PROVA DE EXAME – resolução
de
Programação Orientada por Objetos

INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão
Época Normal – 20 de janeiro de 2014

Grupo I

(8 valores)

No Anexo A do presente enunciado encontra-se o código de um pequeno programa em C++. ...

a) Identifique o tipo de relação existente entre `Documento` e `Frase`.
Agregação. Documento agrega Frase numa relação de 1 para n.

b) Como sabe, para além dos métodos que são definidos de forma explícita em cada classe, ...
Os métodos implícitos que estão presentes em ambas as classes são o construtor de cópia e o operador afetação.

c) O método construtor `Documento::Documento(int m)` tem um problema grave. ...
O método em causa tenta criar um vetor de objetos da classe Frase. Essa operação é impossível dado que a classe Frase não dispõe do construtor por defeito. A solução passaria por definir explicitamente esse construtor: Frase::Frase(){}

d) Apresente o resultado que será visualizado na saída standard após a execução do programa.

```
Criado um documento com um maximo de 5 frases
Criada a frase: Exame
Frase <Exame > destruida
Criada a frase: de POO
Frase <de POO> destruida
Exame de POO
Frase <> destruida
Frase <> destruida
Frase <> destruida
Frase <de POO> destruida
Frase <Exame > destruida
Documento com 2 frases destruido
```

e) Considere agora o seguinte código substituto para a função main(). O programa apresenta um ...
Como o documento doc é construído por cópia do documento apontado por d, através do construtor de cópia implícito, os dois documentos passam, através do apontador frases, a apontar para o mesmo vetor de frases. O problema surge então quando, na última instrução do main, se tenta mostrar as frases de doc: embora o apontador frases ainda exista no doc, o vetor de frases por ele apontado terá já sido desalocado aquando da eliminação do documento apontado por d (delete d).
A solução passaria por definir explicitamente o construtor de cópia da classe Documento tal como se segue:

```
Documento::Documento(const Documento &outro){
      max=outro.max;
      n=outro.n;
      frases= new Frase[max];
      for(int i=0; i<n; i++) frases[i]=outro.frases[i];
}
```

Grupo II

(12 valores)

a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e ... [8.5 val.]

```
class SAP{
    Colecao<Aluno> alunos;
    Colecao<Sala> salas;
public:
    bool SAP::addSala(int n){
          Sala s(n);
          return salas.insert(s);
    }

    bool SAP::addAluno(int n,const string &nom){
          Aluno a(n,nom);
          return alunos.insert(a);
    }

    Aluno *findAluno(int n){
          Aluno  a(n,"");
          return alunos.find(a);
    }

    Sala *findSala(int n){
          Sala s(n);
          return salas.find(s);
    }

    bool registarPresenca(int nsala, int naluno){
          Aluno *pa=findAluno(naluno);
          if(pa!=NULL){
                Sala *ps=findSala(nsala);
                if(ps!=NULL) return ps->registarPresenca(pa);
                else return false;
          }else return false;
    }
};


class Sala{
    int numero;
    Colecao<Presenca> presencas;
public:
    Sala(int n){numero=n;}

    bool registarPresenca(Aluno *a){
          Presenca s(quantidadePresencas()+1,a,this);
          if (presencas.insert(s)) return a->addPresenca(presencas.find(s));
          else return false;
    }

    int quantidadePresencas()const {return presencas.size();}

    bool operator<(const Sala &outra)const {return numero<outra.numero;}
};
```

```
class Aluno{
    int numero;
    string nome;
    Colecao<Presenca *> presencas;
public:
    Aluno(int n,const string &nom): nome(nom){numero=n;}

    bool addPresenca(Presenca *p){return presencas.insert(p);}

    bool operator<(const Aluno &outro)const {return numero<outro.numero;}
};

class Presenca{
    int id;
    TDataHora entrada;
    Aluno* aluno;
    Sala* sala;
public:
    Presenca(int i, Aluno* a, Sala* s){
          entrada=TDataHora::hoje_agora();
          id=i;
          aluno=a;
          sala=s;
    }

    bool operator<(const Presenca &outra)const {return id<outra.id;}
};
```

b) Acrescente ao problema os métodos que permitam mostrar todas as presenças de um ... [2.5 val.]

```
void SAP::printPresencasAluno(int n){
    Aluno *pa=findAluno(n);
    if(pa!=NULL) pa->printPresencas();
    else cout<<"Aluno nao existe!"<<endl;
}

void Aluno::printPresencas()const{
    cout<<"Presencas do aluno "<< nome <<": "<<endl;
    Colecao<Presenca *>::iterator it;
    for(it=presencas.begin(); it!=presencas.end(); it++)
          (*it)->print();
}

void Presenca::print()const{
    cout<<"N."<<id<<" Entrada a "<<entrada<<endl;
};
```

c) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1.0 val.]

```
void main(){
    SAP sap;
    sap.addSala(5);
    sap.addAluno(123,"Ana Rita");
    sap.registarPresenca(5,123);
    sap.printPresencasAluno(123);
}
```