Recibo do Exame de 1ª Época de Introdução à Programação (IGE e ETI), 2003/02/25 **1º semestre de 2002/2003, ISCTE**
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

Exame de 1ª Época

Introdução à Programação

IGE e ETI

2003/02/25 - 1º semestre de 2002/2003

ISCTE

Notas:

- Leia atentamente todo o enunciado (incluindo estas notas) antes de começar a prova.
- Este exame tem quatro questões.
- Duração: 3:30h (três horas e trinta minutos).
- A prova deve ser efectuada sem consulta.
- Sobre a sua secretária só pode estar o enunciado e uma caneta (coloque pastas, telemóveis e outros objectos sobre a cadeira ao seu lado).
- Desligue o telemóvel.
- Deve responder a todas as questões no próprio enunciado, nos espaços reservados.
- Use as páginas em branco ao longo do enunciado para rascunho.
- Não destaque as folhas do enunciado (nem mesmo as de rascunho).
- Preencha o seu nome e número no recibo (não o destaque). Este ser-lhe-á entregue, assinado por um docente, depois da prova.
- Só um recibo assinado servirá de prova de que entregou a resolução deste enunciado.
- As cotações encontram-se abaixo de cada alínea (total de 20 valores).
- As notas serão afixadas até às 18:00h de segunda-feira, dia 3 de Março de 2003.
- A revisão de provas terá lugar das 9:00h às 10:00h de quarta-feira, dia 5 de Março de 2003, no gabinete 9.
- As orais terão lugar a partir das 8:00h de segunda-feira, dia 10 de Março de 2003, no gabinete 9, devendo os interessados inscrever-se enviando uma mensagem de correio electrónico para Manuel.Sequeira@iscte.pt até às 18:00h de sexta-feira, dia 7 de Março de 2003.
- Boa sorte.

Questão 1

Assinale com V (Verdadeiro) as expressões que estão correctas e com F (Falso) as que estão incorrectas.

Os quadrados podem ser deixados em branco ou ser preenchidos com V ou F.

Em geral as alíneas podem ter zero ou mais respostas correctas. Cada resposta correctamente assinalada vale 0,5 valores. Respostas incorrectas correspondem a um desconto de 0,2 valores. Respostas em branco valem 0 valores.

Nas alíneas em que apenas uma resposta está correcta (se existirem estão assinaladas no texto), responder com mais ou menos do que um V anula a cotação. A resposta correcta corresponde à cotação completa. Respostas incorrectas correspondem a um desconto de metade da cotação completa.

Considere as declarações e o programa seguintes:

```
class Movimento {
  public:
    enum Tipo {deposito, levantamento_MB, pagamento_combustivel};

    Movimento(Tipo const tipo, double const montante);

    Tipo tipo() const;
    char abreviaturaDoTipo() const;
    double valorNominal() const;
    double valorEfectivo() const;
    double custo() const;

    static double const taxa_de_combustivel;

   private:
     Tipo tipo_;
     double montante;

     bool cumpreInvariante() const;
};

class ContaAOrdem {
  public:
    ContaAOrdem(int const numero, string const& titular,
                double const saldo_inicial = 0.0);

    int numero() const;
    string const& titular() const;
    double saldo() const;
    int numeroDeMovimentos() const;
    void mostraMovimentos() const;

    void insere(Movimento const& movimento);

  private:
    int numero_;
    string titular_;
    vector<Movimento> movimentos;

    bool cumpreInvariante() const;
};

...

int main()
{
    ContaAOrdem const nova_conta(2, "Rita", 100);

    ...

}
```

Identificação
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

1.1 Quais das seguintes instruções estão correctas, admitindo que fazem parte da função `main()`, no local indicado pelas reticências?

- ☐ `double nova_taxa = taxa_de_combustivel + 20;`
- ☐ `nova_conta.insereMovimento(Movimento(Movimento::levantamento_MB, 20));`
- ☐ `nova_conta.movimentos.push_back(Movimento(Movimento::levantamento_MB, 20));`

[cotação: 1,5]

1.2 Quais dos seguintes conjuntos de instruções estão correctos, admitindo que fazem parte da função `main()`?

- ☐ `int numero = 1; ContaAOrdem conta(numero, "António");`
- ☐
```
Movimento movimento(Movimento::deposito, 100.0);
vector<Movimento> movimentos(10, movimento);
```
- ☐
```
Movimento movimento(0, 0.0);
vector<Movimento> movimentos;
movimentos.push_back(movimento);
```

[cotação: 1,5]

1.3 Quais das seguintes afirmações são verdadeiras e quais são falsas?

- ☐ Os atributos de uma classe são as suas variáveis membro.
- ☐ Um procedimento calcula qualquer coisa, uma função faz qualquer coisa.
- ☐ Deve-se evitar escrever funções que recebem os seus argumentos por referência não-constante.
- ☐ A palavra-chave `static` aplicada a uma operação de uma classe indica que essa operação não altera os atributos da classe.

[cotação: 2]

Identificação
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

Questão 2

Recorde-se que deve ler atentamente todo o enunciado antes de resolver as questões!

Pretende-se implementar, em C++, uma aplicação para apoio à gestão de uma agência de aluguer de automóveis.

A aplicação permite:

- registar na frota um veículo para alugar;
- mostrar todos os veículos da frota bem como as reservas para cada veículo;
- alterar as tarifas (diária e de fim de semana) de aluguer de um veículo;
- registar a reserva de um veículo;
- calcular o preço de uma reserva;
- alterar a data de fim de uma reserva;
- cancelar uma reserva.

Quando se regista um veículo, deve-se fornecer a sua matrícula (que identifica o veículo univocamente), a marca, o modelo, a tarifa diária e a tarifa de fim de semana. As tarifas podem ser posteriormente alteradas.

Quando o utilizador pretende registar uma reserva para um veículo tem que introduzir a matrícula de um veículo existente na frota, e tem que fornecer o seu nome, identificação e tipo de identificação. A identificação é guardada como um inteiro e o tipo de identificação indica se se trata de um bilhete de identidade ou de um passaporte. É também obrigatório introduzir as datas de início e de fim do período da reserva. Considere que as datas são representadas por simples inteiros. Considere ainda que está definido um predicado `bool eFimDeSemana(int const inicio, int const fim)` que recebe duas datas, indicando se as duas datas definem um fim-de-semana, i.e., se o primeiro argumento corresponde a uma sexta-feira e o segundo argumento corresponde a uma segunda-feira.

Não se preocupe em verificar se é possível efectuar uma reserva para uma determinada viatura e um determinado período, i.e., data de início e de fim. I.e., não precisa de lidar com possíveis sobreposições de reservas.

O preço de uma reserva cuja data de início é uma sexta-feira e cuja data de fim é uma segunda-feira é igual à tarifa de fim-de-semana. Nos restantes casos o preço é calculado com base na tarifa diária e no número de dias existentes entre a data de início e de fim da reserva. Para calcular o número de dias entre duas datas, basta subtraí-las.

Para calcular o preço de uma reserva, para alterar a data de fim de uma reserva ou para cancelar uma reserva é necessário fornecer a matrícula da viatura e a data de início da reserva.

Devem existir três classes: `GestorDeAgencia`, `Veiculo` e `Reserva`.

Atenção! Cada veículo deve guardar as respectivas reservas!

2.1 Defina as classes envolvidas neste sistema de gestão de aluguer de automóveis.

Não é necessário nesta alínea definir quaisquer métodos das classes.

Não se esqueça de, para todos os métodos, indicar a pré-condição e a condição objectivo. Para cada classe deve também indicar a sua condição invariante.

[cotação: 2,5]

2.2 Defina completamente a classe `Reserva`.

Todas as condições (pré-condições, condições objectivo e condições invariantes de classe) devem ser testadas através de instruções de asserção (`assert`). Nos métodos que mostram informação no ecrã não é necessário prestar atenção à formatação da saída.

[cotação: 1]

2.3 Defina completamente a classe `Veiculo`.

Todas as condições (pré-condições, condições objectivo e condições invariantes de classe) devem ser testadas através de instruções de asserção (`assert`). Nos métodos que mostram informação no ecrã não é necessário prestar atenção à formatação da saída.

[cotação: 3]

2.4 Defina o método da classe `GestorDeAgencia` que regista uma reserva correspondente a um veículo cuja matrícula é dada.

Todas as condições (pré-condições e condições objectivo) devem ser testadas através de instruções de asserção (`assert`).

[cotação: 1]

2.5 Escreva um programa que permite gerir a agência de aluguer de automóveis.
O programa deve mostrar ao utilizador um menu que permita realizar as operações descritas acima.
[cotação: 1,5]

Identificação
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

Questão 3

Considere um predicado `bool haMultiploEntre(int const valor, int const limite_inferior, int const limite_superior)` que devolve `true` caso exista algum múltiplo de `valor` nos inteiros compreendidos entre `limite_inferior` e `limite_superior` (inclusive) e `false` caso contrário. Admite-se que 0 < `valor`.

Desenvolva, utilizando um ciclo, uma implementação do predicado `haMultiploEntre()`. Siga os passos da metodologia de Dijkstra pedidos nas alíneas desta questão.

Embora este predicado possa ser desenvolvido sem o recurso a um ciclo, pretende-se que apresente uma solução que recorra a um ciclo. Os seguintes cabeçalho e esqueleto do corpo da rotina devem ser respeitados:

```
bool haMultiploEntre(int const valor,
                     int const limite_inferior, int const limite_superior)
{
    bool ha_multiplo = ...;
    // Ciclo:
    ...
    return ha_multiplo;
}
```

3.1 Indique a pré-condição (*PC*) e a condição objectivo (*CO*) da rotina e ainda a condição invariante (*CI*) do seu ciclo interno.

[cotação: 1]

3.2 Indique uma guarda *G* tal que *CI* e ¬*G* implica *CO*. Prove explicitamente que de facto *CI* e ¬*G* implica *CO*.

[cotação: 1]

3.3 Defina uma inicialização *inic* tal que se a *PC* for verdadeira, então a *CI* também é verdadeira após `inic` (inicialização das variáveis) e antes da primeira iteração do ciclo, i.e.,

// *PC*

`inic`

// implica *CI*.

Prove explicitamente que de facto:

// *PC*

`inic`

// implica *CI*.

[cotação: 1]

3.4 Defina um passo *passo* tal que se a guarda G e a CI do ciclo forem verdadeiras no início de um passo, então a CI também é verdadeira no fim desse passo, i.e.,

// *G* e *CI*

passo

// implica *CI*.

Prove explicitamente que de facto:

// *G* e *CI*

passo

// implica *CI*.

[cotação: 1]

Identificação
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

Questão 4

4.1 Quando se invoca uma rotina ocorre aquilo a que se chama "passagem de argumentos". A passagem de argumentos pode ser feita por valor, por referência ou ainda por referência constante. Explique cada um destes três modos de passar argumentos e apresente um exemplo ilustrativo de cada um.

[cotação: 1]

4.2 Os membros de uma classe podem ter uma de três categorias de acesso diferentes. Na disciplina de Introdução à Programação abordaram-se duas dessas categorias. Identifique e explique estas duas categorias de acesso, dando um pequeno exemplo.

[cotação: 1]