Programación Avanzada 2025

Laboratorio 1

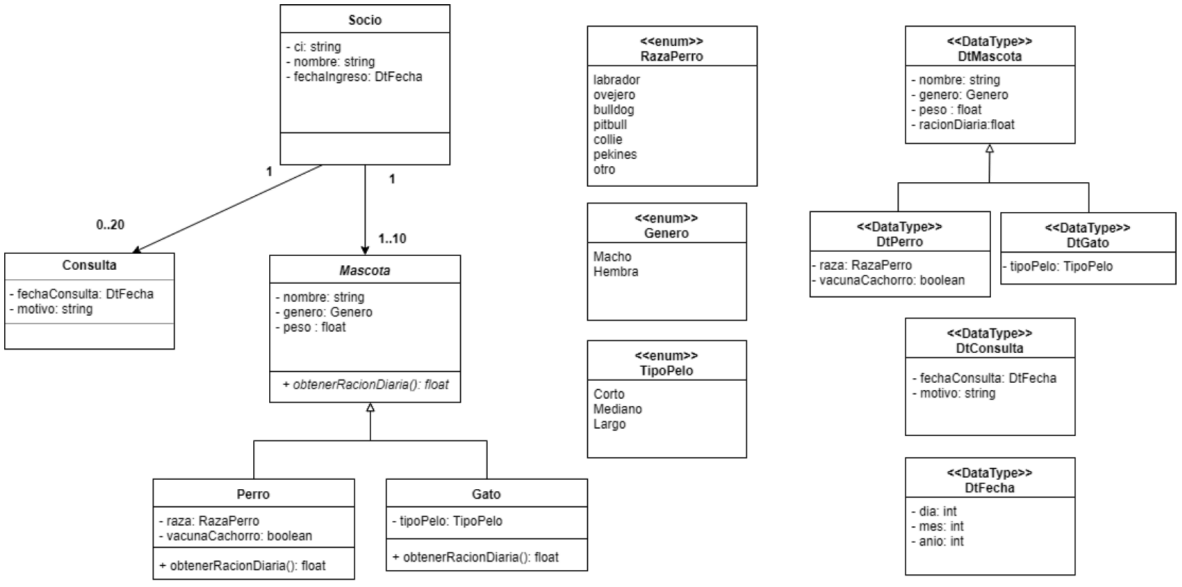
Consideraciones generales:

- La entrega podrá realizarse **hasta el sábado 5/04/25 a las 23:59 hs**.
- Se deberá entregar en el Campus Virtual del curso un archivo con nombre **<número de grupo>_lab1.zip** (o tar.gz) que contenga:
    - El código fuente.
    - Un archivo makefile, que permita compilar y ejecutar el código, independiente de cualquier entorno de desarrollo integrado (IDE).
- El código junto con el makefile debe funcionar en los salones donde se dicta el curso, por lo tanto se recomienda que antes de entregar verifiquen que lo entregado corra en este ambiente.
- Las entregas que no cumplan estos requerimientos no serán consideradas. El hecho de no realizar una entrega implica la insuficiencia del laboratorio completo.

Con este laboratorio se espera que el estudiante adquiera competencias en la implementación de operaciones básicas, el uso básico del lenguaje C++ (que se usará en el laboratorio) y el entorno de programación en linux, así como reafirmar conceptos presentados en el curso. También se espera que el estudiante consulte el material disponible en el Campus Virtual del curso y que recurra a Internet con espíritu crítico, identificando y corroborando fuentes confiables de información.

Ejercicio 1

Se desea implementar un pequeño sistema que permita registrar las consultas que se realizan en una veterinaria. Para ello, interesa contar con información sobre los socios de la veterinaria, las mascotas que estos poseen y las consultas que realizan. A continuación se muestra el modelo de esta realidad.



De los socios se conoce la cédula de identidad (que lo identifica), el nombre y la fecha en que se hizo socio de la veterinaria. Cada socio puede realizar varias consultas, de las cuales se conoce la fecha en que se realizó y el motivo.

Además interesa saber el nombre, el género y el tamaño de las mascotas pertenecientes a cada socio. Estas pueden ser de dos tipos, perro o gato. De los perros se conoce su raza y si se dieron la vacuna que les corresponde de cachorros. De los gatos se sabe su tipo de pelo.

Los socios deben poder consultar la ración diaria que corresponde darle a cada una de sus mascotas. Para ello el sistema cuenta con la operación *`obtenerRacionDiaria()`* que retorna distintos valores según el peso y el tipo de mascota. Para los perros, la ración diaria se calcula como su peso por 0,025 y para los gatos su peso por 0,015.

Se pide:

Implementar en C++,

1. Todas las clases (incluyendo sus atributos, pseudoatributos, getters, setters, constructores y destructores), enumerados y datatypes que aparecen en el diagrama.
2. Una función main que implemente las siguientes operaciones:
    a. `void registrarSocio(string ci, string nombre, DtMascota& dtMascota)`
    Registra un socio con su mascota. El valor el atributo racionDiaria se debe setear en 0.
    b. `void agregarMascota(string ci, DtMascota& dtMascota)`
    Agrega una nueva mascota a un socio ya registrado. Si no existe un socio registrado con esa cédula, se levanta una excepción std::invalid_argument.
    c. `void ingresarConsulta(string motivo, string ci)`
    Crea una consulta con un motivo para un socio. Si no existe un socio registrado con esa cédula, se levanta una excepción std::invalid_argument.
    d. `DtConsulta** verConsultasAntesDeFecha(DtFecha& fecha, string ciSocio, int& cantConsultas)`
    Devuelve las consultas antes de cierta fecha. Para poder implementar esta operación se deberá sobrecargar el
        i. operador < (menor que) para el DataType Fecha. El largo del arreglo está dado por el parámetro cantConsultas.
    e. `void eliminarSocio(string ci)`
    Elimina al socio, sus consultas y sus mascotas. Si no existe un socio registrado con esa cédula, se levanta una excepción std::invalid_argument.
    f. `DtMascota** obtenerMascotas(string ci, int& cantMascotas)`
    Devuelve un arreglo con las mascotas del socio. El largo del arreglo está dado por el parámetro cantMascotas. Si no existe un socio registrado con esa cédula, se levanta una excepción std::invalid_argument.
3. Implementar en el main un menú sencillo que sea interactivo con el usuario para poder probar las funcionalidades requeridas en el punto 2. Al ejecutar el programa debe pedir el ingreso de un número especificando la acción a realizar, y pedir luego los datos necesarios para cada operación. Ejemplo:

```
Bienvenido!
Elija la opción:
    1) Registrar socio
    2) Agregar mascota
    3) Ingresar consulta
    ….
    0) Salir
Opción:
```

4. Sobrecargar el operador de inserción de flujo (ej. <<) en un objeto de tipo std::ostream. Este operador debe “imprimir” las distintas clases de DtMascota (DtPerro, DtGato) con el siguiente formato:

```
- Nombre:
- Género:
- Peso: XXX kg
- Ración Diaria: XXX gramos.
- (En caso de ser Perro):
    - Tiene vacuna del Cachorro: Si / No
- (En caso de ser Gato):
    - Tipo de pelo
```

Notas:

- A los efectos de este laboratorio, la función main mantendrá una colección de socios, implementada como un arreglo de tamaño MAX_SOCIOS. Puede implementar operaciones en las clases dadas en el modelo si considera que le facilitan para la resolución de las operaciones pedidas en el main.
- Se puede utilizar el tipo std::string para implementar los atributos de tipo string.
- El main debe manejar las excepciones lanzadas por las operaciones de los objetos. Por más información consulte las referencias.