



EJERCICIOS 1

- 1) Dada la siguiente definición de clase, escriba una definición apropiada para la función miembro *fijar*.

```
class Temperatura
{
public:
    void fijar (double grados, char escala);
private:
    double grados;
    char escala;    // 'F' para Fahrenheit y 'C' para Celsius
};
```

El miembro *Fijar*, establece los atributos a los valores que se dan como argumentos.

- 2) Supongamos que un programa tiene la siguiente definición de clase:

```
class Automovil
{
public
    void fijar_precio(double nuevo_precio);
    void fijar_margen(double nuevo_margen);
    double obtener_precio();
private:
    double precio;
    double margen;
    double obtener_margen();
};
```

Y suponga que la parte `main()` del programa contiene la siguiente declaración

```
Automovil hyundai, jaguar;
```

Y que de alguna manera el programa asigna ciertos valores a todos los atributos.

¿Cuáles de las siguientes instrucciones son válidas en la parte main() del programa?

```
...
hyundai.precio = 4999;
jaguar.fijar_precio(30000);
double aux_precio, aux_margen;
aux_precio = jaguar.obtener_precio();
aux_margen = hyundai.obtener_margen();
if ( hyundai == jaguar )
    cout << "¿Cambiamos de coche?";
hyundai = jaguar;
```

3) Dé una definición de la función que tiene el siguiente prototipo:

```
float diferencia (Cuenta c1, Cuenta c2);
// devuelve el saldo de la cuenta c1 menos el de la cuenta c2.
```

La definición de la clase Cuenta es la siguiente:

```
class Cuenta
{
    private:
        long int numero_cuenta;
        float saldo;
        float interes_anual;
    public:
        void inicializar(long int num);
        float dar_saldo();
        float dar_interes();
        void mod_saldo(float s);
        void mod_interes(float i);
};
```

- 4) Crear una clase llamada **Card** que guarde la información de cada uno de los libros de una biblioteca. La clase debe guardar el título del libro, autor y número de ejemplares disponibles. Usar una función miembro pública llamada **store** para almacenar la información de un libro y una función miembro pública llamada **show** para mostrar la información. Incluya un **main** breve para probar la clase.
- 5) Crear una clase llamada **Box** con tres atributos de tipo *double* que representen la longitud de los lados del cuadrado. Incluir una función miembro que inicialice el valor

de los atributos. Haga que la clase **Box** calcule el volumen del cubo y guarde el resultado en una variable `double`. Incluir una función miembro llamada `vol` que muestre el volumen del objeto de la clase **Box**.

- 6) Crear una clase que se denomine **Contador** que defina un único atributo `cont` (un entero). El contador se podrá inicializar a cualquier valor no negativo. Otros métodos que han de proporcionar los objetos de ésta clase son: incrementar el contador (en una unidad), decrementar el contador (en una unidad) y mostrar el valor del contador. El contador nunca podrá tener un valor negativo. Prueba la clase en una función `main()`.
- 7) Crea una clase `Nif` que se usará para mantener DNIs con su correspondiente letra. Los atributos de la clase serán el número de DNI (entero largo) y la letra que le corresponde. La clase dispondrá de los siguientes métodos:
 - Inicializador que reciba el número de DNI y establezca automáticamente la letra que le corresponde.
 - Accedentes y mutador para el número de DNI (ajustando automáticamente la letra).
 - `Leer()` : que pida el número de DNI (ajustando automáticamente la letra).
 - Método que nos permita mostrar el NIF (DNI, un guión y la letra en mayúsculas; por ejemplo: 395469-F).

La letra se calculará con un método auxiliar (privado) de la siguiente forma: se obtiene el resto de la división entera del número de DNI entre 23 y se usa la siguiente tabla para obtener la letra correspondiente:

0 - T	12 - N
1 - R	13 - J
2 - W	14 - Z
3 - A	15 - S
4 - G	16 - Q
5 - M	17 - V
6 - Y	18 - H
7 - F	19 - L
8 - P	20 - C
9 - D	21 - K
10 - X	22 - E
11 - B	

- 8) Dadas las clases que se muestran a continuación, averigua qué se mostrará en la pantalla con el programa principal que se encuentra al final.

<pre>#include <iostream.h> // INTERFAZ DE LA CLASE ClaseA // class ClaseA { public: void init(int = 0); int mas(int = 0); int por(int = 1); private: int a; }; // Implementación métodos de la ClaseA // void ClaseA::init(int val) { a = val; } int ClaseA::mas(int val) { return a + val; } int ClaseA::por(int val) { return a * val; }</pre>	<pre>// INTERFAZ DE LA CLASE ClaseB// class ClaseB { public: void init(int = 0); int mas(int = 0); int por(int = 1); void igual(ClaseA); private: ClaseA obj; }; // Implementación métodos de la ClaseB // void ClaseB::init(int val) { obj.init(val); } int ClaseB::mas(int val) { return obj.mas(val); } int ClaseB::por(int val) { return obj.por(val); } void ClaseB::igual(ClaseA otro) { obj = otro; }</pre>
--	--

<pre> // INTERFAZ DE LA CLASE ClaseC// class ClaseC { public: void init(int = 0, int = 0); int mas(); int por(); void obj1(ClaseA); void obj2(ClaseB); private: ClaseA obj1; ClaseB obj2; }; // Implementación de los métodos de la ClaseC // void ClaseC::init(int val1, int val2) { obj1.init(val1); obj2.init(val2); } int ClaseC::mas() { return obj1.mas() + obj2.mas(); } int ClaseC::por() { return obj1.por() * obj2.por(); } void ClaseC::obj1(ClaseA m) { obj1 = m; } void ClaseC::obj2(ClaseB n) { obj2 = n; } </pre>	<pre> // Comienzo del main -----// int main() { ClaseA objA; objA.init(5); cout << objA.por(5) << endl; ClaseB objB; objB.init(10); cout << objB.por(5) + objA.mas(3) << endl; ClaseC objC; objC.init(12, 7); cout << objC.mas() << endl; cout << objC.por() << endl; objB.igual(objA); objC.obj1(objA); objC.obj2(objB); cout << objC.mas() << endl; cout << objC.por() << endl; return 0; } </pre>
--	--