Uma universidade pretende implementar uma aplicação em C++, baseada em classes e objectos, que guarde informação sobre as suas actividades, incluindo dados de projectos e investigadores.
Uma classe abstracta **Identificador** servirá de base a todas as restantes classes e guardará o código e o nome de cada entidade. O código é um número inteiro par único para cada entidade e atribuído sequencialmente a partir de 1000. Um protótipo da classe abstracta **Identificador** é mostrado a seguir:

```
class Identificador {
protected:
    static int id;
    const int codigo;
    string nome;
public:
    Identificador(string nome);
    virtual ~Identificador();
    virtual string Tipo() = 0;
    virtual void Imprime();
    int Codigo() const;
    string Nome();
};
```

Cada **Projecto** tem uma verba máxima atribuída. Cada **Investigador** é caracterizado também pela idade e tem uma lista de Projectos aos quais está ligado. O membro-função **Tipo( )** devolve o nome de cada classe existente. O membro-função **Codigo( )** devolve o código do identificador de cada classe. O membro-função **Imprime( )** regista no monitor todas as características de cada classe. Sempre que possível deve fazer-se uso das funções já definidas.

[2 val] **a)** Implemente o código dos membros-função da classe abstracta **Identificador**. Um identificador deve ter um código único par e sequencial a começar em 1000.

[2 val] **b)** Especifique a classe **Projecto**, enumerando os seus membros-dado, implementando os construtores, todos os membros-função necessários e o membro-função **adicionaVerba(float quantia)** que adiciona à verba do projecto o valor **quantia** recebido como mecenato científico.

void adicionaVerba (float quantia)

[2 val] **c)** Implemente a classe **Investigador**, enumerando os membros-dado, implementando os construtores, os membros-função necessários e o membro-função **adicionaProjecto(Projecto* proj)** que adiciona o projecto *proj* à lista de projectos do investigador.

*void adicionaProjecto (Projecto *proj).*

[2 val] **d)** Implemente na classe **Projecto** o operador > :

bool operator > (const Projecto &proj2) const

Considere que um projecto é maior que outro se a verba atribuída ao projecto for superior.

[2 val] **e)** Implemente o método **ImprimeTudo( )** da classe **Universidade**, que imprime a informação de todas as entidades pertencentes à Universidade. Deve utilizar o método **Imprime( )** de cada uma das entidades.

Teste o código realizado com o programa principal contido no ficheiro "programaTeste.cpp".
Ficheiros anexos:
Universidade.h – protótipo da classe abstracta **Identificador** e da classe **Universidade**
Universidade.cpp – implementação parcial da classe **Universidade**
programaTeste.cpp – programa de teste

[2 val] **f)** Sobre sobrecarga de funções, é correcto afirmar:

A) Acontece quando há duas ou mais funções com o mesmo comportamento mas nomes diferentes;
B) Duas ou mais funções podem ter o mesmo nome, mas os tipos de retorno devem ser diferentes;
C) Duas ou mais funções podem ter o mesmo nome e os mesmos argumentos, mas os nomes das variáveis de argumento devem ser diferentes;
D) Em C++ não é permita a sobrecarga de funções, mas apenas a sobrecarga de operadores;
E) Nenhuma das anteriores.

[2 val] **g)** Relativamente à herança em C++:

A) Uma classe que estenda outra classe herda apenas os membros-função e membros-dado da classe base que estejam declarados como public.
B) Uma classe pode estender múltiplas classes, herdando os membros-função e membros-dado declarados como public ou protected nas classes base.
C) O construtor de uma subclasse tem sempre de invocar explicitamente o construtor da sua classe base.
D) Uma classe não pode estender mais que uma classe base. No entanto a sua classe base pode ser subclasse de outra.
E) Nenhuma das anteriores.

[2 val] **h)** Sobre membros-função destrutores, é correcto afirmar que:

A) São utilizados para evitar acesso indevido a membros privados de uma classe;
B) Podem receber argumentos, ao contrário dos construtores;
C) Executam o comando delete[] como última instrução;
D) Servem normalmente para libertar recursos (memória, p. ex.) associados aos objectos;
E) Nenhuma das anteriores.

[2 val] **i)** Pode afirmar-se que os membros-dado estáticos de uma classe em C++:

A) São variáveis que fazem parte da classe mas não dos objectos da classe
B) Criam um apontador em todos os objectos da classe para o seu valor
C) Guardam cópias diferentes, com o mesmo valor, em cada objecto da classe
D) Podem ser declarados e definidos fora da definição da classe
E) Nenhuma das anteriores.

[2 val] **j)** Se uma classe **Aeroporto** for definida como "friend" de uma classe **CompanhiaAerea**, então:

A) Aeroporto tem acesso apenas aos membros privados de CompanhiaArea;
B) É necessário apenas definir métodos de leitura para atributos privados de CompanhiaAerea;
C) É necessário apenas definir métodos de escrita para atributos privados de Aeroporto;
D) CompanhiaAerea herda todos os membros privados de Aeroporto;
E) Nenhuma das anteriores.