Lista de exercícios – Classes

Professor: Adelardo Adelino Dantas de Medeiros

1) Desenvolva classes que possam representar pontos e retângulos no espaço 2D, de acordo com as seguintes especificações:
   - Defina o tipo `Ponto` para representar um ponto em coordenadas cartesianas.
   - Crie uma função `imprimir`, capaz de imprimir um `Ponto` na tela.
   - Sobrecarregue o operador << usando a função `imprimir`.
   - Forneça uma implementação para uma função `distancia` que calcula a distância Euclidiana entre dois dados do tipo `Ponto`.
   - Defina um tipo `Retangulo` para representar retângulos que são paralelos aos eixos no sistema de coordenadas cartesianas. Represente um retângulo pelos seus pontos superior direito (ponto de coordenadas máximas) e inferior esquerdo (coordenadas mínimas), usando o tipo `Ponto`.
   - Escreva a função `area` que calcula a área de um `Retangulo`.
   - Escreva uma função `pontoInterno` que retorna o resultado do teste (true ou false) sobre se um `Ponto` está ou não dentro de um `Retangulo`.
   - Sobrecarregue o operador < usando a função `pontoInterno`.

2) Construa uma classe capaz de manipular números racionais (frações de dois inteiros). A classe deve ter os construtores apropriados (por cópia, default e específico, a partir de dois inteiros) e permitir as operações usuais (soma, multiplicação, etc.) tanto entre racionais quanto entre racionais e números inteiros. Preveja também a sobrecarga dos operadores de entrada e saída. OTIMIZAÇÃO OPCIONAL: Os números racionais devem estar sempre na sua forma mais simplificada.

3) Programe uma classe capaz de armazenar e manipular vetores de números em ponto flutuante de tamanho arbitrário. A classe deve ter os construtores apropriados e permitir as operações usuais (soma, produto escalar, produto vetorial, norma euclidiana, etc.) tanto entre vetores quanto entre vetores e escalares. Preveja também a sobrecarga dos operadores de entrada e saída. A classe deve permitir o uso do operador de acesso aleatório (colchete), ou seja, devem ser válidas expressões como:

```
Vetor x(8);

cout << x[2];
x[3] = 5.0;
```

4) Defina uma classe que manipule matrizes, usando a classe de vetores definida no exemplo anterior (uma matriz é um array de vetores). Além das operações usuais entre matrizes e entre matrizes e escalares, preveja o produto de matriz por vetor e a transposta de matrizes.

5) Crie uma classe capaz de armazenar e manipular vetores de booleanos de tamanho arbitrário. A classe deve permitir o uso do operador de acesso aleatório (colchete). Quanto ao armazenamento interno, a classe deve ser otimizada no sentido de que cada booleano deve ocupar apenas um bit, e não um byte inteiro. Contudo, esta otimização não deve ser perceptível pelo usuário, que manipulará os dados da classe como se eles fossem booleanos convencionais (bytes).