Facultad de Informática – Universidad Complutense

Fundamentos de la programación – Doble Grado

Curso 2013–2014

Examen final – 10 de junio de 2014

Tiempo disponible: 3 horas

Se pide construir un programa modular que muestre las ventas de productos de una empresa agrupadas por clientes. El programa constará de cuatro módulos: *Producto*, *ListaProductos*, *ListaClientes* y módulo principal (`main.cpp`).

Módulo *Producto* (0,5 puntos)

Declara un tipo de estructura `tProducto` con tres campos: código, precio y unidades. Implementa, al menos, los siguientes subprogramas:

- ✓ `muestra()`: Dado un producto muestra su información en la pantalla con el siguiente formato (código – unidades x precio = total):

```
787654      -      2 x      69.00 $ =      138.00 $
```

- ✓ `totalVenta()`: Dado un producto devuelve el total de la venta de ese producto.

Módulo *ListaProductos* (1,5 puntos) Máx. 200 productos

Declara un tipo `tListaProd` para listas de productos de longitud variable **implementadas con array dinámico** (sin ordenación). Implementa estos subprogramas:

- ✓ `insertaProd()`: Dada una lista de productos y un producto añade el producto al final de la lista. Si la lista está llena, se ignora el nuevo producto.
- ✓ `muestra()`: Dada una lista de productos muestra sus productos en la pantalla.
- ✓ `totalVentas()`: Dada una lista de productos devuelve el total de esas ventas.
- ✓ `destruye()`: Dada una lista de productos libera la memoria dinámica que usa.

Módulo *ListaClientes* (5 puntos) Máx. 100 clientes

Declara un tipo de estructura `tCliente` con dos campos: NIF y ventas (una lista de productos). Declara también un tipo `tListaClientes` para listas de clientes de longitud variable **implementadas con array estático de punteros a datos dinámicos**. La lista se debe mantener **ordenada por NIF** en todo momento.

El módulo implementa, al menos, los siguientes subprogramas:

- ✓ `encuentra()`: Dada una lista de clientes y un NIF localiza en la lista un cliente con ese NIF. Hará uso del algoritmo recursivo de búsqueda binaria.
- ✓ `totalVentas()`: Dada una lista de clientes devuelve el total de las ventas.

- ✓ `insertaCliente()`: Dada una lista de clientes, un NIF y un producto, añade un nuevo cliente con ese NIF (y ese producto como primera venta) en la lista de clientes. La lista de clientes debe seguir estando ordenada por NIF tras insertar. Si la lista está llena, se ignora el nuevo cliente.
- ✓ `carga()`: Dada una lista de clientes, carga en ella la información de un archivo `datos.txt` que contiene en cada línea la información de una venta: NIF del cliente, código de producto, precio y unidades vendidas.

  Ejemplo de archivo `datos.txt`:

```
87654321G 1234 125.95 10
12345678F 44237 16.4 5
32154637K 33768765 58 3
12345678F 5487 63.65 1
21212121Y 9990999 32.8 3
...
XXX
```

  El subprograma lee cada venta y localiza el cliente con el NIF. Si no existe se añade uno nuevo a la lista de clientes con ese primer producto. Si existe, se añade el producto a la lista de productos del cliente existente.
- ✓ `muestra()`: Dada una lista de clientes muestra sus ventas en la pantalla. Termina mostrando el total de las ventas. Por ejemplo:

```
------------------------
Cliente: 12345678F
44237       -      5 x      16.40 € =        82.00 €
5487        -      1 x      63.65 € =        63.65 €
787654      -      2 x      69.00 € =       138.00 €
888         -      4 x     156.50 € =       626.00 €
------------------------
Cliente: 21212121Y
9990999     -      3 x      32.80 € =        98.40 €
...
------------------------
Total ventas: 5305.66 €
```

- ✓ `destruye()`: Dada una lista de clientes libera la memoria dinámica que utiliza.

Módulo principal (0,5 puntos)

Carga la información del archivo `datos.txt` en una lista de clientes, muestra la información cargada y libera la memoria dinámica utilizada.

Se valorará la legibilidad, así como el uso adecuado de los esquemas de recorrido y búsqueda, de la comunicación entre subprogramas y de la memoria (2,5 puntos).

Entrega el código del programa a través del *librero* (sólo .cpp y .h, comprimidos en un ZIP). ¡Asegúrate de entregar una versión sin errores de compilación!