ELE078 - Programação Orientada a Objetos

Atividade Prática 03

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 [ ]:

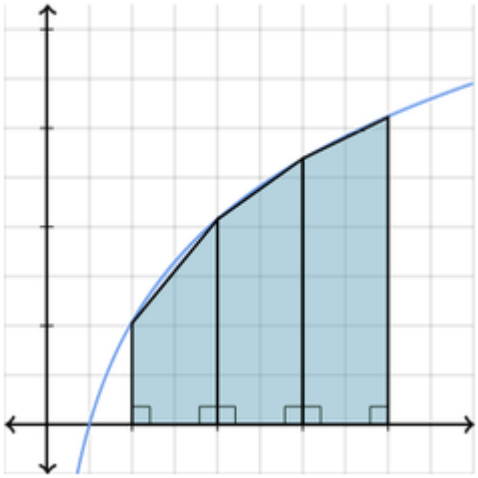
```
// a ser implementada

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 método *getIntegral()* que é capaz de obter a integral de uma função $y = f(x)$ via regra do trapézio. Essa função deve ser polimórfica podendo então mudar seu comportamento dependendo se o objeto apontado é do tipo função quadrática, senoide, linear, entre outros. Para relembrar, o método do Trapézio (ou soma Trapezoidal) é um técnica de integração numérica que busca aproximar o valor da integral a partir do somatório da área dos trapézios que dividem a área total da função.

Como mostrado na Figura a seguir, a integral da função:

$$\int_a^b 3ln(x)\,dx$$



Essa divisão pode ser arbitrária e quanto maior, mais próximo será do valor real da integral. A implementação desse cálculo de integral deve considerar no mínimo três tipos de funções:

- **Quadratica**: $ax^2 + bx + c$
- **Senoidal**: $\frac{sin(x)}{x}$
- **Linear**: $ax + b$

In [ ]:

```
// codigo exemplo

#include <iostream>
#include <cmath>
using namespace std;

class Funcao
{
    public:

    //funcao que obtem a integral da funcao pela regra do trapezio
    double getIntegral(double limiteInferior, double limiteSuperior,
double intervalos);

    // funcao virtual representando a funcao cuja integral deve ser
calculada
    virtual double func(double input) = 0;

    // destrutor
    //virtual ~Funcao(){}

};

...

int main()
{
    double resultado;

    //cria um container de ponteiros do tipo Funcao
    Funcao *f[3];

    f[0] = new Quadratica(1,2,4);
    f[1] = new Senoide();
    f[2] = new Linear(1,4);

    cout << "*** Calculo de integrais usando a regra do trapezio: **
*"<<endl<<endl;
    cout << "*** Funcoes ***" << endl;
    cout << "(1) x^2 + 2x + 4" << endl;
    cout << "(2) sen(x) / x" << endl;
    cout << "(3) x + 4" << endl;
    cout << endl;

    for (int i=0; i<3; i++)
    {
        resultado = f[i]->getIntegral(1,4,1000);
        cout << "Integral da Funcao (" << i+1 << "): " << resultado;
        cout << endl;
    }

    for (int i=0; i<3; i++)
    {
        delete f[i];
    }
```

```
    return 0;
}
```