Programação Orientada a Objectos

2014/15

DEIS – Engenharia Informática

Exame época Normal

1ª Parte

28 / Jan / 2015

Nota: namespaces e #includes não são relevantes para os excertos de código apresentados.

1. (5%) Considere o seguinte código. Se achar que está correcto diga o que aparece no ecrã. Se achar que não compila, identifique os erros e corrija-os e por fim diga qual o output.

```
class A {
public:
     virtual void funcV(){ cout << "\n A: funcV "; }
     void g(){ cout << "\n A: g "; }
};
class B : public A {
public:
     virtual void funcV(){ cout << "\n B: funcV "; }
     void g(){ cout << "\n B: g "; }
};
void listar1(A *  x){
     x->funcV();
     x->g();
}
void listar2(B *  x){
     x->funcV();
     x->g();
}
void main(){
     listar1(new A);
     listar1(new B);
     listar2(new A);
     listar2(new B);
}
```

2. (5%) Considere as seguintes definições:

```
class SerVivo {
public:
     virtual void funcA() = 0;
     virtual void funcB() = 0;
};
class Vegetal : public SerVivo{
public:
     virtual void funcB(){ cout << "funcB em Vegetal \n"; }
     virtual void funcC() = 0;
};
```

Defina a classe Arvore, derivada de Vegetal, de modo que seja concreta (possam existir objectos do tipo Arvore). A nova classe deve ter apenas o estritamente necessário para ser concreta.

3. (5%) Qual será a saída resultante da execução deste programa?

```
class Animal {
public:
   virtual ~Animal(){ cout << " ~Animal ";}
};

class Panda: public Animal {
public:
   virtual ~Panda(){ cout << " ~Panda ";}
};


class Planta {
public:
   ~Planta(){ cout << " ~Planta ";}
};

class Arvore: public Planta {
public:
   ~Arvore(){ cout << " ~Arvore ";}
};

int main(){
   Animal * a = new Panda;
   delete a;
   Planta * p = new Arvore;
   delete p;
}
```

4. **.** (5%) Considere o seguinte programa:

```
class A {
     int n;
public:
     A(int nn = 0){ n = nn; }
     // ...
};
void main(){
     A x(2), y(3), z;
     z = x + y;
     getN() = 4;
     int n = x;
}
```

Escreva os protótipos (apenas os protótipos) das funções que necessitam de estar definidas para que este programa corra sem erros. As funções que indicar devem estar devidamente contextualizadas, ou seja, deve ficar claro se pertencem à classe A ou se são globais.

5. (5%) Considerando as seguintes definições, e mexendo apenas na classe C, garanta que o programa seguinte corre. Na função main() deve estar definido um objecto da classe objc com o valor 5 para a altura e 10 para peso.

```
class A {
     int altura;
public:
     A(int a ){ altura = a; }
     // ...
};

class B {
     int peso;
public:
     B(int p){ peso = p; }
     // ...
};

class C : public A{
     B carga;
public:
     // . . .
};

void main(){
     C objc . . . // objecto objc fica com 5 de altura e 10 de peso
}
```

6. (5%) A classe FichaAluno representa a ficha de um aluno de um curso. A classe Curso representa o conjunto dos alunos de um curso (fichas de inscrição). Quando um objecto da classe Curso é destruído, as fichas dos seus alunos deixam de ser necessárias. Os membros pedag1 e pedag2 apontam para dois dos alunos que pertencem à colecção representada pelo vector. Escreva o que falta <u>da classe</u> para garantir o funcionamento correcto do programa <u>tal como ele está agora</u> (ou seja, sem mexer nas funções globais nem nos membros variáveis da classe Curso).

```
class FichaAluno{
     // ...
};
class Curso{
     vector<FichaAluno *> alunos;
     FichaAluno * pedag1;
     FichaAluno * pedag2;
public:
     // …
};
void func() {
     Curso c;
     // etc – insere alguns alunos
}
int main() {
     func();
}
```

7. (10%) O programa seguinte tem erros de execução. Escreva uma versão corrigida deste programa. Não pode alterar o membro variável, nem intenção e protótipo das funções, nem o destrutor, nem a função main().

```
class Conjunto{
     vector<string *> elementos;
public:
     Conjunto() { cout << “teste”; }
     ~Conjunto(){
        for (unsigned int i = 0; i < elementos.size(); i++)
           delete elementos[i];
     }
     Conjunto(const Conjunto & ob){
        for (unsigned int i = 0; i < elementos.size(); i++)
           elementos.push_back(ob.elementos[i]);
     }
     void acrescenta(string s){
        elementos.push_back(&s);
     }
     void eliminaPrimeiro(){
        if (elementos.size() > 0){
           delete elementos[0];
        }
     }
};
int main(){
     Conjunto a;
     a.acrescenta("aaa");
     a.acrescenta("bbb");
     a.acrescenta("ccc");
     a.eliminaPrimeiro();
     Conjunto b = a;
}
```