UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

Programación II – Curso 24/25

Práctica 1: Enunciado

1. El problema

El problema a resolver en esta Práctica 1 consiste en implementar una serie de funcionalidades para una plataforma de subastas de consolas de videojuegos retro. Será necesario diseñar una estructura de datos que permita almacenar conjuntamente toda la información asociada a las consolas. En esta primera práctica se abordarán las tareas de gestión de consolas, 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.

Para facilitar la elaboración de esta primera práctica, se recomienda organizar el trabajo siguiendo las fases que se detallan a continuación.

2. Fase 1

Esta primera fase se centrará en el TAD. Para ello: (1) implementaremos una librería donde se incluyen los tipos de datos necesarios para el problema a resolver; y (2) implementaremos el TAD Lista en sus dos versiones, estática y dinámica.

2.1 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 consoleId (constante).
`tUserId`	Identificador de un usuario (`string`).
`tConsoleId`	Identificador de una consola (`string`).
`tConsoleBrand`	Marca de la consola (tipo enumerado: `{nintendo, sega}`).
`tConsolePrice`	Precio de la consola (`float`).
`tBidCounter`	Contador de pujas (`int`).
`tItemL`	Datos de un elemento de la lista (una consola). Compuesto por: • `seller`: de tipo `tUserId`

	• `consoleId` de tipo `tConsoleId` • `consoleBrand` de tipo `tConsoleBrand` • `consolePrice` de tipo `tConsolePrice` • `bidCounter` de tipo `tBidCounter`

2.2. TAD Lista

Para mantener la lista de consolas 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`).

2.2.1. Tipos de datos incluidos en el TAD Lista

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

2.2.2. 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 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 (tConsoleId, tList) → tPosL`
  Devuelve la posición **del primer elemento de la lista** cuyo identificador de consola se corresponda con el indicado (o `LNULL` si no existe tal elemento).

2.2.3. Testeo de la implementación del TAD

Una vez terminada la implementación del TAD Lista, es necesario comprobar el correcto funcionamiento de ésta mediante el fichero de prueba facilitado (`test.c`).

3. Fase 2

Una vez implementado el TAD, nos centraremos en el programa principal. La tarea consiste ahora en implementar un único programa (`main.c`) que procese las peticiones recibidas por la plataforma, que tienen el siguiente formato:

`N consoleId userId consoleBrand consolePrice`	***[N]ew:*** Alta de una nueva consola.
`D consoleId`	***[D]elete***: Baja de una consola.
`B consoleId userId consolePrice`	***[B]id***: Puja de un pujador por una determinada consola.
`S`	***[S]tats:*** Listado de las consolas 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 necesitará introducir 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 hace lo siguiente:

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:␣console␣CC␣seller/bidder␣UU␣brand␣BB␣price␣PP
```

donde `NN` es el número de petición, `T` es el tipo de operación (`N`, `D`, `B` o `S`), `CC` es el identificador de la consola, `UU` es identificador del usuario que vende la consola (*seller*) en la operación `[N]ew` o el pujador (*bidder*) en la operación `[B]id`, `BB` es la marca de la consola y `PP` 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: console Console1 seller User2 brand nintendo price 100.00`".

2. **Procesa la petición** correspondiente:

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

  ```
  *␣New:␣console␣CC␣seller␣UU␣brand␣BB␣price␣PP
  ```

  El resto de los mensajes siguen el mismo formato.

  Si ya existiese una consola 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: console CC seller UU brand BB price PP bids II
  ```

  donde `II` es el número de pujas que había recibido dicha consola.

  Si no existiese ninguna consola con ese identificador, se imprime el siguiente mensaje:

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

- Si la operación es **`[B]id`**, se buscará la consola, 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: console CC seller UU brand BB price PP bids II
  ```

  Si no existiese ninguna consola con ese identificador, si el vendedor de la consola 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 consolas actuales de la siguiente forma:

  ```
  Console CC1 seller UU1 brand BB1 price PP1 bids II1
  Console CC2 seller UU2 brand BB2 price PP2 bids II2
  ...
  Console CCn seller UUn brand BBn price PPn bids IIn
  ```

  A continuación, se mostrarán, para cada marca de consola, el número de consolas que se ofertan de esa marca, la suma de precios de todos las consolas de dicha marca y el precio medio, con el formato siguiente:

  ```
  Brand_____Consoles____Price__Average
  Nintendo__%8d_%8.2f_%8.2f
  Sega______%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 consolas estuviese vacía se imprimirá el mensaje:

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

4. Ejecución de la práctica

Para facilitar el desarrollo de la práctica se proporciona el siguiente material de especial interés: (1) un directorio `CLion` que incluye un proyecto plantilla (`P1.zip`) junto con un fichero que explica cómo hacer uso de éste (`Como_usar_IDE.pdf`); y (2) un directorio `script` que contiene 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 dicho *script* de pruebas no dé problemas, se recomienda encarecidamente **NO copiar-pegar directamente texto desde este documento al fichero de código**, ya que el formato PDF puede incluir caracteres invisibles que darían por incorrectas salidas (aparentemente) válidas.

5. Documentación del código

El código deberá estar convenientemente **comentado**, incluyendo las variables empleadas. Los comentarios han de ser concisos pero explicativos. Asimismo, después de la cabecera de cada procedimiento o función del programa y/o librería, se incluirá la siguiente información correspondiente a su **especificación**, tal y como se explicó en el TGR correspondiente:

- *Objetivo* del procedimiento/función.
- *Entradas* (identificador y breve descripción, una por línea).
- *Salidas* (identificador y breve descripción, una por línea).
- *Precondiciones* (condiciones que han de cumplir las entradas para el correcto funcionamiento de la subrutina).
- *Postcondiciones* (otras consecuencias de la ejecución de la subrutina que no quedan reflejadas en la descripción del objetivo o de las salidas).

6. Información importante

El documento `NormasEntrega_CriteriosEvaluacion.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:

1. **Entrega parcial #1**: viernes **7 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` (solamente dichos ficheros).

2. **Entrega parcial #2:** viernes **14 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` (solamente dichos ficheros).

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

Fecha límite de entrega de la práctica: viernes **21 de marzo** a las 22:00 horas.