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**
Exame Final Época de Recurso - Data 11/07/2013, Duração 2h00m		

Nome:_____ Nº Al:_____

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	16

GRUPO I (6 Valores)

1. Qual a melhor definição do conceito de Herança:

A) Uma classe, chamada derivada, pode ser derivada de outra classe (classe base). Esta classe herda todos os métodos da classe base e permite realizar operações conjuntas em que métodos partilhados são aplicados (i.e. simulataneamente são aplicados os métodos de ambas as classes verificando o sistema qual é o melhor).
B) Uma classe, chamada derivada, pode reutilizar parte ou todas as propriedades e comportamentos de outra classe (classe base). Métodos da classe base pode ser estendidos pela adição de novos propriedades ou métodos.
C) Uma classe, chamada subclasse é criada utilizando as propriedades da classe base. Esta classe tem menor dimensão do que a classe base e pode ser utilizada de modo mais simples.
D) Uma classe, chamada subclasse é criada utilizando as propriedades da classe base. Esta classe herda as propriedades e métodos privados da classe base que depois podem ser utilizados na classe derivada.
E) Nenhuma das respostas anteriores.

2. Defina Encapsulamento:

A) Encapsulamento refere-se a colocar uma proteção (plástica ou fibra de vidro) que encapsula o objecto evitando ataques externos.
B) Encapsulamento refere-se à colocação de cápsulas nos objectos de modo a que estes não possam ser acedidos exteriormente.
C) Parte da informação sobre um objeto está escondida do seu utilizador. O encapsulamento isola a funcionalidade interna do resto da aplicação.
D) Os objectos ficam estanques de modo a que a sua permeabilidade é reduzida. Deste modo a herança é mais simples e pode ser realizada de modo privado.
E) Nenhuma das respostas anteriores.

3. Uma pilha (stack) e uma fila (queue) de objectos:

A) Permitem implementar árvores binárias.
B) São tipos de dados disponíveis na Standard Template Library.
C) São tipos de dados standard do C++.
D) Têm todos os métodos de acesso tipicamente utilizados nas listas ligadas.
E) Nenhuma das respostas anteriores.

4. Quais dos seguintes ciclos são infinitos (i.e. nunca terminam):

```
a. while (0) {}
b. for (;;) {}
c. do {} while{1};
d. for (i=0;i<1;i++) {}
e. while (1) {}
```

A) Os ciclos b, c, d, e
B) Os ciclos a, b, c, e
C) Os ciclos a, b, c
D) Todos!
E) Nenhuma das anteriores

5. No método de ordenação por partição (Quick Sort):

A) No caso médio (valores aleatoriamente distribuídos) tem complexidade no tempo O(n*log(n)) e complexidade no espaço O(n*log(n)).
B) Com um método de escolha do Pivot que retorne o valor menor do array, tem complexidade no tempo $O(n^2)$ e complexidade no espaço O(n).
C) Com um método de escolha do Pivot que retorne a mediana do array, tem complexidade no tempo O(n*log(n)) e complexidade no espaço O(1).
D) No caso médio tem uma complexidade no espaço de ordem inferior à ordenação por "buble sort" pois ocupa menos memória.
E) Nenhuma das anteriores.

6. Relativamente aos Construtores de classes em C++:

A) São funções membro especiais chamadas pelo sistema no momento da criação de um objeto. Podem existir vários construtores com parâmetros distintos para um dado objecto.
B) O construtor tem de ser único pois é o responsável pela construção do objeto;
C) Podem ser criados vários construtores mas tem de existir sempre um destrutor por cada construtor criado;
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.

7. No método de ordenação por partição (Quick Sort) para ordenar um vetor X com um total de 9999 elementos distintos com valores entre 1 e 9999 (exemplo códigos de alunos da Universidade do Minho), qual o valor ideal para o primeiro pivot?

A) X[5000]
B) 5000
C) (X[1]+X[9999])/2
D) 9999
E) Nenhuma das Anteriores

8. Considere o seguinte fragmento de código C++:

```
vector<int> v1; int n;  cin >> n;
for(int i=1; i<10; i++)
     v1.push_back(i);
```

A complexidade temporal do fragmento de código é:

A) T(n)=O(n*log(n)).
B) T(n)=O(n^3).
C) T(n)=O(n^2).
D) T(n)=O(n).
E) Nenhuma das anteriores.

9. Considere o seguinte fragmento de código C++:

```
int n; cin >> n;
for(int i=0; i<n*n; i++)
   for(int j=1; j<n; j*=2) {
       cout << i << " - " << j << endl;
   }
```

A complexidade temporal do fragmento de código é:

A) T(n)=O(n*log(n)).
B) T(n)=O(n^2*log(n)).
C) T(n)=O(n^2).
D) T(n)=O(n).
E) Nenhuma das anteriores.

10. Considere o seguinte fragmento de código C++:

```
vector<int> v1; int n; cin >> n;
for(int i=0; i<n; i++)
   for(int j=n; j<n+100 ; j++)
      for(int k=n; k>2 ; k = k/2)
         v1.push_back(i+j+k);
for(int i=0; i<n; i++)
   for(int j=i; j<i+2 ; j++)
      for(int k=5; k<n; k+=2)
         v1.push_back(i+j+k);
```

A complexidade temporal do fragmento de código é:

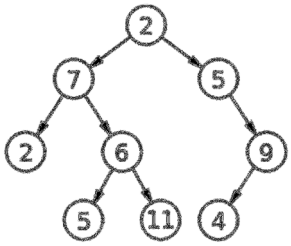
A) A complexidade temporal é T(n)=O(n*log(n)).
B) A complexidade temporal é T(n)=O(n^2*log(n)).
C) A complexidade temporal é T(n)=O(n^2).
D) A complexidade temporal é T(n)=O(n^3).

Nenhuma das anteriores.

11. 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.

Suponha a seguinte árvore binária (não balanceada):



12. A travessia em pós-ordem da árvore resulta na sequência:

A) 2 7 2 6 5 11 5 9 4
B) 2 5 11 6 7 4 9 5 2
C) 2 7 5 6 11 2 5 4 9
D) 2 7 6 5 11 5 9 4 2
E) Nenhuma das anteriores.

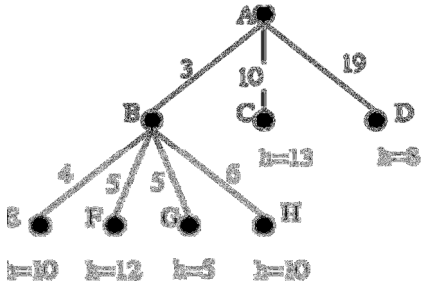
13. A travessia em-ordem da árvore resulta na sequência:

A) 2 7 2 6 5 11 5 9 4
B) 2 5 11 6 7 4 9 5 2
C) 2 7 5 6 11 2 5 4 9
D) 2 7 6 5 11 5 9 4 2
E) Nenhuma das anteriores.

14. A operação de pesquisa de um elemento numa lista implementada utilizando listas ligadas:

A) Possui complexidade temporal T(n)=O(n^2).
B) Possui complexidade temporal T(n)=O(n).
C) Possui complexidade temporal T(n)=O(1).
D) Não é possível utilizando listas ligadas pois os nós teriam de estar em posições consecutivas.
E) Nenhuma das anteriores.

Supondo a seguinte árvore de pesquisa em que cada arco apresenta o custo do operador correspondente e h é uma função heurística que estima um custo para a solução,



15. Diga qual o nó expandido em seguida utilizando cada um dos seguintes métodos: i) Pesquisa em largura; ii) Pesquisa Gulosa/Ganciosa;

A) i) Nó C ii) Nó C
B) i) Nó E ii) Nó D
C) i) Nó C ii) Nó F
D) i) Nó C ii) Nó G
E) Nenhuma das anteriores.

16. Diga qual o nó expandido em seguida utilizando o Algoritmo A*:

A) Nó C
B) Nó D
C) Nó E
D) Nó G
E) Nenhuma das anteriores.

GRUPO II (9 Valores)

2. Pretende-se implementar um programa de gestão de uma clínica veterinária. Entre outras, existem as classes `Tratamento` e `Animal`. A classe `Tratamento` inclui atributos e métodos relativos a tratamentos efectuados. A classe `Animal` inclui atributos e métodos relativos aos animais. Existem uma relação de composição entre as duas classes. Implemente os métodos especificados em cada uma das alíneas. Suponha que foram feitos todos os `#include` necessários e que todos os atributos privados têm métodos set e get definidos. A declaração das classes inclui o seguinte código:

```
int numAnimais = 0;
class Tratamento {
  public:
    Tratamento(int=1, int=1, int=1900, int=0);
    void setTratamento(string="", string="",
                       bool=false, int=0);
    bool operator==(const Tratamento &);
    //... outros metodos
  private:
    int dia, mes, ano, hora;
    string problema, solucao;
    bool resolvido; //true - sim; false - Nao
    int preco;
};
class Animal {
  public:
    Animal(string="", string="", string="",
           string="", string="")
    void adicionaTratamento(int=1, int=1, int=1900,
                            int=0);
    int procuraTratamento(Tratamento &);
    //...outros metodos
    vector <Tratamento> tratamentos;
  private:
    int codigo;
    string nome, nomeDono, telefDono, especie, raca;
};
```

2.1) Implemente o construtor `Animal::Animal(string n, string nomeD, string telefD, string esp, string rac)` O construtor deve atribuir um código ao animal. O código (variável codigo) é um inteiro que é atribuído sequencialmente a cada animal que é registado. A variável numAnimais contém o valor relativo ao número de animais registados, pelo que deve ser incrementada quando um novo animal se regista (detém o código do último animal registado).

2.2) Implemente o construtor `Tratamento(int d, int m, int a, int h)` da classe `Tratamento`, que recebe como parâmetros o dia, mês, ano e hora do tratamento. Considere que os valores dos restantes atributos são inseridos posteriormente recorrendo ao método `setTratamento`.

2.3) Implemente método `adicionaTratamento(int d, int m, int a, int h)` que adiciona um novo tratamento na primeira posição do vector `tratamentos`, chegando todos os restantes tratamentos para a posição seguinte.

2.4) Implemente o *overloading* do operador "==" que permite comparar as datas e horas de dois tratamentos. (deve resultar `true` apenas se as datas e as horas forem iguais)

2.5) Implemente o método `procuraTramento(Tratamento tata&)`, que retorna o índice do tratamento no vector `Tratamentos`. Retorna -1 caso o tratamento não exista. (Sugestão: utilize o operador "==" sobrecarregado)

2.6) Substitua o vector por uma stack da STL.

2.6.a) Declare a variável `tratamentos`;

2.6.b) Implemente um novo método da classe `Animal` `removeTratamentoStack()` que remove um tratamento da stack. O método deve gerar uma mensagem de erro caso a stack esteja vazia.

2.6.c) Implemente um novo método `copiaTratamentos(stack<Tratamento> &trata)` que recebe como parâmetro uma stack de tratamentos e a concatena com a stack `Tratamentos`.

2.6.d) Implemente um novo método `guardaAnimalFichStack()` que insere no ficheiro com nome `animal.txt` a informação relativa ao animal e seus tratamentos.

2.7) Substitua o vector por uma lista ligada. A classe Animal terá um apontador `head` para o início da lista e a classe Tratamento terá um apontador `next` para o próximo nodo. Considere que ambos os apontadores foram inicializados a NULL.

2.7.a) Implemente o método `adicionaTratamentoLista(int d, int m, int a, int h)` que adiciona um novo tratamento na primeira posição da lista ligada `tratamentos`.

2.7.b) Implemente o método `procuraTramentoLista(Tratamento &trata)`, que retorna o apontador para o nodo da lista que se situa na posição imediatamente anterior ao do tratamento procurado. Retorna -1 caso o tratamento não exista. (Sugestão: utilize o operador "==" sobrecarregado)

2.7.c) Implemente o método `removeTratamentoLista(int d, int m, int a, int h)` que remove um determinado tratamento da lista. Se o tratamento não existir é enviada uma mensagem de erro ao utilizador. (Sugestão: utilize o método desenvolvido na alínea 2.7.b), mesmo que não o tenha implementado)

GRUPO III (5 Valores)

3. Considere a classe quadrado representada a seguir:

```
class Quadrado {
public:
  virtual double getArea() const { return 0.0; }
  virtual double getVolume() const {return 0.0;}
private:
  double lado;
};
```

Considere a classe `Cubo`, subclasse de `Quadrado`.

3.1) Declare a classe `Cubo` (Cubo.h).

3.2) Implemente o método virtual `getVolume()` da classe `Cubo`.

3.3) Suponha que agora está a trabalhar na `main()`. Declare um vector `formas` que tanto albergue quadrados como cubos.

3.4) Supondo que o vector formas está declarado, escreva o código necessário à introdução de informação sobre um quadrado e dois cubos no vector.

3.5) Supondo que o vector formas está declarado, escreva o código necessário à listagem de todos os elementos do vector, incluindo informação sobre a área e volume da cada figura geométrica.