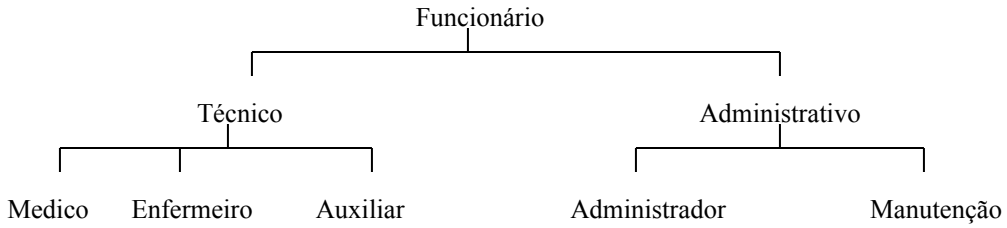
Lista de Exercícios de Programação III

Um Hospital de Vitória te contratou para desenvolver um protótipo de um sistema de informação que cadastre os funcionários e salas do hospital e gere alguns relatórios.

1. Considere que durante a especificação do sistema foram identificadas as seguintes categorias de funcionários classificadas na hierarquia abaixo:



Considere também que:

- Todo funcionário possui um código identificador único (um número inteiro), um nome (uma string), um salário (um número de ponto flutuante) e uma data de contratação.
- Todo técnico possui um atributo representando o número de anos que exerce sua atividade profissional (um número inteiro).
- Todo médico possui uma especialidade (uma string) e um identificador único (um número inteiro) no Conselho Regional de Medicina (CRM). São exemplos de especialidades: “otorrinolaringologia”, “pediatria”, “cirurgia geral”, etc.

Com base nessas informações, implemente as classes *Funcionario*, *Tecnico* e *Medico*. Implemente também uma classe *Data* com atributos específicos para dia, mês e ano. Garanta que todas subclasses concretas de *Funcionario* implementem obrigatoriamente os métodos de leitura de dados da entrada padrão e de impressão dos dados na saída padrão.

2. Implemente uma classe *Sala* que possui como atributos o seu número (um número inteiro), o código identificador do funcionário responsável e sua disponibilidade (um booleano).

3. Implemente uma classe *Lista* genérica na qual os elementos da lista são armazenados em um vetor de ponteiros para os elementos.

4. Considerando que todas as classes da hierarquia foram implementadas (as que você não implementou na primeira questão foram implementadas por outro programador), utilize-as juntamente com a classe *Lista* para implementar um programa que:

   - leia da entrada padrão o número total de funcionários do hospital
   - leia da entrada padrão os dados de cada funcionário do hospital
   - leia da entrada padrão o número total de salas
   - leia da entrada padrão os dados de cada sala do hospital
   - imprima os dados de todas as salas do hospital ordenadas crescentemente por seu número
   - imprima os dados de todos os médicos do hospital cujo especialidade é “cirurgia plastica” e que foram contratados nos últimos três anos a partir da data de hoje.
   - imprima os dados de todas as salas ocupadas do hospital que têm médicos como funcionário responsável
   - imprima os dados de todos os funcionários marajás do hospital ordenados decrescentemente por salário (um marajá tem salário superior a 10.000 reais)
   - imprima os dados de todos os médicos pediatras ordenados crescentemente por anos de atividade

5. Considere as seguintes definições de funções e classes em C++:

```
template <class T >
T xpto (T x, T y) {
  return y;
}

class tdata {
        int d, m, a;
}
```

```
template <class T,
class U>
U ypto (T x, U y) {
  return y;
}

class thorario {
        int h, m, s;
}
```

```
template <class T,
class U, class V>
V zpto (T x, U y) {
  return ((V) y);
}

class tdimensao {
        int h, l, w;
}
```

Indique quais das linhas de código de *main* abaixo são legais e explique as que não são.

```
main () {
        tdata a;
        thorario b;
        tdimensao c;
        a = xpto (a, a);
        b = xpto (a, a);
        c = xpto (a, b);
        a = ypto (a, a);
        b = ypto (a, a);
        b = ypto (a, b);
        c = ypto (a, b);
        a = zpto (a, a);
        b = zpto (a, a);
        b = zpto (a, b);
        c = zpto (a, b);
}
```

6. Considere as classes *Ponto* e *Linha* apresentadas abaixo. (a) Implemente o método construtor de cópia e o método que sobrecarrega o operador de atribuição da classe *Linha*. (b) Explique porque é importante implementar esses métodos de *Linha* em C++. (c) Explique porque esses métodos não precisam ser implementados na classe *Ponto*.

```
class Ponto {
        int x, y;
public:
        Ponto (int x, int y);
}

class Linha {
        Ponto *o, *d;
public:
        Linha (Ponto *or, Ponto *dt);
        Linha (const Linha&l);
        Linha operator=(const Linha &l) const;
}
```

Uma firma de Engenharia te contratou para desenvolver um protótipo em C++ de um sistema de desenho auxiliado por computador no plano cartesiano bidimensional. Para implementar esse protótipo você deve atentar para os aspectos de uma boa programação orientada a objetos (em especial, ao ocultamento de informação) e deve:

7. Criar uma classe *Ponto* representada por duas coordenadas inteiras.
8. Criar uma classe abstrata *FormaGeometrica*, a qual contém um ponto em sua origem. A classe *FormaGeometrica* deve possuir pelo menos um método abstrato para impressão dos dados de uma forma geométrica.
9. Criar uma classe concreta *Linha*, subclasse de *FormaGeometrica*. Além de um ponto de origem, a classe *Linha* tem um outro ponto representando sua outra extremidade. Um método específico dessa classe que deve ser implementado é aquele que calcula o tamanho da linha. O tamanho da linha é dado pela seguinte fórmula

$$d = \sqrt{((x_o - x_1)^2 + (y_o - y_1)^2)}$$

10. Criar uma classe concreta *Circulo*, subclasse de *FormaGeometrica*. Além de sua origem, a classe Circulo possui um raio e uma área. Um método específico dessa classe que deve ser implementado é aquele que calcula a área do círculo. A área do círculo é dada pela fórmula

    $a = \pi R^2$

    Você pode considerar o valor 3.1416 para o número π.

11. Criar uma classe concreta *Triangulo*, subclasse de *FormaGeometrica*. Além de um ponto de origem representando um vértice do triângulo, esta classe possui ainda dois outros pontos representando os seus outros vértices. Essa classe deve possuir um método que responda se um triângulo é equilátero (todos os seus lados são iguais).

12. Criar uma classe *Desenho*, a qual pode conter um número indefinido de formas geométricas (isto é, linhas, círculos e triângulos).

    Você deve implementar os seguintes métodos dessa classe:

a) Ordenar as formas pertencentes ao desenho segundo algum critério de ordenação a ser especificado na hora da chamada do método.
b) Imprimir os dados de todas as formas do desenho ordenados pela coordenada x da origem da forma geométrica (voce deve usar o método definido em a).
c) Imprimir os dados de todos os circulos de área superior a 20 (com exceção daqueles com raio negativo ou zero, para os quais, em vez dos dados, deve ser impressa uma mensagem indicativa dessa anomalia) ordenados crescentemente pelo sua área (voce deve usar o método definido em a)

13. Execute cada um dos programas seguintes, escritos em C++, passo a passo, mostrando o resultado apresentado:

```
// programa 1
#include <iostream.h>

class base {
   public:
      virtual void show1() {
         cout << "base 1\n";
      }
      void show2 () {
         cout << "base 2 \n";
      }
};
class derivada1: public base {
   public:
      void show1() {
         cout << "derivada 1\n";
      }
};
class derivada2: public base {
   public:
      void show2 () {
         cout << "derivada 2 \n";
      }
};
void prt(base *q) {
   q->show1();
   q->show2();
}
void main( ) {
   base b;
   base *p;
   derivada1 dv1;
   derivada2 dv2;

   p = &b;
   prt(p);

   dv1.show1();
   p = &dv1;
   prt(p);

   dv2.show2();
   p = &dv2;
   prt(p);
}
```

```
// programa 2
#include <iostream.h>

class teste {
    int d;
 public:
    teste () {
       d = 0;
       cout << "default \n";
    }
    teste (int p, int q = 0) {
       d = p + q;
       cout << "soma \n";
    }
    teste (teste & p) {
       d = p.d;
       cout << "copia \n";
    }
    teste & operator = (teste & p)
    {
       cout << "atribuicao 1\n";
       d = p.d; return *this;
    }
    teste & operator = (int i)  {
       cout << "atribuicao 2\n";
       d = i; return *this;
    }
    void mostra ( ) {
       cout <<  d << " \n";
    }
};
void main ( ) {
   teste e1 (2, 6);  e1.mostra ( );
   teste e2;           e2.mostra ( );
   teste e3 (73);    e3.mostra ( );
   teste e4;
   e4 = e1;  e4.mostra();
   teste e5 (e2);    e5.mostra ( );
   e5 = 55;           e5.mostra ( );
   teste e6 = e3;   e6.mostra ( );
   teste e7 = 21;   e7.mostra ( );
   e1 = e3 = e5 = e7;
   e1.mostra();     e5.mostra();
}
```

```
// programa 3
#include <iostream.h>

template <class T>
class pilha {
    T* v;
    T* p;
public:
    pilha (int i) {
       cout << "cria "<< i << "\n";
       v = p = new T[i];
    }
    ~pilha () {
       delete[ ] v;
        cout << "tchau \n";
    }
    void empilha (T a) {
       cout << "emp "<< a << "\n";
       *p++ = a;
    }
    T desempilha () {
       return *--p;
    }
    int vazia() {
      if (v == p) return 1;
      else  return 0;
    }
};
void main ( ) {
   pilha<int> p(40);
   pilha<char> q(30);
   p.empilha(11);
   q.empilha('x');
   p.empilha(22);
   q.empilha('y');
   p.empilha(33);
   do {
      cout << p.desempilha() <<
      \n";
   } while (!p.vazia());
   do {
      cout << q.desempilha() <<
      "\n";
   } while (!q.vazia());
}
```