Algoritmos e Estruturas de Dados I

Lista 10: Registros

Nome:_____

Parte I: Pontos

1. Implemente a seguinte estrutura de dados em C++ que representa pontos em um espaço euclidiano:

```
struct Ponto {
  float x;
  float y;


  float distancia(Ponto& p);

  void atribuir(float a, float b);
};
```

2. Implemente o método **float Ponto::distancia(Ponto& p)** que calcula a distância entre o ponto corrente (aquele que chama o método) e o ponto **p** passado como parâmetro.

3. Implemente o método **void Ponto::atribuir(float a, float b)** que atribui respectivamente os valores **a** e **b** as coordenadas **x** e **y** do ponto corrente.

Parte II: Triângulos**.**

4. Utilizando o tipo de dados Ponto, defina um tipo de dado **Triangulo**, que representa um triângulo num espaço euclidiano.

5. Implemente um método **float Triangulo::perimetro()** que retorna o perímetro do triângulo corrente (aquele que chama o método).

6. Implemente um método **float Triangulo::area()** que retorna a área do triângulo corrente.

7. Implemente um método **bool Triangulo::equilatero()** que testa se o triângulo corrente é equilátero.

8. Implemente um método **bool Triangulo::semelhante(Triangulo& t)** que testa se o triângulo **t** é semelhante ao triângulo corrente.

Parte III: Retângulos**.**

9. Utilizando o tipo de dados Ponto, defina um tipo de dado **Retangulo**, que representa um retângulo num espaço euclidiano.

10. Implemente um método **float Retangulo::perimetro()** que retorna o perímetro do retângulo corrente (aquele que chama o método).

11. Implemente um método **float Retangulo::area()** que retorna a área do retângulo corrente.

12. Implemente um método **bool Retangulo::quadrado()** que testa se o retângulo corrente é quadrado.

Parte IV: Circunferência**.**

13. Defina o tipo de dados **Circunferencia**. Pense bem em quais os dados que definem uma circunferência.

14. Implemente um método **float Circunferencia::perimetro()** que retorna o perímetro da circunferência corrente (aquele que chama o método).

15. Implemente um método **float Circunferencia::area()** que retorna a área da circunferência corrente.

16. Implemente um método **bool Circunferencia::contem(Ponto& p)** que testa se o ponto **p** está dentro da circunferência corrente.

17. Implemente um método **bool Circunferencia::contem(Triangulo& p)** que testa se o triângulo **t** está dentro da circunferência corrente.

18. Implemente um método **bool Circunferencia::contem(Retangulo& r)** que testa se o retângulo **r** está dentro da circunferência corrente.

19. Implemente um método **bool Circunferencia::pertence(Ponto& p)** que testa se o ponto p está na linha definida pela circunferência corrente.

20. Implemente um método **bool Circunferencia::circunscrita(Triangulo& t)** que testa se a circunferência corrente é circunscrita ao triângulo **t**.

21. Implemente um método **bool Circunferencia::circunscrita(Retangulo& r)** que testa se a circunferência corrente é circunscrita ao retângulo **r**.