UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

Programación II – Curso 2021/22

Práctica 1: Enunciado

1. El problema

En esta práctica se implementarán una serie de funcionalidades para BIDFIC, una plataforma de subastas de productos. Será necesario diseñar una estructura de datos que permita almacenar conjuntamente toda la información asociada a los productos. En esta primera práctica se abordarán las tareas de gestión de productos, incluyendo altas, bajas y modificaciones de precio.

Como el objetivo de este trabajo es practicar la independencia de la implementación en los Tipos Abstractos de Datos (TADs), se pide crear dos implementaciones de una LISTA NO ORDENADA, las cuales deberán funcionar de manera intercambiable: una implementación ESTÁTICA y otra DINÁMICA. De este modo el programa principal no deberá realizar ninguna suposición sobre la forma en que está implementado el TAD.

2. Librería Types

Algunos tipos de datos se definirán en esta librería (`types.h`) ya que son necesarios para el problema a resolver y los usará tanto el TAD como el programa principal.

`NAME_LENGTH_LIMIT`	Longitud máxima de userId y productId (constante)
`tUserId`	Identificador de un usuario (`string`)
`tProductId`	Identificador de un producto (`string`)
`tProductCategory`	Categoría de producto (tipo enumerado: `{book, painting}`)
`tProductPrice`	Precio del producto (`float`)
`tBidCounter`	Contador de pujas (`int`)
`tItemL`	Datos de un elemento de la lista (un producto). Compuesto por: • `seller`: de tipo `tUserId` • `productId` de tipo `tProductId` • `productCategory` de tipo `tProductCategory` • `productPrice` de tipo `tProductPrice` • `bidCounter` de tipo `tBidCounter`

3. TAD Lista

Para mantener la lista de productos y su información asociada, el sistema utilizará un TAD Lista. Se realizarán dos implementaciones:

1. ESTÁTICA con arrays (`static_list.c`) con tamaño máximo 25.
2. DINÁMICA, simplemente enlazada, con punteros (`dynamic_list.c`).

3.1. Tipos de datos incluidos en el TAD Lista

`tList`	Representa una lista de productos
`tPosL`	Posición de un elemento de la lista
`LNULL`	Constante usada para indicar posiciones nulas

3.2. Operaciones incluidas en el TAD Lista

Una precondición común para todas estas operaciones (salvo `createEmptyList`) es que la lista debe estar previamente inicializada:

- `createEmptyList (tList) → tList`

Crea una lista vacía.
PostCD: La lista queda inicializada y no contiene elementos.

- `isEmptyList (tList) → bool`

Determina si la lista está vacía.

- `first (tList) → tPosL`

Devuelve la posición del primer elemento de la lista.
PreCD: La lista no está vacía.

- `last (tList) → tPosL`

Devuelve la posición del último elemento de la lista.
PreCD: La lista no está vacía.

- `next (tPosL, tList) → tPosL`

Devuelve la posición en la lista del elemento siguiente al de la posición indicada (o `LNULL` si la posición no tiene siguiente).
PreCD: La posición indicada es una posición válida en la lista.

- `previous (tPosL, tList) → tPosL`

Devuelve la posición en la lista del elemento anterior al de la posición indicada (o `LNULL` si la posición no tiene anterior).
PreCD: La posición indicada es una posición válida en la lista.

- `insertItem (tItemL, tPosL, tList) → tList, bool`

  Inserta un elemento en la lista antes de la posición indicada. Si la posición es `LNULL`, entonces se añade al final. **Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario.**

  PreCD: La posición indicada es una posición válida en la lista o bien nula (`LNULL`).

  PostCD: Las posiciones de los elementos de la lista posteriores a la del elemento insertado pueden haber variado.

- `deleteAtPosition (tPosL, tList) → tList`

  Elimina de la lista el elemento que ocupa la posición indicada.

  PreCD: La posición indicada es una posición válida en la lista.

  PostCD: Las posiciones de los elementos de la lista posteriores a la de la posición eliminada pueden haber variado.

- `getItem (tPosL, tList) → tItemL`

  Devuelve el contenido del elemento de la lista que ocupa la posición indicada.

  PreCD: La posición indicada es una posición válida en la lista.

- `updateItem (tItemL, tPosL, tList) → tList`

  Modifica el contenido del elemento situado en la posición indicada.

  PreCD: La posición indicada es una posición válida en la lista.

  PostCD: El orden de los elementos de la lista no se ve modificado.

- `findItem (tProductId, tList) → tPosL`

  Devuelve la posición **del primer elemento de la lista** cuyo identificador de producto se corresponda con el indicado (o `LNULL` si no existe tal elemento).

4. Descripción de la tarea

La tarea consiste en implementar un único programa principal (`main.c`) que procese las peticiones de los usuarios de BIDFIC con el siguiente formato:

`N productId userId productCategory productPrice`	***[N]ew**:* Alta de un nuevo producto
`D productId`	***[D]elete***: Baja de un producto
`B productId userId productPrice`	***[B]id***: Puja por un determinado producto
`S`	***[S]tats:*** Listado de los productos actuales y sus datos

En el programa principal se implementará un bucle que procese una a una las peticiones de los usuarios. Para simplificar tanto el desarrollo como las pruebas, el programa no obtendrá ningún dato por teclado, sino que leerá y procesará las peticiones de usuarios contenidas en un fichero (ver documento `EjecucionScript.pdf`) En cada iteración del bucle, el programa leerá del fichero una nueva petición y la procesará. Para facilitar la corrección de la práctica todas las peticiones del fichero van numeradas correlativamente.

Para cada línea del fichero de entrada, el programa:

1. **Muestra una cabecera con la operación a realizar.** Esta cabecera está formada por una primera línea con 20 asteriscos y una segunda línea que indica la operación tal y como se muestra a continuación:

```
********************
NN␣T:␣product␣PP␣seller/bidder␣UU␣category␣CC␣price␣ZZ
```

donde `NN` es el número de petición, `T` es el tipo de operación (`N`, `D`, `B` o `S`), `PP` es el identificador del producto, `UU` es identificador del usuario que vende el producto (seller) en la operación `[N]ew` o el pujador en la operación `[B]id`, `CC` es la categoría del producto y `ZZ` es el precio de este (con dos decimales), `␣` indica un espacio en blanco. Sólo se imprimirán los parámetros necesarios; es decir, para una petición `[S]tats` se mostrará únicamente "`01 S`", mientras que para una petición `[N]ew` se mostrará "`01 N: product Product1 seller User2 category book price 15.00`".

2. **Procesa la petición** correspondiente:

- Si la operación es **`[N]ew`**, se incorporará el producto al **final** de la lista de productos, poniendo el identificador de usuario, la categoría y el precio indicado. El contador de pujas se inicializará a 0. Además, se imprimirá el mensaje:

  ```
  *␣New:␣product␣PP␣seller␣UU␣category␣CC␣price␣ZZ
  ```

  El resto de los mensajes siguen el mismo formato.

  Si ya existiese un producto con ese identificador o no se haya podido realizar la inserción se imprimirá el siguiente mensaje:

  ```
  + Error: New not possible
  ```

- Si la operación es **`[D]elete`**, se buscará el producto en la lista, se borrará y se imprimirá el siguiente mensaje:

  ```
  * Delete: product PP seller UU category CC price ZZ bids II
  ```

  donde `II` es el número de pujas que había recibido dicho producto. Si no existiese ningún producto con ese identificador, se imprimirá el siguiente mensaje:

  ```
  + Error: Delete not possible
  ```

- Si la operación es **`[B]id`**, se buscará el producto, se modificará el precio y el contador de pujas y se mostrará el siguiente mensaje con el precio y el contador de pujas actualizado:

  ```
  * Bid: product PP seller UU category CC price ZZ bids II
  ```

  Si no existiese ningún producto con ese identificador, si el vendedor del producto es el mismo que el pujador o si el precio de la puja no es superior al precio actual, se imprimirá el mensaje:

```
+ Error: Bid not possible
```

- Si la operación es **[S]tats**, se mostrará la lista completa de productos actuales de la siguiente forma:

```
Product PP1 seller UU1 category CC1 price ZZ1 bids II1
Product PP2 seller UU2 category CC2 price ZZ2 bids II2
...
Product PPn seller UUn category CCn price ZZn bids IIn
```

A continuación, se mostrarán, para cada categoría de producto, el número de productos que se ofertan en esa categoría, la suma de precios de todos los productos de dicha categoría y el precio medio, con el formato siguiente:

```
Category  Products    Price  Average
Book      %8d %8.2f %8.2f
Painting  %8d %8.2f %8.2f
```

Donde `%8d` indica que el entero correspondiente está justificado a la derecha (con 8 dígitos) y `%8.2f` indica un tamaño total de 8 dígitos, con 2 decimales. Si la lista de usuarios estuviese vacía se imprimirá el mensaje:

```
+ Error: Stats not possible
```

5. Lectura de ficheros

Para facilitar el desarrollo de la práctica se proporciona el siguiente material: (1) Un directorio `CLion` que incluye un proyecto plantilla (`P1.zip`) junto con un fichero que explica cómo hacer uso del mismo (`Presentacion_uso_IDE.pdf`) y, (2) Un directorio `script` donde se proporciona un fichero (`script.sh`) que permite probar de manera conjunta los distintos archivos proporcionados. Además, se facilita un documento de ayuda para su ejecución (`EjecucionScript.pdf`). Nótese que, para que el `script` no dé problemas se recomienda **NO copiar directamente el código de este documento**, ya que el formato PDF puede incluir caracteres invisibles que darían por incorrectas salidas válidas.

6. Información importante

El documento `NormasEntrega.pdf`, disponible en la página web de la asignatura detalla claramente las normas de entrega. Para un adecuado **seguimiento de la práctica** se realizarán **entregas obligatorias parciales** antes de las fechas y con los contenidos que se indican a continuación:

- Seguimiento 1: viernes **4 de marzo** a las 22:00 horas. Implementación dinámica y prueba del TAD Lista (entrega de los ficheros `types.h`, `dynamic_list.c` y `dynamic_list.h`).

- Seguimiento 2: viernes **11 de marzo** a las 22:00 horas. Implementación estática y prueba del TAD Lista (entrega de los ficheros `types.h`, `static_list.c` y `static_list.h`).

Para comprobar el correcto funcionamiento de los TAD se facilita el fichero de prueba `test_list.c`. Se realizará una corrección automática usando el script proporcionado para ver si se supera o no el seguimiento (véase el documento `CriteriosEvauacion.pdf`).

Fecha límite de entrega de la práctica: viernes **25 de marzo** a las 22:00 horas.