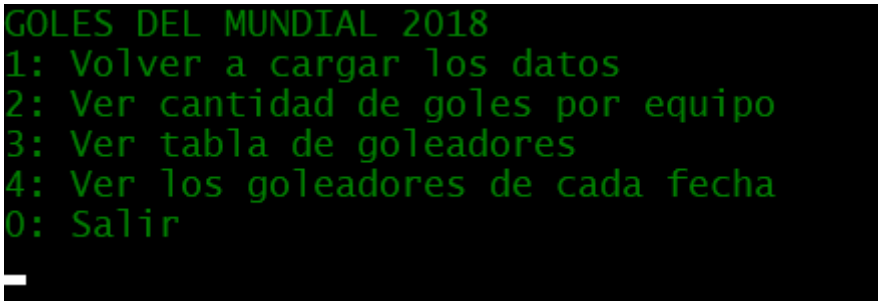
Aclaraciones

- Se interpretó que la funcionalidad del ejercicio 3 era una especificación de cómo debía funcionar una funcionalidad del ejercicio 2. Se utiliza la palabra “emitir” pero no se especifica si se refiere a “emitir por pantalla” o a “emitir un archivo”, así que tomamos la decisión de optar por emitir por pantalla. Es por eso por lo que no existe una carpeta con el ejercicio 3, ya que el mismo se encuentra dentro del 2.
- Los equipos están ordenados según sus posiciones una vez finalizado el mundial.
- Los nombres de los equipos están almacenados en un array de strings, ya que por consigna no hay ninguna estructura que los contenga.
- No se consideraron los goles en contra para la tabla de goleadores, en su lugar, se decidió por contarlos aparte y mostrarlos al final de esta.

Ejercicio 2 y 3 – Instructivo

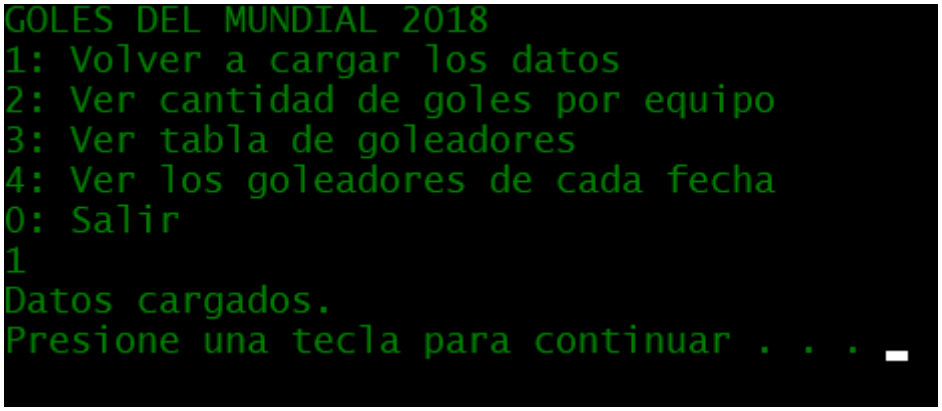
Ejecución

Opciones del menú:



El menú principal muestra al usuario 5 opciones principales.

Opción 1:

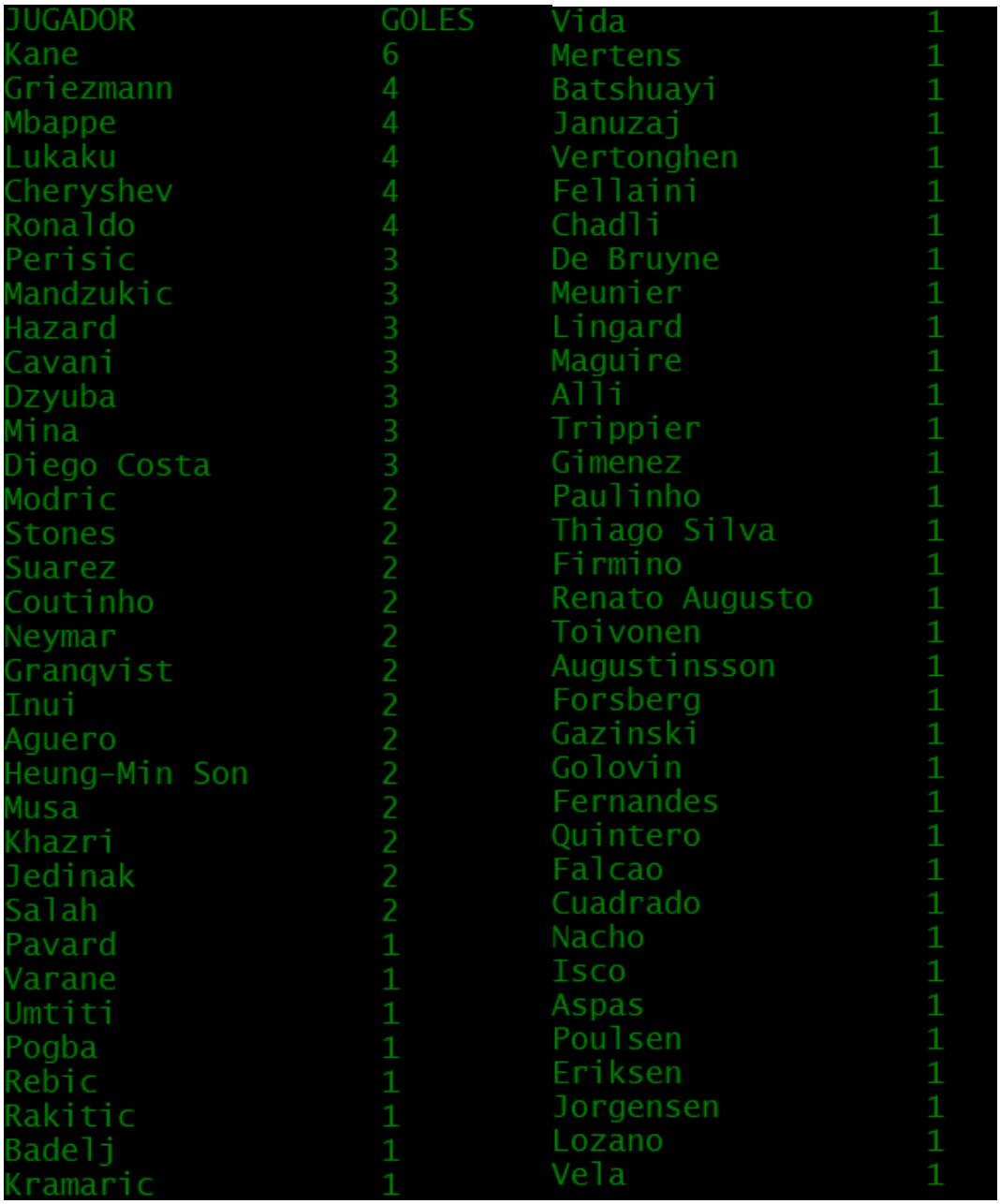


Se muestra ante el usuario con una leyenda de “Datos cargados” al momento en que se cargaron los datos del archivo para ser utilizados.

Opción 2:

Se muestra ante el usuario la cantidad de goles por equipos, ordenados en forma descendiente.

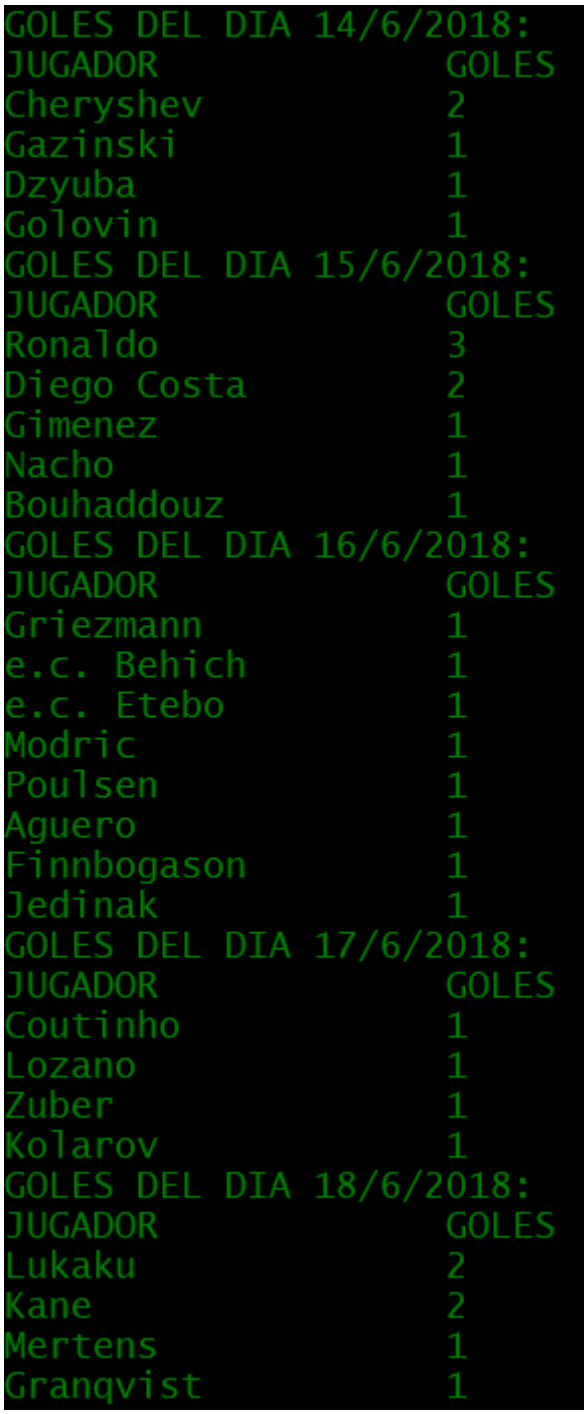
Opción 3:



JUGADOR	GOLES	Vida	1
Kane	6	Mertens	1
Griezmann	4	Batshuayi	1
Mbappe	4	Januzaj	1
Lukaku	4	Vertonghen	1
Cheryshev	4	Fellaini	1
Ronaldo	4	Chadli	1
Perisic	3	De Bruyne	1
Mandzukic	3	Meunier	1
Hazard	3	Lingard	1
Cavani	3	Maguire	1
Dzyuba	3	Alli	1
Mina	3	Trippier	1
Diego Costa	3	Gimenez	1
Modric	2	Paulinho	1
Stones	2	Thiago Silva	1
Suarez	2	Firmino	1
Coutinho	2	Renato Augusto	1
Neymar	2	Toivonen	1
Granqvist	2	Augustinsson	1
Inui	2	Forsberg	1
Aguero	2	Gazinski	1
Heung-Min Son	2	Golovin	1
Musa	2	Fernandes	1
Khazri	2	Quintero	1
Jedinak	2	Falcao	1
Salah	2	Cuadrado	1
Pavard	1	Nacho	1
Varane	1	Isco	1
Umtiti	1	Aspas	1
Pogba	1	Poulsen	1
Rebic	1	Eriksen	1
Rakitic	1	Jorgensen	1
Badelj	1	Lozano	1
Kramaric	1	Vela	1

Se muestra ante el usuario los goleadores del mundial, ordenados descendientemente.

Opción 4:



Se muestra ante el usuario los goles, ordenados por las fechas de los partidos.

Las siglas “e.c.”, hacen referencia a “en contra”.

Funcionamiento del main (explicación en el respectivo orden)

```
376 int main(){
377     Nodo<Jugador>* partidos[CANTEQUIPOS][CANTPARTIDOS] = { NULL };
378     int menu = 0;
379     cargarArchivo(partidos, "../goles.DAT");
380     while(true){
381         cout<< "GOLES DEL MUNDIAL 2018"<<endl;
382         cout<< "1: Volver a cargar los datos" <<endl;
383         cout<< "2: Ver cantidad de goles por equipo" <<endl;
384         cout<< "3: Ver tabla de goleadores" << endl;
385         cout<< "4: Ver los goleadores de cada fecha" << endl;
386         cout<< "0: Salir" <<endl;
387         cin>> menu;
388         switch(menu){
389             case 1: cargarArchivo(partidos, "goles.DAT"); cout<<"Datos cargados."<<endl; break;
390             case 2: golesPorEquipo(partidos); break;
391             case 3: goleadores(partidos); break;
392             case 4: goleadoresPorFecha(partidos); break;
393             case 0: return 0; break;
394         }
395         system("pause");
396         system("CLS");
397     }
398     return 0;
399 }
```

Línea 377: declaración de la matriz de 32x7 (32 equipos, 7 partidos como máximo). Previamente se definieron las estructuras “Nodo” (template) y “Jugador”.

```
43  template<typename T>
44  struct Nodo {
45      T info;
46      Nodo<T>* sig;
47  };
48
```

```
65  struct Jugador{
66      char nombre_jugador[20];
67      int fecha;
68      int goles;
69  };
```

43 a 48: declaración de la estructura template Nodo

65 a 69: declaración de la estructura Jugador

379: ejecución de la función “cargarArchivo”, la cual recibe la matriz previamente declarada y la ruta al archivo a ser leído.

Dentro de cargarArchivo

```
217 void cargarArchivo(Nodo<Jugador>* partidos[][CANTPARTIDOS], const char* ruta){
218     for(int i=0; i<CANTEQUIPOS;i++){
219         for(int j=0; j<CANTPARTIDOS; j++){
220             liberar<Jugador>(partidos[i][j]);
221         }
222     }
223     int partido = 0;
224     int equipo = 0;
225     Nodo<Jugador>* aux = NULL;
226     FILE* f = fopen(ruta, "rb+");
227     Gol lectura = read<Gol>(f);
228     Gol ant;
229     while(!feof(f)){
230         if(strcmp(lectura.nombre_jugador,"NULL")!=0){
231             aux = buscar<Jugador,Gol>(partidos[equipo][partido],lectura,criterioJugadorGol);
232             if(aux==NULL){
233                 agregarNodo<Jugador>(partidos[equipo][partido],crearJugador(lectura));
234             } else {
235                 aux->info.goles++;
236             }
237         }
238         ant = lectura;
239         lectura = read<Gol>(f);
240         if(ant.cod_equipo != lectura.cod_equipo){
241             partido = 0;
242             equipo++;
243         } else if(ant.id_partido != lectura.id_partido) {
244             partido++;
245         }
246     }
247     fclose(f);
248     return;
249 }
```

218 a 222: Se limpian los datos de la matriz, para luego ser cargada.

226: Se declara el puntero hacia el archivo binario de la ruta especificada, de tipo lectura binaria.

227 a 246: Se leen secuencialmente los registros de estructura “Gol” (declarada previamente) hasta el final del archivo, utilizando el template de la cátedra “read”. Para ello se utiliza la variable “lectura”.

```
56  struct Gol{
57      int id_gol;
58      int cod_equipo;
59      int fecha;
60      char nombre_jugador[20];
61      int id_partido;
62
63  };
```

```
116 template<typename T> T read(FILE* f)
117 {
118     T buff;
119     fread(&buff,sizeof(T),1,f);
120     return buff;
121 }
```

223 y 224: los enteros “equipo” y “partido” son utilizados para recorrer la matriz, ubicando los datos en la misma.

230: Si el nombre del jugador es “NULL”, significa que en ese partido el equipo no hizo goles, entonces se omite el paso de ubicar datos en el casillero actual de la matriz.

231: Un auxiliar busca si ya existen registros de un jugador con el mismo nombre dentro de la posición actual de la matriz (utilizando el criterio de comparación criterioJugadorGol, dentro del template de la cátedra “buscar”).

```
123 int criterioJugadorGol(Jugador j, Gol g){
124     return strcmp(j.nombre_jugador,g.nombre_jugador)==0?0:-1;
125 }
```

```
184 template <typename T, typename K>
185 Nodo<T>* buscar(Nodo<T>* p, K v, int (*criterio)(T,K)) {
186     Nodo<T>* aux = p;
187     while( aux!=NULL && criterio(aux->info,v)!=0 ) {
188         aux=aux->sig;
189     }
190     return aux;
191 }
```

232: En caso de no encontrarlo, se agrega un nuevo jugador al partido (utilizando el template “agregarNodo” de la cátedra y la función crearJugador, que recibe la lectura tipo Gol y devuelve una variable de tipo Jugador).

```
139 template <typename T>
140 void agregarNodo(Nodo<T>*& p, T v) {
141     Nodo<T>* nuevo = new Nodo<T>();
142     nuevo->info = v;
143     nuevo->sig = NULL;
144     if( p==NULL ) {
145         p = nuevo;
146     } else {
147         Nodo<T>* aux = p;
148         while(aux->sig!=NULL ) {
149             aux = aux->sig;
150         }
151         aux->sig = nuevo;
152     }
153 }
```

```
91 Jugador crearJugador(Gol g){
92     Jugador j;
93     j.goles = 1;
94     j.fecha = g.fecha;
95     strcpy(j.nombre_jugador, g.nombre_jugador);
96     return j;
97 }
```

235: En caso de encontrarlo, incrementa su variable “goles” en 1.

238 a 245: Se procede a definir si ya se leyeron todos los goles de un partido (y todos los partidos de un equipo), asignando a la variable “ant” la “lectura” hecha, ejecutando una nueva “lectura” y comparando el código de equipo y el código de partido de ambos. Primero, en caso de que el código de equipo sea distinto, se retorna a la columna 0 de “partidos”, y se pasa a la fila siguiente de “equipos”. Caso contrario, se verifica si la nueva lectura corresponde a un partido distinto (aunque el equipo fuera el mismo). En caso de no serlo, se incrementan en 1 los “partidos”.

247: Una vez ejecutada la lectura de la totalidad de los elementos del archivo, se cierra el mismo.

Retornando al main

```
376 int main(){
377     Nodo<Jugador>* partidos[CANTEQUIPOS][CANTPARTIDOS] = { NULL };
378     int menu = 0;
379     cargarArchivo(partidos, "../goles.DAT");
380     while(true){
381         cout<< "GOLES DEL MUNDIAL 2018"<<endl;
382         cout<< "1: Volver a cargar los datos" <<endl;
383         cout<< "2: Ver cantidad de goles por equipo" <<endl;
384         cout<< "3: Ver tabla de goleadores" << endl;
385         cout<< "4: Ver los goleadores de cada fecha" << endl;
386         cout<< "0: Salir" <<endl;
387         cin>> menu;
388         switch(menu){
389             case 1: cargarArchivo(partidos, "goles.DAT"); cout<<"Datos cargados."<<endl; break;
390             case 2: golesPorEquipo(partidos); break;
391             case 3: goleadores(partidos); break;
392             case 4: goleadoresPorFecha(partidos); break;
393             case 0: return 0; break;
394         }
395         system("pause");
396         system("CLS");
397     }
398     return 0;
399 }
```

380 a 387 (y 397): Se ejecuta un menú de opciones, el cual se repite hasta que el usuario lo desee.

395 y 396: Se efectúa una pausa para que el usuario pueda visualizar lo que desee, y luego se limpia la pantalla.

389: En caso de escoger la opción 1, se vuelve a ejecutar la función cargarArchivo.

390: En caso de escoger la opción 2, se pasa a la función golesPorEquipo, que recibe la matriz con los datos cargados.

Dentro de golesPorEquipo

```
274 void golesPorEquipo(Nodo<Jugador>* partidos[][CANTPARTIDOS]){
275     int goles=0;
276     Nodo<Jugador>* jug = NULL;
277     Nodo<Equipo>* equipos = NULL;
278     for(int equipo=0; equipo<CANTEQUIPOS; equipo++){
279         for(int partido=0; partido<CANTPARTIDOS; partido++){
280             jug = partidos[equipo][partido];
281             while(jug!=NULL){
282                 goles += jug->info.goles;
283                 jug=jug->sig;
284             }
285         }
286         agregarNodo<Equipo>(equipos, crearEquipo(equipo, goles));
287         goles=0;
288     }
289     Nodo<Equipo>* aux = equipos;
290     Arbol<Equipo>* raiz = NULL;
291     while(aux!=NULL){
292         insertarEquipo(raiz, aux->info);
293         aux = aux->sig;
294     }
295     system("CLS");
296     cout << left << setw(15) << "PAIS" << setw(5) << "GOLES" <<endl;
297     inOrden(raiz);
298     return;
299 }
```

275 a 277: se inicializan las variables necesarias, incluyendo una de tipo Nodo de estructura Equipo, previamente declarada.

```
76  struct Equipo{
77      int cod_equipo;
78      int goles;
79  };
```

278 a 288: Se recorre la matriz, fila por fila.

280: Se asigna a un auxiliar “jug” el puntero a la lista del casillero de la matriz indicado.

281 a 284: Se recorre la lista y se suman a la variable “goles” la cantidad que está descripta en cada elemento de esta.

286 y 287: Una vez finalizado con un equipo, se cargan los datos en una lista de equipos, y se reinicia el contador de goles a cero. Se utiliza la función crearEquipo.

```
109 Equipo crearEquipo(int cod_equipo, int goles){
110     Equipo e;
111     e.cod_equipo = cod_equipo;
112     e.goles = goles;
113     return e;
114 }
```

Lo siguiente pertenecería al ejercicio 3.

---

289 a 294: Se recorre la lista de equipos, y se va ingresando cada uno dentro de un árbol de equipos (previamente declarada su estructura), utilizando la función insertarEquipo.

296 y 297: Se muestra por pantalla el contenido del árbol, ordenados de mayor a menor, utilizando la función inOrden.

```
251 void insertarEquipo(Arbol<Equipo>*& nodo, Equipo v) {
252     if(nodo==NULL) {
253         nodo = new Arbol<Equipo>();
254         nodo->dato = v;
255         nodo->izq=NULL;
256         nodo->der=NULL;
257     } else {
258         if(v.goles > nodo->dato.goles) {
259             insertarEquipo(nodo->izq,v);
260         } else {
261             insertarEquipo(nodo->der,v);
262         }
263     }
264 }
265
266 void inOrden(Arbol<Equipo>* nodo){
267     if(nodo!=NULL){
268         inOrden(nodo->izq);
269         cout << left << setw(15) << EQUIPOS[nodo->dato.cod_equipo] << setw (5) << nodo->dato.goles << endl;
270         inOrden(nodo->der);
271     }
272 }
```

Ambas funciones son similares a las funciones de árbol dadas por el profesor Agüero.

Volviendo al main

```
376 int main(){
377     Nodo<Jugador>* partidos[CANTEQUIPOS][CANTPARTIDOS] = { NULL };
378     int menu = 0;
379     cargarArchivo(partidos, "../goles.DAT");
380     while(true){
381         cout<< "GOLES DEL MUNDIAL 2018"<<endl;
382         cout<< "1: Volver a cargar los datos" <<endl;
383         cout<< "2: Ver cantidad de goles por equipo" <<endl;
384         cout<< "3: Ver tabla de goleadores" << endl;
385         cout<< "4: Ver los goleadores de cada fecha" << endl;
386         cout<< "0: Salir" <<endl;
387         cin>> menu;
388         switch(menu){
389             case 1: cargarArchivo(partidos, "goles.DAT"); cout<<"Datos cargados."<<endl; break;
390             case 2: golesPorEquipo(partidos); break;
391             case 3: goleadores(partidos); break;
392             case 4: goleadoresPorFecha(partidos); break;
393             case 0: return 0; break;
394         }
395         system("pause");
396         system("CLS");
397     }
398     return 0;
399 }
```

391: En caso de haberse escogido la opción 3, se pasa a la función “goleadores”, que recibe la matriz.

Función goleadores

```
301 void goleadores(Nodo<Jugador>* partidos[][CANTPARTIDOS]){
302     Nodo<Jugador>* jugadores = NULL;
303     Nodo<Jugador>* jug = NULL;
304     Nodo<Jugador>* jugEnc = NULL;
305     int enContra = 0;
306     for(int equipo=0; equipo<CANTEQUIPOS; equipo++){
307         for(int partido=0; partido<CANTPARTIDOS; partido++){
308             jug= partidos[equipo][partido];
309             while(jug!=NULL){
310                 if(!esEnContra(jug->info)){
311                     jugEnc = buscar<Jugador,Jugador>(jugadores, jug->info, criterioJugador);
312                     if(jugEnc==NULL){
313                         agregarNodo<Jugador>(jugadores, jug->info);
314                     } else {
315                         jugEnc->info.goles += jug->info.goles;
316                         ordenar<Jugador>(jugadores, criterioGoles);
317                     }
318                 } else {
319                     enContra++;
320                 }
321                 jug = jug->sig;
322             }
323         }
324     }
325     jug = jugadores;
326     system("CLS");
327     cout<< left << setw(20) << "JUGADOR" << setw(5) << "GOLES" <<endl;
328     while(jug!=NULL){
329         cout<< left << setw(20) << jug->info.nombre_jugador << setw(3) << jug->info.goles <<endl;
330         jug = jug->sig;
331     }
332     cout << left << setw(20) << "GOLES EN CONTRA" << enContra << endl;
333     liberar<Jugador>(jugadores);
334     return;
335 }
```

302 a 305: Se inicializan las variables necesarias.

306 a 324: Se recorre la matriz.

308 a 322: Se recorre la lista dentro del casillero de la matriz

310 a 318: Si el elemento apuntado no es en contra, según la función “enContra”, se busca al jugador dentro de la lista de jugadores que convirtieron goles utilizando “criterioJugador”.

```
213  bool esEnContra(Jugador j){
214      return strncmp(j.nombre_jugador,"e.c.",4)==0? true:false;
215  }
```

214: Se comparan las primeras 4 letras del nombre de jugador, si comienza con la cadena “e.c.”, es en contra, si no, no lo es.

```
127  int criterioJugador(Jugador j1, Jugador j2){
128      return strcmp(j1.nombre_jugador,j2.nombre_jugador)==0?0:-1;
129  }
```

313: Si el jugador no fue encontrado, se agrega a la lista

315 y 316: Si el jugador fue encontrado, se suma a la variable “goles” la cantidad que indica el “jug”. Posteriormente, se ordena la lista según el criterio criterioGoles, bajo el template de la cátedra “ordenar”.

```
193  template <typename T>
194  void ordenar(Nodo<T>*& p, int (*criterio)(T,T)) {
195      Nodo<T>* q = NULL;
196      while(p!=NULL) {
197          T v = eliminarPrimerNodo<T>(p);
198          insertarOrdenado<T>(q,v,criterio);
199      }
200      p = q;
201  }
```

```
131  int criterioGoles(Jugador j1, Jugador j2){
132      return j2.goles-j1.goles;
133  }
```

319: Si el gol es en contra, se incrementa el contador de goles “enContra”.

325 a 332: Se muestran por pantalla los goleadores, recorriendo la lista e incorporando al final la cantidad almacenada de goles en contra.

333: Se libera la lista de jugadores para ahorrar espacio en memoria.

Regresando al main

```
376  int main(){
377      Nodo<Jugador>* partidos[CANTEQUIPOS][CANTPARTIDOS] = { NULL };
378      int menu = 0;
379      cargarArchivo(partidos, "../goles.DAT");
380      while(true){
381          cout<< "GOLES DEL MUNDIAL 2018"<<endl;
382          cout<< "1: Volver a cargar los datos" <<endl;
383          cout<< "2: Ver cantidad de goles por equipo" <<endl;
384          cout<< "3: Ver tabla de goleadores" << endl;
385          cout<< "4: Ver los goleadores de cada fecha" << endl;
386          cout<< "0: Salir" <<endl;
387          cin>> menu;
388          switch(menu){
389              case 1: cargarArchivo(partidos, "goles.DAT"); cout<<"Datos cargados."<<endl; break;
390              case 2: golesPorEquipo(partidos); break;
391              case 3: goleadores(partidos); break;
392              case 4: goleadoresPorFecha(partidos); break;
393              case 0: return 0; break;
394          }
395          system("pause");
396          system("CLS");
397      }
398      return 0;
399  }
```

392: En caso de haberse escogido la opción 4, se pasa a la función “goleadoresPorFecha”, que recibe la matriz.

```
337 void goleadoresPorFecha(Nodo<Jugador>* partidos[][CANTPARTIDOS]){
338     Nodo<Fecha>* fechas = NULL;
339     Nodo<Fecha>* aux = NULL;
340     Nodo<Jugador>* jug = NULL;
341     Nodo<Jugador>* ant = NULL;
342     for(int partido = 0; partido<CANTPARTIDOS; partido++){
343         for(int equipos = 0; equipos<CANTEQUIPOS; equipos++){
344             jug = partidos[equipos][partido];
345             while(jug!=NULL){
346                 aux = fechas;
347                 while(aux!=NULL && aux->info.fecha != jug->info.fecha){
348                     aux = aux->sig;
349                 }
350                 if(aux == NULL){
351                     insertarOrdenado<Fecha>(fechas, crearFecha(jug->info), criterioFecha);
352                 } else {
353                     insertarOrdenado<Jugador>(aux->info.sub,jug->info, criterioGoles);
354                 }
355                 ant = jug;
356                 jug = jug->sig;
357             }
358         }
359     }
360     aux = fechas;
361     system("CLS");
```

```
360     aux = fechas;
361     system("CLS");
362     while(aux!=NULL){
363         cout<< "GOLES DEL DIA " << (aux->info.fecha)%100 << "/" << ((aux->info.fecha)/100)%100 << "/" << (aux->info.fecha)/10000 << ": " << endl;
364         cout<< left << setw(20) << "JUGADOR" << setw(5) << "GOLES" << endl;
365         jug = aux->info.sub;
366         while(jug!=NULL){
367             cout<< left << setw(20) << jug->info.nombre_jugador << setw(5) << jug->info.goles << endl;
368             jug = jug->sig;
369         }
370         aux=aux->sig;
371     }
372     liberar<Fecha>(fechas);
373     return;
374 }
```

342 a 359: Se recorre la matriz, de columna en columna

344 a 357: Se recorre el jugador en el casillero.

346: En caso de que se haya cambiado de fecha, y ninguna coincida con alguna ya almacenada, se crea una nueva. Caso contrario, se agrega el jugador a la lista de jugadores de esa fecha.