Septiembre 2016

*Ejercicio 1:

A) No compila en el main declaramos una clase Foo<const char*>, de tipo char puntero por lo que no podemos utilizar la función mar(), que suma el factor pasado al comando, en este caso no se puede debido a que a un puntero no se le pueden sumar regiones de memoria, es decir cadenas como si fuera un string.

B) No compila pero no se el porque

*Ejercicio 2:

A) Define con tus propias palabras el concepto de polimorfismo (dí qué es, no cuándo se utiliza ni para qué sirve).

Se denomina polimorfismo a la característica de un lenguaje de programación que permite escribir código que utilice el mismo identificador para denominar diferentes elementos del lenguaje. El lenguaje debe ser capaz de diferenciar entre cada uno de dichos elementos en función del contexto.

B) Los lenguajes orientados a objetos vistos en la asignatura (C++ y Java) permiten polimorfismo. **Explica** brevemente qué mecanismo de dichos lenguajes permite **polimorfismo por inclusión**, y qué otro mecanismo permite **polimorfismo paramétrico**, y cómo lo permiten.

Polimorfismo por inclusión

Mediante la definición de una relación de subtipado entre diferentes tipos de datos, se puede definir código que trabaje con el tipo base y sus subtipos relacionados. En POO el mecanismo de subtipado se denomina herencia.

Polimorfismo paramétrico

El mismo código trabaja con cualquier tipo de datos. También denominado programación genérica.

**NOTA:

AL referersi que expliques brevemente los mecanismo se refiera a poner un ejemplo en lenguaje por ejemplo C++ de un polimorfismo de inclusión que seria una clase herencia, y en polimorfismo parametrico una clase de tipo genérico ¿???

C) Ilustra mediante un ejemplo en un lenguaje orientado a objetos a tu elección (C++ o Java) simultáneamente los dos tipos de polimorfismo identificados en (b). Dicho ejemplo deberá ser lo más pequeño posible (preferiblemente con solo dos o tres clases involucradas) y deberá incluir un pequeño programa principal.

Polimorfismo por inclusión

C++

```
Class Identificador{
        int Id;
        Identificador (int _id) : id(_id) { }
        Virtual int consultar_id(){
                Return this->Id;
        }
};
Class matricula: public Identificador{
        Matricula(int _idmatricula) :
                Identificador(_idmatricula){}
        Int consultar _id(){
                Return this->Id;
        }
};
```

+ Programa principal

Polimorfismo parametrico

C++

```
Template<typename T>
Class Identificador{
        T Id;
        Identificador (T _id) id(_id) { }
        T consultar_id(){
                Return this->Id;
        }
};
```

D) **Justifica** dónde aparecen las modalidades de polimorfismo descritas de (b) en tu ejemplo de (c).

E) De los dos tipos de polimorfismo de (b), sólo uno de ellos está representado mediante un mecanismo de **Haskell** (programación funcional). **¿Cuál?**

El polimorfismo paramétrico que permite poner métodos genéricos en haskell.