```
// Programação Orientada a Objectos 2020/2021
// Ana Rita Videira - 5012012218

//Exame 2020 - Época Normal - Parte 2 - Exercicio 1, 2 e 3


#include <stdio.h>
#include <vector>
#include <string>
#include <sstream>
#include <iostream>

using namespace std;

/*

 A classe Fig2D tem com objetivo armazenar informação sobre figuras geometricas
 com duas dimensoes (quadrados, trapézios, triangulos, pentágonos, circulos,
 retangulos, entre outros). Os seus atributos permitem armazenar a cor da figura
 e a data de criação.

 A classe Fig2D servirá de base para as classes derivadas que representam figuras
 concretas. Ao responder ás alineas, tenha em consideração o seguinte:

 i. Todos os objectos sao criados indicando obrigatoriamente todos os seus atributos.

 ii. Pode adicinoar novos métodos às classes já escrita, mas não pode eliminar
 ou alterar o que já se encontra nas classes Data e Fig2D.

 iii. Todas as classes do problema ( as já disponibilizadas e as que vai escrever
 devem poder ser utilizadas em todas as situações habituais.

 1 / a) Implemente 2 classes derivadas de Fig2D. Cada uma dessas classes deve
 corresponder a uma figura geometrica correta.

 2 / b) Exemplifique com codigo como poderia criar 2 objectos dinâmicos das
 classes derivadas implementadas na alinea a)

 3 / c) Cria uma classe Desenho que contanha como unico atributo um contentor STL
 de figuras 2D (inicialmente vazio). Adicione os seguintes métodos a esta classe:

    i. Adicionar uma nova figura ao Desenho, garantindo que as figuras ficam
    ordenadas no contantor de acordo com um determinado critério. O criterio fica à
    sua escolha e deve referi-lo explicitamente, indicando igualemento como resolverá
     eventuais empates;
    ii. Implementar uma operacao polimorfica, à sua escolha, envolvendo todas as
    figuras armazenadas no contentor. Refira explicitamente qual o objetivo do método
    que implementar.
    iii. Os objetos da classe Desenho tomam posse das figuras 2D que são armzenadas
    no contentor. Adicione o que for neessário à sua classe para garantir a correção
    do código.

 */

class Data{
    int dia, mes,ano;
public:
    Data(int a, int b, int c): dia(a), mes(b), ano(c){ };
};

class Fig2D{
    const Data criacao;
    string cor;

public:

    Fig2D( Data a, string c):criacao(a),cor(c){};    // alinea a

    virtual float area() const=0;          // calculo da area da figura;
    virtual float perimetro() const=0;     // calcuo do perimetro da figura
    void mudaCor(string a){ cor=a; };
    Data getCriacao() const { return criacao; };
    virtual Fig2D * clone()const = 0;
```

```
};


// alinea a

class Rectangulo : public Fig2D {
    float ladoMaior;
    float ladoMenor;

public:

    Rectangulo(float ma, float m, Data d, string c) :ladoMaior(ma),ladoMenor(m),
    Fig2D(d, c) {
          cout << "\nConstuiu Rectangulo " << c << endl;
    };

    float area() const {
        return ladoMaior*ladoMenor;
    };

    float perimetro() const {
        return 2 * ladoMaior + 2 * ladoMenor;
    };

    void setLados(int maior, int menor){
        ladoMaior=maior;
        ladoMenor=menor;
    };

    float getLadoMaior() const{
        return ladoMaior;
    };
    float getLadoMenor() const{
        return ladoMenor;
    };

    Fig2D * clone() const{
        return new Rectangulo(*this);
    };



};


class Quadrado : public Fig2D{
    float lado;

public:
     Quadrado (float l,Data d, string c) : lado(l), Fig2D(d,c) {
         cout << "\nConstuiu Quadrado " << c << endl;
     };


     float area() const {
         return lado*lado;
     };
     float perimetro() const {
         return 4*lado;
     };

     void setLado(int l){
         lado=l;
     };

     float getLado() const{
         return lado;
     };

    Fig2D * clone() const {
        return new Quadrado(*this);
    };

```

```
};


//alinea c
#include <bits/stdc++.h>

class Desenho {
private:
    vector < Fig2D * > vf;

    void ordenaVector() {  // ordena vetor por tamanho de areas

      //   std::sort(s.begin(), s.end(), [](int a, int b) {
       //         return a > b; } );

         sort( vf.begin(), vf.end(), [](Fig2D * a, Fig2D * b)
         {
             float aa = a->area();
             float ab = b->area();

             if(aa = ab)
                 return a < b;
             else
                 return a > b;  }
         );

    };

public:

    Desenho() {
        cout << "\nConstruiu Desenho" << endl;
    };

    Desenho(const Desenho &d){
        *this = d;
         cout << "\nConstruiu Desenho por Copia" << endl;
    };

    Desenho & operator=(const Desenho & orig) { // operador atribuicao
        if (this == &orig)  // se na memória já são o mesmo
            return *this;   // então já são iguais

        //Devolde os recursos ocupados pelo obejtcto alvo da atribuição
        for (auto x : this->vf) {
            delete x;
        }
        this->vf.clear();

        //Cria recursos iguais / copia os do outro objecto
        for (auto x : orig.vf) {
           // this->vf.push_back(new Fig2D*);  //CloneMe() é metodo virtual de Fig2D
        }
        return *this;
    };



    ~Desenho(){
        for(const Fig2D * x : this->vf )
            delete x;
        this->vf.clear();
    };

    void addFigura(const Fig2D &nova){
        vf.push_back(nova.clone());
        ordenaVector();
    };

    float totalAreas() const {
        float contador = 0;

        for (const Fig2D * x : this->vf) {
```

```
            contador += x->area();
        }
        return contador;
    };


    float totalPerimetros() const {
    double contador = 0;

    for (const Fig2D * x : this->vf) {
        contador += x->perimetro();
    }
    return contador;
    };


};



int main(int argc, char** argv) {


    // testando alinea a
    Data a(24,1,1996);

    Quadrado q(5,a,"rosa");
    cout << "\nQuadrado: " << q.area() << endl;
    Rectangulo r(2,3,a,"azul");
    cout << "\nRectangulo: " << r.area() << endl;


    // alinea b
    Data d(25,1,1996);
    Rectangulo* e = new Rectangulo(2,3,a,"vermelho");
    Quadrado* u = new Quadrado(2,a,"amarelo");


    //testando alinea c
    Desenho da;
    da.addFigura(q);
    Desenho db = da;




    return 0;
}
```