {
    public:
            Ponto (double = 1.0, double = 1.0, double = 1.0);

            double getCoordX ();
```

```
            double getCoordY ();
            double getCoordZ ();

            void setCoordX (double);
            void setCoordY (double);
            void setCoordZ (double);

    private:
            double coordX, coordY, coordZ;
};
/*****************************************************************************/
/***************************** Arquivo ponto.cpp *****************************/

#include "ponto.h"

Ponto::Ponto (double x, double y, double z) {
    coordX = x; coordY = y; coordZ = z;
}

double Ponto::getCoordX () { return coordX; }
double Ponto::getCoordY () { return coordY; }
double Ponto::getCoordZ () { return coordZ; }

void Ponto::setCoordX (double x) { coordX = x; }
void Ponto::setCoordY (double y) { coordY = y; }
void Ponto::setCoordZ (double z) { coordZ = z; }
/*****************************************************************************/
/***************************** Arquivo linha.h *******************************/

#include "ponto.h"
#include <cmath>

using namespace std;

class Linha {
    public:
            Linha (Ponto, Ponto);

            void setP1 (Ponto);
            void setP2 (Ponto);

            double getComprimento ();

    private:
            Ponto p1, p2;
};
/*****************************************************************************/
/***************************** Arquivo linha.cpp *****************************/

#include "linha.h"

Linha::Linha (Ponto p1_, Ponto p2_) {
    p1 = p1_; p2 = p2_;
}

void Linha::setP1 (Ponto p) {
    p1 = p;
}
void Linha::setP2 (Ponto p) {
    p2 = p;
}

double Linha::getComprimento () {
    return sqrt (pow (p1.getCoordX() - p2.getCoordX(), 2) +
                 pow (p1.getCoordY() - p2.getCoordY(), 2) +
                 pow (p1.getCoordZ() - p2.getCoordZ(), 2));
}
/*****************************************************************************/
```