Recibo do Exame de 2ª Época de Programação Orientada para Objectos (IGE e ETI), 2003/7/30 **2º semestre de 2002/2003, ISCTE**
Nome do aluno: .....
Número do aluno: .....
Assinatura do docente: .....

Exame de 2ª Época

Programação Orientada para Objectos

IGE e ETI

2003/7/30 - 2º 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.
- Preencha o seu nome e número em todos os outros espaços reservados para identificação.
- 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 19:00h de segunda-feira, dia 1 de Setembro de 2003.
- A revisão de provas terá lugar das 8:00h às 9:00h de quarta-feira, dia 3 de Setembro de 2003, no gabinete D6.18.
- As orais terão lugar a partir das 8:00h de sexta-feira, dia 5 de Setembro de 2003, no gabinete D6.18, devendo os interessados inscrever-se enviando uma mensagem de correio electrónico para Manuel.Sequeira@iscte.pt até às 18:00h de quarta-feira, dia 3 de Setembro 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.

1.1 Considere o seguinte programa:

```
#include <string>
#include <iostream>

struct A {
    virtual std::string const f() const;
    std::string const g() const;
    virtual std::string const h() const;
};

std::string const A::f() const {
    return "A::f()";
}

std::string const A::g() const {
    return "A::g()";
}

std::string const A::h() const {
    return "A::h()";
}

struct B : public A {
    std::string const f() const;
    std::string const g() const;
    std::string const h() const;
};

std::string const B::f() const {
    return "B::f()";
}

std::string const B::g() const {
    return "B::g()";
}

std::string const B::h() const {
    return "B::h()";
}

struct C : public B {
    std::string const f() const;
    std::string const g() const;
};

std::string const C::f() const {
    return "C::f()";
}

std::string const C::g() const {
    return "C::g()";
}

int main()
{
    B b;
    A* ponteiro1 = &b;
    C c;
    A* ponteiro2 = &c;

    …

}
```

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

Admitindo que as instruções referidas fazem parte da função `main()`, a seguir às declarações e no local indicado pelas reticências, quais das seguintes afirmações são verdadeiras e quais são falsas?

- ☐ Após a instrução `cout << ponteiro1->f() << endl;` aparece no ecrã "A::f()".
- ☐ Após a instrução `cout << ponteiro1->g() << endl;` aparece no ecrã "A::g()".
- ☐ Após a instrução `cout << ponteiro2->h() << endl;` aparece no ecrã "B::h()".
- ☐ Após a instrução `cout << b.g() << endl;` aparece no ecrã "A::g".

[cotação: 2,0]

1.2 Considere o seguinte programa:

```
int main()
{
    int numero = 5;
    int* ponteiro = &numero;
    int numeros[] = {0, 5, 10, 15, 20, 25};
    int* outro_ponteiro = numeros + 5;

    …
}
```

Admitindo que as instruções referidas fazem parte da função `main()`, a seguir às declarações e no local indicado pelas reticências, quais das seguintes afirmações são verdadeiras e quais são falsas?

- ☐ Após a instrução `cout << numeros[1] << endl;` aparece no ecrã "0".
- ☐ Após a instrução `cout << *(numeros + *ponteiro) << endl;` aparece no ecrã "25".
- ☐ Após a instrução `cout << ponteiro[numeros[0]] << endl;` aparece no ecrã "5".
- ☐ Após a instrução `cout << outro_ponteiro[-2] << endl;` aparece no ecrã "20".

[cotação: 2,0]

1.3 Considere, agora as seguintes declarações e o seguinte programa:

```
#include <iostream>

template<typename I, typename F>
I arredondamentoDe(F const& valor)
{
    if(0 <= valor)
        return I(valor + F(0.5));
    else
        return I(valor - F(0.5));
}

int main()
{
    int x = 2;
    double z = 0.5;
    float f = 0.5f;

    ...
}
```

Admitindo que as instruções referidas fazem parte da função `main()`, a seguir às declarações e no local indicado pelas reticências, quais das seguintes instruções estão correctas?

- ☐ `x = arredondamentoDe(z);`
- ☐ `std::cout << arredondamentoDe<int, double>(f) << std::endl;`

[cotação: 1,0]

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 emissão de canais de televisão. O gestor de emissão gere um catálogo de programas, disponíveis para serem emitidos, e uma grelha de programação. A grelha de programação consiste num conjunto de períodos, em que a cada período corresponde a um programa. Os períodos identificam o momento de emissão dos programas.

A aplicação permite:

- registar um programa no catálogo;
- introduzir um programa do catálogo na grelha de programação, num novo período da grelha de programação, com indicação do instante de início (dado em segundos desde o início dos tempos);
- mostrar todos os dados de um programa independentemente de estar ou não na grelha de programação;
- mostrar todos os programas da grelha entre dois instantes, por ordem crescente do instante de início da emissão do programa;
- verificar se um dado programa é transmitido em directo;
- alterar instante de início do período em que um determinado programa é emitido;
- retirar um programa da grelha de programação;
- retirar um programa do catálogo de programas.

Todos os programas são registados no catálogo de programas.

Os programas disponíveis no catálogo podem ser introduzidos na grelha de programação através do registo de um novo período na grelha. Os períodos da grelha de programação referenciam o programa do catálogo a emitir e o instante de início da sua emissão.

Acerca de cada programa conhece-se o título (único para cada programa), a duração (em segundos) e a sua descrição. Os programas podem ser noticiários ou de entretenimento. Os programas de entretenimento podem por sua vez ser comprados ou de produção própria.

No caso dos noticiários é preciso registar os nomes dos jornalistas que pertencem à sua equipa. O mesmo se passa em relação aos comentadores para ele acreditados. Os noticiários são sempre emitidos em directo.

No caso dos programas comprados é necessário registar a entidade a quem foram adquiridos e o preço de aquisição. Os programas comprados nunca são emitidos em directo.

Para os programas de produção própria é preciso indicar o nome do seu responsável e se se trata de um directo.

Todos os programas de entretenimento têm uma categoria que pode ser filme, concurso, série, musical, desporto ou infantil.

Para registar um período novo na grelha de programação e lhe atribuir um programa, é necessário fornecer o instante de início e indicar o programa a emitir.

Quando se insere um período na grelha, é preciso garantir que este não se sobreponha a nenhum outro período existente.

Considere a existência da classe genérica `list` disponível na biblioteca padrão do C++ e utilize variáveis dinâmicas quando for adequado.

2.1 Desenhe um diagrama de classes, utilizando a notação UML, que corresponda ao modelo da aplicação descrita. É necessário incluir no diagrama a representação das classes, as suas relações e as respectivas multiplicidades. Não é necessário inscrever nas classes os respectivos atributos e operações.

[cotação: 2]

2.2 Defina as classes que incluiu no diagrama de classes, excepto as que representam programas de produção própria e programas comprados. Indique qual a condição invariante de cada classe (*CIC*) e quais as pré-condições (*PC*) e as condições objectivo (*CO*) das suas operações. Adicione a operação necessária para verificar a *CIC*. Não é necessário nesta alínea definir quaisquer métodos das classes.

[cotação: 3]

2.3 Defina todos os métodos responsáveis pela inserção de um programa na grelha de programação desde a interface com o utilizador até à sua inserção efectiva. Considere apenas o caso de o programa a inserir ser um noticiário. Todas as condições (pré-condições, condições objectivo e condições invariantes de classe) devem ser verificadas através de instruções de asserção (`assert`).

[cotação: 1,5]

2.4 Defina todos os métodos envolvidos na remoção de um programa desde a interface com o utilizador até à sua remoção efectiva. Caso o programa esteja inserido num período da grelha de programação, a aplicação deve pedir a confirmação para a sua remoção. Todas as condições (pré-condições e condições objectivo) devem ser verificadas através de instruções de asserção (`assert`).

[cotação: 1,5]

2.5 Defina os métodos da classe que representa o conceito de noticiário que permitem:

- mostrar num canal de saída o noticiário e
- saber se um noticiário é emitido em directo.

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

[cotação: 1]

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

Questão 3

3.1 Considere a seguinte estrutura

```
struct Elo {
    typedef int Item;

    Item item;     // o item propriamente dito.
    Elo* anterior; // ponteiro para o elo anterior na cadeia.
    Elo* seguinte; // ponteiro para o elo seguinte na cadeia.

    /// Construtor de elos.  Simplifica a construção de novos elos.
    Elo(Elo* anterior = 0, Elo* seguinte = 0, Item const& item = Item());
};
```

usada para representar os elos de uma cadeia duplamente ligada (não-circular) com guardas. Implemente um método que permita esvaziar uma cadeia duplamente ligada (não-circular) com guardas dados os ponteiros para as duas guardas.

[cotação: 1,0]

3.2 Considere a seguinte definição de uma pilha de inteiros:

```
class PilhaDeInt {
  public:
    typedef int Item;

    PilhaDeInt();
    PilhaDeInt(PilhaDeInt const& original);

    ~PilhaDeInt();

    PilhaDeInt& operator=(PilhaDeInt const& modelo);

    bool estáVazia() const;

    int altura() const;

    Item const& topo() const;
    Item& topo();

    void põe(Item const& novo_item);
    void tiraItem();

  private:
    static int const capacidade_inicial = 32;
    int capacidade_actual;
    Item* itens;
    int número_de_itens;

    bool cumpreInvariante() const;
};
```

Defina os construtores e os seguintes métodos da classe `PilhaDeInt`:

- `void põe(Item const& novo_item);`
- `void tiraItem();`

fornecendo-os de garantia forte face a excepções. Não é necessário reduzir o espaço reservado no caso de se retirar um item.

[cotação: 3,0]

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

Questão 4

4.1 Explique o que são directivas de pré-processamento. Apresente dois exemplos de directivas de pré-processamento e explique para que serve cada uma dessas directivas.

[cotação: 1]

4.2 Qualquer operação polimórfica é também abstracta? Responda afirmativa ou negativamente e justifique a sua resposta, explicando os conceitos associados e exemplificando com pedaços de código C++.

[cotação: 1]