INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão

**Época Normal – 20 de janeiro de 2014**

---

**Notas importantes**:

1. *A duração da prova é de 1 hora e 30 minutos.*
2. *Comece por preencher a zona reservada à sua identificação na folha de exame.*
3. *Resolva os dois grupos em folhas de exame separadas.*

---

## Grupo I

(8 valores)

No Anexo A do presente enunciado encontra-se o código de um pequeno programa em C++. Depois de o analisar com a devida atenção, responda às seguintes questões.

a) Identifique o tipo de relação existente entre `Documento` e `Frase`.

b) Como sabe, para além dos métodos que são definidos de forma explícita em cada classe, existirão outros definidos automaticamente pelo C++. Diga quais são os métodos implícitos que estarão presentes em cada uma das duas classes do problema, ainda que não se vejam.

c) O método construtor `Documento::Documento(int m)` tem um problema grave. Identifique-o e proponha uma solução adequada, sem alterar o método em questão (correções que deverá levar em conta na resolução das restantes alíneas).

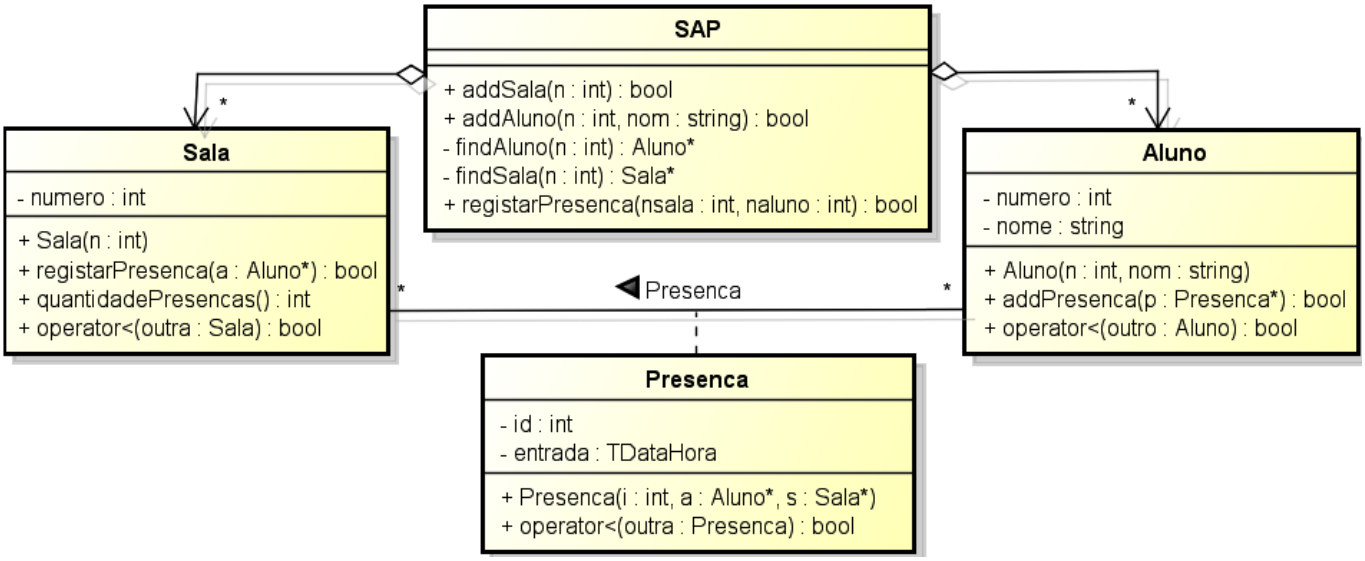
d) Apresente o resultado que será visualizado na saída standard após a execução do programa.

e) Considere agora o seguinte código substituto para a função main(). O programa apresenta um erro sério de execução. Identifique-o e resolva-o convenientemente, alterando apenas o código da classe `Documento`.

```
void main(){
    Documento *d= new Documento(5);
    d->addFrase("Exame ");
    d->addFrase("de POO");
    Documento doc(*d);
    delete d;
    doc.print();
}
```

## Grupo II
(12 valores)

Suponha que, para efetuar a implementação do atual Sistema Automático de Presenças (SAP), a ESTiG começou por solicitar a criação de um protótipo, o qual, deveria permitir registar, para além das presenças, os alunos e as salas de aula. Nesse protótipo, não deveriam ser ainda consideradas as unidades curriculares, as turmas, as salas, nem tão pouco as aulas lecionadas; apenas o registo do dia e da hora de entrada dos alunos nas salas, sem preocupações com a validação da hora e data de entrada. Para evitar a definição de chaves múltiplas, sugeriu-se considerar um identificador na presença, automaticamente incrementado. Segue-se o diagrama de classes UML, para descrever de forma precisa a solução que se pretende para a aplicação.




a) Defina em C++ todas as classes do problema, respeitando integralmente os métodos e atributos descritos no diagrama. Não implemente quaisquer outros métodos ou atributos. [8.5 val.]

*NOTA 1: Para os tipos data hora é usada a classe TDataHora, estudada nas aulas da UC, e da qual se apresentam, no Anexo B deste enunciado, alguns métodos e operadores;*

*NOTA 2: Imagine que a aplicação está correr em tempo real. Por isso, para a hora e data das presenças devem ser consideradas a hora e data correntes. Use, para o efeito, as funcionalidades da classe TDataHora;*

*NOTA 3: As coleções deverão ser implementadas com base no template de classes Colecao ou ColecaoHibrida que utilizou nas aulas de POO. Para efeitos de consulta, disponibilizam-se os protótipos dos seus principais métodos no Anexo B deste enunciado;*

*NOTA 4: Defina os métodos no momento em que está a declarar as classes (não separe a declaração da implementação) e considere que todo o código reside num único ficheiro;*

*NOTA 5: Nesta e nas restantes alíneas, não precisa de indicar as diretivas #include.*

b) Acrescente ao problema os métodos que permitam mostrar todas as presenças de um dado aluno, com a indicação do número da presença e da respetiva data hora de entrada. [2.5 val.]

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

**ANEXO A**

```
#include<string>
#include<iostream>
using namespace std;

class Frase{
	string conteudo;
public:
	Frase(const string &c):conteudo(c){
		cout << "Criada a frase: " << conteudo << endl;
	}

	void print(){cout << conteudo;}

	~Frase(){cout << "Frase <" << conteudo << "> destruida"<<endl;}
};

class Documento{
	Frase *frases;
	int n, max;
public:
	Documento(int m){
		max=m; n=0;
		frases=new Frase[max];
		cout << "Criado um documento com um maximo de "
			<< max << " frases" <<endl;
	}

	void addFrase(const string &c){
		if(n<max){
			Frase f(c);
			frases[n++]=f;
		}
	}

	void print(){
		for(int i=0;i<n;i++) frases[i].print();
		cout << endl;
	}

	~Documento(){
		delete []frases;
		cout << "Documento com " << n << " frases destruido"<<endl;
	}
};

void main(){
	Documento doc(5);
	doc.addFrase("Exame ");
	doc.addFrase("de POO");
	doc.print();
}
```

## ANEXO B

### Template Colecao

```
#include<set>
using namespace std;

template<class K>
class Colecao: public set<K>{
public:
	bool insert(const K &c);
	K *find(const K &c);
	int size() const;
	void erase(const K &);
	//void clear();
	//bool empty() const;
	//iterator begin();
	//iterator end();
};
```

### Template ColecaoHibrida

```
#include<set>
using namespace std;

template<class T>
class less_pointers{
public:
	bool operator()(const T &left, const T &right) const {
		return (*left < *right);
	}
};

template<class K>
class ColecaoHibrida: public set<K, less_pointers<K>>{
public:
	bool insert(const K &c);
	K find(const K &c);
	int size() const;
	void erase(const K &);
	//void clear();
	//bool empty() const;
	//iterator begin();
	//iterator end();
};
```

### Classe TDataHora

```
#include"TData.h"
#include"THora.h"

class TDataHora: public TData, public THora{
public:
	TDataHora();
	string getStr() const; //devolve uma string c/ data e tempo ("x/x/xx h:m:s")
	...
	static TDataHora hoje_agora();//devolve data e hora corrente
};
ostream &operator<<(ostream &os, const TDataHora &dt);
istream &operator>>(istream &is, TDataHora &dt);
```