INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Engenharia Informática
Informática de Gestão

Época de Recurso – 9 de fevereiro de 2013

Notas importantes:

1. *O Grupo III é de resolução facultativa e destina-se a substituir as notas do trabalho prático e miniteste*
2. *Duração da prova: 1h30 (Grupos I e II) + 1h (Grupo III)*
3. *Comece por preencher a zona reservada à sua identificação na folha de exame;*
4. *Resolva os três 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) Diga quais são os métodos construtores que estarão presentes em cada uma das classes do problema. [1.4 val.]

b) Apresente o resultado que será visualizado na saída standard com a execução do programa. [2.7]

c) O que seria visualizado se o método `get` da classe `I` não fosse virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea anterior. [0.5 val.]

d) O que seria visualizado se o método `print` da classe `I` não fosse virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea b). [0.5 val.]

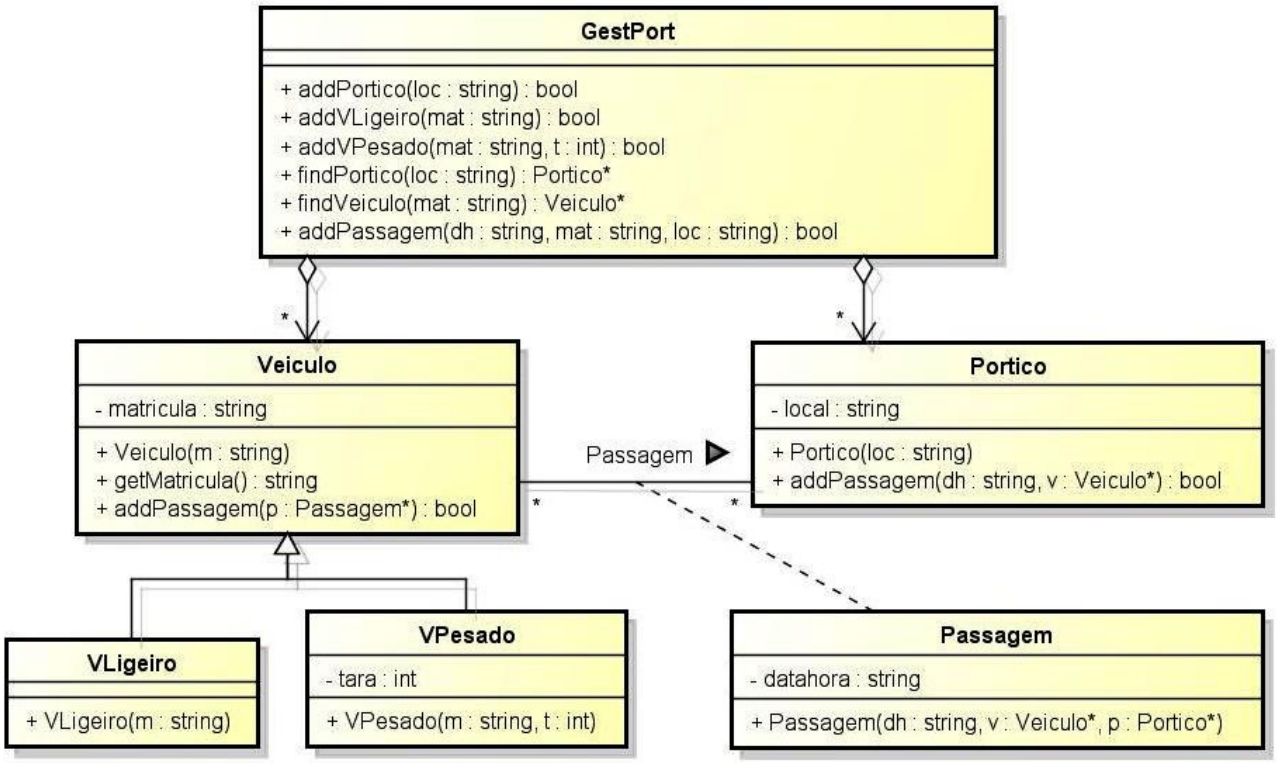
e) E o que seria visualizado se fosse o método destrutor da classe `I` a não ser virtual? Refira-se apenas à parte da visualização que resultaria diferente em relação ao output da alínea b). [0.5 val.]

f) Diga se as declarações `I V1[10]; I *V2[20];` são ou não válidas no problema considerado. Caso não sejam, introduza as alterações necessárias nas classes do problema para que as declarações passem a ser válidas. Diga, por fim, quantos construtores serão invocados com as referidas declarações. [1.2 val.]

g) Faça as alterações que julgue necessárias para que a função *main* do problema considerado possa finalizar com a seguinte instrução: `if (a>0) cout<<a;` [1.2 val.]

Grupo II
(12 valores)

Suponha que a *GestPort*, *Lda* é a empresa responsável pelo sistema eletrónico de cobrança de portagens nas antigas SCUTs. Pretende-se uma pequena aplicação informática que lhe permita registar a data e a hora (apenas uma string) da passagem de veículos pelos pórticos de cobrança eletrónica colocados estrategicamente ao longo das inúmeras vias concessionadas. Portanto, nesta primeira fase, o sistema ainda não deve assegurar a gestão de cobranças. Segue-se o diagrama de classes UML que descreve de forma precisa a solução que se pretende para a aplicação descrita.



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; apenas acrescente, nas classes em que se justifique, o operador que permita que os objetos sejam colecionáveis. [11 val]

NOTA 1: Incluindo as autoestradas duas ou mais vias de trânsito, admita que num mesmo pórtico poderão ser registados veículos que passem na mesma data e hora;

NOTA 2: 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 3: 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 4: Nesta e nas restantes alíneas, não precisa de indicar as diretivas #include.

b) Implemente um pequeno main que faça uso de todas as funcionalidades da aplicação. [1 val.]

Grupo III
(20 valores)

NOTA: O problema que se segue é um exercício suplementar e facultativo que se destina a substituir a classificação obtida durante o período letivo (miniteste+trabalho). Com a sua resolução, ou tentativa de resolução, essa componente de avaliação deixa de contar para a época de recurso.

As alíneas que se seguem destinam-se a acrescentar novas funcionalidades ao problema que começou a implementar no grupo de questões anterior. Por isso, nas suas repostas deve ter sempre em conta o código que já implementou anteriormente para o problema.

a) Implemente o(s) método(s) necessário(s) que permita(m) listar todos os veículos registados no sistema, mostrando, para cada um deles, a matrícula, se se trata de um veículo ligeiro ou pesado e, tratando-se deste segundo caso, a respetiva tara. [4 val.]

b) Implemente o(s) método(s) necessário(s) que permita(m) listar todas as passagens registadas num dado pórtico, mostrando, para cada uma delas, a matrícula do veículo e a data e a hora da sua passagem. [4 val.]

c) Para que a aplicação se revele verdadeiramente útil, vamos agora acrescentar-lhe os mecanismos que lhe permitam gerir a cobrança de portagens. Para o efeito, nesta alínea, acrescente-lhe o(s) atributo(s) e método(s) que permitam definir uma tarifa unitária específica para cada pórtico (tarifa que será posteriormente usada para o cálculo da taxa da portagem de cada veículo que passe no pórtico). [3 val.]

d) Implemente o(s) método(s) necessário(s) para calcular o valor total das portagens faturado pela empresa GestPort, admitindo o seguinte: tratando-se de um veículo ligeiro, a taxa de portagem tem sempre o valor da tarifa unitária; caso se trate de um pesado, a taxa a cobrar será o dobro da tarifa unitária mais o valor dessa tarifa por cada 5 toneladas da sua tara (exemplo: sendo a tarifa 3€ e a tara 21000Kg, a taxa será 2*3€+4*3€ = 18€). [5 val.]

e) Implemente o método destrutor, com o comportamento adequado, para a classe GestPort. [1 val.]

f) Implemente o(s) método(s) necessário(s) para remover do sistema um dado veículo. A remoção só deverá ser levada a cabo se o veículo em causa ainda não tiver passado em nenhum pórtico. [2 val.]

g) Escreva uma função main() que faça uso de todas as novas funcionalidades do problema. [1 val.]

ANEXO A

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

class I{
public:
      int virtual get()const{return 0;}
      void virtual print()const{cout<<"I::print"<<endl;}
      virtual ~I(){}
};

class A: public I{
      int val;
      static int n;
public:
      A(){val=++n; cout<<"novo A "<<val<<endl;}
      int get()const{return val;}
      void print()const{cout<<"A::print "<<val<<endl;}
      ~A(){cout<<"A "<<val<<" eliminado"<<endl;}
};

int A::n=0;

class B: public I{
      int val;
      A a[2];
public:
      B(): val(0){cout<< "novo B "<<endl;}
      B(A &x):val(x.get()){cout<< "A => B "<<val<<endl;}
      void print()const{
            cout<<"B::print "<<val<<endl;
            a[0].print();
            a[1].print();
      }
      ~B(){cout<<"B "<<val<<" eliminado"<<endl;}
};


void main(){
      cout<<"<1>"<<endl;
      B b1;
      cout<<"<2>"<<endl;
      I *p=&b1;
      cout<<"<3>"<<endl;
      p->print();
      cout<<"<4>"<<endl;
      A a;
      cout<<"<5>"<<endl;
      B b2(a);
      cout<<"<6>"<<endl;
}
```

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();
};
```