Universidade do Minho	**Escola de Engenharia da Universidade do Minho** Mestrado Integrado em Eng. Electrónica Industrial e Computadores Complementos de Programação de Computadores	**2012/2013** **MIEEIC** **(1º Ano)** **2º Sem**

Docentes: Luis Paulo Reis e Ana Alice Baptista

Teste Nº1 - Época Normal - Data 19/04/2012, Duração 1h45m+15min (com consulta)

Nome:_____ Nº Aluno:_____

Responda às seguintes questões, preenchendo a tabela com a **opção correcta (em maiúsculas)** **(Correcto:x Val / Errado: -x/3 Val).** Suponha que foram realizados as inclusões das bibliotecas necessárias.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

GRUPO I (6.0 Valores)

1) Suponha a seguinte classe `CPonto`:

```
1  class CPonto {
2   private:
3     double x, y;
5   public:
6     CPonto();
7     CPonto(double x, double y);
8     CPonto(const CPonto& out);
9     ~CPonto();
10    void Transla(double dx, double dy);
11    void Escala(double fx, double fy);
12    double DistAte(const CPonto& out);
13 };
```

Qual das seguintes alternativas é verdadeira:

A) A linha 13 tem um erro (ponto e virgula no final que nunca se usa depois de uma chaveta)
B) As linhas 3 e 4 declaram dois membros função da classe `CPonto` (uma função `double x` e outra `double y`).
C) As linhas 6 a 12 declaram os sete membros dados e membros função da classe.
D) A classe `CPonto` tem quatro construtores (linhas 6, a 9)
E) Nenhuma das anteriores.

2) Suponha a seguinte definição dos membros-função da classe da alínea anterior que, respetivamente, realizam: a translação de um ponto, escalamento de um ponto, cálculo da distância de um ponto até a um outro ponto.

```
1  void CPonto::Transla(double dx, double dy) {
2       x += dx; y += dy; }
3  void CPonto::Escala(double fx, double fy) {
4       x *= fx; y *= fy; }
5  double CPonto::DistAte(const CPonto&out){
6       return sqrt((out.x-x)*(out.x-x)+
7              (out.y-y)*(out.y-y)); }
```

Qual das seguintes alternativas é verdadeira:

A) As linhas 2 e 4 contêm erros pois os operadores `+=` e `*=` não estão definidos para a classe `CPonto`.
B) As linhas 6 e 7 contém erros pois `x` e `y` não estão definidos na função.
C) As linhas 1, 3 e 5 contêm erros pois `CPonto::` só é utilizado fora das classes e não na definição de uma função de uma classe.
D) As linhas 6 e 7 contém erros pois out.`x` e out.`y` não estão definidos na função.
E) Nenhuma das anteriores.

3) Considere o seguinte fragmento de código C++:

```
1  class CRectangle {
2       float width, height;
3    public:
4       void convert_square(CSquare a) {
5            width = a.side; height = a.side;
6       }
7       void set_values (float a, float b) {
8            width=a; height=b; }
9  };
10  class CSquare {
11      float side;
12   public:
13     void convert_rect(CRectangle a) {
14         side = (a.width+a.height)/2.0; }
15     friend class CRectangle;
16 };
```

A) A função `friend class` (linha 15) não está corretamente declarada;
B) A função `convert_square` (linhas 4 a 6) não pode usar o membro-dado `side`, da classe `CSquare`, (que é privado) por isso está erradamente declarada;
C) A função `convert_rect` (linhas 13 a 14) não pode usar os membros-dado `width` e `height`, da classe `CRectangle`, (que são privados) por isso está erradamente declarada;
D) O programa está correto e permite converter quadrados em retângulos e retângulos em quadrados;
E) Nenhuma das anteriores.

4) Para que membros de uma classe possam ser somente acessíveis pela própria classe e pelas subclasses respetivas (i.e. classes derivadas):

A) Devem estar declarados na zona `public` da classe
B) Devem estar declarados na zona `protected` da classe
C) Devem estar fora de qualquer zona da classe (i.e. declarados entre a classe e a sub-classe de modo a estarem acessíveis a ambas);
D) Devem ser constantes globais;
E) Nenhuma das anteriores.

5) A pesquisa binária é tipicamente mais rápida do que a pesquisa sequencial.

A) A pesquisa binária só é mais rápida se o vetor não estiver ordenado caso contrário é melhor pesquisar sequencialmente.
B) A pesquisa binária só é mais rápida para vetores de grandes dimensões (mais de 1000 elementos).
C) Para um vetor ordenado de mil elementos a pesquisa binária é mais de dez vezes mais rápida (em média) do que a pesquisa sequencial.
D) Normalmente é duas vezes mais rápida pois divide em cada iteração o vetor em duas partes.
E) Nenhuma das anteriores.

6) Considere a classe base `CPolygon` e a classe derivada `CRectangle` declaradas no seguinte fragmento de código:

```
1   class CPolygon {
2     protected:
3       int width, height;
4     public:
5       void set_values (int a, int b) {
              width=a; height=b; }
6   };
7   class CRectangle: public CPolygon {
8     public:
9       int area () { return (width * height); }
10  };
11  int main () {
12    CRectangle rect;
13    CPolygon *ppoly1 = &rect;
14    ppoly1->set_values (4,5);
15    cout << rect.area() << endl;
16    return 0;
17  }
```

A) O programa está correto;
B) Para ficar correto é necessário alterar a instrução: '`ppoly1->set_values(4,5);`' para a instrução: '`ppoly1.set_values(4,5)`';
C) Para o programa ficar correto é necessário alterar a instrução '`ppoly1->set_values(4,5);`' para a instrução: '`rect->set_values(4,5);`'
D) Para o programa ficar correto é necessário alterar a instrução: '`cout << rect.area() << endl;`' para: '`cout << ppoly1.area() << endl;`'
E) Nenhuma das anteriores

7) Quando se usa um `vector` de inteiros em C++ (usando `#include <vector>`) declarado como '`vector <int> v;`' e se pretende adicionar um dado elemento ao `vector`:

A) Tem que se realizar, quando se adiciona um elemento, um `resize(v)` para garantir que existe memória para inserção de cada um dos novos elementos (memória dinâmica);
B) Pode-se usar o operador "<<" para colocar esses mesmos elementos no `vector`, i.e.: '`v << elemento;`'
C) Pode-se utilizar a instrução '`v.push_back(elemento)`'.
D) Pode-se usar o operador ">>" para colocar o `elemento` no `vector`, fazendo: '`elemento >> v;`'
E) Nenhuma das anteriores.

8) Relativamente aos Construtores de classes em C++:

A) Construtores são funções membro especiais chamadas pelo sistema no momento da criação de um objeto que possuem sempre um valor de retorno;
B) Podem ser criados vários construtores mas tem de existir sempre um destrutor por cada construtor criado;
C) Tem de ser único pois é o responsável pela construção do objeto;
D) O construtor tem de ter um nome diferente do objeto que constrói (de modo a não se confundir com este)
E) Nenhuma das anteriores.

9) Suponha que deseja representar pessoas, carros e datas.

A) Devem ser utilizadas três classes distintas mas não faz sentido utilizar o conceito de herança;
B) Devem ser utilizadas classes: pessoa, carro e data, sendo que pessoa pode ser derivada de data e de carro dado que cada pessoa pode ter um carro e tem uma data de nascimento;
C) O mais lógico será utilizar uma classe pessoa derivada de data e veículo também derivada de data (dado que os veículos têm data de construção.
D) Dado que as pessoas têm carros e datas o melhor será representar tudo na mesma classe;
E) Nenhuma das anteriores.

10) Dois membros-função de uma dada classe com o mesmo nome, mas diferente número de argumentos:

A) São funções distintas;
B) Constituem um exemplo de sobrecarga (overload) de funções;
C) São herdados de uma outra classe base;
D) Não podem ser implementados, a menos que tenham diferentes tipos de retorno;
E) Nenhuma das anteriores.

11) Dois membros-função de duas classes distintas em que uma classe é derivada da outra com o mesmo nome, argumentos e tipo de retorno:

A) São amigos (friend) entre si;
B) São funções virtuais de classes abstratas;
C) São herdados de uma outra classe base;
D) Não podem ser implementados, a menos que tenham diferentes tipos de retorno;
E) Nenhuma das anteriores.

12) A classe vector da STL de C++ é um tipo abstrato de dados.

A) Sim, mas só se se declarar no código fonte que vector é do tipo abstrato.
B) Não, pois o tipo de dados abstrato é a lista (list da STL).
C) Sim, pois a classe contém dados e operações sobre esses dados.
D) Não, pois é um dos tipos base de dados do C++ (o array).
E) Não pois encapsula os dados e operações.

13) Considere a seguinte definição do objeto nomes: `vector<string> nomes;` Qual poderá ser o protótipo da função `pesq()`, que tenta averiguar se um dado nome já existe num vetor de nomes?

```
A)  bool pesq(const vector<string> v1, const object
    nom);
B)  void pesq(vector<char *> lista, const str x);
C)  bool pesq(const vector<string>, const string);
D)  bool pesq(vector<string>, void string);
E)  Nenhuma das respostas anteriores.
```

14) Qual é o método de ordenação preferível para vetores de média/grande dimensão?

A) Ordenação por Partição (QuickSort) pois tem complexidade temporal O(n*log n);
B) Ordenação por Inserção pois tem complexidade temporal o(n*log n);
C) Ordenação por MergeSort pois tem uma complexidade espacial melhor que o Quicksort;
D) Ordenação por BubbleSort pois o processo de ordenação deste método é muito rápido;
E) Nenhuma das Anteriores

15) Uma classe pode herdar dados e métodos de mais do que uma classe base?

A) Pode herdar dados e métodos somente de uma única classe base;
B) Pode herdar dados e métodos de uma ou duas classes base;
C) Pode herdar dados e métodos de várias classes base;
D) Pode herdar dados e métodos de várias classes desde que a herança seja privada (private) e não pública (public)
E) Nenhuma das anteriores

GRUPO II (10.0 Valores)

2) Considere uma imobiliária que possui um software de gestão desenvolvido em C++ que inclui uma classe `CImobiliaria` cuja interface está definida no ficheiro `imobiliaria.h` que a seguir se apresenta. O nome das funções e das variáveis indica a sua funcionalidade no programa. Na rotina `'main()'` do programa de gestão de informação da imobiliária, entre outras coisas, consta a declaração (e definição) `'CImobiliaria imob;'`. Suponha que foram incluídos todos os `#include` necessários.

```
1  class CImovel {
2    friend ostream &operator<<(ostream&, const
3               CImovel&);
4    public:
5       CImovel(string, string, string, long int, long
6               int);
7       string getTipo() const;
8       string getNomeProp() const;
9       string getTelefProp() const;
10      long int getValorPedido() const;
11      long int getValorMinimo() const;
12   private:
13      string tipo; //vivenda, apartamento, terreno...
14      string nomeProp, telefProp;
18      long int valorPedido, valorMinimo;  // em euros
19 };
20
21 class CImobiliaria {
22      vector <CImovel> imoveis;
23   public:
24      void adicionarImovel();
25      void ordenarImobiliaria();
26      int procuraImovelProp(string);
27      void removeImovelProp(string);
28 };
```

2.1) Escreva o código da função `long int getValorPedido() const;` que retorna o valor de um dado imóvel.

2.2) Escreva o código da função `'void adicionarImovel()'`, que acrescenta ao vector `imoveis` a informação sobre um novo imóvel. Nota: Esta informação é introduzida pelo utilizador na função.

2.3) Escreva o código da função `friend ostream &operator<<(ostream &, const CImovel &)`, que permite efetuar o *overloading* do operador `<<` na classe `CImovel`. Esta função deve colocar na *stream* de retorno o valor contido em cada um dos atributos privados da classe `CImovel`.

2.4) Escreva o código da função `int procuraImovelProp(string nome)` que procura no vector `imoveis` o primeiro imóvel cujo proprietário é dado pela string `nome`. A função deve retornar a posição do referido imóvel no vector ou retornar o valor -1 caso o proprietário não exista.

2.5) Escreva o código da função `void removeImovelProp(string nome)` que remove do vector `imoveis` todos os imóveis que têm como proprietário `nome`. **Sugestão**: use a função da alínea 2.4, mesmo que não a tenha implementado.

2.6) Escreva o código da função '`void ordenarImobiliaria()`', que altera o vector com a informação dos imóveis, ordenando-o internamente segundo o valor pedido dos imóveis, do mais para o menos valioso (decrescente).

2.7) Escreva o código de uma nova função membro da classe `CImobiliaria`, `gravarImobiliaria` que receba o nome de um ficheiro (por exemplo "`imobVidaSimples.txt`") e grave o conteúdo do vector `imoveis` ordenado segundo o valor dos imóveis. **Sugestão**: use as funções e operadores das alíneas 2.3 e 2.6, mesmo que não os tenha implementado.

2.8) Suponha que deseja criar uma classe derivada `CVivenda` (sendo que interessa saber a área da vivenda e o nome dos diversos edifícios que a compõem -strings). A classe deve ter um construtor que construa uma vivenda (recebendo como parâmetros a `área`, `nomeProp`, `telefProp`, `valorPedido` e `valorMinimo` e supondo que não tem ainda qualquer edifício introduzido. Apresente a declaração da classe `CVivenda`, incluindo o respetivo construtor.

GRUPO III (4.0 Valores)

Um programa de gestão de uma equipa de futebol robótico pretende guardar informação sobre o campo, jogo, jogadores e bola.

- Cada equipa tem uma identificação, nome e é composta por 11 jogadores.
- Em cada instante, para cada jogador pretendemos saber o nome, a posição (x,y), a velocidade (vx, vy), a orientação (teta) e a energia (valor real entre 0.0 e 100.0). Para a bola só pretendemos saber a identificação/nome, posição (x,y) e a velocidade (vx, vy). Para os objetos e marcadores fixos do campo (exemplo: postes das balizas) pretendemos saber o nome e posição (x,y).Deste modo temos objectos, sendo que alguns são móveis e sendo que nos móveis alguns são humanos.
- Pretende-se armazenar o filme do jogo que contém as posições, velocidades e outras informações de todos os objetos móveis (jogadores de ambas as equipas e bola) ao longo do jogo (4500 informações: 90 min - 60 informações por minuto).
- Cada jogo, para além do filme respetivo possui as identificações das equipas envolvidas, data (dia, mês e ano), hora exata de início (hora, min, seg) e campo onde decorreu (identificação).
- Cada campo tem uma identificação, nome, identificação do clube a que pertence e dimensões (comprimento e largura).

Defina uma hierarquia de classes que na sua opinião melhor descreve o cenário acima, especificando os dados e métodos de cada classe. Justifique as suas escolhas. *Nota: Não é necessário implementar qualquer método, simplesmente definir os ficheiros *.h das classes respetivas.*