3ª aula prática – Herança e subclasses; Polimorfismo; Membros Estáticos; Documentação (Doxygen)

Faça download do ficheiro ***aeda1819_fp03.zip*** da página da disciplina e descomprima-o (contém os ficheiros ***zoo.h***, ***zoo.cpp***, ***animal.h***, ***animal.cpp***, ***veterinario.h***, ***veterinario.cpp***, ***Test.cpp*** e ***vets.txt***)

- Note que os testes unitários deste projeto estão comentados. Retire os comentários à medida que vai implementando os testes.
- *Deverá realizar esta ficha respeitando a ordem das alíneas*. Poderá executar o projeto como CUTE Test quando quiser saber se a implementação que fez é suficiente para passar no teste correspondente.

Enunciado

Pretende-se escrever um programa em C++ para gestão de um Jardim Zoológico. Por uma questão de simplificação, admita que o jardim zoológico só possui cães e morcegos. O programa deve conter as classes a seguir descritas parcialmente:

```
class Animal {
 protected:
    string nome;
    int idade;
    Veterinario *vet;
    static int maisJovem;
public:
    // nome, idade do animal,  info de veterinário
    virtual string getInformacao() const;
    virtual bool eJovem() const = 0;
    static int getMaisJovem();
};
```

```
class Zoo {
    vector<Animal *> animais;
    vector<Veterinario *> veterinarios;
  public:
    void adicionaAnimal(Animal *a1);
    void alocaVeterinarios(istream &isV);
    string getInformacao() const;
    bool animalJovem(string nomeA);
 };
```

```
class Voador {
    int velocidade_max;
    int altura_max;
public:
    //imprime também valor velocidade e altitude máxima
    virtual string getInformacao() const;
};
```

```
class Cao: public Animal {
    string raca;
public:
    //...
};
```

```
class Morcego: public Animal, public Voador {
            // ...
};
```

```
class Veterinario {
    string nome;
    long codOrdem;
    //...
};
```

a)

- Implemente os construtores de todas as classes.
- Implemente também o **membro função estático**

  `int Animal::getMaisJovem()`

  que retorna a idade do mais jovem dos animais. Atualize convenientemente o membro estático `Animal::maisJovem`
- Implemente também, nas classes respetivas, o membro função: `bool eJovem()const`

  Considere que um cão é jovem se tiver menos de 5 anos, e um morcego é jovem se tiver menos de 4 anos.

b) Implemente o membro-função

```
void Zoo::adicionaAnimal(Animal *a1)
```

que adiciona o animal *a1 ao conjunto de animais do zoo (vetor *animais*).

c) Implemente o membro-função

```
string Zoo::getInformacao() const
```

que retorna numa *string* a informação sobre todos os animais do Zoo. A informação sobre um animal é o conjunto dos valores de todos os seus membros dado (incluindo o veterinário).

Nota: este teste não falha, deve verificar a escrita correta da informação.

d) Implemente o membro-função

```
bool Zoo::animalJovem(string nomeA)
```

que verifica se o animal de nome *nomeA* existe e é jovem.

e) Implemente o membro-função

```
void Zoo::alocaVeterinarios(istream &isV)
```

Este método carrega a informação sobre os veterinários existentes no ficheiro *isV* que contém a informação sobre cada veterinário em duas linhas: nome e código da Ordem (para teste, use o ficheiro *vets.txt*). Os veterinários devem ser distribuídos uniformemente para os animais do zoológico.

f) Adicione o membro-função

```
bool Zoo::removeVeterinario (string nomeV)
```

que remove o veterinário de nome *nomeV* do vetor de veterinários do Zoo. Se algum animal tiver este veterinário, deve passar para o veterinário seguinte do vetor *veterinarios*. Se não existir nenhum veterinário de nome *nomeV*, esta função retorna ***false***.

g) Implemente o operador

```
bool Zoo::operator < (Zoo& zoo2) const
```

para comparar dois jardins zoológicos. Um jardim zoológico é menor que um segundo se a soma das idades dos seus animais for inferior à soma das idades dos animais do segundo zoo.

h) Efetue a documentação dos membros-função implementados (use Doxygen).