FACULTAD DE INFORMÁTICA

# Ejercicios de repaso

## SOLUCIONES DEL TALLER

Programación orientada a objetos — Unidad 5

Autor: Luis Hernández Yáñez

FdI UCM

## Ejercicio nº 1

Crea una clase **Vehiculo** que modele la información que se mantiene en un taller sobre cada vehículo que reparan: modelo, propietario (una persona), importe de las piezas usadas en la reparación (sin IVA), fecha de la reparación, hora de inicio de la reparación y hora de finalización de la reparación. Ninguna reparación lleva más de un día. Además de otros métodos que consideres adecuados, la clase debe proporcionar un servicio **total()** que devuelva el total de la reparación (piezas más mano de obra), teniendo en cuenta que se cobran 0,375 € por minuto de mano de obra y que el IVA es del 16%. También habrá un método **mostrar()** que muestre la información de la forma siguiente:

```
jueves 14 de noviembre de 2002
Vehículo: SEAT 600
Propietario: Luis Sanz Sanz
Hora de inicio: 17:15:25
Hora de final: 19:07:12
Importe de las piezas: 213.35 eur
Total de la reparación: 295.771 eur (IVA inc.)
```

Prueba la clase en una función **main()**.



FdI UCM

## Ejercicio nº 1 — vehiculo.h

```
#ifndef vehiculo_h // Evitar inclusiones múltiples
#define vehiculo_h

#include <iostream>
#include <string>
using namespace std;
#include "persona.h"
#include "fecha.h"
#include "hora.h"

class Vehiculo {
public:
  Vehiculo(string = "", Persona = Persona(), double = 0,
           Fecha = Fecha(), Hora = Hora(), Hora = Hora());
    // Datos: modelo, propietario, coste de las piezas, fecha,
    //        hora de inicio y hora de fin
  Vehiculo(const Vehiculo&);
  Vehiculo& operator=(const Vehiculo&);
  ~Vehiculo();
  string modelo() const;
  void modelo(string);
```

(*continúa*)



FdI UCM

## Ejercicio nº 1 — vehiculo.h

```
  Persona propietario() const;
  void propietario(Persona);
  double piezas() const;
  void piezas(double);
  Fecha fecha() const;
  void fecha(Fecha);
  Hora inicio() const;
  void inicio(Hora);
  Hora fin() const;
  void fin(Hora);
  double total() const;
  void mostrar() const;
private:
  static float _porMin; // euros por min. - atributo de clase
  string _modelo;
  Persona _propietario;
  double _piezas;
  Fecha _fecha;
  Hora _inicio, _fin;
};
#endif
```

## Ejercicio nº 1

vehiculo.cpp

```
// Clase Vehiculo - Implementación

#include "vehiculo.h"

float Vehiculo::_porMin = 0.375;

Vehiculo::Vehiculo(string modelo, Persona propietario,
   double piezas, Fecha fecha, Hora inicio, Hora fin)
  : _modelo(modelo), _propietario(propietario), _piezas(piezas),
    _fecha(fecha), _inicio(inicio), _fin(fin) { }

Vehiculo::Vehiculo(const Vehiculo& otro) {
  _modelo = otro._modelo;
  _propietario = otro._propietario;
  _piezas = otro._piezas;
  _fecha = otro._fecha;
  _inicio = otro._inicio;
  _fin = otro._fin;
}
```

(*continúa*)





## Ejercicio nº 1

vehiculo.cpp

```
Vehiculo& Vehiculo::operator=(const Vehiculo& otro) {
  _modelo = otro._modelo;
  _propietario = otro._propietario;
  _piezas = otro._piezas;
  _fecha = otro._fecha;
  _inicio = otro._inicio;
  _fin = otro._fin;
  return *this;
}

Vehiculo::~Vehiculo() { }

string Vehiculo::modelo() const { return _modelo; }

void Vehiculo::modelo(string m) { _modelo = m; }

Persona Vehiculo::propietario() const { return _propietario; }

void Vehiculo::propietario(Persona p) { _propietario = p; }
```

(*continúa*)





## Ejercicio nº 1

vehiculo.cpp

```
double Vehiculo::piezas() const { return _piezas; }

void Vehiculo::piezas(double p) { _piezas = p; }

Fecha Vehiculo::fecha() const { return _fecha; }

void Vehiculo::fecha(Fecha f) { _fecha = f; }

Hora Vehiculo::inicio() const { return _inicio; }

void Vehiculo::inicio(Hora h) { _inicio = h; }

Hora Vehiculo::fin() const { return _fin; }

void Vehiculo::fin(Hora h) { _fin = h; }

double Vehiculo::total() const {
  const float IVA = 1.16;
  long int minutos = (_fin - _inicio) / 60;
  return (_piezas + minutos * _porMin) * IVA;
}
```

(*continúa*)





## Ejercicio nº 1

vehiculo.cpp

```
void Vehiculo::mostrar() const {
  cout << endl;
  cout << _fecha.completa() << endl;
  cout << "Vehículo: " << _modelo << endl;
  cout << "Propietario: " << _propietario.nombreCompleto()
       << endl;
  cout << "Hora de inicio: " << _inicio.hora() << endl;
  cout << "Hora de final: " << _fin.hora() << endl;
  cout << "Importe de las piezas: " << _piezas << " eur"
       << endl;
  cout << "Total de la reparación: " << total()
       << " eur (IVA inc.)" << endl;
}
```

## Ejercicio nº 1

prueba1.cpp

```
#include "vehiculo.h"

int main() {
  Persona prop("223344G", 30, "Luis", "Sanz Sanz");
  Vehiculo v1("SEAT 600", prop, 213.35, Fecha(14,11,2002),
              Hora(17, 15, 25), Hora(19, 7, 12));
  v1.mostrar();
  Vehiculo v2(v1);
  v2.mostrar();
  Vehiculo v3;
  v3 = v2;
  v3.mostrar();

  return 0;
}
```

```
jueves 14 de noviembre de 2002
Vehículo: SEAT 600
Propietario: Luis Sanz Sanz
Hora de inicio: 17:15:25
Hora de final: 19:07:12
Importe de las piezas: 213.35 eur
Total de la reparación: 295.771 eur (IVA inc.)
```



## Ejercicio nº 2

Crea una clase **Mueble** que modele la información que se mantiene en una carpintería sobre cada mueble que fabrican: descripción, carpintero (una persona), cantidad de madera utilizada (en $m^3$), día de inicio de la fabricación y día de finalización de la fabricación. Los muebles se tardan en fabricar días completos. Además de otros métodos que consideres adecuados, la clase debe proporcionar un servicio **total()** que devuelva el total de la fabricación (material más mano de obra), teniendo en cuenta que la madera cuesta 1,233 € por $m^3$, que los carpinteros cobran 150 € por día de trabajo y que el IVA es del 16%. También se dispondrá de un método **mostrar()** que muestre la información de la forma siguiente:

```
Mueble: Estantería castellana de tres cuerpos
Carpintero: Luis Sanz Sanz
Día de inicio: 17/11/2002
Día de final: 19/11/2002
Importe de la madera: 1491.93 eur
Importe de la mano de obra: 300 eur
Total de la fabricación: 2078.64 eur (IVA inc.)
```

Prueba la clase en una función **main()**.



## Ejercicio nº 2

mueble.h

```
// Clase Mueble - Archivo de cabecera "mueble.h"

#ifndef mueble_h // Evitar inclusiones múltiples
#define mueble_h

#include <iostream>
#include <string>
using namespace std;
#include "persona.h"
#include "fecha.h"

class Mueble {
public:
  Mueble(string = "", Persona = Persona(), double = 0,
         Fecha = Fecha(), Fecha = Fecha());
  Mueble(const Mueble&);
  Mueble& operator=(const Mueble&);
  ~Mueble();
  string descripcion() const;
  void descripcion(string);
```

(*continúa*)



## Ejercicio nº 2

mueble.h

```
  Persona carpintero() const;
  void carpintero(Persona);
  double madera() const;
  void madera(double);
  Fecha inicio() const;
  void inicio(Fecha);
  Fecha fin() const;
  void fin(Fecha);
  double total() const;
  void mostrar() const;
private:
  static float _porM3, _porDia; // Atributos de clase
  string _descripcion;
  Persona _carpintero;
  double _madera;
  Fecha _inicio, _fin;
};

#endif
```

## Ejercicio nº 2

mueble.cpp

```
// Clase Mueble - Implementación "mueble.h"

#include "mueble.h"

float Mueble::_porM3 = 1233;
float Mueble::_porDia = 150;

Mueble::Mueble(string desc, Persona carpintero, double madera,
               Fecha inicio, Fecha fin)
  : _descripcion(desc), _carpintero(carpintero), _madera(madera),
   _inicio(inicio), _fin(fin) { }

Mueble::Mueble(const Mueble& otro) {
  _descripcion = otro._descripcion;
  _carpintero = otro._carpintero;
  _madera = otro._madera;
  _inicio = otro._inicio;
  _fin = otro._fin;
}
```

(*continúa*)





## Ejercicio nº 2

mueble.cpp

```
Mueble& Mueble::operator=(const Mueble& otro) {
  _descripcion = otro._descripcion;
  _carpintero = otro._carpintero;
  _madera = otro._madera;
  _inicio = otro._inicio;
  _fin = otro._fin;
  return *this;
}

Mueble::~Mueble() { }

string Mueble::descripcion() const { return _descripcion; }

void Mueble::descripcion(string m) { _descripcion = m; }

Persona Mueble::carpintero() const { return _carpintero; }

void Mueble::carpintero(Persona p) { _carpintero = p; }

double Mueble::madera() const { return _madera; }
```

(*continúa*)





## Ejercicio nº 2

mueble.cpp

```
void Mueble::madera(double p) { _madera = p; }

Fecha Mueble::inicio() const { return _inicio; }

void Mueble::inicio(Fecha f) { _inicio = f; }

Fecha Mueble::fin() const { return _fin; }

void Mueble::fin(Fecha f) { _fin = f; }

double Mueble::total() const {
  const float IVA = 1.16;
  long int dias = _fin - _inicio;
  return (_madera * _porM3 + dias * _porDia) * IVA;
}
```

(*continúa*)





## Ejercicio nº 2

mueble.cpp

```
void Mueble::mostrar() const {
  cout << endl;
  cout << "Mueble: " << _descripcion << endl;
  cout << "Carpintero: " << _carpintero.nombreCompleto()
       << endl;
  cout << "Día de inicio: " << _inicio.reducida() << endl;
  cout << "Día de final: " << _fin.reducida() << endl;
  cout << "Importe de la madera: " << _madera * _porM3 << " eur"
       << endl;
  cout << "Importe de la mano de obra: "
       << (_fin - _inicio) * _porDia << " eur" << endl;
  cout << "Total de la fabricación: " << total()
       << " eur (IVA inc.)" << endl;
}
```

## Ejercicio nº 2

prueba2.cpp

```
#include "mueble.h"

int main() {
  Persona carp("223344G", 30, "Luis", "Sanz Sanz");
  Mueble m1("Estantería castellana de tres cuerpos", carp, 1.21,
            Fecha(17,11,2002), Fecha(19,11,2002));
  m1.mostrar();
  Mueble m2(m1);
  m2.mostrar();
  Mueble m3;
  m3 = m2;
  m3.mostrar();

  return 0;
}
```

```
Mueble: Estantería castellana de tres cuerpos
Carpintero: Luis Sanz Sanz
Día de inicio: 17/11/2002
Día de final: 19/11/2002
Importe de la madera: 1491.93 eur
Importe de la mano de obra: 300 eur
Total de la fabricación: 2078.64 eur (IVA inc.)
```



## Ejercicio nº 3

Crea una clase **Pedido** que modele la información que se mantiene en un comercio sobre cada pedido que atienden: código, descripción, cliente (una persona), fecha del pedido, hora del pedido y prioridad (un número). Además de otros métodos que consideres adecuados, la clase debe proporcionar operadores de incremento y decremento que aumenten o decrementen, respectivamente, en una unidad la prioridad del pedido. Y un método **mostrar()** que muestre la información de la siguiente forma:

```
Código: G629X002   Prioridad: 2
Descripción: Fungibles
Cliente: Luis Sanz Sanz (NIF: 223344G)
14/11/2002 17:15:25
```

Prueba la clase en una función **main()**.



## Ejercicio nº 3

pedido.h

```
// Clase Pedido - Archivo de cabecera "pedido.h"

#ifndef pedido_h // Evitar inclusiones múltiples
#define pedido_h

#include <iostream>
#include <string>
using namespace std;
#include "persona.h"
#include "fecha.h"
#include "hora.h"

class Pedido {
public:
  Pedido(string = "", string = "", Persona = Persona(),
         Fecha = Fecha(), Hora = Hora(), int = 0);
  Pedido(const Pedido&);
  Pedido& operator=(const Pedido&);
  ~Pedido();
  string codigo() const;
  void codigo(string);
```

(*continúa*)



## Ejercicio nº 3

pedido.h

```
  string descripcion() const;
  void descripcion(string);
  Persona cliente() const;
  void cliente(Persona);
  Fecha fecha() const;
  void fecha(Fecha);
  Hora hora() const;
  void hora(Hora);
  int prioridad() const;
  void prioridad(int);
  Pedido& operator++();
  Pedido& operator--();
  void mostrar() const;
private:
  string _codigo, _descripcion;
  Persona _cliente;
  Fecha _fecha;
  Hora _hora;
  int _prioridad;
};
#endif
```

## Ejercicio nº 3

pedido.cpp

```
// Clase Pedido - Implementación "pedido.h"
#include "pedido.h"

Pedido::Pedido(string codigo, string desc, Persona cliente,
           Fecha fecha, Hora hora, int prioridad)
  : _codigo(codigo), _descripcion(desc), _cliente(cliente),
    _fecha(fecha), _hora(hora), _prioridad(prioridad) { }

Pedido::Pedido(const Pedido& otro) {
  _codigo = otro._codigo;
  _descripcion = otro._descripcion;
  _cliente = otro._cliente;
  _fecha = otro._fecha;
  _hora = otro._hora;
  _prioridad = otro._prioridad;
}

Pedido& Pedido::operator=(const Pedido& otro) {
  _codigo = otro._codigo;
  _descripcion = otro._descripcion;
  _cliente = otro._cliente;
```

(*continúa*)





## Ejercicio nº 3

pedido.cpp

```
  _fecha = otro._fecha;
  _hora = otro._hora;
  _prioridad = otro._prioridad;
  return *this;
}

Pedido::~Pedido() { }

string Pedido::codigo() const { return _codigo; }

void Pedido::codigo(string s) { _codigo = s; }

string Pedido::descripcion() const { return _descripcion; }

void Pedido::descripcion(string s) { _descripcion = s; }

Persona Pedido::cliente() const { return _cliente; }

void Pedido::cliente(Persona p) { _cliente = p; }
```

(*continúa*)





## Ejercicio nº 3

pedido.cpp

```
Fecha Pedido::fecha() const { return _fecha; }

void Pedido::fecha(Fecha f) { _fecha = f; }

Hora Pedido::hora() const { return _hora; }

void Pedido::hora(Hora h) { _hora = h; }

int Pedido::prioridad() const { return _prioridad; }

void Pedido::prioridad(int i) { _prioridad = i; }

Pedido& Pedido::operator++() {
  _prioridad++;
  return *this;
}

Pedido& Pedido::operator--() {
  _prioridad--;
  return *this;
}
```

(*continúa*)





## Ejercicio nº 3

pedido.cpp

```
void Pedido::mostrar() const {
  cout << endl;
  cout << "Código: " << _codigo << "   Prioridad: "
       << _prioridad << endl;
  cout << "Descripción: " << _descripcion << endl;
  cout << "Cliente: " << _cliente.nombreCompleto() << endl;
  cout << _fecha.reducida() << " " << _hora.hora() << endl;
}
```

## Ejercicio nº 3

prueba3.cpp

```
#include "pedido.h"

int main() {
  Persona cli("223344G", 30, "Luis", "Sanz Sanz");
  Pedido p1("G629X002", "Fungibles", cli, Fecha(14,11,2002),
            Hora(17, 15, 25));
  p1.mostrar();
  Pedido p2(p1);
  p2.mostrar();
  Pedido p3;
  p3 = p2;
  p3.mostrar();
  (p3++)++;
  p3.mostrar();

  return 0;
}
```

```
Código: G629X002   Prioridad: 0
Descripción: Fungibles
Cliente: Luis Sanz Sanz
14/11/2002 17:15:25

Código: G629X002   Prioridad: 2
Descripción: Fungibles
Cliente: Luis Sanz Sanz
14/11/2002 17:15:25
```



## Ejercicio nº 4

Crea una clase **Deuda** que modele la información que mantiene un prestamista sobre cada deuda que contraen con él: deudor (una persona), fecha del último pago, interés diario y deuda (cantidad que se debe). Además de otros métodos que consideres adecuados, la clase debe proporcionar operadores de incremento y decremento que aumenten o decrementen una milésima el interés diario. Además, habrá un método **pago()** que reciba una cantidad a descontar de la deuda y la fecha del pago; antes de descontar la cantidad de lo debido, calculará y sumará los intereses devengados (deuda por interés diario por número de días transcurridos). Por último, habrá un método **mostrar()** que muestre la información de la siguiente forma:

```
Deudor: Luis Sanz Sanz (223344G), 30 años
Deuda pendiente: 3000 eur (Interés: 0.003%)
Desde el lunes 14 de octubre de 2002
```

Prueba la clase en una función **main()**.



## Ejercicio nº 4

deuda.h

```
// Clase Deuda - Archivo de cabecera "deuda.h“

#ifndef deuda_h // Evitar inclusiones múltiples
#define deuda_h
#include <iostream>
#include <string>
using namespace std;
#include "persona.h"
#include "fecha.h"

class Deuda {
public:
  Deuda(Persona = Persona(), Fecha = Fecha(), double = 0,
        double = 0);
    // Datos: deudor, fecha, interés y deuda contraída
  Deuda(const Deuda&);
  Deuda& operator=(const Deuda&);
  ~Deuda();
  Persona deudor() const;
  void deudor(Persona);
```

(*continúa*)



## Ejercicio nº 4

deuda.h

```
  Fecha fecha() const;
  void fecha(Fecha);
  double interes() const;
  void interes(double);
  double deuda() const;
  void deuda(double);
  Deuda& operator++();
  Deuda& operator--();
  void pago(Fecha, double);
  void mostrar() const;
private:
  Persona _deudor;
  Fecha _fecha;
  double _interes, _deuda;
};

#endif
```

## Ejercicio nº 4

deuda.cpp

```
// Clase Deuda - Implementación "deuda.h"
#include "deuda.h"

Deuda::Deuda(Persona deudor, Fecha fecha, double interes,
  double deuda) : _deudor(deudor), _fecha(fecha),
  _interes(interes), _deuda(deuda) { }

Deuda::Deuda(const Deuda& otra) {
  _deudor = otra._deudor;
  _fecha = otra._fecha;
  _interes = otra._interes;
  _deuda = otra._deuda;
}

Deuda& Deuda::operator=(const Deuda& otra) {
  _deudor = otra._deudor;
  _fecha = otra._fecha;
  _interes = otra._interes;
  _deuda = otra._deuda;
  return *this;
}
```

(*continúa*)





## Ejercicio nº 4

deuda.cpp

```
Deuda::~Deuda() { }

Persona Deuda::deudor() const { return _deudor; }

void Deuda::deudor(Persona p) { _deudor = p; }

Fecha Deuda::fecha() const { return _fecha; }

void Deuda::fecha(Fecha f) { _fecha = f; }

double Deuda::interes() const { return _interes; }

void Deuda::interes(double i) { _interes = i; }

double Deuda::deuda() const { return _deuda; }

void Deuda::deuda(double d) { _deuda = d; }

Deuda& Deuda::operator++() {
  _interes += 0.001;
  return *this;
}
```

(*continúa*)





## Ejercicio nº 4

deuda.cpp

```
Deuda& Deuda::operator--() {
  _interes -= 0.001;
  return *this;
}

void Deuda::pago(Fecha fp, double cantidad) {
  long int dias = fp - _fecha;
  double intereses = _deuda * _interes / 100 * dias;
  _deuda += intereses - cantidad;
  _fecha = fp;
}

void Deuda::mostrar() const {
  cout << endl;
  cout << "Deudor: " << _deudor.nombreCompleto() << " ("
       << _deudor.nif() << "), " << _deudor.edad() << " años"
       << endl;
  cout << "Deuda pendiente: " << _deuda
       << " eur (Interés: " << _interes << "%)" << endl;
  cout << "Desde el " << _fecha.completa() << endl;
}
```





## Ejercicio nº 4

prueba4.cpp

```
#include "deuda.h"

int main() {
  Persona p("223344G", 30, "Luis", "Sanz Sanz");
  Deuda d1(p, Fecha(14,10,2002), 0.003, 3000);
  d1.mostrar();
  d1.pago(Fecha(14,11,2002),250);
  d1.mostrar();
  (d1++)++;
  d1.mostrar();
  d1.pago(Fecha(14,12,
          2002),250);
  d1.mostrar();

  return 0;
}
```

```
Deudor: Luis Sanz Sanz (223344G), 30 años
Deuda pendiente: 3000 eur (Interés: 0.003%)
Desde el lunes 14 de octubre de 2002

Deudor: Luis Sanz Sanz (223344G), 30 años
Deuda pendiente: 2752.79 eur (Interés: 0.003%)
Desde el jueves 14 de noviembre de 2002

Deudor: Luis Sanz Sanz (223344G), 30 años
Deuda pendiente: 2752.79 eur (Interés: 0.005%)
Desde el jueves 14 de noviembre de 2002

Deudor: Luis Sanz Sanz (223344G), 30 años
Deuda pendiente: 2506.92 eur (Interés: 0.005%)
Desde el sábado 14 de diciembre de 2002
```

## Ejercicio nº 5

Crea una clase **Cuenta** que modele la información que se mantiene en un banco sobre cada cuenta corriente: cliente (una persona), fecha y hora de apertura de la cuenta, y saldo de la misma. Además de otros métodos que consideres adecuados, la clase debe proporcionar operadores suma y resta que devuelvan nuevas cuentas con la cantidad que se pase sumada o restada al saldo de la original. También se quiere disponer de las abreviaturas **+=** y **-=**, así como de un método **mostrar()** que muestre la información así:

```
Cliente: Luis Sanz Sanz (223344G), 30 años
Cuenta abierta el jueves 14 de noviembre de 2002 a las 11:15:25
Saldo actual: 2500 eur
```

Prueba la clase en una función **main()**.



## Ejercicio nº 5 — cuenta.h

```
// Clase Cuenta - Archivo de cabecera "cuenta.h"
#ifndef cuenta_h // Evitar inclusiones múltiples
#define cuenta_h

#include <iostream>
#include <string>
using namespace std;
#include "persona.h"
#include "fecha.h"
#include "hora.h"

class Cuenta {
public:
  Cuenta(Persona = Persona(), Fecha = Fecha(), Hora = Hora(),
         double = 0);
  Cuenta(const Cuenta&);
  Cuenta& operator=(const Cuenta&);
  ~Cuenta();
  Persona cliente() const;
  void cliente(Persona);
```

(*continúa*)



## Ejercicio nº 5 — cuenta.h

```
  Fecha fecha() const;
  void fecha(Fecha);
  Hora hora() const;
  void hora(Hora);
  double saldo() const;
  void saldo(double);
  Cuenta operator+(double) const;
  Cuenta operator-(double) const;
  Cuenta& operator+=(double);
  Cuenta& operator-=(double);
  void mostrar() const;
private:
  Persona _cliente;
  Fecha _fecha;
  Hora _hora;
  double _saldo;
};

#endif
```



## Ejercicio nº 5 — cuenta.cpp

```
// Clase Cuenta - Implementación "cuenta.h"
#include "cuenta.h"

Cuenta::Cuenta(Persona cliente, Fecha fecha, Hora hora,
  double saldo) : _cliente(cliente), _fecha(fecha), _hora(hora),
  _saldo(saldo) { }

Cuenta::Cuenta(const Cuenta& otra) {
  _cliente = otra._cliente;
  _fecha = otra._fecha;
  _hora = otra._hora;
  _saldo = otra._saldo;
}

Cuenta& Cuenta::operator=(const Cuenta& otra) {
  _cliente = otra._cliente;
  _fecha = otra._fecha;
  _hora = otra._hora;
  _saldo = otra._saldo;
  return *this;
}
```

(*continúa*)

## Ejercicio nº 5

cuenta.cpp

```
Cuenta::~Cuenta() { }

Persona Cuenta::cliente() const { return _cliente; }

void Cuenta::cliente(Persona p) { _cliente = p; }

Fecha Cuenta::fecha() const { return _fecha; }

void Cuenta::fecha(Fecha f) { _fecha = f; }

Hora Cuenta::hora() const { return _hora; }

void Cuenta::hora(Hora h) { _hora = h; }

double Cuenta::saldo() const { return _saldo; }

void Cuenta::saldo(double s) { _saldo = s; }
```

(*continúa*)





## Ejercicio nº 5

cuenta.cpp

```
Cuenta Cuenta::operator+(double cantidad) const {
  Cuenta tmp = *this;
  tmp._saldo += cantidad;
  return tmp;
}

Cuenta Cuenta::operator-(double cantidad) const {
  Cuenta tmp = *this;
  tmp._saldo -= cantidad;
  return tmp;
}

Cuenta& Cuenta::operator+=(double cantidad) {
  _saldo += cantidad;
  return *this;
}

Cuenta& Cuenta::operator-=(double cantidad) {
  _saldo -= cantidad;
  return *this;
}
```

(*continúa*)





## Ejercicio nº 5

cuenta.cpp

```
void Cuenta::mostrar() const {
  cout << endl;
  cout << "Cliente: " << _cliente.nombreCompleto() << " ("
       << _cliente.nif() << "), " << _cliente.edad()
       << " años" << endl;
  cout << "Cuenta abierta el " << _fecha.completa()
       << " a las " << _hora.hora() << endl;
  cout << "Saldo actual: " << _saldo << " eur" << endl;
}
```





## Ejercicio nº 5

prueba5.cpp

```
#include "cuenta.h"

int main() {
  Persona cli("223344G", 30, "Luis", "Sanz Sanz");
  Cuenta c1(cli, Fecha(14,11,2002), Hora(11, 15, 25), 2500);
  c1.mostrar();
  Cuenta c2(c1);
  c2.mostrar();
  Cuenta c3;
  c3.mostrar();
  c3 = c2;
  c3.mostrar();
  Cuenta c4;
  c4 = c3 + 234.57;
  c4.mostrar();
  c4 += 76;
  c4.mostrar();

  return 0;
}
```

```
Cliente: Luis Sanz Sanz (223344G), 30 años
Cuenta abierta el jueves 14 de noviembre de ...
Saldo actual: 2500 eur

Cliente: Luis Sanz Sanz (223344G), 30 años
Cuenta abierta el jueves 14 de noviembre de ...
Saldo actual: 2734.57 eur

Cliente: Luis Sanz Sanz (223344G), 30 años
Cuenta abierta el jueves 14 de noviembre de ...
Saldo actual: 2810.57 eur
```

## Ejercicio nº 6

Crea una clase **Evento** que modele la información que se mantiene en un programa sobre los eventos que maneja: responsable (una persona), fecha y hora de inicio del evento, fecha y hora de fin del evento, y nivel de servicio requerido. Además de otros métodos que consideres adecuados, la clase debe proporcionar operadores de incremento y decremento que incrementen o decrementen en una unidad el nivel de servicio. También habrá un método **duracion()** que devolverá la duración del evento en minutos, así como un método **mostrar()** que muestre la información de esta forma:

```
Responsable: Luis Sanz Sanz
Evento iniciado el jueves 13 de junio de 2002 a las 11:15:25
Evento terminado el miércoles 23 de octubre de 2002 a las 19:03:54
Duración: 190548 min.
Nivel de servicio: 8
```

Prueba la clase en una función **main()**.



## Ejercicio nº 6 — evento.h

```
// Clase Evento - Archivo de cabecera "evento.h"
#ifndef evento_h // Evitar inclusiones múltiples
#define evento_h

#include <iostream>
#include <string>
using namespace std;
#include "persona.h"
#include "fecha.h"
#include "hora.h"

class Evento {
public:
  Evento(Persona = Persona(), Fecha = Fecha(), Hora = Hora(),
         Fecha = Fecha(), Hora = Hora(), int = 0);
  Evento(const Evento&);
  Evento& operator=(const Evento&);
  ~Evento();
  Persona responsable() const;
  void responsable(Persona);
```

(*continúa*)



## Ejercicio nº 6 — evento.h

```
  Fecha fechaInicio() const;
  void fechaInicio(Fecha);
  Hora horaInicio() const;
  void horaInicio(Hora);
  Fecha fechaFin() const;
  void fechaFin(Fecha);
  Hora horaFin() const;
  void horaFin(Hora);
  int nivel() const;
  void nivel(int);
  Evento& operator++();
  Evento& operator--();
  long int duracion() const;
  void mostrar() const;
private:
  Persona _responsable;
  Fecha _fechaInicio, _fechaFin;
  Hora _horaInicio, _horaFin;
  int _nivel;
};
#endif
```



## Ejercicio nº 6 — evento.cpp

```
// Clase Evento - Implementación "evento.h"
#include "evento.h"

Evento::Evento(Persona responsable, Fecha fi, Hora hi,
               Fecha ff, Hora hf, int nivel)
  : _responsable(responsable), _fechaInicio(fi), _horaInicio(hi),
    _fechaFin(ff), _horaFin(hf), _nivel(nivel) { }

Evento::Evento(const Evento& otro) {
  _responsable = otro._responsable;
  _fechaInicio = otro._fechaInicio;
  _horaInicio = otro._horaInicio;
  _fechaFin = otro._fechaFin;
  _horaFin = otro._horaFin;
  _nivel = otro._nivel;
}

Evento& Evento::operator=(const Evento& otro) {
  _responsable = otro._responsable;
```

(*continúa*)

## Ejercicio nº 6

evento.cpp

```
  _fechaInicio = otro._fechaInicio;
  _horaInicio = otro._horaInicio;
  _fechaFin = otro._fechaFin;
  _horaFin = otro._horaFin;
  _nivel = otro._nivel;
  return *this;
}

Evento::~Evento() { }

Persona Evento::responsable() const { return _responsable; }

void Evento::responsable(Persona p) { _responsable = p; }

Fecha Evento::fechaInicio() const { return _fechaInicio; }

void Evento::fechaInicio(Fecha f) { _fechaInicio = f; }

Hora Evento::horaInicio() const { return _horaInicio; }

void Evento::horaInicio(Hora h) { _horaInicio = h; }
```

(*continúa*)





## Ejercicio nº 6

evento.cpp

```
Fecha Evento::fechaFin() const { return _fechaFin; }

void Evento::fechaFin(Fecha f) { _fechaFin = f; }

Hora Evento::horaFin() const { return _horaFin; }

void Evento::horaFin(Hora h) { _horaFin = h; }

int Evento::nivel() const { return _nivel; }

void Evento::nivel(int n) { _nivel = n; }

Evento& Evento::operator++() {
  _nivel++;
  return *this;
}

Evento& Evento::operator--() {
  _nivel--;
  return *this;
}
```

(*continúa*)





## Ejercicio nº 6

evento.cpp

```
long int Evento::duracion() const {
  const long int SPD = 86400;
  long int seg = (_fechaFin - _fechaInicio) * SPD +
                 (_horaFin - _horaInicio);
  return seg / 60;
}

void Evento::mostrar() const {
  cout << endl;
  cout << "Responsable: " << _responsable.nombreCompleto()
       << endl;
  cout << "Evento iniciado el " << _fechaInicio.completa()
       << " a las " << _horaInicio.hora() << endl;
  cout << "Evento terminado el " << _fechaFin.completa()
       << " a las " << _horaFin.hora() << endl;
  cout << "Duración: " << duracion() << " min." << endl;
  cout << "Nivel de servicio: " << _nivel << endl;
}
```





## Ejercicio nº 6

prueba6.cpp

```
#include "evento.h"

int main() {
  Persona resp("223344G", 30, "Luis", "Sanz Sanz");
  Evento e1(resp, Fecha(13,6,2002), Hora(11, 15, 25),
            Fecha(23,10,2002), Hora(19, 3, 54), 8);
  e1.mostrar();
  Evento e2(e1);
  e2.mostrar();
  Evento e3;
  e3.mostrar();
  e3 = e2;
  e3++;
  e3.mostrar();

  return 0;
}
```

```
Responsable: Luis Sanz Sanz
Evento iniciado el jueves 13 de junio de 2002 a las ...
Evento terminado el miercoles 23 de octubre de 2002 ...
Duración: 190548 min.
Nivel de servicio: 8

Responsable: Luis Sanz Sanz
Evento iniciado el jueves 13 de junio de 2002 a las ...
Evento terminado el miercoles 23 de octubre de 2002 ...
Duración: 190548 min.
Nivel de servicio: 9
```