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 de Recurso – 8 de fevereiro de 2014

Grupo I

(8 valores)

a) Identifique o tipo de relação existente entre *Instituto* e *Escola* e entre *Escola* e *ESTiG*. [0.6 val.]
Respetivamente, associação simples (Instituto associa Escola) e Herança (ESTiG deriva de Escola). *(do ponto de vista conceptual, seria também aceitável considerar agregação em vez de associação simples)*

b) Diga quais são os métodos construtores que estarão presentes em cada uma das classes... [1.0 val.]
Tanto o construtor por defeito como o de cópia estão presentes em todas as classes; as classes Escola e Instituto dispõem ainda dum construtor de conversão cada.

c) Nas classes do problema existem métodos que deveriam ser definidos constantes? Se ... [0. 6 val.]
Sim, o método print() de todas as classes.

d) Apresente o resultado que será visualizado na saída standard com a execução do programa. [3.6 val.]

```
Criada a ESTiG. A melhor escola do pais!
Criada a ESA. A 2a melhor escola do pais!
A ESTiG e' mesmo fixe!
A ESA e' mesmo fixe!
Encerrada a 2a melhor escola do pais!
ESA encerrada!
Instituto encerrado
Instituto encerrado
Encerrada a melhor escola do pais!
ESTiG encerrada!
```

e) O que seria visualizado se o método *print* da classe *Escola* não fosse virtual? Refira-se... [1.0 val.]

```
No lugar das mensagens
        A ESTiG e' mesmo fixe!
        A ESA e' mesmo fixe!
surgiria:
        A escola e' mesmo fixe!
        A escola e' mesmo fixe!
```

f) E se fosse o método destrutor da classe *Escola* a não ser virtual? Refira-se apenas ... [0. 6 val.]

```
Não era mostrada a mensagem
        Encerrada a 2a melhor escola do pais!
```

g) Diga se as declarações `ESTiG V1[5]; ESA *V2[5];` são ou não válidas no problema ... [0. 6 val.]
Ambas são válidas. Na primeira declaração são invocados 10 construtores; na segunda não é invocado qualquer construtor.

Grupo II
(12 valores)

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

```
class Mundial{
      Colecao<Estadio> estadios;
      Colecao<Selecao> selecoes;
public:
      bool addSelecao(string p){
            Selecao s(p);
            return selecoes.insert(p);
      }

      bool addEstadio(string nom, string loc){
            Estadio e(nom,loc);
            return estadios.insert(e);
      }

      bool addJogo(string estad, TData d, string s1, string s2){
            Estadio *pe=findEstadio(estad);
            if(pe!=NULL){
                  Selecao *ps1=findSelecao(s1);
                  if(ps1!=NULL){
                        Selecao *ps2=findSelecao(s2);
                        if(ps2!=NULL)
                              return pe->addJogo(d,ps1,ps2);
                  }
            }
            return false;
      }

      Selecao *findSelecao(string p){
            Selecao s(p);
            return selecoes.find(s);
      }

      Estadio *findEstadio(string nom){
            Estadio e(nom, "");
            return estadios.find(e);
      }
};


class Selecao{
      string pais;
      Colecao<Jogo *> jogos;
public:
      Selecao(string p): pais(p){}
      bool addJogo(Jogo *j){return jogos.insert(j);}
      bool operator<(const Selecao &outra)const {return pais<outra.pais;}
};
```

```
class Estadio{
      string nome;
      string local;
      Colecao<Jogo> jogos;
public:
      Estadio(string nom, string loc):nome(nom),local(loc){}

      bool addJogo(TData d, Selecao *s1, Selecao *s2){
            Jogo jogo(d,this,s1,s2);
            jogos.insert(jogo);
            Jogo *pj=jogos.find(jogo);
            if (pj!=NULL)
                  if (s1->addJogo(pj))
                        if (s2->addJogo(pj)) return true;
            return false;
      }

      bool operator<(const Estadio &outro)const {return nome<outro.nome;}
};


class Jogo{
      TData data;
      Estadio *estadio;
      Selecao *opositor1, *opositor2;
public:
      Jogo(TData d, Estadio *e, Selecao *s1, Selecao *s2): data(d){
            estadio=e;
            opositor1=s1; opositor2=s2;
      }

      bool operator<(const Jogo &outro)const {return data<outro.data;}
};
```

b) Acrescente ao problema os métodos que permitam mostrar todas os jogos realizados ... [2.5 val.]

```
void Mundial::printJogosDeEstadio(string estad){
      Estadio *pe=findEstadio(estad);
      if(pe!=NULL) pe->printJogos();
      else cout<<"Estadio inexistente!"<<endl;
}

void Estadio::printJogos()const{
      cout<<"Jogos do estadio "<< nome<<":"<<endl;
      Colecao<Jogo>::iterator it;
      for(it=jogos.begin(); it!=jogos.end(); it++)
            it->print();
}

void Jogo::print()const{
      cout<<data<<": "<<opositor1->getPais()<<" x "<<opositor2->getPais()<<endl;
}

string Selecao::getPais(){return pais;}
```

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

```
void main(){
	Mundial brasil2014;
	brasil2014.addEstadio("Maracana", "Rio de Janeiro");
	brasil2014.addSelecao("Portugal");
	brasil2014.addSelecao("Brasil");
	brasil2014.addJogo("Maracana","13/07/14", "Brasil", "Portugal");
	brasil2014.printJogosDeEstadio("Maracana");
}
```

Grupo III

(20 valores)

a) Defina em C++ as novas classes Jogador e Presenca, e acrescente às já existentes ... [9.0 val]

```
class Presenca{
	int minutos;
	Jogo *jogo;
	Jogador *jogador;
public:
	Presenca(int m, Jogo *jg, Jogador *jgr){
		minutos=m;
		jogo=jg; jogador=jgr;
	}
	int getMinutos(){return minutos;}
	bool Presenca::operator<(const Presenca &outra)const{
		if(jogador->getSelecao()->getPais()!=
					outra.jogador->getSelecao()->getPais())
			return jogador->getSelecao()->getPais()<
					outra.jogador->getSelecao()->getPais();
		else return jogador->getNumero()<outra.jogador->getNumero();
	}
};

class Jogador{
	string nome;
	int numero;
	Selecao *selecao;
	Colecao<Presenca *> presencas;
public:
	Jogador(string nom, int n, Selecao *s):nome(nom){
		numero=n;
		selecao=s;
	}
	int getNumero(){return numero;}
	Selecao *getSelecao(){return selecao;}
	bool addPresenca(Presenca *p){return presencas.insert(p);}
	float mediaMinutPorJogo(){
		float m=0.0;
		Colecao<Presenca *>::iterator it;
		for(it=presencas.begin(); it!=presencas.end(); it++)
			m+=(*it)->getMinutos();
		m/=presencas.size();
		return m;
	}
	bool operator<(const Jogador &outro)const {return numero<outro.numero;}
};
```

```
class Selecao{
      string pais;
      Colecao<Jogo *> jogos;
      Colecao<Jogador> jogadores;
public:
      ...
      bool addJogador(string nom, int n){
            Jogador j(nom, n, this);
            return jogadores.insert(j);
      }
      float mediaMinutPorJogo(int jgr){
            Jogador *pj=findJogador(jgr);
            if(pj!=NULL) return pj->mediaMinutPorJogo();
            else return 0.0;
      }
      Jogador *findJogador(int n){
            Jogador j("",n,NULL);
            return jogadores.find(j);
      }
};

class Jogo{
      TData data;
      Estadio *estadio;
      Selecao *opositor1, *opositor2;
      Colecao<Presenca> presencas;
public:
      ...
      bool addPresenca(Jogador *jgr, int m){
            Presenca p(m,this,jgr);
            if(jgr->getSelecao()!=opositor1 && jgr->getSelecao()!=opositor2){
                  cout<<"A Selecao desse jogador nao jogou esse encontro!"<<endl;
                  return false;
            }else
                  if (presencas.insert(p))
                        return jgr->addPresenca(presencas.find(p));
                  else return false;
      }
};

class Estadio{
      string nome; string local; Colecao<Jogo> jogos;
public:
      ...
      bool addPresenca(TData d, Jogador *jgr, int m){
            Jogo *pj=findJogo(d);
            if(pj!=NULL) return pj->addPresenca(jgr,m);
            else return false;
      }
      Jogo *findJogo(TData d){
            Jogo j(d,NULL,NULL,NULL);
            return jogos.find(j);
      }
};
```

```
class Mundial{
      Colecao<Estadio> estadios;
      Colecao<Selecao> selecoes;
public:
      ...
      bool addJogador(string sel, string nom, int n){
            Selecao *ps=findSelecao(sel);
            if(ps!=NULL) return ps->addJogador(nom, n);
            else {cout<<"Selecao inexistente!"<<endl; return false;}
      }
      bool addPresenca(string estad, TData d, string sel, int jgr, int m){
            Estadio *pe=findEstadio(estad);
            if(pe!=NULL){
                  Selecao *ps=findSelecao(sel);
                  if(ps!=NULL){
                        Jogador *pj=ps->findJogador(jgr);
                        if(pj!=NULL)
                              return pe->addPresenca(d,pj,m);
                  }
            }
            return false;
      }
      float mediaMinutPorJogo(string sel, int jgr){
            Selecao *ps=findSelecao(sel);
            if(ps!=NULL) return ps->mediaMinutPorJogo(jgr);
            else return 0.0;
      }
};
```

b) Implemente um pequeno main que faça uso das novas funcionalidades da aplicação. [1.0 val.]

```
void main(){
      Mundial brasil2014;
      brasil2014.addEstadio("Maracana", "Rio de Janeiro");
      brasil2014.addSelecao("Portugal");
      brasil2014.addSelecao("Brasil");
      brasil2014.addJogo("Maracana","13/07/14", "Brasil", "Portugal");
      brasil2014.printJogosDeEstadio("Maracana");
      brasil2014.addJogador("Portugal","Criastiano Ronaldo",7);
      brasil2014.addPresenca("Maracana","13/07/14", "Portugal", 7, 85);
      cout<<"Media dos Minutos/Jogo de CR7: " <<
                                    brasil2014.mediaMinutPorJogo("Portugal",7)<<endl;
}
```