Universidade do Minho	**Escola de Engenharia da Universidade do Minho** Mestrado Integrado em Eng. Electrónica Industrial e Computadores Programação	**2011/2012** **MIEEIC** **(1º Ano)** **2º Sem**
Teste Nº1 - Época Normal - Data 19/04/2012, Duração 1h45m		

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

GRUPO I (5 Valores)

1) Suponha a seguinte classe `CPonto`:

```
1  class CPonto {
2   private:
3     double x;
4     double 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) As linhas 3 e 4 declaram dois membros função da classe `CPonto` (uma função `double x` e outra `double y`).
B) As linhas 6 a 12 declaram os sete membros dados e membros função da classe.
C) A classe `CPonto` tem três construtores (linhas 6, 7 e 8) e um destrutor (linha 9).
D) A linha 13 tem um erro (ponto e virgula final)
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    }
4  void CPonto::Escala(double fx, double fy) {
5       x *= fx; y *= fy;
6    }
7  double CPonto::DistAte(const CPonto&out){
8     return sqrt((out.x-x)*(out.x-x)+
            (out.y-y)*(out.y-y));
9  }
```

Qual das seguintes alternativas é verdadeira:

A) As linhas 2 e 5 contêm erros pois os operadores `+=` e `*=` não estão definidos para a classe `CPonto`.
B) A linha 8 contém um erro pois `x` e `y` não estão definidos na função.
C) As linhas 1, 4 e 7 contêm erros pois `CPonto::` só é utilizado fora das classes e não na definição de uma função de uma classe.
D) As linhas 3, 6 e 9 contêm erros pois deviam ter um ponto e virgula após a chaveta.
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     }
16     friend class CRectangle;
17 };
```

A) A função `friend class` (linha 16) não está correctamente declarada;
B) A função `convert_square` (linhas 4 a 6) não pode usar o membro-dado `side`, da classe `CSquare`, por isso está erradamente declarada;
C) A função `convert_rect` (linhas 13 a 15) não pode usar os membros-dado `width` e `height`, da classe `CRectangle`, 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) 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.

6) Supondo as classes `CPolygon` e `CRectangle` definidas na alínea anterior:

A) Resulta em erro de compilação fazer:
```
CPolygon *ppoly1 =
    new CRectangle; ppoly1->set_values(4,5);
```
B) Resulta em erro de compilação fazer:
```
CPolygon *ppoly2 =
    new CPolygon; ppoly2->set_values(2,2);
```
C) Resulta em erro de compilação fazer:
```
CRectangle *rect1 =
    new CRectangle; rect1->set_values(2,3);
```
D) Resulta em erro de compilação fazer:
```
CPolygon ppoly3; ppoly3.set_values(1,3);
```
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 (utilização de 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 carros, motorizadas e camiões, ou seja veículos em geral.

A) Devem ser utilizadas quatro classes distintas mas não faz sentido utilizar o conceito de herança;
B) Devem ser utilizadas classes: carro, motorizada e camião, sendo que camião dado o seu tamanho superior é uma superclasse;
C) O mais lógico será utilizar uma classe veículo e três classes derivadas desta (carro, motorizada e camião).
D) Dado que os carros, camiões e motorizadas são muito semelhantes será melhor representar tudo na mesma classe.
E) Nenhuma das anteriores.

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

A) São amigos (friend) entre si;
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) 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.

GRUPO II (10 Valores)

2) Suponha o seguinte código representando a interface de um Jardim que se encontra contido num ficheiro `Jardim.h` e se destina a gerir a informação sobre todas as plantas de um Jardim/Horto. O nome das funções e das variáveis indica a sua funcionalidade no programa. Na rotina `'main()'` do programa de gestão de informação do Jardim, entre outras coisas, consta a declaração (e definição) `'CJardim jard;'`. Suponha que foram incluídos todos os #include necessários.

```
1  // #includes e declarações gerais
2  const int MINVALOR = 0;
3  const int MAXVALOR = 10000;
4
5  class CPlanta {
6    protected:
7       string nome;
8       string especie;
9       int valor, custo;   // em euros
10   public:
11      CPlanta(string, string, int, int);
12      string getEspecie() const;
13      string getNome() const;
14      int getValor() const;
15      int getCusto() const;
16 };
17
18 class CJardim{
19      vector <CPlanta> plantas;
20    public:
21      void adicionarPlanta(string, string,
             int, int);
22      void imprimirJardim(ostream &);
23      void ordenarJardim();
24 };
```

2.1) Escreva o código da função `int getValor() const;` que retorna o valor de uma dada planta.

2.2) Escreva o código da função `'void adicionarPlanta(string nom, string esp, int val, int cus)'`, que acrescenta à estrutura de dados a informação sobre uma nova planta adquirida.

2.3) Escreva o código de `'imprimirJardim(ostream &os)'`, função que, se invocada em `'main()'` como `'jardim.imprimirJardim(cout);'`, vai permitir mostrar no ecrã do computador a lista de plantas de `'jardim'`, mostrando para cada uma a sua espécie, nome, valor e custo.

2.4) Escreva o código da função `'void ordenarJardim()'`, que altera a estrutura de dados com a informação das plantas, ordenando-a internamente segundo o valor das plantas, da menos para a mais valiosa.

2.5) Implemente um novo operador == que compara duas plantas retornando verdadeiro caso tenham o mesmo nome, espécie, valor e custo.

2.6) Escreva o código de uma nova função membro da classe `CJardim`, `gravarJardim` que receba o nome de um ficheiro (por exemplo `"jardimEstrela.txt"`) e grave o conteúdo do jardim ordenado segundo o valor das plantas e não incluindo plantas repetidas. Sugestão: use as funções e operadores das alíneas 2.4 e 2.5 (mesmo que não os tenha implementado)

2.7) Construa uma função membro da classe `CJardim` designada: `void substJardim(string nomeParte, string nomeSubst);` que substitua no nome de todas as plantas do jardim que incluam `nomeParte`, essa parte do nome por `nomeSubst`. Exemplo `substJardim("Brasil", "América")` transforma `"Cana do Brasil"` em `"Cana da América"`, `"Planta Brasil Nova"` em `"Planta América Nova"` e assim por diante.

GRUPO III (5.0 Valores)

Um programa de gestão de um clube desportivo guarda informação sobre o seu *pessoal*, que inclui *atletas*, *professores* e *sócios*. Sabendo que:

- Todo o *pessoal* do clube desportivo é identificado por um código único e sequencial e possui um nome e uma data de admissão no clube.
- Existem diversas *turmas*, cada qual identificada por uma letra. Cada *turma* tem uma única *modalidade*. A *turma* deve especificar o seu horário semanal composto por várias *aulas* (número de aulas indefinido). *Sugestão: use um vetor de aulas.*
- Cada *aula* tem um dia da semana, uma hora de início e uma duração.
- Existem diversas *modalidades*, cada qual com um número e uma designação.
- Os *atletas* estão inscritos numa única *modalidade* e numa determinada *turma*.
- Os *professores* lecionam uma determinada modalidade a várias *turmas* e têm um salário. Lecionam todas as aulas da respetiva turma.
- Os *sócios* pagam uma determinada quota mensal e podem ser de três tipos: normal, privilegiado e vip.

Implemente a hierarquia de classes que na sua opinião melhor descreve o cenário acima, especificando os dados e métodos de cada classe. Justifique a sua escolha. *Nota: Não é necessário implementar qualquer método, simplesmente definir os ficheiros *.h das classes respectivas.*