Programação Orientada a Objectos

2015/16

DEIS – Engenharia Informática

Exame época Normal

1ª Parte

3 / Fev / 2016

Nota: namespaces e #includes não são relevantes para os excertos de código apresentados.
Na resolução não pode: usar dados públicos ou protected, nem usar funções protected (porque não)

1. a) (5%) Qual será a saída resultante da execução deste programa?

```
class Figura {
public:
    int perimetro() { return 0; }
    virtual int area() { return 0; }
};
class Rectangulo : public Figura {
    int comp;
    int larg;
public:
    Rectangulo( int c = 1, int l = 1):comp(c), larg(l){}
    int perimetro() {  return 2 * comp + 2 * larg; }
    virtual int area() { return comp * larg;}
    virtual void f(){}
};
class Circulo: public Figura { };
void descreve( Figura * p){
    cout << "\n perimetro : " << p->perimetro();
    cout << "\n      area : " << p->area();
}
int main() {
    cout << "\n ---1---";
    descreve( new Figura);
    cout << "\n ---2---";
    descreve( new Rectangulo(4,2));
    cout << "\n ---3---";
    descreve( new Circulo);
    cout << "\n ---4---";
}
```

1. b) (5%) Indique quais as linhas deste programa em que ocorrem erros de compilação.

```
int main() {
    Figura * p1 = new Rectangulo;
    cout << p1->area();
    p1->f();
    Rectangulo * p2 = new Figura;
}
```

2. (5%) O programa pretende declarar um objecto relógio que é um item de luxo sujeito a determinadas taxas. Verifique se existe algum problema no código. Se houver, explique e resolva-o. A função main está correcta e não deve ser modificada.

```
class ItemLuxo {
    int escalao;
public:
    virtual bool obtemTaxa() = 0;
    bool atribuiEscalao(int e){};
    ItemLuxo(int esc) { escalao = esc; }
};
class RelogioLuxo : public ItemLuxo {
    int preco;
    int categ;
public:
    RelogioLuxo(int pre) { preco = pre; categ = 3; }
    virtual bool atribuiEscalao(int e) { escalao = e; }
};

int main(int argc, char** argv) {
    RelogioLuxo a(25000);
    return 0;
}
```

3. (5%) O programa seguinte refere-se ao registo de coisas. Há uma classe que mantém o nome da coisa, o BI do dono e o número de vezes que é copiado ou atribuido. O resultado da atribuição pode ser observado no programa seguinte. Faça o que for preciso para que o output da função main seja o indicado (não pode mexer na função main).

```
class Registo {
    int DonoBI;
    string NomeItem;
    int copias;
public:
    Registo(int dbi, string ni) {
        DonoBI = dbi;
        NomeItem = ni;
        copias = 0;
    }
};

int main() {
    Registo a(123,"carro"), b(456,"casa"), c(789,"mesa");
    a = b = c;
    a = c;
    cout << a << "\n" << b << "\n" << c;
}
// --- OUTPUT ---
123 / mesa (copiado:0)
456 / mesa (copiado:1)
789 / mesa (copiado:2)
```

4. (5%) Existem duas classes Aluno e Prof para representar os conceitos homónimos da vida real. A forma de cumprimentar alunos e professores é diferente. O código abaixo apresenta o que já existe.

```
class Aluno {
    string nome;
    /* outros dados aluno – não interessa quais */
public:
    Aluno(string n) { nome = n; }
    void cumprimenta() { cout << "Menino " << nome; }
};

class Prof {
    string nome;
    /* dados prof – não relevantes para a pergunta */
public:
    Prof(string n) { nome = n; }
    void cump() { cout << "Sr. Prof. " << nome; }
};
```

Crie uma classe Escola que armazena alunos e professores, e que permita cumprimentar todos os alunos e professores. Nesta pergunta, deve preocupar-se apenas quais as variáveis necessárias para guardar os dados, bem como com o mecanismo que permite cumprimentar toda a gente. Ignore os outros aspectos da classe. Pode modificar as classes Aluno e Prof, pode acrescentar mais classes e o que mais entender. *Deve, no entanto, usar correctamente os conceitos de orientação a objectos.*

5. (5%) Indique quais as linhas deste programa em que ocorrem erros de compilação? Justifique.

```
class Num {
    int n;
public:
    Num(){ n = 0; }
    int * obterN(){ return &n; }
    const int * receberN(){ return &n; }
};
int main() {
    Num ob;                      //linha 1
    *ob.obterN()= 5;             //linha 2
    *ob.receberN()= 5;           //linha 3
    int * x = ob.obterN();       //linha 4
    int * y = ob.receberN();     //linha 5
    const int * w = ob.obterN(); //linha 6
    const int * z = ob.receberN();//linha 7
}
```

6. (5%) Considere as duas classes seguintes. Usando essas classes, defina a classe Fato. Um fato é uma roupa e tem sempre uma gravata. Não pode modificar as classes Gravata nem Roupa.

```
class Gravata {
    string cor;
public:
    Gravata(string s) { cor = s; }
};

class Roupa {
    string marca;
public:
    Roupa(string m) { marca = m;
    }
};
```

7. (5%) A classe Consultas representa a lista de consultas (string com dia e hora) de um determinado médico. Cada médico tem as suas consultas, que variam com o horário que tem e portanto podem ser em número não conhecido. Analise a implementação desta classe e das funções que a utilizam, (código apresentado em seguida). Aponte as deficiências que encontrar e diga o como se resolvem. A sua resposta incluirá: qual o problema (de forma a que se entenda) e as funções (código) a fazer ou modificar de forma a resolver esse problema.

```
class Consultas {
     string * p;  // array dinâmico de consultas (cada consulta: dia e hora)
     int n;       // numero de consultas
public:
     Consultas(){ p[0]="vazio";}
     Consultas( const Consultas & ob){ p=ob.p; }
     Consultas & operator=(Consultas &ob){ p=ob.p; return *this;}
     ~Consultas(){ delete [] p; }
};
Consultas & func1(Consultas ob){ return ob; }
Consultas & func2(Consultas & ob){ return ob; }
Consultas func3(Consultas & ob){ return ob; }

void func4(Consultas esta){
    Consultas aux[5];
    aux[0] = esta;
}
```