UNIVERSIDAD NEBRIJA

PROGRAMACIÓN II - GRUPO C

PRÁCTICA Nº5

Miércoles, 12 de mayo 2023

INSTRUCCIONES

1. Sólo se podrá utilizar una hoja con el "chuletario" elaborado por cada alumno con sintaxis del lenguaje. No se permitirá la consulta de ningún otro tipo de material durante la realización de la práctica.

2. La entrega de la práctica sólo se admitirá a través de la **actividad** disponible en el campus virtual de la asignatura de prácticas de PII del grupo C **antes de la hora de finalización de la sesión de prácticas**.

3. Aunque las prácticas se tengan que realizar en grupos de dos integrantes, para su evaluación, **ambos deberán hacer la entrega a través de su campus virtual, la misma práctica**. En otro caso, la práctica quedará **sin evaluar** y supondrá un **0 en la calificación del estudiante que no la haya entregado**.

4. Se debe entregar **los ficheros** correspondientes al ejercicio en el formato más adecuado (**.cpp** o **.h), sin comprimir**.

5. Los ficheros entregados deben **incluir, al principio del cada fichero entregado, el nombre de los integrantes del equipo**.

6. **El incumplimiento de alguna de las instrucciones sobre la realización/entrega de la Práctica supondrá su descalificación.**

Criterios generales de evaluación

Funciones/Métodos: Si no se usa el paso por referencia constante cuando las variables de los parámetros de entrada no son de tipo simple.	40%
Tipos de datos y variables: • Uso de variables globales (fuera del ámbito de una función). • Si no se usan los tipos contenedor vistos en clase (std::array; std::vector; std::set; std::string, etc.) para las variables que lo necesiten.	 0% 0%
El **programa no compila** o **no se asemeja** a lo pedido**.**	0%
Si no se cumplen los **criterios de entrega** indicados en la **actividad/examen.**	0%

Criterios particulares de evaluación

El elemento evaluable no compila o no se asemeja a lo que se pide	0%
El elemento evaluable no se aproxima suficientemente a lo pedido	40%
El elemento evaluable se aproxima suficientemente a lo pedido	60%
El elemento evaluable funciona correctamente y las estrategias y elementos de código elegidos son adecuados.	100%

IMPORTANTE:

- Todos los ejercicios del examen deberán ser resueltos de forma **algorítmica**, es decir, la **solución** propuesta tendrá que ser **general** y **no particular** para unos determinados datos/valores.
- Todos los **ejercicios resueltos sin utilizar funciones** cuando sea apropiado se valorarán con una **nota máxima del 60% sobre la calificación prevista**.
- Se recomienda una **primera lectura del enunciado de la práctica** para planificar su realización.
- Conviene ir **probando los diferentes recursos programados** de forma gradual. Programar todo seguido y realizar las pruebas al final cuando quedan 10 minutos para la entrega, suele acabar con **errores de compilación sin resolver en la entrega** y por lo tanto con una calificación de 0 puntos.

Esta práctica consiste en desarrollar en C++11 los elementos adecuados para que el código correspondiente al **programa** y **ficheros cabecera proporcionados** en la actividad funcionen correctamente y cumplan con las especificaciones que se indican.

IMPORTANTE: Se pueden comentar las partes del programa que no se consigan implementar para **evitar errores de compilación**, en cuyo caso, la evaluación de la práctica tendrá su correspondiente **penalización proporcional a la parte que no se haya desarrollado**.

Completar de forma adecuada los ficheros proporcionados para crear la jerarquía de clases que aparecen en la **Figura 1**:

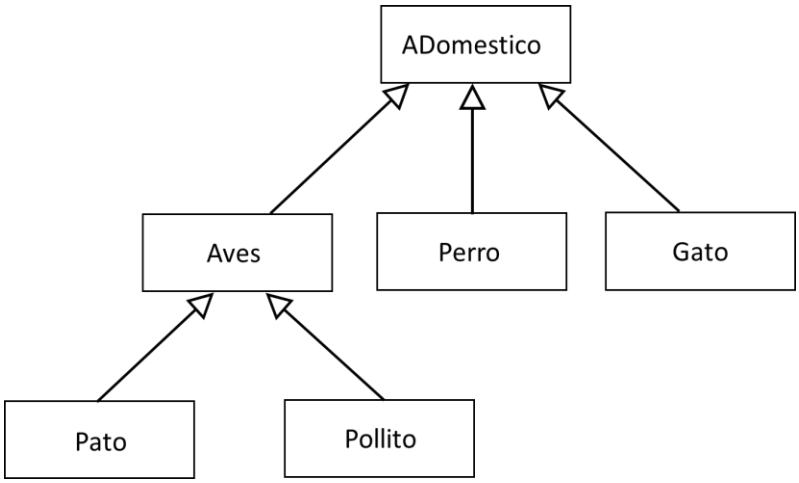

Figura 1. Jerarquía de clases para tipos de animaels domésticos

(4 puntos) El programa que se proporciona debe permitir el manejo de un vector para acceder a los objetos correspondientes a las clases de la jerarquía de tipos de animales domésticos (**Figura 1**). Se presenta un menú que permite al usuario seleccionar la operación que desea realizar teniendo en cuenta lo siguiente:

- Las cuatro primeras opciones permitirán **mostrar la onomatopeya**, siendo además necesario solicitar al usuario determinados datos de cada animal doméstico y almacenar los datos creando un objeto que debe ser accesible desde un elemento del vector de punteros de la clase **ADomestico**.
- La opción 5, ***mostrarADomesticos***, deberá recorrer el vector de punteros a los objetos animales domésticos, generado con las opciones anteriores, para mostrar por pantalla el **tipo de animal** (“Ave”, “Perro” o “Gato”) que corresponda, además del **valor de sus atributos**. Nota: utilizar ***downcasting*** *(dynamic_cast).*

- El programa terminará cuando la opción seleccionada del menú sea la correspondiente a **FIN PROGRAMA**.

Cada opción del menú, siempre que sea adecuado, deberá implementarse a través de una/s función/es, evitando que dicho código quede en el ***main***.

1. Datos del Pato.
2. Datos del Pollito.
3. Datos del Perro.
4. Datos del Gato.
5. Mostrar los Animales Domesticos.
6. FIN PROGRAMA

(6 puntos) Para organizar la información se han definido las clases que aparecen en la **Figura 1** y, a continuación se indican algunas de las características que deben de cumplir:

- Métodos ***pedirIdentificacion, conocerOnomatopeya*** y ***verTipo*** que serán diferentes según el objeto que los invoque:
  - Método ***pedirIdentificacion***: se solicitará la **fecha de nacimiento**, mes y año, que se guardará en un atributo de tipo **string** con el formato "MM/AAAA". Se lanzará una excepción si se comprueba que el ***mes*** no es un valor válido. Además, cuando se trate de un perro o un gato, se solicitará el **nombre** que se guardará en otro atributo.

    NOTA: ***to_string*** permite convertir un **entero a string**.
  - Método ***conocerOnomatopeya***: permitirá mostrar por pantalla la onomatopeya del animal doméstico (el pato "**CUA CUA**", el pollito "**PIO PIO**", el perro "**GUAU GUAU**" y el gato "**MIAU MIAU**"), según la opción del menú seleccionada. Además, cuando se trata del perro o el gato, se escribirá el nombre y luego la onomatopeya, mientras que si se trata del pato o pollito, se pondrá sólo el sonido que hacen.

```
El perro Tambor dice: GUAU GUAU...

Presione una tecla para continuar . . .
```

```
Mi pollito hace: PIO PIO...

Presione una tecla para continuar . . .
```

- Método ***verTipo***: devolverá un valor de tipo ***string*** con la descripción del animal doméstico seleccionado en el menú. Sus posibles valores serán: "**Ave**", "**Perro**" o "**Gato**".

Clase Ave

- Atributos: ***string*** fecha.
- Métodos:
  - Constructor por defecto con inicializadores.
  - Constructor con parámetros.
  - Métodos **pedirIdentificacion**.
  - Método **get** para su atributo.

Clase Pato

- Métodos:
  - Constructor por defecto con inicializadores.
  - Constructor con parámetros.
  - Métodos **conocerOnomatopeya** y **verTipo**.

Clase Pollito

- Métodos:
  - Constructor por defecto con inicializadores.
  - Constructor con parámetros.
  - Métodos **conocerOnomatopeya** y **verTipo**.

Clase Perro

- Atributo*:* ***string*** fecha.y ***string*** *nombre*.
- Métodos:
  - Constructor por defecto con inicializadores.
  - Constructor con parámetros.
  - Métodos **pedirIdentificacion**, **conocerOnomatopeya** y **verTipo.**
  - Métodos **gets** para sus atributos.

Clase Gato

- Atributo*:* ***string*** fecha.y ***string*** *nombre*.
- Métodos:
  - Constructor por defecto con inicializadores.
  - Constructor con parámetros.
  - Métodos **pedirIdentificacion**, **conocerOnomatopeya** y **verTipo.**
  - Métodos **gets** para sus atributos.

Si se considera necesario, se pueden definir nuevos métodos en cada una de las clases anteriores.