1. Cree una jerarquía de "figuras": una clase base llamada **`Figura`** y clases derivadas, **`Cuadrado`** y **`Triangulo`**. En la clase base haga una función virtual **`dibujar()`**, y sobrescríbala en las clases derivadas. Haga un arreglo de punteros a **`Figura`** e inicialícelo con objetos creados sobre el heap, llame a **`dibujar()`** a través de punteros a la clase base para verificar el comportamiento de las funciones virtuales.
2. Modifique el ejercicio anterior haciendo **`dibujar()`** una función virtual pura. Intente crear un objeto de la clase **`Figura`**. Sale error por ser abstracta
3. Cree una jerarquía de roedores: Raton, Gerbo, Hamster, etc. En la clase base, provea métodos que son comunes a todos los roedores y redefina éstos en las clases derivadas con comportamientos propios del tipo específico de roedor. Cree un arreglo de punteros a roedores, inicialícelo con diferentes tipos de roedores y llame a métodos de la clase base para ver qué ocurre. ver
4. Modifique el ejercicio anterior parta utilizar un **`vector<Roedor *>`** en lugar de un arreglo. Asegúrese que la memoria es liberada correctamente. ver
5. Modifique el ejercicio anterior agregando un destructor no virtual y verifique que ocurre.
6. Cree una clase con datos miembros y funciones virtuales. Escriba una función que "mire" la memoria de un objeto de esa clase e imprima las distintas piezas del mismo. Tendrá que experimentar e iterativamente descubrir dónde está la **`VPTR`** en el objeto.
7. Cree una clase que tenga un dato miembro y una clase derivada que agrega otro dato miembro. Escriba una función no miembro que tome un objeto de la clase base por valor e imprima el tamaño del objeto utilizando **`sizeof()`**. En **`main( )`** cree un objeto de la clase derivada, imprima su tamaño y llame a su función. Explique que ocurre.
8. Cree una clase base conteniendo una función **`clone( )`** que retorna un puntero a una copia de objeto. Derive dos subclases que sobrescriban **`clone()`** para retornar copias de sus tipos específicos. En **`main`**, cree objetos y realice **`upcast`** de sus tipos derivados, luego llame a **`clone()`** para cada uno para verificar que son clonados los subtipos correctos. Experimente con **`clone()`** para que retornen un puntero al tipo base y trate de retornar el mismo tipo derivado. ver

4) Para borrar tengo que crear un * temporal que apunte al .back(), hacer el .pop() y despues delete temporal.

5)me sale que: deleting object of polymorfic class type roedores wich has non-virtual destructor can cause undefined behavior.

7) hace un upcast a clase base ya que la funcion espera un objeto de la clase base y por eso me sale el tamaño de la base?