



Ejercicio 27

Docente: Jimmy Nataniel Requena Llorentty

Materia: Programación III

Carrera: Ingeniería En Sistemas

Estudiantes: Joaquin Marcos Maita Flores

Santa Cruz – Bolivia

2025

DE ANIMAL A PERRO: HERENCIA EN CÓDIGO

```
#include <iostream>
#include <string>

// Clase Base
class Animal {
protected: // Hacemos nombre protected para que Perro pueda accederlo si quisiera
    std::string nombre;
    int edad; // Podría ser private si solo se accede vía getters/setters
públicos de Animal

public:
    Animal(const std::string& n, int e) : nombre(n), edad(e) {
        std::cout << " CONSTRUCTOR Animal: Nace un animal llamado '" << nombre <<
        "' de " << edad << " año(s)." << std::endl;
    }

    ~Animal() {
        std::cout << " DESTRUCTOR Animal: Muere el animal '" << nombre << "'." <<
        std::endl;
    }

    void comer() const { // const porque no modifica el estado del Animal
        std::cout << nombre << " esta comiendo." << std::endl;
    }

    void dormir() const {
        std::cout << nombre << " esta durmiendo." << std::endl;
    }

    std::string getNombre() const { return nombre; }
};

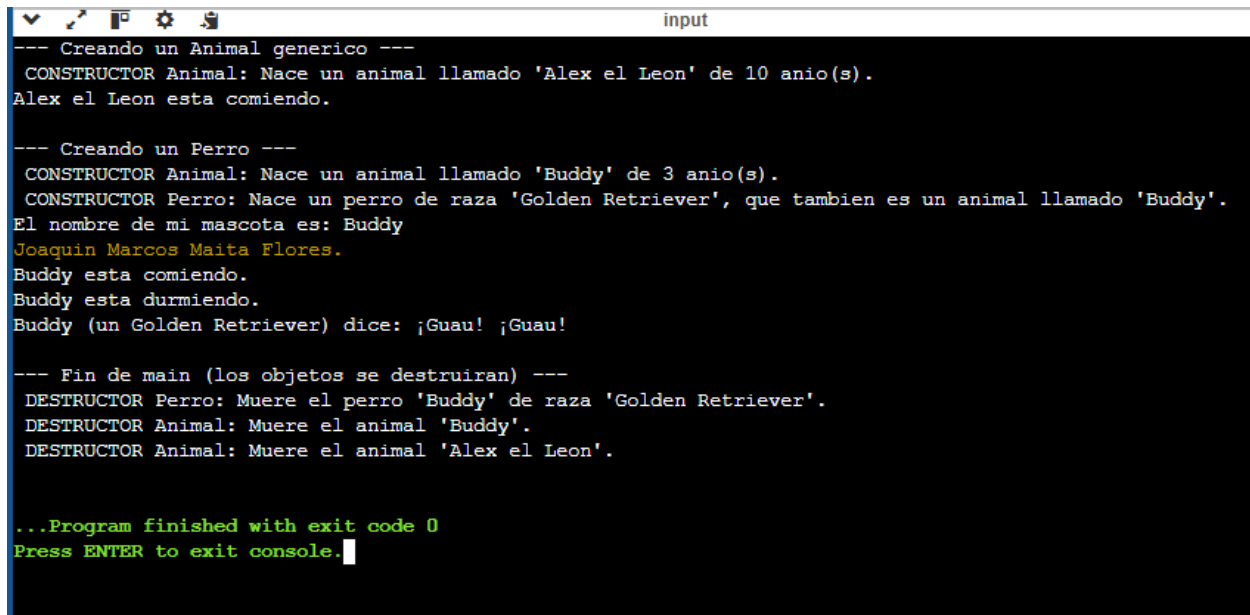
// Clase Derivada
class Perro : public Animal { // Perro "es un tipo de" Animal
private:
    std::string raza; // Atributo específico de Perro

public:
    // Constructor de Perro llama explícitamente al constructor de Animal
    Perro(const std::string& n, int e, const std::string& r)
        : Animal(n, e), raza(r) {
```

```
        std::cout << " CONSTRUCTOR Perro: Nace un perro de raza '" << raza << "',  
que tambien es un animal llamado '" << nombre << "'.'" << std::endl;  
        // 'nombre' es accesible aquí porque es 'protected' en Animal  
    }  
  
    ~Perro() {  
        // 'nombre' es accesible aquí  
        std::cout << " DESTRUCTOR Perro: Muere el perro '" << nombre << "' de  
raza '" << raza << "'.'" << std::endl;  
    }  
  
    // Método específico de Perro  
    void ladrar() const {  
        std::cout << getNombre() << " (un '" << raza << ") dice: ¡Guau! ¡Guau!" <<  
std::endl;  
        // Usamos getNombre() para acceder al nombre, buena práctica aunque sea  
protected  
    }  
};  
  
int main() {  
    std::cout << "--- Creando un Animal generico ---" << std::endl;  
    Animal animalGenerico("Alex el Leon", 10);  
    animalGenerico.comer();  
  
    std::cout << "\n--- Creando un Perro ---" << std::endl;  
    Perro miMascota("Buddy", 3, "Golden Retriever");  
    std::cout << "El nombre de mi mascota es: " << miMascota.getNombre() <<  
std::endl;  
  
    std::cout << "\033[33mJoaquin Marcos Maita Flores.\033[0m" << std::endl;  
    // Métodos heredados de Animal  
    miMascota.comer();  
    miMascota.dormir();  
  
    // Método propio de Perro  
    miMascota.ladrar();  
  
    std::cout << "\n--- Fin de main (los objetos se destruirán) ---" <<  
std::endl;
```

```
    return 0;  
}
```

CÓDIGO EJECUTADO



```
input  
--- Creando un Animal generico ---  
CONSTRUCTOR Animal: Nace un animal llamado 'Alex el Leon' de 10 anio(s).  
Alex el Leon esta comiendo.  
  
--- Creando un Perro ---  
CONSTRUCTOR Animal: Nace un animal llamado 'Buddy' de 3 anio(s).  
CONSTRUCTOR Perro: Nace un perro de raza 'Golden Retriever', que tambien es un animal llamado 'Buddy'.  
El nombre de mi mascota es: Buddy  
Joaquin Marcos Maita Flores.  
Buddy esta comiendo.  
Buddy esta durmiendo.  
Buddy (un Golden Retriever) dice: ¡Guau! ¡Guau!  
  
--- Fin de main (los objetos se destruirán) ---  
DESTRUCTOR Perro: Muere el perro 'Buddy' de raza 'Golden Retriever'.  
DESTRUCTOR Animal: Muere el animal 'Buddy'.  
DESTRUCTOR Animal: Muere el animal 'Alex el Leon'.  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Este ejemplo enseña:

1. La relación "es-un" (un perro *es un* animal).
2. Cómo compartir atributos comunes (nombre, edad) entre clases relacionadas.
3. El orden de construcción/destrucción (primero la base, luego la derivada).
4. La extensión de funcionalidad (el perro añade su método ladrar()).

Anexo

<https://onlinegdb.com/J4ejuQbPQJ>