Módulo de biblioteca «venta»

Problema 1.º – Definición de un módulo denominado «venta»

Los problemas planteados en esta clase van a utilizar vectores y registros de un tipo denominado «Venta», que representan información relativa a las distintas ventas de productos que realiza una empresa. Por cada venta, la empresa registra información acerca del código del producto vendido, el código del cliente a quien se ha vendido el producto, la cantidad de producto vendido y el precio unitario al que se ha vendido.

A continuación se muestra el fichero de cabecera («venta.h») de un módulo denominado venta, que incluye la definición del tipo Venta y de unas cuantas funciones para trabajar con registros de ese tipo. Se pide el código del fichero de implementación («venta.cpp») del módulo venta.

```
/*
 * Los registros de tipo «Venta» gestionan la información asociada a ventas
 * realizadas por una empresa, según se especifica en el enunciado.
 */
struct Venta {
    int producto;           // Código del producto vendido
    int cliente;            // Código del cliente a quien se ha vendido
    int cantidad;           // Cantidad de producto que se ha vendido
    double precioUnitario;  // Precio unitario al que se ha vendido el producto
};

/*
 * Pre:  ---
 * Post: Ha devuelto el precio total de venta del producto de la venta «venta».
 */
double precioTotal(const Venta venta);

/*
 * Pre:  ---
 * Post: Ha devuelto true si y solo si la venta «venta» es errónea, es decir,
 *       si la cantidad o el precio unitario o ambos son negativos.
 */
bool esErronea(const Venta venta);
```

Ficheros binarios de ventas

Un fichero binario almacena los datos de cada una de las ventas realizadas por la empresa referida anteriormente. La estructura del fichero responde al siguiente esquema:

```
<fichero_ventas> ::= { <venta> }
<venta> ::= registro de tipo Venta
```

Por ejemplo, un fichero que respondiera a dicho formato podría tener el siguiente contenido, donde los datos se representan en base diez, agrupados por llaves y separados por espacios en blanco y comas **solo** para facilitar la comprensión del contenido del mismo:

```
{ {117, 120552, 120, 3.15}, {122, 130922, 65, 6.40}, {105, 120552, 100, 3.16}, …,
{154, 137054, 75, 0.98} }
```

Problema 2.º

Diseñad la función totalFactura que se especifica a continuación. No es necesario utilizar ninguna estructura de datos indexada adicional (vector) para su resolución.

```
/*
 * Pre:  Existe un fichero binario de ventas con el nombre «nombreFichero» accesible
 *       para su lectura.
 * Post: Ha devuelto la cantidad total a facturar al cliente cuyo código es igual a
 *       «clienteFactura» por las ventas que le corresponden registradas en
 *       el fichero de ventas de nombre «nombreFichero».
 */
double totalFactura(const char nombreFichero[], const int clienteFactura);
```

Problema 3.º

Diseñad la función `eliminarErroneos` que se especifica a continuación. No es necesario utilizar ninguna estructura de datos indexada adicional (vector) para su resolución.

```
/*
 * Pre:  Existe un fichero binario de ventas con el nombre «nombreFicheroOriginal»
 *       accesible para su lectura y es posible crear o reescribir el fichero de
 *       nombre «nombreFicheroFinal» para su escritura.
 * Post: Ha copiado en el fichero de nombre «nombreFicheroFinal» las ventas
 *       almacenadas en el fichero de nombre «nombreFicheroOriginal» que no son
 *       erróneas y solo esas. Una venta es considerada errónea cuando la cantidad o
 *       el precio unitario o ambos son negativos.
 */
void eliminarErroneos(const char nombreFicheroOriginal[],
                      const char nombreFicheroFinal[]);
```

Problema 4.º

```
/*
 * Pre:  Existe un fichero binario de ventas con el nombre
 *       «nombreFichero» accesible para su lectura.
 * Post: Ha devuelto el número de clientes diferentes cuyas ventas están
 *       registradas en el fichero de ventas de nombre «nombreFichero».
 */
int numClientesDistintos(const char nombreFichero[]);
```

Problema 5.º

Diseñad la función `leerVentas` que se especifica a continuación:

```
/* Pre:  Existe un fichero binario de ventas de nombre «nombreFichero» accesible para
 *       su lectura y el número de ventas almacenados en el mismo es menor o igual a
 *       la dimensión del vector «ventas».
 * Post: Ha asignado a «nVentas» el número de ventas del fichero y ha almacenado las
 *       primeras «nVentas» componentes del vector «ventas» la información de las
 *       ventas almacenadas en el fichero.
 */
void leerVentas(const char nombreFichero[], Venta ventas[], int& nVentas);
```

Problema 6.º

Diseñad la función `guardarVentas` que se especifica a continuación:

```
/* Pre:  n ≥ 0
 * Post: Ha creado un fichero de nombre «nombreFichero» en el que ha almacenado la
 *       información codificada en binario de las «n» primeras componentes del
 *       vector «ventas».
 */
void guardarVentas(const char nombreFichero[], const Venta t[], const int n);
```