UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación

Programación de Sistemas y Concurrencia

Final Junio. 2015/2016
Ing. Computadores
Ing. Informática
Ing. Software

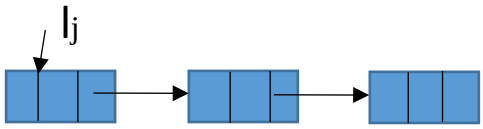
APELLIDOS _____ NOMBRE _____
DNI _____ ORDENADOR _____ GRUPO _____

Lenguaje C

Durante la celebración de la Eurocopa se ha establecido una forma de almacenar los goles que se van marcando en los partidos. Nos proporcionan un archivo binario con toda la información, pero a nosotros solo nos interesan los máximos goleadores. Por tanto tendremos que hacer un programa que lea el fichero y construya la lista de goleadores. Para ello se pide:

1. ListaJugadores.h

En el fichero ListaJugadores.h, define el tipo TListaJugadores para que sea capaz de almacenar secuencias de nodos, cada uno de los cuales tiene un campo con un unsigned int, que es el **número de jugador**, otro unsigned int que es el **número de goles marcados** y un campo para almacenar la dirección del nodo siguiente en la lista. La lista estará **ordenada** por el identificador del jugador.



2. ListaJugadores.c:

Implementar las siguientes funciones:

```
//crea una lista vacía (sin ningún nodo)
void crear(TListaJugadores *lj);

//inserta un nuevo jugador en la lista de jugadores, poniendo 1 en el número
de goles marcados. Si ya existe añade 1 al número de goles marcados.
void insertar(TListaJugadores *lj,unsigned int id);

//recorre la lista de jugadores escribiendo los identificadores y los goles
// marcados
void recorrer(TListaJugadores lj);

//devuelve el número de nodos de la lista
int longitud(TListaJugadores lj);


//Eliminar. Toma un número de goles como parámetro y elimina todos los
// jugadores que hayan marcado menos que ese número de goles
void eliminar(TListaJugadores *lj,unsigned int n);
```

```
// Devuelve el ID del máximo goleador. Si la lista está vacía, devuelve 0. Si
//hay más de un jugador con el mismo número de goles que el máximo devuelve
//el de mayor ID. Hay que devolver el identificador, no el número de goles
//que ha marcado
unsigned int maximo(TListaJugadores *lj);

//Destruye la lista y libera la memoria)
void destruir(TListaJugadores *lj);
```

3. CargarFichero

En el programa Principal.c implementa un procedimiento denominado *cargarFichero* que lea el fichero binario cuyo nombre se pasa por parámetro e inserte los identificadores de los jugadores que se encuentre en el fichero, llamando al procedimiento *insertar* del segundo apartado. En el campus virtual se puede encontrar el fichero Principal.c pero con el procedimiento cargarFichero sin terminar.
Hay que tener en cuenta que el archivo binario aparece otra información aparte del identificador del jugador. En concreto, aparece el identificador de partido, el número de jugador y el minuto en el que se ha marcado. Todos estos campos son de tipo unsigned int.

Así por ejemplo, si en el partido de id 12, el jugador de número 104 ha marcado en el minuto 87 y en el partido de identificador 13 se han marcado 2 goles, ambos por el jugador de número 227 en los minutos 20 y 55, en el fichero aparecerá la información:

12	104	87
13	227	20
13	227	55

Por tanto, para este ejemplo el procedimiento cargar fichero deberá llamar al procedimiento insertar 3 veces, con los valores 104, 227 y 227.

Concurrencia en memoria compartida

Supón que un Centro Comercial dispone de unos aseos *suficientemente* grandes para dar servicio a sus **clientes**. Además de los clientes, el Centro Comercial tiene **un Equipo de Limpieza** que periódicamente limpia los aseos. El sistema debe satisfacer las siguientes condiciones de sincronización:

a) Cualquier número de clientes puede estar simultáneamente utilizando los aseos. Se supone que son tan grandes que, cuando un cliente quiere entrar, siempre hay sitio disponible.
b) Mientras el equipo de limpieza trabaja en los aseos, no puede haber ningún cliente dentro.

Implementa este sistema de forma injusta y justa con semáforos y monitores. Es decir,

1. **Semáforos Binarios**.
   a) Implementa una versión injusta de este sistema, en la que el Equipo de Limpieza podría estar siempre esperando para entrar en los aseos.
   b) Modifica la solución anterior para que sea justa para el Equipo de Limpieza, de manera que un cliente que llega, si ve que el Equipo está esperando, porque hay clientes en los aseos, espera a que el Equipo de Limpieza haya terminado su trabajo para entrar.
2. **Métodos Sincronizados/Locks.**
   a) Implementa una solución injusta del sistema anterior utilizando métodos sincronizados o locks.

b) Implementa una solución justa para el Equipo de Limpieza del sistema anterior utilizando métodos sincronizados o locks.

Para todas las soluciones, utiliza el esqueleto que se encuentra en el campus virtual (es el mismo para todos los ejercicios), en el que existe una clase **Aseos** que proporciona los siguientes métodos:

```
/**
 * Utilizado por el cliente id cuando quiere entrar en los aseos
 * CS Version injusta: El cliente espera si el equipo de limpieza
 * está trabajando
 * CS Version justa: El cliente espera si el equipo de limpieza está
 * trabajando o está esperando para poder limpiar los aseos

 */
public void entroAseo(int id){}

/**
 * Utilizado por el cliente id cuando sale de los aseos
 *
 */
public void salgoAseo(int id){}

/**
 * Utilizado por el Equipo de Limpieza cuando quiere entrar
 *  en los aseos
 * CS: El equipo de trabajo está solo en los aseos, es decir, espera
 * hasta que no  haya ningún cliente.
 *
 */

 public void entraEquipoLimpieza(){}


/**
 * Utilizado por el Equipo de Limpieza cuando  sale de los aseos
 */
 public void saleEquipoLimpieza(){}
```