Programación 2009

Tarea 15

Ejercicio 1 *Verificar que el código siguiente no compila y explicar por qué. Hacerle compilar añadiendole* ***un número mínimo de caracteres****.*

```
#include <iostream>
class Base {
public:
   virtual void funcion();
};
class Derivada : public Base{
public:
   void funcion();
};
int main() {
   Base *bp = new Derivada();
   bp->funcion();
   delete bp;
   return 0;
}
void Derivada::funcion(){
   std::cout << "Cogito ergo sum" << std::endl;
}
```

Ejercicio 2** *Explicar por qué, dado el mecanismo de polimorfismo, un método declarado a la vez* ***inline y virtual*,*

- *no estará implementado como inline por el compilador en el caso que manipulas apuntadores o referencias a objetos,*
- *puede estar implementado como inline si lo manipulas como objeto, directamente.*

Ejercicio 3 *En el ejemplo siguiente, los métodos create y clone suelen ser denominadas como "constructores virtuales". Explicar por qué. En un segundo momento, completar la clase Circle para que se pueda compilar las dos clases, dar un ejemplo de uso de los contructores virtuales. En un tercer momento, crear e imlementar una clase Square heredando de Shape, de la misma manera que Circle.*

```
class Shape {
public:
  virtual ~Shape() { };
  virtual Shape* clone()  const = 0;
  virtual Shape* create() const = 0;
  virtual double getArea() const = 0;
  virtual double getPerimeter() const = 0;
  virtual void print() const = 0;
};
class Circle : public Shape {
  double x;
  double y;
  double r;
public:
  Circle();
  Circle(const double &x,const double &y,const double &r);
  Circle* clone()  const;
  Circle* create() const;
  ...
};
```

Ejercicio 4 *Inspirarse del ejemplo de operador binario virtual de la clase para resolver el problema siguiente. Se implementará una clase abstracta Animal, que contenga una string con el nombre del animal. El método virtual puro será un operador *, que corresponderá al hecho de que dos animales* ***pueden co-existir en una casa****. Regresará un booleano (false si la cohabitación es imposible entre los dos animales, true sino). En caso que la coexistencia sea posible, imprimirá un mensaje que el animal X y el animal Y pueden vivir juntos. Implementar clases Gato, Perro, Caballo, Pez, heredando de Animal, y el operador virtual *, para poder manejar todos los casos:*

- *un Caballo puede coexisitir con cualquier otro animal,*
- *un Gato no puede coexistir con un Pez o un Perro,*
- *un Perro o un Pez pueden coexistir con todos excepto Gato.*

Ejercicio 5 *Escribir una clase virtual* ***Base*** *con un elemento entero x y un método (virtual, redefinido en Derivada) setX para modificar el valor de este elemento. Ahora, poner valores* ***por default*** *(diferentes!) en este método, en la Base Y en la Derivada. Llamarle desde un objeto Derivada, (1) a través de un apuntador a Derivada y (2) a través de un apuntador a Base. Comparar los resultados y intentar dar una explicación.*

Ejercicio 6 *Explicar por qué, si una clase hereda de dos clases* ***sin herencia común****, pero con tablas virtuales, se necesita tener las* ***dos*** *tablas virtuales presentes en el objeto heredando. En particular, explicar porque no se podría mezclar las dos tablas virtuales en una sola.*