

Una empresa de distribución desea implementar un sistema de gestión de ventas. Para ello se considerarán las siguientes entidades:

```
class Cliente {
    private
    int idCliente;
    double descuento;
    string nombre;
    string NIF;
    .....
};

class Venta {
    private:
    int idCliente;
    int idProducto;
    double total;
    .....
};

class Clientes {
    private:
    int numClientes;
    Cliente *losClientes;
    .....
};

class Ventas {
    private:
    int numVentas;
    Venta * lasVentas;
    .....
};
```

Los objetos de las clases Clientes y Ventas son **vectores dinámicos** de objetos de clase Cliente y Venta, respectivamente. Paralelamente a los datos que residen en memoria durante la ejecución del programa existen ficheros de texto que contienen datos de clientes y ventas. En la figura 1 mostramos un ejemplo de fichero de clientes (A) y ventas (B). Observe que los ficheros contienen una *palabra mágica* al principio y después los datos se organizan en líneas, separando los distintos valores con espacios:

- Ficheros de clientes: cada cliente aparece en una línea separada, conteniendo los valores de los campos idCliente, descuento, nombre y NIF (separados por espacios).
- Ficheros de ventas: cada venta aparece en una línea separada, conteniendo los valores de los campos idCliente, idProducto y total (separados por espacios).

FICHEROCLIENTES	FICHEROVENTAS
1 10.00 PINGUS S.A. A1234567	1 4 153
43 5.00 FERG INTEGRAL S.L. B5654521	1 3 450
7 3.24 JOSE MARIA PIO C876787653	43 1 95
.....

Figure 1: (A) Fichero de clientes y (B) fichero de ventas

Los datos de los clientes y ventas se guardan en ficheros de manera que se pueden crear objetos de la clase Clientes y Ventas a partir de su contenido, y después se puede guardar el contenido de los objetos en ficheros. Suponga que dispone de los siguientes métodos públicos:

- Para la clase Cliente y Clientes: ver Cliente.h y Clientes.h.
- Para las clases Venta y Ventas : ver Venta.h y Ventas.h.

1. Deberá implementar cualquier otra función/método que pudiera necesitar.
2. Si no se especificara la clase a la que pertenece un método deberá determinar cuál es la adecuada.

◁ Ejercicio 1 ▷ Métodos básicos [2 puntos]

Defina los siguientes métodos:

1. (0.5 puntos) **Constructor sin argumentos** para las clases Clientes y Ventas para crear una colección vacía. Indique si es necesario implementar el constructor sin argumentos de las clases Cliente y Venta. Si lo fuera, impleméntelo y justifique porqué es necesario. Si no lo fuera, explique la razón. Tenga en cuenta que en la clase Venta ya se proporciona un constructor con un argumento tipo string.
2. (0.5 puntos) Implemente el **destructor** de las clases Cliente, Clientes, Venta y Ventas si fueran necesarios. En caso contrario justifique porqué no es/son necesario(s).
3. (1 puntos) Implemente el **constructor de copia** y el **operador de asignación** para las clases Cliente, Clientes, Venta y Ventas si fueran necesarios. En caso contrario justifique porqué no es/son necesario(s).

◁ **Ejercicio 2** ▷ **Operadores aritméticos y combinados [3 puntos]**

1. (1.5 puntos) Sobrecargue el **operador** `+=` para las clases `Clientes` y `Ventas`. El objetivo es incorporar un nuevo objeto al final de la colección correspondiente. El operador recibe un dato `Cliente` o `Venta` según el caso. Para el caso de clientes, hay que tener en cuenta que no puede haber en la colección dos clientes con el mismo NIF. Si el cliente ya está en la colección no se hace nada.
2. (1.5 puntos) Sobrecargue el **operador binario** `+` para combinar dos objetos de la clase `Clientes` (podrían ser, por ejemplo, los clientes de dos empresas del grupo). El resultado debe ser un nuevo objeto de la clase `Clientes`. Tenga en cuenta que los objetos `Clientes` que se combinan podrían tener valores *repetidos* (p.e. un cliente que está en los dos objetos `Clientes` que se combinan aparece en ambos con el mismo NIF -aunque puede tener diferentes `idCliente`-). Si se detecta un cliente repetido (el mismo NIF) se mantienen los datos del cliente de la primera colección.

◁ **Ejercicio 3** ▷ **Operadores de flujo [1 punto]**

1. (0.5 puntos) Sobrecargue el operador `<<` para la clase `Clientes`. Cada cliente deberá aparecer en una línea separada, conteniendo los valores de los campos `idCliente`, `descuento`, `nombre` y `NIF` (separados por espacios). En la figura 1.A puede ver un ejemplo (no tenga en cuenta la primera línea). Debe considerarse que se dispone del método `toString` en la clase `Cliente`. Este método genera una cadena con los datos de la clase separados por espacios en blanco, con el formato correspondiente.
2. (0.5 puntos) Sobrecargue el operador `>>` para la clase `Ventas`. Los datos de cada venta aparecen en el flujo dispuestos en una línea con el formato indicado anteriormente. Tenga en cuenta que la clase `Venta` ya dispone de un constructor que recibe como argumento una cadena con los datos de una venta tal y como vienen en el fichero correspondiente.

◁ **Ejercicio 4** ▷ **Métodos y funciones para E/S [1.5 puntos]**

1. (0.75 puntos) Implemente un constructor de la clase `Ventas` con un argumento que indica el nombre de un fichero de texto que contiene una serie de líneas de texto con los datos de las ventas. La primera línea del archivo contiene la cadena **FICHEROVENTAS**. A continuación vendría la información de las ventas (figura 1.B).
2. (0.75 puntos) Implementar un método de la clase `Clientes` para guardar en el fichero nombre el contenido del objeto. El contenido de la colección no se modifica.

```
void escribirClientes (const string nombre) const;
```

◁ **Ejercicio 5** ▷ **Sobrecarga de los operadores relacionales `>` y `==` [1 punto]**

Implemente los operadores `>` y `==` para la clase `Ventas`. El criterio se basa en el valor de la *venta media*. La venta media de un objeto `Ventas` se calcula sumando el importe de todas las ventas y dividiendo por el número de ventas. Es decir, si `ventasEnero` y `ventasFebrero` son dos objetos de la clase `Ventas`, entonces `ventasEnero > ventasFebrero` si el valor de la venta media de `ventasEnero` es mayor que el de `ventasFebrero`.

El operador `==` se construye también de acuerdo al criterio del valor de la venta media.

◁ **Ejercicio 6** ▷ **Aplicación [1.5 puntos]**

Escriba un programa *completo* que reciba un número indeterminado de nombres de ficheros de ventas e informe de cual es el *mejor*. La comparación se efectuará en base al valor de la *venta media* y usando los operadores relacionales necesarios (pregunta 5).

El programa mostrará también el valor de la venta media y en el caso en el que hubiera más de un fichero con el mismo valor de venta media deberá mostrar los nombres de todos ellos.