Práctica 1: La liga robótica en casa

Programación 2

Curso 2024-2025

Esta práctica del curso consiste en el desarrollo de un simulador de liga de fútbol robótica. **Para resolver esta práctica solo puedes usar los elementos de C++ vistos en las transparencias del *Tema 1* de teoría.**

Condiciones de entrega

- La fecha límite de entrega para esta práctica es el **viernes 21 de febrero**, hasta las **23:59**
- Debes entregar un único fichero llamado `prac1.cc` con el código de todas las funciones

Código de honor

Si se detecta copia (total o parcial) en tu práctica, tendrás un **0** en la entrega y se informará a la dirección de la Escuela Politécnica Superior para que adopte medidas disciplinarias

Está bien discutir con tus compañeros posibles soluciones a las prácticas
Está bien apuntarte a una academia si sirve para obligarte a estudiar y hacer las prácticas

Está mal copiar código de otros compañeros o pedirle a ChatGPT que te haga la práctica
Está mal apuntarte a una academia para que te hagan las prácticas

Si necesitas ayuda acude a tu profesor/a
No copies

Normas generales

- Debes entregar la práctica exclusivamente a través del servidor de prácticas del Departamento de Lenguajes y Sistemas Informáticos (DLSI). Se puede acceder a él de dos maneras:
  - Página principal del DLSI (`https://www.dlsi.ua.es`), opción "ENTREGA DE PRÁCTICAS"
  - Directamente en la dirección `https://pracdlsi.dlsi.ua.es`
- Cuestiones que debes tener en cuenta al hacer la entrega:
  - El usuario y la contraseña para entregar prácticas son los mismos que utilizas en UACloud
  - Puedes entregar la práctica varias veces, pero sólo se corregirá la última entrega
  - No se admitirán entregas por otros medios, como el correo electrónico o UACloud
  - No se admitirán entregas fuera de plazo
- Tu práctica debe poder ser compilada sin errores con el compilador de C++ existente en la distribución de Linux de los laboratorios de prácticas

- Si tu práctica no se puede compilar su calificación será 0
- Al comienzo de todos los ficheros fuente entregados debes incluir un comentario con tu NIF (o equivalente) y tu nombre. Por ejemplo:

prac1.cc

```
// DNI 12345678X GARCIA GARCIA, JUAN MANUEL
...
```

- Superar **todas** las pruebas del autocorrector te da derecho a presentarte al examen de prácticas. Si falla alguna prueba no podrás presentarte al examen y tu calificación en esta práctica será 0
- El cálculo de la nota de la práctica y su relevancia en la nota final de la asignatura se detallan en las transparencias de presentación de la asignatura (*Tema 0*)

1 Descripción de la práctica

El objetivo de esta práctica es programar un simulador de competición de fútbol robótico, al estilo RoboCup.[1] En este simulador tendrás la opción de introducir equipos que contarán cada uno de ellos con cinco jugadores virtuales. El programa simulará los resultados de los partidos creando tablas de clasificación y de mejores jugadores por cada equipo.

2 Detalles de implementación

En el Moodle de la asignatura se publicarán varios ficheros que necesitarás para la correcta realización de la práctica:

- `prac1.cc`. Este fichero contiene un esqueleto de programa sobre el que realizar tu práctica. Descárgalo y añade tu código en él. Este fichero contiene la siguiente información:
  - Los registros (`struct`) necesarios para hacer la práctica
  - Un tipo enumerado `Error` que contiene todas las posibles clases de error que se pueden dar en esta práctica (por ejemplo, `ERR_OPTION`)
  - Una función `error` que se encarga de mostrar el correspondiente mensaje de error por pantalla en función del parámetro que se le pase. Por ejemplo, cuando la función recibe el parámetro `ERR_OPTION`, mostrará por pantalla el mensaje `ERROR: wrong menu option`
  - Una función `showMenu` que muestra por pantalla el menú del programa
  - Una función `simulateGoals` que simula los goles marcados por los jugadores robóticos de cada equipo
  - Una función `main` que implementa la gestión del menú principal y llama a las funciones correspondientes dependiendo de la opción elegida por el usuario
- `autocorrector-prac1.tgz`. Contiene los ficheros del autocorrector para evaluar la práctica con distintas pruebas de entrada. La corrección automática de la práctica, tras la entrega, se realizará con estas mismas pruebas y será necesario superarlas todas para poder presentarte al examen de esta práctica
- `prac1`. Fichero ejecutable de la práctica (compilado para máquinas Linux de 64 bits) desarrollado por el profesorado de la asignatura, para que puedas probarlo con las entradas que quieras y ver la salida correcta esperada

[1] `https://www.robocup.org/`.

- En el autocorrector de la práctica se introducirán siempre datos del tipo correcto, aunque con valores que pueden ser incorrectos. Por ejemplo, se puede meter una opción incorrecta en el menú, que podría ser `z` o `9`, pero nunca se introducirá un valor de otro tipo (cadena de caracteres, número real, ...)
- El resto de decisiones en la implementación de la práctica quedan a tu criterio

3 Funcionamiento del programa

El programa que vas a desarrollar mostrará un menú que te permitirá crear y eliminar equipos de tu liga robótica, además de mostrar estadísticas de los equipos, como los partidos jugados, ganados, empatados y el número de puntos conseguido. También permitirá ver la lista de mejores jugadores de cada equipo (los que más goles han conseguido).

Una vez definida la lista de equipos, el programa hará todas las combinaciones posibles de partidos (todos los equipos jugarán contra todos en algún momento) y simulará los resultados utilizando números aleatorios. Además, se podrá ver la clasificación final de los equipos tras la simulación.

4 Componentes

El programa a desarrollar debe poder gestionar dos elementos fundamentales: los *equipos*, que representan a cada uno de los conjuntos robóticos que participan en la competición, y los *jugadores*, que son cada uno de los robots que juegan en cada equipo. Los siguientes apartados describen la estructura de cada uno de estos componentes en detalle.

4.1 Jugador

Las estructuras de tipo `Player` almacenan la información relativa a los jugadores. Cada jugador contendrá un nombre (`name`), el número de goles marcados (`goals`) y una variable que indica si es el mejor jugador del equipo (`best`), es decir, el que más goles ha marcado en la competición. Cada equipo tendrá un único mejor jugador que tendrá este campo a `true`. El resto del equipo tendrá ese campo a `false`. Toda esta información se almacenará en un registro con el siguiente formato:

```
struct Player{
    char name[kPLAYERNAME];
    unsigned int goals;
    bool best;
};
```

La constante `kPLAYERNAME` es un entero que tiene el valor `50`, lo que implica que como máximo el nombre del robot podrá tener 49 caracteres (ten en cuenta que hay que guardar también el `'\0'`). En ninguna de las pruebas del autocorrector se introducirán cadenas de más de 49 caracteres para el nombre, por lo que no es necesario comprobarlo.

4.2 Equipo

Un equipo tendrá asociado un identificador único (`id`), un nombre (`name`), el número de victorias (`wins`), el número de derrotas (`losses`), el número de empates (`draws`), el número de puntos conseguidos (`points`) y un array con los jugadores que pertenecen al equipo (`players`). Para almacenar toda esta información se usará un registro de tipo `Team` con la siguiente estructura:

```
struct Team{
    unsigned int id;
    char name[kTEAMNAME];
    unsigned int wins;
    unsigned int losses;
```

```
    unsigned int draws;
    unsigned int points;
    Player players[kPLAYERS];
};
```

La constante `kTEAMNAME` es un entero que tiene el valor `40`. La constante `kPLAYERS` indica el número máximo de jugadores que puede tener un equipo. En esta práctica ese valor será `5`.

5 Menú

Al iniciar el programa, se mostrará por pantalla el menú principal, quedando a la espera de que el usuario elija una opción:

Terminal

```
1- Add team
2- Add all teams
3- Delete team
4- Show teams
5- Play league
6- Show standings
7- Show best players
q- Quit
Option:
```

Las opciones válidas que puede introducir el usuario son los números del `1` al `7` y la letra `q`. Si la opción elegida no es ninguna de éstas (por ejemplo un `9`, una `x` o un `;`), se emitirá el error `ERR_OPTION` llamando a la función `error` con dicho parámetro. Cuando el usuario elige una opción correcta, se debe ejecutar el código asociado a dicha opción. Al finalizarla, volverá a mostrar el menú principal y a pedir otra opción, hasta que el usuario decida salir del programa utilizando la opción `q`.

6 Opciones

En los siguientes apartados se describe el funcionamiento que deberá tener cada una de las siete opciones que se muestran en el menú principal del programa.

6.1 Add team

Esta opción permite al usuario introducir un nuevo equipo en el simulador haciendo uso de la función `addTeam` que deberás implementar. Esta función se activa cuando el usuario elige la opción `1` del menú principal.

Lo primero que hará el programa es comprobar que no se ha superado el número máximo de equipos, que es de 20. Este valor está definido en la constante `kMAXTEAMS` del fichero `prac1.cc` proporcionado en Moodle. Si ya existen 20 equipos en el programa, se deberá emitir el error `ERR_MAX_TEAMS` y se volverá al menú principal. Si no se ha llegado al límite, el programa asignará un identificador único al equipo. El primer equipo que se introduzca en el programa tendrá el valor `id` igual a `0`, el segundo tendrá el valor `1` y así sucesivamente.

A continuación, el programa pedirá el nombre del equipo con el siguiente mensaje:

Terminal

```
Enter team name:
```

Si se introduce una cadena vacía (el usuario pulsa la tecla *enter* sin haber escrito nada), el programa asignará un nombre automático al equipo que empezará por `Team_` y tendrá a continuación el valor del `id` del equipo. Por ejemplo, si el equipo tiene el `id` igual a `12`, el equipo se llamará `Team_12`. Los campos `wins`, `losses`, `draws` y `points` deberán inicializarse a `0` al crear el equipo.

A continuación, el programa deberá crear la lista de jugadores y almacenarlos en el campo `players` del registro `Team`. El nombre del jugador se generará automáticamente, comenzando por el nombre del equipo, seguido de la cadena `-R` y finalmente el número de jugador (valores de `1` a `5`). Por ejemplo, si el equipo se llama `Team_12`, el primer jugador creado se llamará `Team_12-R1`, el segundo `Team_12-R2` y así sucesivamente. El número de goles de cada jugador se inicializará a `0` y el campo `best` a `false`.

Tras crear los jugadores y asignarlos al equipo se volverá al menú principal.

- Para transformar un número entero en su correspondiente array de caracteres puedes usar la función `sprintf` de la siguiente manera:

  ```
  char number[3];
  sprintf(number,"%d",12);
  ```

  En este ejemplo, se guardaría la cadena `"12"` en la variable `number`.

6.2 Add all teams

¿Te puede la pereza y no te apetece meter los equipos uno a uno? Esta opción permite crear de manera automática toda la lista de equipos. El código se implementará en la función `addAllTeams` y se activará cuando el usuario elija la opción `2` del menú. Lo primero que hará la función es comprobar si ya existe algún equipo en el sistema, en cuyo caso advertirá de que se eliminarán todos antes de crear los nuevos mediante el mensaje:

Terminal
```
Do you want to delete existing teams (y/n)?
```

Si el usuario responde `n` o `N`, el programa volverá al menú principal sin hacer ningún cambio. Si contesta `y` o `Y`, se descartarán los equipos existentes para añadir los nuevos. Si el usuario introduce cualquier otra cosa, el programa volverá a mostrar el mensaje pidiendo confirmación.

Cuando el usuario confirme que quiere borrar los equipos, el programa preguntará el número de equipos nuevos que se quieren crear, mostrando para ello el siguiente mensaje:

Terminal
```
Enter number of teams:
```

Si el número introducido es menor que 2 o mayor que 20, se emitirá el error `ERR_NUM_TEAMS` y se volverá a solicitar que introduzca el número con el mismo mensaje anterior. Si el número es correcto, se procederá a crear tantos equipos como se hayan indicado, incluidos sus jugadores.

Los nombres de los equipos seguirán el mismo formato que se indicó en la sección anterior cuando el usuario dejaba el nombre en blanco: `Team_+id`. Por ejemplo, si el usuario ha indicado que quiere crear tres equipos, el primero se llamará `Team_0`, el segundo `Team_1` y el tercero `Team_2`. Los nombres de los robots, igualmente, seguirán el formato indicado en la sección anterior. Por ejemplo, el primer robot del primero equipo creado se llamará `Team_0-R1`.

Tras introducir los datos de todos los equipos y sus jugadores el programa volverá al menú principal. Ten en cuenta que el usuario podrá seguir añadiendo equipos de manera individual a esa lista (si no ha llegado al máximo) mediante la opción *Add team*, además de poder borrar los que no quiera con la opción *Delete team* que se explica a continuación.

- Si existían equipos antes de elegir esta opción del menú, al borrarlos e insertar nuevos el identificador del primer equipo que se crea comenzará de nuevo desde `0`

6.3 Delete team

Esta opción permite eliminar un equipo junto con todos sus jugadores. Para ello, deberás implementar la función `deleteTeam`. Ésta se activa cuando el usuario elige la opción `3` del menú. En primer lugar se comprobará que haya equipos para borrar. Si no hay equipos, se emitirá el error `ERR_NO_TEAMS` y se volverá al menú principal. Si existe algún equipo, se pedirá al usuario que introduzca el nombre del equipo que se desea eliminar con el siguiente mensaje:

Terminal

```
Enter team name:
```

Si el usuario introduce una cadena vacía se emitirá el error `ERR_EMPTY` y se volverá al menú principal. En caso de no existir el equipo introducido, se emitirá el error `ERR_NOT_EXIST` y se volverá a pedir al usuario que introduzca el valor mostrando de nuevo el mensaje anterior. Si se introduce un equipo existente, se eliminará del programa junto con todos sus jugadores y se volverá al menú principal.

- Aunque borres equipos, el siguiente `id` que debes asignar no se verá afectado. Es decir, si has añadido cinco equipos, el siguiente que añadas tendrá `id=5` (recuerda que empiezan en `0`). Si borras entonces tres equipos, el siguiente que añadas tendrá `id=6` (no se ve afectado porque hayas borrado)

6.4 Show Teams

Esta opción permite mostrar toda la información almacenada para los equipos, implementando para ello la función `showTeams`. Ésta se activa cuando el usuario elige la opción `4` del menú principal. En primer lugar se comprobará que haya equipos para mostrar. Si no hay equipos, se emitirá el error `ERR_NO_TEAMS` y se volverá al menú principal. A continuación, se pedirá al usuario que introduzca el nombre del equipo del que se desea mostrar datos con el siguiente mensaje:

Terminal

```
Enter team name:
```

Si no existe un equipo con ese nombre, se emitirá el error `ERR_NOT_EXIST` y se volverá a pedir de nuevo el nombre mostrando el mensaje anterior. Si el usuario introduce una cadena vacía se mostrará la información de todos los equipos con el formato que se indica más abajo. Si el nombre introducido es correcto, se mostrará toda la información del equipo introducido con el siguiente formato:

Terminal

```
Name: Los delincuentes
Wins: 12
Losses: 3
Draws: 1
Points: 37
Los delincuentes-R1: 12 goals
Los delincuentes-R2: 3 goals
Los delincuentes-R3: 0 goals
Los delincuentes-R4: 4 goals
Los delincuentes-R5: 9 goals
```

Como muestra el ejemplo, además del nombre, victorias, derrotas, empates y puntos, se mostrará el nombre de los cinco jugadores, ordenados del primero al último, y el número de goles que ha conseguido cada uno. Una vez mostrada la información, se volverá al menú principal.

6.5 Play league

Esta opción realizará la simulación de la competición, donde cada equipo jugará una vez contra todos los demás. Deberás implementar la función `playLeague`, que se activa cuando el usuario elige la opción `5` del menú principal. Lo primero que deberá comprobar esta función es que el número de equipos sea mayor que `1` (si no, no puede jugar contra nadie). En caso contrario, se mostrará el error `ERR_NUM_TEAMS` y se volverá al menú principal.

El programa deberá generar todas las combinaciones posibles de partidos entre los equipos. Por ejemplo, si tenemos cuatro equipos, *A*, *B*, *C* y *D*, los posibles partidos a disputar serían *A vs B*, *A vs C*, *A vs D*, *B vs C*, *B vs D* y *C vs D*. Como puedes ver, si hay un partido *A vs B* no tiene que haber uno *B vs A*. Los equipos se enfrentan entre ellos una sola vez.

Para cada partido, se utilizará la función `simulateGoals` desarrollada por el profesorado y que se incluye en el fichero `prac1.cc` que hay en Moodle. Esta función recibe como parámetro un equipo (`Team`) y para cada jugador decide aleatoriamente si ha metido un gol o no, sumando el total de goles y devolviéndolo como resultado de la función.

Por ejemplo, dados dos equipos, el siguiente código calcula los goles de cada uno y comprueba quién ha metido más:

```
Team a,b; // Imaginamos que los equipos tienen todos sus datos ya introducidos

// Simulamos el partido A vs B
int goalsA=simulateGoals(a);
int goalsB=simulateGoals(b);

if(goalsA>goalsB){
    // Ha ganado A
}
```

Tras la simulación del partido, el programa deberá actualizar las victorias (`wins`), derrotas (`losses`), empates (`draws`) y puntos (`points`) de cada equipo. Cada victoria suma 3 puntos, cada empate 1 y cada derrota 0.

Por otra parte, como se indicó en la Sección 4.1, cada equipo tiene un mejor jugador que tendrá el campo `best` a `true`. Al final de la competición se deberá determinar quién ha sido el mejor jugador de cada equipo. En caso de empate a goles entre varios jugadores, se elegirá al jugador que primero aparezca en la lista de jugadores del equipo.

Si el usuario decide volver a ejecutar esta opción del programa, se borrará la clasificación anterior de los equipos y se generará una nueva con todas las estadísticas desde cero (se eliminarán las victorias, derrotas, goles de los jugadores, etc.).

6.6 Show standings

Esta opción mostrará por pantalla la clasificación final del torneo después de haber jugado los partidos. Para mostrar esta información deberás implementar la función `showStandings`. Se activa cuando el usuario elige la opción `6` del menú principal. Si no se ha jugado todavía ninguna competición, es decir, el usuario no ha ejecutado nunca la opción `Play league`, la función deberá emitir el error `ERR_NO_LEAGUE` y volver al menú principal.

En caso contrario, se mostrará por pantalla la información de la clasificación final de la competición, ordenando a los equipos por puntuación, de mayor a menor. En caso de empate a puntos entre dos equipos, se mostrará primero el equipo que se hubiera introducido antes en el programa.

Para cada equipo se mostrará la siguiente información: `name|wins|draws|losses|points`. A continuación se muestra un ejemplo de salida por pantalla (se muestran solo tres equipos de ejemplo, pero tu código tiene que mostrarlos todos):

Terminal

```
Los delincuentes|15|3|2|48
Imbatibles|12|8|0|44
Kaka de Luxe|8|8|4|32
```

- Ten en cuenta que esta ordenación es solo a la hora de mostrar por pantalla. Internamente no debes reordenar los equipos y tienes que mantener el mismo orden en el que fueron introducidos

6.7 Show best players

Esta opción mostrará por pantalla la lista de los mejores jugadores de cada equipo. Para mostrar esta información deberás implementar la función `showBestPlayers` que se activa cuando el usuario elige la opción 7 del menú principal. Al igual que sucedía con `Show standings`, Si no se ha jugado todavía ninguna competición el programa deberá mostrar el error `ERR_NO_LEAGUE` y volver al menú principal.

En caso contrario, se mostrará la lista de equipos (en el orden en el que fueron creados, no en el orden de puntuación de la competición) y el mejor jugador de cada uno de ellos, junto con el número de goles marcado. El formato para cada equipo será el siguiente: `team name|robot name|goals`. A continuación se muestra un ejemplo de salida por pantalla con tres equipos de ejemplo, aunque tu código debe mostrar todos los equipos:

Terminal

```
UA|UA-R3|12
Torreznos Digitales|Torreznos Digitales-R2|18
Sopa de ajo|Sopa de ajo-R1|22
```

En este ejemplo, el mejor jugador del equipo `UA` es el robot llamado `UA-R3` que ha marcado `12` goles.