ELE078 - Programação Orientada a Objetos

Atividade Prática 07 - Herança e Composição de Classes ¶

Exercício 1

Escreva uma classe chamada Ponto3D capaz de manipular objetos com coordenadas cartesianas nos eixos x, y e z (tridimensionais). Para a implementação desta classe Ponto3D, vc deve fazer reuso do código da classe Ponto2D, cujo código é fornecido logo em seguida.

In [ ]:

```
#include<iostream>

class Ponto2D{

     public:
        Ponto2D(int xx = 0.0, int yy = 0.0):x(xx),y(yy){ };
        friend ostream& operator<< (ostream &op, const Ponto2D &p);
        Ponto2D& operator= (const Ponto2D &p);
        ~Ponto2D(){};
        double get_x(void) { return x; }
        double get_y(void) { return y; }
        void set (double nx, double ny) { x=nx; y=ny; }

    private:
        double x;
        double y;
};

ostream& operator<< (ostream &op, const Ponto2D &p){
            op << endl;
            op << "x = " << p.x << endl;
            op << "y = " << p.y << endl;
            return op;
}

Ponto2D& Ponto2D::operator= (const Ponto2D &p){
    x = p.x;
    y = p.y;
    return *this;
}
```

In [ ]:

```
// a ser implementada

class Ponto3D: public Ponto2D{

    public:
        Ponto3D(double xx = 0, double yy = 0, double zz=0);
        friend ostream& operator<< (ostream &op, const Ponto3D &p);
        Ponto3D& operator= (const Ponto2D &p);
        void set(double nx = 0, double ny = 0, double nz=0);
        double get_z();


    private:
        double z;
};


// código para teste da classe Ponto3D

int main()
{
    Ponto2D p1(3,4), p2;
    p2.set(2,1.5);
    cout << p1 << endl;
    cout << p2 << endl;

    p2 = p1;
    cout << p2 << endl;


    Ponto3D p3(2,4.5,5), p4;
    p4.set(1,0.3,12);
    cout << p3 << endl;
    cout << p4 << endl;

    p4 = p3;
    cout << p4 << endl;

    p4 = p1;
    cout << p4 << endl;

    return 0;
}
```

Exercício 2:

Crie uma classe Box definida como um paralelepípedo retangular, uma figura tridimensional formada por seis paralelogramos. Os atributos de um objeto Box são largura, altura e profundidade. Defina funções membro para o cálculo da área e do volume da Box. Crie pelo menos um construtor de forma que seja possível inicializar um objeto Box a partir das coordenadas de seus vértices, ou seja, objetos do tipo Ponto 3D.

Exercício 3:

Crie duas classes chamadas *Traveler* e *Pager* sem construtores default, mas com construtores que recebem uma string como parâmetro e copiam para uma variável interna. Para cada classe, escreva o construtor de cópia e o operador de atribuição. Crie uma classe chamada BusinessTraveler derivada de Traveler e insira um membro da classe Pager. Para essa classe crie:

- Construtor default que inicializa ambos os strings de Traveler e Pager com “1“
- Construtor que recebe uma string e inicializa ambos os strings de Traveler e Pager com o string recebido
- Construtor de cópia
- Operador de atribuição

Exercício 4:

Considere o código a seguir, responda as seguintes questões e justifique sua resposta:

In [ ]:

```
class Base{

    int i;

    protected:
        int read() const { return i; }
        void set(int ii) { i = ii; }

    public:
        Base(int ii = 0) : i(ii) {}
        int value(int m) const { return m*i; }
};


class Derived : public Base{

        int j;
        public:
            Derived(int jj = 0) : j(jj) {}
            void change(int x) { set(x); }
};
```

- É possível adicionar uma função membro na classe Derived que chama a função read()?
- Alterando a herança para private, ainda assim é possível adicionar uma função membro na classe Derived que chama a função read()?
- É possível chamar a função read() a partir de um objeto do tipo Derived?
- Modifique o código de forma que a classe Derived use herança protected. Crie uma classe Derived2 que seja derivada da classe Derived utilizando herança public. É possível chamar read() a partir de uma função membro da classe Derived2? E o método value()?