FACULTADE DE INFORMÁTICA
Programación II – Curso 2024/25

Práctica 2: Enunciado

1. El Problema

La segunda práctica consistirá en añadir un conjunto de nuevas funcionalidades al programa desarrollado en la primera práctica. En concreto, se incluirá una **pila de pujas para cada consola**.

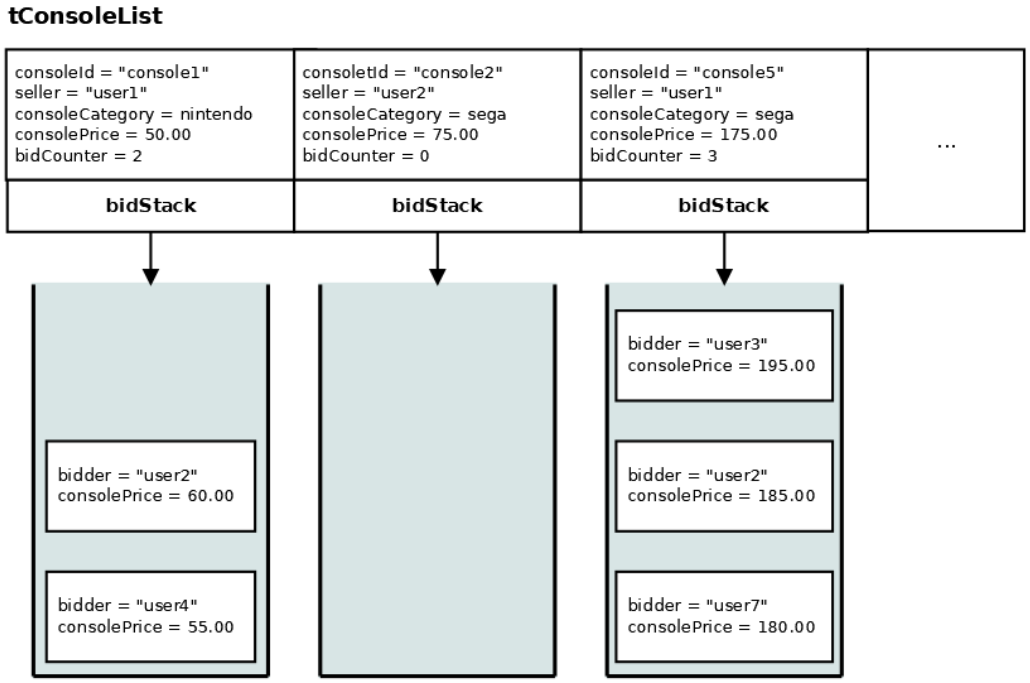
El objetivo de esta práctica es comprender el funcionamiento e implementar varios tipos abstractos de datos (TADs) así como manejar interdependencias entre ellos.

2. Fase 1

Para resolver este problema se utilizarán 2 tipos abstractos de datos (TADs):

- Una Lista Ordenada (TAD ConsoleList) para almacenar la lista de consolas.
- Una Pila (TAD BidStack) para almacenar las pujas por cada consola en la subasta.

Estos dos TADs se unen para permitir que cada consola, esto es, cada nodo de la lista ordenada, tenga su propia pila de pujas, tal y como representa la siguiente figura:



2.1 Librería `types.h`

Algunos tipos de datos comunes se definirán en el fichero `types.h`, ya que se utilizarán en varios TADs.

`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`).

2.2 TAD ConsoleList

Para mantener la lista de consolas, el sistema utilizará un TAD Lista ordenada con implementación **dinámica, simplemente enlazada** (`console_list.c` y `console_list.h`).

2.2.1. Tipos de datos incluidos en el TAD ConsoleList

`tList`	Representa una lista de consolas **ordenada alfabéticamente** por el identificador de consola (`consoleId`).
`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` • `bidStack`: de tipo `tStack` (pila con las pujas recibidas por la consola)
`tPosL`	Posición de un elemento de la lista de consolas.
`LNULL`	Constante usada para indicar posiciones nulas de la lista de consolas.

2.2.2. Operaciones incluidas en el TAD ConsoleList

Las operaciones de este TAD son idénticas a las indicadas para el TAD Lista en la Práctica 1 (consulte el enunciado de dicha práctica), únicamente cambian de especificación las siguientes operaciones:

- `insertItem (tItemL, tList) → tList, bool`
  Inserta un elemento en la lista de forma **ordenada** por el campo `consoleId`. Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario.
  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 **y la consola en dicha posición tiene una pila de pujas vacía.**
  PostCD: Las posiciones de los elementos de la lista posteriores a la de la posición eliminada pueden haber variado.

Asimismo, la operación `findItem` no cambia su especificación, pero en su implementación debería tener en cuenta que la lista está **ordenada**.

2.3 TAD BidStack

Este TAD permitirá almacenar las pujas recibidas por una consola, y su implementación corresponde a una **pila estática** para un máximo de 25 pujas (`bid_stack.c` y `bid_stack.h`).

2.3.1. Tipos de datos incluidos en el TAD BidStack

`tStack`	Representa una pila de pujas.
`tItemS`	Datos de un elemento de la pila, es decir, una puja. Compuesto por: • `bidder`: de tipo `tUserId` (el usuario que hace la puja). • `consolePrice`: de tipo `tConsolePrice` (el importe que ofrece).
`tPosS`	Posición de un elemento de la pila de pujas.
`SNULL`	Constante usada para indicar posiciones nulas de la pila.

2.3.2. Operaciones incluidas en el TAD BidStack

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

- `createEmptyStack (tStack) → tStack`
  Crea una pila vacía.
  PostCD: la pila no tiene elementos.
- `push (tItemS, tStack) → tStack, bool`
  Inserta un elemento en la cima de la pila. Devuelve un valor `true` si el elemento fue apilado; `false` en caso contrario.
- `pop (tStack) → tStack`
  Elimina de la pila el elemento situado en la cima.
  PreCD: La pila no está vacía.
- `peek (tStack) → tItemS`
  Recupera el elemento de la cima de la pila sin eliminarlo.
  PreCD: La pila no está vacía.
- `isEmptyStack (tStack) → bool`
  Determina si una pila está vacía o no.

3. Fase 2

Una vez implementados los TADs, nos centraremos en el programa principal. La tarea consiste en implementar un único programa (`main.c`) que **haga uso de ConsoleList** y de **BidStack** y 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 por una determinada consola.
`A consoleId`	***[A]ward:*** Se asigna el ganador de la puja por una consola.
`I`	***[I]nvalidateBids***: Se eliminan las pujas de aquellas consolas con un número de pujas *excesivo*.
`R`	***[R]emove:*** Elimina las consolas sin pujas.
`S`	***[S]tats:*** Listado de las consolas y sus datos.

En el programa principal se implementará un bucle que procese una a una las peticiones sobre las consolas. Para simplificar el desarrollo y las pruebas, el programa no necesitará introducir ningún dato por teclado, sino que leerá y procesará las peticiones contenidas en un fichero (ver documento `EjecucionScript_P2.pdf`). En cada iteración del bucle, el programa leerá del fichero una nueva petición y la procesará. Para facilitar la corrección de la práctica 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:␣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`, `A`, `I`, `R` o `S`); `CC` es el identificador de la consola; `UU` es identificador del usuario (precedido de la palabra `seller` o `bidder` dependiendo de la operación); `BB` es la marca de la consola; `PP` es su precio (con dos decimales); y `␣` 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 añadirá la consola a la lista de consolas, con su correspondiente identificador de consola, el identificador de usuario de su vendedor, su marca y su precio. El contador de pujas de la consola se inicializará a 0 y se le asociará una nueva pila de pujas vacía. Además, se imprimirá el mensaje:

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

  Si ya existiese una consola con ese identificador o la consola no se pudiera insertar, se imprimirá el siguiente mensaje:

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

- Si la operación es ***[D]elete***, se buscará la consola en la lista, se borrará de la lista, **eliminando todas las pujas existentes**, y se imprimirá el siguiente mensaje:

  ```
  * Delete: console CC seller UU brand BB price PP bids II
  ```

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

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

- Si la operación es ***[B]id***, se buscará la consola y, si la nueva puja supera la puja más alta actual (o el precio original, si no hubiese pujas), entonces: (i) se añadirá la nueva puja a su pila; (ii) se actualizará el contador de pujas; y (iii) se mostrará el siguiente mensaje con el precio de la puja:

  ```
  * Bid: console CC bidder UU brand BB price PP bids II
  ```

  Si no existiese ninguna consola con dicho identificador (`consoleId`), el pujador (`bidder`) y el vendedor (`seller`) fuesen la misma persona, el precio de la puja no fuese superior a la puja más alta actual, o la pila estuviese llena, se imprimirá entonces el mensaje:

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

- Si la operación es ***[A]ward***, se buscará la consola y se mostrará el siguiente mensaje:

  ```
  * Award: console CC bidder UU brand BB price PP
  ```

  donde `UU` es el ganador de la puja y `PP` es el precio final. Dado que la consola ha sido vendida, deberá ser eliminada de la lista de consolas tras vaciar su pila de pujas.

  Si no existiese ninguna consola con dicho identificador (`consoleId`) o su pila de pujas estuviese vacía, se imprimirá el mensaje:

  ```
  + Error: Award not possible
  ```

  En este último caso, con la pila vacía, no se eliminará la consola de la lista.

- Si la operación es ***[I]nvalidateBids*** se eliminan todas las pujas de aquellas consolas con un número de pujas superior a dos veces la media del número de pujas de todas las consolas de la lista. Para aquellas consolas que cumplan esta condición, se vaciará su pila de pujas y se mostrará el siguiente mensaje:

  ```
  * InvalidateBids: console CC seller UU brand BB price PP bids
  II average bids AA
  ```

  Donde `CC` es el identificador de consola, `UU` es el vendedor de la consola, `BB` la marca, `PP` el precio original, `II` el número actual de pujas por la consola y `AA` el número medio de pujas de la lista de consolas (con 2 cifras decimales).

  Si no es posible invalidar las pujas de ninguna consola, bien porque la lista está vacía o porque no se cumple la condición descrita, se imprimirá el mensaje:

  ```
  + Error: InvalidateBids 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 top bidder BB1
  Console CC2 seller UU2 brand BB2 price PP2. No bids
  Console CC3 seller UU3 brand BB3 price PP3 bids II3 top bidder BB3
  ...
  Console CCn seller UUn brand BBn price PPn bids IIn top bidder BBn
  ```

  donde `PPi` es el precio original. A continuación, se mostrarán, para cada marca, el número de consolas ofertadas de esa marca, la suma de precios originales de esas consolas, así como su precio medio; todo ello con el siguiente formato:

  ```
  Brand_____Consoles____Price__Average
  Nintendo__%8d_%8.2f_%8.2f
  Sega______%8d_%8.2f_%8.2f
  ```

  Por último, se mostrará la consola para la que el importe de su mejor puja suponga el mayor incremento (en términos porcentuales) sobre su precio original:

  ```
  Top bid: console CC seller UU brand BB price PP bidder BB top price
  TT increase RR%
  ```

  donde `RR` es el porcentaje de dicho incremento (con 2 cifras decimales). Si no existiera ninguna puja, se mostrará el mensaje:

  ```
  Top bid not possible
  ```

  Si la lista de consolas estuviese vacía, se imprimirá el mensaje:

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

- Si la operación es ***[R]emove***, se eliminarán de la lista aquellas consolas que no tengan pujas. Se mostrará el siguiente mensaje:

```
Removing console CC1 seller UU1 brand BB1 price PP1 bids II1
Removing console CC2 seller UU2 brand BB2 price PP2 bids II2
Removing console CC3 seller UU3 brand BB3 price PP3 bids II3
...
Removing console CCn seller UUn brand BBn price PPn bids IIn
```

  Si no existiese ninguna consola sin pujas, se mostrará el mensaje:

```
+ Error: Remove 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 (`Plantilla-Template_P2.zip`) 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_P2.pdf`). Nótese que, para que el `script` no dé problemas, se recomienda **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 dos **entregas parciales obligatorias** antes de las fechas y con los contenidos que se indican a continuación:

- **Entrega parcial #1: viernes 4 de Abril a las 22:00 horas.** Implementación y prueba del TAD ConsoleList y el TAD BidStack (entrega de los ficheros `types.h`, `console_list.c`, `console_list.h`, `bid_stack.c` y `bid_stack.h`). Para comprobar el correcto funcionamiento de los TAD se facilitarán los ficheros de prueba `test_console_list.c` y `test_bid_stack.c`.
- **Entrega parcial #2: viernes 25 de Abril a las 22:00 horas.** Implementación y prueba de las siguientes funcionalidades del programa principal: ***[N]ew***, ***[B]id*** y ***[S]tats*** (entrega de los ficheros `types.h`, `console_list.c`, `console_list.h`, `bid_stack.c`, `bid_stack.h` y `main.c`). Para comprobar el correcto funcionamiento de las operaciones se facilitarán los ficheros de prueba `new.txt` y `bid.txt`.

Se realizará una corrección automática usando el script proporcionado (véase documento `NormasEntrega_CriteriosEvaluacion.pdf`).

Fecha límite de entrega: **viernes 2 de mayo de 2025 a las 22:00 horas.**