Algoritmos y Estructura de Datos

Trabajo Práctico

Simulación de Torneo de Fútbol

Se deberá desarrollar un sistema que permita simular los resultados de los partidos de un torneo de fútbol y ver la tabla de posiciones a cada fecha.

Pantallas a desarrollar

Menú principal

Se debe presentar un menú con las siguientes opciones:

```
1. Generar fixture.
2. Simular partidos.
3. Ver equipo.
4. Ver tabla de posiciones.
5. Guardar fixture.
6. Cargar fixture.
7. Salir.
Elija una opción:
```

Luego de ejecutar la opción correspondiente debe volver a mostrar este menú (excepto la opción de salir).

Generar fixture

Esta opción debe generar un fixture a partir de los equipos participantes. Un fixture es la asignación de partidos a cada fecha del torneo de modo tal que todos los equipos jueguen contra todos los demás.

Simular partidos

Esta opción simula los resultados de los partidos hasta una fecha ingresada por el usuario. En caso de ya haberse simulado algunos partidos debe preguntar si comenzar la simulación de cero o continuar la anterior desde donde quedó.

```
Ingrese una fecha: 9
Ya se han simulado partidos. ¿Continuar? S
Se han simulado los partidos hasta la fecha 9.
```

Ver equipo

Aquí el usuario puede ver los partidos correspondientes a un equipo junto con los resultados, en caso de haber sido simulados.

```
Ingrese un equipo: Sacachispas FC
Partidos de Sacachispas FC
Sacachispas FC            3 – 1 Excursionistas
Deportivo Riestra         1 – 1 Sacachispas FC
Sacachispas FC            3 – 0 Sportivo Dock Sud
```

```
Defensores de Cambaceres    -   Sacachispas FC
                          ...
```

Ver tabla de posiciones

Se muestra al usuario la tabla de posiciones a una fecha en particular.

La tabla debe estar ordenada por puntos descendente, diferencia de gol descendente y goles a favor descendente.

```
Ingrese una fecha: 3
Tabla de posiciones
Equipo                        Puntos  PJ  PG  PE  PP  GF  GC  DG
Defensores de Cambaceres         9     3   3   0   0   5   1   4
Sacachispas FC                   7     3   2   1   0   7   2   5
Excursionistas                   7     3   2   1   0   5   0   5
...
```

Si el usuario ingresa 0 como fecha, debe mostrarse la tabla actualizada hasta la última fecha simulada.

Guardar fixture

Se guarda el estado actual del fixture en un archivo, de manera tal que pueda ser leído al usar la opción “Cargar fixture”.

Cargar fixture

Lee de un archivo el fixture completo. En caso de tener un fixture en memoria este se descarta antes de leer el archivo.

Salir

Sale de la aplicación.

Información de los equipos

La información de los equipos estará almacenada en un archivo de texto llamado equipos.txt. Para la lectura del mismo se usará la función leerEquipo:

```
bool leerEquipo(FILE *archivo, char nombre[31], int &paramA, int &paramB);
```

El primer parámetro es el archivo ya abierto para lectura. En caso de haber podido leer se devuelve en el resto de los parámetros el nombre del equipo y dos valores de rendimiento del equipo que deben ser usados durante la simulación de los partidos. Retorna verdadero si pudo leer y falso en caso contrario. Puede usarse de la siguiente manera:

```
while (leerEquipo(arch, nombre, paramA, paramB)) {
    // Hacer algo con nombre, paramA y paramB
}
```

Se provee un archivo de equipos, pero puede ser modificado por el grupo para probar distintos escenarios del sistema.

Restricciones

- Hay a lo sumo 100 equipos en un torneo y al menos 2.

Persistencia

Deben guardarse los datos de todos los partidos de cada fecha junto con los resultados, en caso de haber sido simulados. Al cargarse un fixture se debe verificar que la cantidad de equipos en el fixture corresponda con la cantidad de equipos existentes, para evitar inconsistencias en caso de haber modificado el archivo de equipos.

El archivo de fixtures puede ser binario o de texto, a decisión del grupo.

Generación del fixture

Para la generación del fixture se provee el siguiente procedimiento:

```
void generarPartido(int equipos, int fecha, int partido, short &local, short
&visitante);
```

El parámetro *equipos* es la cantidad de equipos en el torneo y *fecha* y *partido* la fecha y número de partido que se quiere generar. Devuelve dos números entre 0 (inclusive) y *equipos* (excluído), *local* y *visitante*, siendo estos dos valores que pueden ser usados como índice de los equipos correspondiente al local y visitante del partido.

La cantidad de fechas de un torneo depende de la cantidad de equipos (n). Si esta es par son $n - 1$. Si es impar es n. La cantidad de partidos por fecha es $n / 2$, redondeado hacia abajo.

Dado que no es aleatoria la generación de partidos, utilizando solo el procedimiento *generarPartido* siempre se genera el mismo fixture. Para evitar esto, y con el fin de que los fixtures varíen al ser generados, se deben "mezclar" los equipos o las fechas (o ambos) antes de comenzar la generación del fixture. Para realizar el proceso aleatorio pueden usarse las funciones *rand* y *srand* (investigar su uso).

Simulación de los partidos

Para la simulación se provee el procedimiento *simularPartido*:

```
void simularPartido(int localA, int localB, int visitanteA, int visitanteB,
int &golesLocal, int &golesVisitante);
```

Los primeros cuatro parámetros corresponden a los valores *paramA* y *paramB* del local y visitante respectivamente. Devuelve en *golesLocal* y *golesVisitante* la cantidad de goles que hizo cada equipo en el partido. La simulación sí es aleatoria, pero ponderada según los parámetros de cada equipo, por lo tanto, para dos equipos, diferentes invocaciones de simularPartido pueden dar resultados distintos.

Información adicional

- Tras un partido se asignan 3 puntos al equipo que ganó y 0 al que perdió. Si hubo empate cada uno lleva 1 punto.
- Si el número de equipos es impar, en cada fecha uno queda libre. Salvo en la última fecha, algunos equipos van a tener un partido jugado más que otros.

Condiciones generales

- El trabajo práctico será realizado en grupos integrados por 5 alumnos como máximo.
- Presentar una carpeta con carátula que especifique los datos de los integrantes con toda la documentación del programa: enunciado, estrategia, representación gráfica del algoritmo, codificación en C/C++, juegos de datos de entrada y resultados de la ejecución.