UNIVERSIDADE DE AVEIRO

DEPARTAMENTO DE ELECTRÓNICA E TELECOMUNICAÇÕES

Exame Teórico de Paradigmas de Programação I

16/Janeiro/2002 (duração uma hora).

I. Vamos assumir que a classe `CircSquare` (circunferência dentro do quadrado) foi derivada da classe `shape` (forma). Um objecto da classe `CircSquare` recebe uma mensagem de redimensionamento, que significa que o agregado de figuras será redimensionado por um factor que é especificado (por exemplo, **1.5**). A classe `CircSquare` tem o código seguinte:

```
class CircSquare : public shape
{     Circle c;
      Rectangle r;
public:
      void draw()
      {     c.draw(); r.draw(); }
      void rotate()
      {     c.rotate(); r.rotate();  }
      void setA(int newA)
      {     c.setRadius(newA/2);
            r.setA(newA);        //modificar lado A
            r.setB(newA);  }     //modificar lado B
      void resize(float factor)
      {     setA(r.getA()*factor);    }
};
```

Apresente o diagrama de sequência para redimensionamento considerado acima.

Valor 0-4.

II. Para a classe seguinte:

```
template <class Type> class set  // a classe "conjunto" que se chama "set"
{
protected:
      Type* conjunto;   // ponteiro para conjunto do tipo "Type"
      unsigned tamanho; // tamanho do conjunto
public:
// construtores da classe "set" reservam memória dinamicamente
// funções da classe "set"
      Type Max();                  };
```

II.1. Implementar a função *Max* de acordo com as seguintes regras:

a) Para `set<char*>` a função *Max* devolve uma "string" máxima que deve ser comparada com as outras "strings" utilizando a função *strcmp*;

b) Para `set<float>` a função *Max* deve devolver o número que tem o valor máximo.

c) Para `set<pessoa>` a função *Max* deve devolver a pessoa que tem a altura máxima.

II.2. Implementar todas as funções da classe "set" que são necessárias para realizar as operações da função `Max` correctamente.

Valor 0-4.

III. Definir um manipulador de formatos `mp(n,m,c)` novo com três argumentos que são:

n - largura;
m - precisão;
c - caracter para preencher espaços

Valor 0-3.

IV. Com base no código seguinte, explique a diferença entre a redefinição do operador = (atribuição) e o construtor de cópia. Pode só marcar as respectivas linhas do código onde o operador "=" ou o construtor de cópia vão ser activados. Utilize as marcas "CC" (para o construtor de cópia) e "=" para atribuição.

```
class SET { ... };
SET SET::operator+=(const SET &) { ... }
const SET& operator =(const SET &S) { ... };
.....................................
void function(SET my_set)
{  SET my_set1(...);
   SET my_set2=my_set1;
   ....................................
   my_set1=my_set;
   my_set2 +=my_set;
   ....................................
}
```

Valor 0-3.

V. Relativamente a cada uma das seguintes afirmações sobre a linguagem C++, indique (à esquerda da numeração) se é verdadeira (**V**) ou falsa (**F**).

1. Uma referência pode ser considerada como lvalue.

2. As funções membro estáticas podem ser virtuais.

3. Vamos assumir que o computador tem uma memória. Isto é o caso de agregação.

4. Os membros estáticos não podem ser inicializados antes da execução da função *main*.

5. Herança sem polimorfismo é possível.

6. As funções virtuais podem ser declaradas com o especificador *friend* para outras classes.

7. É possível obter um ponteiro para o construtor.

8. O operador = pode ser herdado.

9. Quando se constrói um objecto de uma classe derivada o primeiro construtor a ser executado o é o da classe derivada e só depois o construtor da classe base.

10. Uma referência é um ponteiro explícito.

11. A lista das excepções faz parte da função.

12. Esta linha está correcta: *template<class M, classA> A converte(M)*;

13. É permitido declarar referências para classes *abstractas*

14. As funções membro estáticas têm acesso ao ponteiro *this*.

15. É impossível redefinir um operador caso os seus argumentos sejam ponteiros.

16. As variáveis membro estáticas podem ser inicializadas na sua declaração.

17. As regras adoptadas nas linguagens C/C++ para converter os tipos dos argumentos podem ser usadas para templates.

18. Para declarar um array de objectos de uma classe, esta tem de ter um construtor por defeito.

19. Podemos usar expressões constantes como argumentos de template.

20. Uma classe derivada pode ter as mesmas classes base virtuais e não virtuais que podem ser herdadas através de classes intermédias.

21. Para sair dos blocos *catch* ou *try* é permitido usar a instrução *goto*.

22. Uma classe **rua** pode ser derivada de uma classe **cidade**.

23. A seguinte linha está correcta: *int &a;*.

24. O operador `[]` é considerado um operador binário.

O valor para cada pergunta pode ser ***-0.25*** (a resposta é errada), ***0*** (não há nenhuma resposta) ***0.25*** (a resposta é correcta).

Valor 0-6.