Archivo Jugador.cpp

```cpp
#include "Jugador.h"
#ifndef NULL
#define NULL      0
#endif
#include "string"
/*-------------------------------------------------------------------------------*/
//                            IMPLEMENTACION DE PRIMITIVAS
/*-------------------------------------------------------------------------------*/
void crearJugador(Jugador &jugador){
jugador.id=0;
jugador.nombre="-";
jugador.goles=0;
jugador.idEquipo=0;
}
/*-------------------------------------------------------------------------------*/
int getId(Jugador &jugador){
return jugador.id;
}
/*-------------------------------------------------------------------------------*/
string getNombre(Jugador &jugador){
return jugador.nombre;
}
/*-------------------------------------------------------------------------------*/
int getGoles(Jugador &jugador){
return jugador.goles;
}
/*-------------------------------------------------------------------------------*/
void setNombre(Jugador &jugador,string nombre){
jugador.nombre=nombre;
}
/*-------------------------------------------------------------------------------*/
/*-------------------------------------------------------------------------------*/
void setId(Jugador &jugador, int id){
jugador.id=id;
}
/*-------------------------------------------------------------------------------*/
void setGoles(Jugador &jugador, int goles){
jugador.goles=goles;
}
/*-------------------------------------------------------------------------------*/
void setIdEquipo(Jugador &jugador,int id){
jugador.idEquipo=id;
}
/*-------------------------------------------------------------------------------*/
int getIdEquipo(Jugador &jugador){
return jugador.idEquipo;
}
/*-------------------------------------------------------------------------------*/
void destructor(Jugador &jugador){
jugador.id=0;
jugador.nombre="-";
jugador.goles=0;
jugador.idEquipo=0;
}
/*-------------------------------------------------------------------------------*/
/*********************************************************************************/
```

Archivo ListaEquipo.cpp

```cpp
#include "ListaEquipo.h"

#ifndef NULL
#define NULL      0
#endif
/*-------------------------------------------------------------------------*/
//                           IMPLEMENTACION DE PRIMITIVAS
/*-------------------------------------------------------------------------*/
void crearListaEquipo(ListaEquipo &lista){
lista.primero=finEquipo();

}
/*-------------------------------------------------------------------------*/
bool listaVaciaEquipo(ListaEquipo &lista){
return(lista.primero==finEquipo());
}
/*-------------------------------------------------------------------------*/
PtrNodoListaEquipo finEquipo() {
  return NULL;
}
/*-------------------------------------------------------------------------*/
PtrNodoListaEquipo primeroEquipo(ListaEquipo &lista) {
  return lista.primero;
}
/*-------------------------------------------------------------------------*/
PtrNodoListaEquipo siguienteEquipo(ListaEquipo &lista, PtrNodoListaEquipo ptrNodo){

if((!listaVaciaEquipo(lista)) && (ptrNodo->siguiente!=finEquipo())){
    return ptrNodo->siguiente;
}
else
    return finEquipo();

}
/*-------------------------------------------------------------------------*/
PtrNodoListaEquipo anteriorEquipo(ListaEquipo &lista,PtrNodoListaEquipo ptrNodo){
PtrNodoListaEquipo ptrAnterior=finEquipo();
PtrNodoListaEquipo ptrCursor=primeroEquipo(lista);

while(ptrCursor!=finEquipo() && ptrCursor !=ptrNodo){

    ptrAnterior=ptrCursor;
    ptrCursor=siguienteEquipo(lista,ptrCursor);

}
return ptrAnterior;

}
/*-------------------------------------------------------------------------*/
PtrNodoListaEquipo ultimoEquipo(ListaEquipo &lista){
return anteriorEquipo(lista,finEquipo());
}
/*-------------------------------------------------------------------------*/
PtrNodoListaEquipo crearNodoListaEquipo(Equipo equipo){
PtrNodoListaEquipo ptrNodo = new NodoListaEquipo;
ptrNodo->equipo=equipo;
ptrNodo->siguiente=finEquipo();

return ptrNodo;
}
/*-------------------------------------------------------------------------*/
PtrNodoListaEquipo adicionarAlPrincipio(ListaEquipo &lista, Equipo equipo){
PtrNodoListaEquipo ptrNodo = crearNodoListaEquipo(equipo);

ptrNodo->siguiente=lista.primero;
lista.primero=ptrNodo;
```

```
67
68  return ptrNodo;
69  }
70  /*-------------------------------------------------------------------------*/
71  PtrNodoListaEquipo adicionarDespues(ListaEquipo &lista, Equipo equipo, PtrNodoListaEquipo ptrNodo){
72
73  PtrNodoListaEquipo ptrAux = finEquipo();
74  if(listaVaciaEquipo(lista)){
75      adicionarAlPrincipio(lista,equipo);
76      }
77     if(ptrNodo!=finEquipo()){
78
79      ptrAux=crearNodoListaEquipo(equipo);
80
81      ptrAux->siguiente=ptrNodo->siguiente;
82      ptrNodo->siguiente=ptrAux;
83     }
84
85      return ptrAux;
86
87   }
88  /*-------------------------------------------------------------------------*/
89   PtrNodoListaEquipo adicionarFinal(ListaEquipo &lista,Equipo equipo){
90
91   return adicionarDespues(lista,equipo,ultimoEquipo(lista));
92     }
93  /*-------------------------------------------------------------------------*/
94   PtrNodoListaEquipo adicionarAntes(ListaEquipo &lista, Equipo equipo, PtrNodoListaEquipo ptrNodo){
95   PtrNodoListaEquipo ptrNodoNuevo=finEquipo();
96   if(!listaVaciaEquipo(lista)){
97
98       if(ptrNodo!=primeroEquipo(lista))
99           ptrNodoNuevo=adicionarDespues(lista,equipo,anteriorEquipo(lista,ptrNodo));
100       else
101          ptrNodoNuevo=adicionarAlPrincipio(lista,equipo);
102     }
103     return ptrNodoNuevo;
104
105  }
106 /*-------------------------------------------------------------------------*/
107  void colocarDato(ListaEquipo &lista,Equipo equipo, PtrNodoListaEquipo ptrNodo){
108  if(!listaVaciaEquipo(lista) && ptrNodo!=finEquipo()){
109     ptrNodo->equipo=equipo;
110  }
111  }
112 /*-------------------------------------------------------------------------*/
113  void obtenerDato(ListaEquipo &lista, Equipo &equipo, PtrNodoListaEquipo ptrNodo) {
114
115   if ((! listaVaciaEquipo(lista)) && (ptrNodo != finEquipo()))
116     equipo = ptrNodo->equipo;
117 }
118
119 /*-------------------------------------------------------------------------*/
120 void eliminarNodo(ListaEquipo &lista, PtrNodoListaEquipo ptrNodo) {
121
122   PtrNodoListaEquipo ptrPrevio;
123
124   /* verifica que la lista no esté vacia y que nodo no sea fin*/
125   if ((! listaVaciaEquipo(lista)) && (ptrNodo != finEquipo())) {
126
127     if (ptrNodo == primeroEquipo(lista))
128       lista.primero = siguienteEquipo(lista,primeroEquipo(lista));
129
130     else {
131       ptrPrevio = anteriorEquipo( lista , ptrNodo );
132       ptrPrevio->siguiente = ptrNodo->siguiente;
```

```
133     }
134     // Si el dato es un TDA, acá habría que llamar al destructor.
135
136     delete ptrNodo;
137   }
138 }
139
140 /*-------------------------------------------------------------------------*/
141 void eliminarNodoPrimero(ListaEquipo &lista) {
142
143   if (! listaVaciaEquipo(lista))
144     eliminarNodo(lista,primeroEquipo(lista));
145 }
146
147 /*-------------------------------------------------------------------------*/
148 void eliminarNodoUltimo(ListaEquipo &lista) {
149
150   if (! listaVaciaEquipo(lista))
151     eliminarNodo(lista,ultimoEquipo(lista));
152 }
153
154 /*-------------------------------------------------------------------------*/
155 void eliminarLista(ListaEquipo &lista) {
156
157   /* retira uno a uno los nodos de la lista */
158   while (! listaVaciaEquipo(lista))
159     eliminarNodo(lista,primeroEquipo(lista));
160 }
161 /*-------------------------------------------------------------------------*/
162 PtrNodoListaEquipo localizarDato(ListaEquipo &lista, Equipo equipo) {
163
164    bool encontrado = false;
165    Equipo equipoCursor;
166    PtrNodoListaEquipo ptrCursor = primeroEquipo(lista);
167
168   /* recorre los nodos hasta llegar al último o hasta
169      encontrar el nodo buscado */
170   while ((ptrCursor != finEquipo()) && (! encontrado)) {
171
172     /* obtiene el dato del nodo y lo compara */
173     obtenerDato(lista,equipoCursor,ptrCursor);
174     if (compararDatoEquipo(equipoCursor,equipo) == IGUAL_EQUIPO)
175       encontrado = true;
176     else
177       ptrCursor = siguienteEquipo(lista,ptrCursor);
178   }
179
180   /* si no lo encontró devuelve fin */
181   if (! encontrado)
182     ptrCursor = finEquipo();
183
184   return ptrCursor;
185 }
186
187 /*-------------------------------------------------------------------------*/
188 ResultadoComparacionEquipo compararDatoEquipo(Equipo equipo1, Equipo equipo2) {
189     if (equipo1.id > equipo2.id) {
190         return MAYOR_EQUIPO;
191     }
192     else if (equipo1.id < equipo2.id) {
193         return MENOR_EQUIPO;
194     }
195     else {
196         return IGUAL_EQUIPO;
197     }
198 }
```

```
/*-------------------------------------------------------------------------*/
bool estaPrimero(Equipo equipo1, Equipo equipo2) {
    if (equipo1.puntos > equipo2.puntos) {
        return true;
    }
    else if (equipo1.puntos < equipo2.puntos) {
        return false;
    }
    else {
        if(equipo1.golesAFavor> equipo2.golesAFavor){
            return true;
        }
        else return false;

    }
}
/*-------------------------------------------------------------------------*/
void eliminarDato(ListaEquipo &lista, Equipo equipo) {

  /* localiza el dato y luego lo elimina */
  PtrNodoListaEquipo ptrNodo = localizarDato(lista,equipo);
  if (ptrNodo != finEquipo())
    eliminarNodo(lista,ptrNodo);
}
/*-------------------------------------------------------------------------*/
PtrNodoListaEquipo insertarDato(ListaEquipo &lista, Equipo equipo) {

  PtrNodoListaEquipo ptrPrevio = primeroEquipo(lista);
  PtrNodoListaEquipo ptrCursor = primeroEquipo(lista);
  PtrNodoListaEquipo ptrNuevoNodo;
  Equipo equipoCursor;
  bool ubicado = false;

  /* recorre la lista buscando el lugar de la inserción */
  while ((ptrCursor != finEquipo()) && (! ubicado)) {

    obtenerDato(lista,equipoCursor,ptrCursor);
    if (estaPrimero(equipoCursor,equipo) == true)
      ubicado = true;

    else {
      ptrPrevio = ptrCursor;
      ptrCursor = siguienteEquipo(lista,ptrCursor);
    }
  }

  if (ptrCursor == primeroEquipo(lista))
    ptrNuevoNodo = adicionarDespues(lista,equipo,ptrPrevio);
  else
    ptrNuevoNodo = adicionarAlPrincipio(lista,equipo);

  return ptrNuevoNodo;
}
/*-------------------------------------------------------------------------*/
void reordenar(ListaEquipo &lista) {

  ListaEquipo temp = lista;
  PtrNodoListaEquipo ptrCursor = primeroEquipo(temp);
  crearListaEquipo(lista);
  while ( ptrCursor != finEquipo() ) {
        Equipo equipo;
        obtenerDato( temp, equipo, ptrCursor);
        insertarDato( lista, equipo );
        eliminarNodo( temp, ptrCursor );
        ptrCursor = primeroEquipo(temp);
  }
```

```
265    eliminarLista( temp );
266  }
267  /*****************************************************************************/
```

Archivo ListaGrupo.cpp

```cpp
#include "ListaGrupo.h"

#ifndef NULL
#define NULL      0
#endif
/*-------------------------------------------------------------------------------*/
//                            IMPLEMENTACION DE PRIMITIVAS
/*-------------------------------------------------------------------------------*/
void crearListaGrupo(ListaGrupo &lista){
    lista.primero=finGrupo();
}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo finGrupo(){
    return NULL;
}
/*-------------------------------------------------------------------------------*/
bool listaVaciaGrupo(ListaGrupo lista){
    return(lista.primero==finGrupo());
}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo primeroListaGrupo(ListaGrupo &lista){
return lista.primero;
}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo siguienteListaGrupo(ListaGrupo &lista,PtrNodoGrupo ptrNodoGrupo){
    if(!listaVaciaGrupo(lista) && ptrNodoGrupo->siguiente!=finGrupo()){

        return ptrNodoGrupo->siguiente;
        }
        else
            return finGrupo();

}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo anteriorListaGrupo(ListaGrupo &lista,PtrNodoGrupo ptrNodoGrupo){

PtrNodoGrupo ptrCursor = primeroListaGrupo(lista);
PtrNodoGrupo ptrAnterior=finGrupo();
    while(ptrCursor!=finGrupo() && ptrCursor!=ptrNodoGrupo){
        ptrAnterior=ptrCursor;
        ptrCursor=siguienteListaGrupo(lista,ptrCursor);
    }

    return ptrAnterior;

}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo ultimoListaGrupo(ListaGrupo &lista){
return anteriorListaGrupo(lista,finGrupo());

}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo crearNodoGrupo(Grupo grupo){
    PtrNodoGrupo ptrNodo = new NodoGrupo;
    ptrNodo->grupo=grupo;
    ptrNodo->siguiente=finGrupo();

    return ptrNodo;

}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo adicionarAlPrincipio(ListaGrupo &lista,Grupo grupo){
PtrNodoGrupo ptrNodo=crearNodoGrupo(grupo);
ptrNodo->siguiente=lista.primero;
lista.primero=ptrNodo;

```

```
return ptrNodo;
}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo adicionarDespues(ListaGrupo &lista, Grupo grupo,PtrNodoGrupo ptrNodo){
PtrNodoGrupo ptrAux=finGrupo();
if(listaVaciaGrupo(lista)){
    adicionarAlPrincipio(lista,grupo);
}
if(ptrNodo!=finGrupo()){
    PtrNodoGrupo ptrAux=crearNodoGrupo(grupo);
    ptrAux->siguiente=ptrNodo->siguiente;
    ptrNodo->siguiente=ptrAux;
}
return ptrAux;
}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo adicionarFinal(ListaGrupo &lista,Grupo grupo){
return adicionarDespues(lista,grupo,ultimoListaGrupo(lista));

}
/*-------------------------------------------------------------------------------*/
PtrNodoGrupo adicionarAntes(ListaGrupo &lista,Grupo grupo,PtrNodoGrupo ptrNodoGrupo){
PtrNodoGrupo ptrNodoNuevo=finGrupo();
 if(!listaVaciaGrupo(lista)){

     if(ptrNodoNuevo!=primeroListaGrupo(lista))
         ptrNodoNuevo=adicionarDespues(lista,grupo,anteriorListaGrupo(lista,ptrNodoNuevo));
      else
         ptrNodoNuevo=adicionarAlPrincipio(lista,grupo);
    }
    return ptrNodoNuevo;

}
/*-------------------------------------------------------------------------------*/
 void colocarDato(ListaGrupo &lista,Grupo grupo, PtrNodoGrupo ptrNodoGrupo){
 if(!listaVaciaGrupo(lista) && ptrNodoGrupo!=finGrupo()){
    ptrNodoGrupo->grupo=grupo;
 }
 }
/*-------------------------------------------------------------------------------*/
 void obtenerDato(ListaGrupo &lista, Grupo &grupo, PtrNodoGrupo ptrNodo) {

  if ((! listaVaciaGrupo(lista)) && (ptrNodo != finGrupo()))
    grupo = ptrNodo->grupo;
}

/*-------------------------------------------------------------------------------*/
void eliminarNodo(ListaGrupo &lista, PtrNodoGrupo ptrNodo) {

  PtrNodoGrupo ptrPrevio;

  /* verifica que la lista no esté vacia y que nodo no sea fin*/
  if ((! listaVaciaGrupo(lista)) && (ptrNodo != finGrupo())) {

    if (ptrNodo == primeroListaGrupo(lista))
      lista.primero = siguienteListaGrupo(lista,primeroListaGrupo(lista));

    else {
      ptrPrevio = anteriorListaGrupo( lista , ptrNodo );
      ptrPrevio->siguiente = ptrNodo->siguiente;
    }
    // Si el dato es un TDA, acá habría que llamar al destructor.

    delete ptrNodo;
  }
}
```

```
133
134 /*--------------------------------------------------------------------------*/
135 void eliminarNodoPrimero(ListaGrupo &lista) {
136
137   if (! listaVaciaGrupo(lista))
138     eliminarNodo(lista,primeroListaGrupo(lista));
139 }
140
141 /*--------------------------------------------------------------------------*/
142 void eliminarNodoUltimo(ListaGrupo &lista) {
143
144   if (! listaVaciaGrupo(lista))
145     eliminarNodo(lista,ultimoListaGrupo(lista));
146 }
147
148 /*--------------------------------------------------------------------------*/
149 void eliminarLista(ListaGrupo &lista) {
150
151   /* retira uno a uno los nodos de la lista */
152   while (! listaVaciaGrupo(lista))
153     eliminarNodo(lista,primeroListaGrupo(lista));
154 }
155
156 ResultadoComparacionGrupo compararDatoGrupo(Grupo grupo1, Grupo grupo2) {
157     if (grupo1.id > grupo2.id) {
158         return MAYOR_GRUPO;
159     }
160     else if (grupo1.id < grupo2.id) {
161         return MENOR_GRUPO;
162     }
163     else {
164         return IGUAL_GRUPO;
165     }
166 }
167 /*--------------------------------------------------------------------------*/
168 PtrNodoGrupo localizarDato(ListaGrupo &lista, Grupo grupo) {
169
170    bool encontrado = false;
171    Grupo grupoCursor;
172    PtrNodoGrupo ptrCursor = primeroListaGrupo(lista);
173
174   /* recorre los nodos hasta llegar al último o hasta
175      encontrar el nodo buscado */
176   while ((ptrCursor != finGrupo()) && (! encontrado)) {
177
178     /* obtiene el dato del nodo y lo compara */
179     obtenerDato(lista,grupoCursor,ptrCursor);
180     if (compararDatoGrupo(grupoCursor,grupo) == IGUAL_GRUPO)
181       encontrado = true;
182     else
183       ptrCursor = siguienteListaGrupo(lista,ptrCursor);
184   }
185
186   /* si no lo encontró devuelve fin */
187   if (! encontrado)
188     ptrCursor = finGrupo();
189
190   return ptrCursor;
191 }
192 /*--------------------------------------------------------------------------*/
193 void eliminarDato(ListaGrupo &lista, Grupo grupo) {
194
195   /* localiza el dato y luego lo elimina */
196   PtrNodoGrupo ptrNodo = localizarDato(lista,grupo);
197   if (ptrNodo != finGrupo())
198     eliminarNodo(lista,ptrNodo);
```

```
199  }
200  /*---------------------------------------------------------------------------*/
201  /*****************************************************************************/
```

Archivo ListaJugadores.cpp

```cpp
#include "ListaJugadores.h"

#ifndef NULL
#define NULL      0
#endif
/*-------------------------------------------------------------------------*/
//                        IMPLEMENTACION DE PRIMITIVAS
/*-------------------------------------------------------------------------*/
void crearListaJugador(ListaJugador &lista) {
  lista.primero = finJugador();
}
/*-------------------------------------------------------------------------*/
bool listaVaciaJugador(ListaJugador &lista) {

  return (primeroJugador(lista) == finJugador());
}
/*-------------------------------------------------------------------------*/
PtrNodoListaJugador finJugador() {
  return NULL;
}
/*-------------------------------------------------------------------------*/
PtrNodoListaJugador primeroJugador(ListaJugador &lista) {
  return lista.primero;
}
/*-------------------------------------------------------------------------*/
PtrNodoListaJugador siguienteJugador(ListaJugador &lista, PtrNodoListaJugador ptrNodo) {

  /* verifica si la lista está vacia o si ptrNodo es el último */
  if ((! listaVaciaJugador(lista)) && (ptrNodo != finJugador()))
    return ptrNodo->sgte;
  else
    return finJugador();
}
/*-------------------------------------------------------------------------*/
PtrNodoListaJugador anteriorJugador(ListaJugador &lista, PtrNodoListaJugador ptrNodo) {

  PtrNodoListaJugador ptrPrevio = finJugador();
  PtrNodoListaJugador ptrCursor = primeroJugador(lista);

  while (( ptrCursor != finJugador()) && (ptrCursor != ptrNodo)) {
    ptrPrevio = ptrCursor;
    ptrCursor = siguienteJugador(lista,ptrCursor);
  }
  return ptrPrevio;
}
/*-------------------------------------------------------------------------*/
PtrNodoListaJugador ultimoJugador(ListaJugador &lista) {

  /* el último nodo de la lista es el anterior al fin() */
  return anteriorJugador(lista,finJugador());
}
/*-------------------------------------------------------------------------*/
PtrNodoListaJugador crearNodoListaJugador(Jugador jugador) {

  /* reserva memoria para el nodo y luego completa sus datos */
  PtrNodoListaJugador ptrAux = new NodoListaJugador;

  ptrAux->jugador = jugador;
  ptrAux->sgte = finJugador();

  return ptrAux;
}

/*-------------------------------------------------------------------------*/
PtrNodoListaJugador adicionarPrincipio(ListaJugador &lista, Jugador jugador) {

```

```
67    /* crea el nodo */
68    PtrNodoListaJugador ptrNuevoNodo = crearNodoListaJugador(jugador);
69
70    /* lo incorpora al principio de la lista */
71    ptrNuevoNodo->sgte = lista.primero;
72    lista.primero = ptrNuevoNodo;
73
74    return ptrNuevoNodo;
75  }
76  /*-------------------------------------------------------------------------*/
77  PtrNodoListaJugador adicionarDespues(ListaJugador &lista, Jugador jugador, PtrNodoListaJugador ptrNodo) {
78
79    PtrNodoListaJugador ptrNuevoNodo = finJugador();
80
81    /* si la lista está vacia se adiciona la principio */
82    if (listaVaciaJugador(lista))
83      ptrNuevoNodo = adicionarPrincipio(lista,jugador);
84
85    else {
86      if (ptrNodo != finJugador()) {
87
88        /* crea el nodo y lo intercala en la lista */
89        ptrNuevoNodo = crearNodoListaJugador(jugador);
90
91        ptrNuevoNodo->sgte = ptrNodo->sgte;
92        ptrNodo->sgte = ptrNuevoNodo;
93      }
94    }
95    return ptrNuevoNodo;
96  }
97  /*-------------------------------------------------------------------------*/
98  PtrNodoListaJugador adicionarFinal(ListaJugador &lista, Jugador jugador) {
99
100   /* adiciona el dato después del último nodo de la lista */
101   return adicionarDespues(lista,jugador,ultimoJugador(lista));
102 }
103 /*-------------------------------------------------------------------------*/
104 PtrNodoListaJugador adicionarAntes(ListaJugador &lista, Jugador jugador, PtrNodoListaJugador ptrNodo) {
105
106   PtrNodoListaJugador ptrNuevoNodo = finJugador();
107
108   if (! listaVaciaJugador(lista)) {
109     if (ptrNodo != primeroJugador(lista))
110       ptrNuevoNodo = adicionarDespues(lista,jugador,anteriorJugador(lista,ptrNodo));
111     else
112       ptrNuevoNodo = adicionarPrincipio(lista,jugador);
113    }
114    return ptrNuevoNodo;
115 }
116 /*-------------------------------------------------------------------------*/
117 void colocarDato(ListaJugador &lista, Jugador &jugador, PtrNodoListaJugador ptrNodo) {
118
119   if ( (! listaVaciaJugador(lista)) && (ptrNodo != finJugador()))
120     ptrNodo->jugador = jugador;
121 }
122 /*-------------------------------------------------------------------------*/
123 void obtenerDato(ListaJugador &lista, Jugador &jugador, PtrNodoListaJugador ptrNodo) {
124
125   if ((! listaVaciaJugador(lista)) && (ptrNodo != finJugador()))
126     jugador = ptrNodo->jugador;
127 }
128 /*-------------------------------------------------------------------------*/
129 void eliminarNodo(ListaJugador &lista, PtrNodoListaJugador ptrNodo) {
130
131   PtrNodoListaJugador ptrPrevio;
132
```

```
133   /* verifica que la lista no esté vacia y que nodo no sea fin*/
134   if ((! listaVaciaJugador(lista)) && (ptrNodo != finJugador())) {
135
136     if (ptrNodo == primeroJugador(lista))
137       lista.primero = siguienteJugador(lista,primeroJugador(lista));
138
139     else {
140       ptrPrevio = anteriorJugador( lista , ptrNodo );
141       ptrPrevio->sgte = ptrNodo->sgte;
142     }
143     // Si el dato es un TDA, acá habría que llamar al destructor.
144
145     delete ptrNodo;
146   }
147 }
148 /*-------------------------------------------------------------------------*/
149 void eliminarNodoPrimero(ListaJugador &lista) {
150
151   if (! listaVaciaJugador(lista))
152     eliminarNodo(lista,primeroJugador(lista));
153 }
154 /*-------------------------------------------------------------------------*/
155 void eliminarNodoUltimo(ListaJugador &lista) {
156
157   if (! listaVaciaJugador(lista))
158     eliminarNodo(lista,ultimoJugador(lista));
159 }
160 /*-------------------------------------------------------------------------*/
161 void eliminarLista(ListaJugador &lista) {
162
163   /* retira uno a uno los nodos de la lista */
164   while (! listaVaciaJugador(lista))
165     eliminarNodo(lista,primeroJugador(lista));
166 }
167 /*-------------------------------------------------------------------------*/
168 ResultadoComparacionJugador compararDatoJugador(Jugador jugador1, Jugador jugador2) {
169     if (jugador1.id > jugador2.id) {
170         return MAYOR_JUGADOR;
171     }
172     else if (jugador1.id < jugador2.id) {
173         return MENOR_JUGADOR;
174     }
175     else {
176         return IGUAL_JUGADOR;
177     }
178 }
179 /*-------------------------------------------------------------------------*/
180 PtrNodoListaJugador localizarDato(ListaJugador &lista, Jugador jugador) {
181
182    bool encontrado = false;
183    Jugador jugadorCursor;
184    PtrNodoListaJugador ptrCursor = primeroJugador(lista);
185
186   /* recorre los nodos hasta llegar al último o hasta
187      encontrar el nodo buscado */
188   while ((ptrCursor != finJugador()) && (! encontrado)) {
189
190     /* obtiene el dato del nodo y lo compara */
191     obtenerDato(lista,jugadorCursor,ptrCursor);
192     if (compararDatoJugador(jugadorCursor,jugador) == IGUAL_JUGADOR)
193       encontrado = true;
194     else
195       ptrCursor = siguienteJugador(lista,ptrCursor);
196   }
197
198   /* si no lo encontró devuelve fin */
```

```
199   if (! encontrado)
200     ptrCursor = finJugador();
201
202   return ptrCursor;
203 }
204 /*-------------------------------------------------------------------------*/
205 void eliminarDato(ListaJugador &lista, Jugador jugador) {
206
207   /* localiza el dato y luego lo elimina */
208   PtrNodoListaJugador ptrNodo = localizarDato(lista,jugador);
209   if (ptrNodo != finJugador())
210     eliminarNodo(lista,ptrNodo);
211 }
212 /*-------------------------------------------------------------------------*/
213 /*************************************************************************/
```

Archivo ListaPartidos.cpp

```cpp
#include <iostream>
#include <stdlib.h>
#include <string>
#include <sstream>
#include <fstream>
#include "ListaPartidos.h"
#ifndef NULL
#define NULL      0
#endif

using namespace std;

/*-------------------------------------------------------------------------*/
//                        IMPLEMENTACION DE PRIMITIVAS
/*-------------------------------------------------------------------------*/
void crearListaPartido(ListaPartido &lista) {
  lista.primero = finListaPartido();
}
/*-------------------------------------------------------------------------*/
bool listaVaciaPartido(ListaPartido &lista) {

  return (primeroPartido(lista) == finListaPartido());
}
/*-------------------------------------------------------------------------*/
PtrNodoPartido finListaPartido() {
  return NULL;
}
/*-------------------------------------------------------------------------*/
PtrNodoPartido primeroPartido(ListaPartido &lista) {
  return lista.primero;
}
/*-------------------------------------------------------------------------*/
PtrNodoPartido siguientePartido(ListaPartido &lista, PtrNodoPartido ptrNodo) {

  /* verifica si la lista está vacia o si ptrNodo es el último */
  if ((! listaVaciaPartido(lista)) && (ptrNodo != finListaPartido()))
    return ptrNodo->siguiente;
  else
    return finListaPartido();
}
/*-------------------------------------------------------------------------*/
PtrNodoPartido anteriorPartido(ListaPartido &lista, PtrNodoPartido ptrNodo) {

  PtrNodoPartido ptrPrevio = finListaPartido();
  PtrNodoPartido ptrCursor = primeroPartido(lista);

  while (( ptrCursor != finListaPartido()) && (ptrCursor != ptrNodo)) {
    ptrPrevio = ptrCursor;
    ptrCursor = siguientePartido(lista,ptrCursor);
  }
  return ptrPrevio;
}
/*-------------------------------------------------------------------------*/
PtrNodoPartido ultimoPartido(ListaPartido &lista) {

  /* el último nodo de la lista es el anterior al fin() */
  return anteriorPartido(lista,finListaPartido());
}
/*-------------------------------------------------------------------------*/
PtrNodoPartido crearNodoListaPartido(Partido partido) {

  /* reserva memoria para el nodo y luego completa sus datos */
  PtrNodoPartido ptrAux = new NodoListaPartido;

  ptrAux->partido = partido;
  ptrAux->siguiente = finListaPartido();
```

```
67
68    return ptrAux;
69  }
70  /*--------------------------------------------------------------------------*/
71  PtrNodoPartido adicionarPrincipio(ListaPartido &lista, Partido partido) {
72
73    /* crea el nodo */
74    PtrNodoPartido ptrNuevoNodo = crearNodoListaPartido(partido);
75
76    /* lo incorpora al principio de la lista */
77    ptrNuevoNodo->siguiente = lista.primero;
78    lista.primero = ptrNuevoNodo;
79
80    return ptrNuevoNodo;
81  }
82  /*--------------------------------------------------------------------------*/
83  PtrNodoPartido adicionarDespues(ListaPartido &lista, Partido partido, PtrNodoPartido ptrNodo) {
84
85    PtrNodoPartido ptrNuevoNodo = finListaPartido();
86
87    /* si la lista está vacia se adiciona la principio */
88    if (listaVaciaPartido(lista))
89      ptrNuevoNodo = adicionarPrincipio(lista,partido);
90
91    else {
92      if (ptrNodo != finListaPartido()) {
93
94        /* crea el nodo y lo intercala en la lista */
95        ptrNuevoNodo = crearNodoListaPartido(partido);
96
97        ptrNuevoNodo->siguiente = ptrNodo->siguiente;
98        ptrNodo->siguiente = ptrNuevoNodo;
99      }
100   }
101   return ptrNuevoNodo;
102 }
103 /*--------------------------------------------------------------------------*/
104 PtrNodoPartido adicionarFinal(ListaPartido &lista, Partido partido) {
105
106   /* adiciona el dato después del último nodo de la lista */
107   return adicionarDespues(lista,partido,ultimoPartido(lista));
108 }
109 /*--------------------------------------------------------------------------*/
110 PtrNodoPartido adicionarAntes(ListaPartido &lista, Partido partido, PtrNodoPartido ptrNodo) {
111
112   PtrNodoPartido ptrNuevoNodo = finListaPartido();
113
114   if (! listaVaciaPartido(lista)) {
115     if (ptrNodo != primeroPartido(lista))
116       ptrNuevoNodo = adicionarDespues(lista,partido,anteriorPartido(lista,ptrNodo));
117     else
118       ptrNuevoNodo = adicionarPrincipio(lista,partido);
119    }
120    return ptrNuevoNodo;
121 }
122 /*--------------------------------------------------------------------------*/
123 void colocarDato(ListaPartido &lista, Partido &partido, PtrNodoPartido ptrNodo) {
124
125   if ( (! listaVaciaPartido(lista)) && (ptrNodo != finListaPartido()))
126     ptrNodo->partido = partido;
127 }
128 /*--------------------------------------------------------------------------*/
129 void obtenerDato(ListaPartido &lista, Partido &partido, PtrNodoPartido ptrNodo) {
130
131   if ((! listaVaciaPartido(lista)) && (ptrNodo != finListaPartido()))
132     partido = ptrNodo->partido;
```

```
}
/*-------------------------------------------------------------------------*/
void eliminarNodo(ListaPartido &lista, PtrNodoPartido ptrNodo) {

  PtrNodoPartido ptrPrevio;

  /* verifica que la lista no esté vacia y que nodo no sea fin*/
  if ((! listaVaciaPartido(lista)) && (ptrNodo != finListaPartido())) {

    if (ptrNodo == primeroPartido(lista))
      lista.primero = siguientePartido(lista,primeroPartido(lista));

    else {
      ptrPrevio = anteriorPartido( lista , ptrNodo );
      ptrPrevio->siguiente = ptrNodo->siguiente;
    }
    // Si el dato es un TDA, acá habría que llamar al destructor.

    delete ptrNodo;
  }
}
/*-------------------------------------------------------------------------*/
void eliminarNodoPrimero(ListaPartido &lista) {

  if (! listaVaciaPartido(lista))
    eliminarNodo(lista,primeroPartido(lista));
}
/*-------------------------------------------------------------------------*/
void eliminarNodoUltimo(ListaPartido &lista) {

  if (! listaVaciaPartido(lista))
    eliminarNodo(lista,ultimoPartido(lista));
}
/*-------------------------------------------------------------------------*/
void eliminarLista(ListaPartido &lista) {

  /* retira uno a uno los nodos de la lista */
  while (! listaVaciaPartido(lista))
    eliminarNodo(lista,primeroPartido(lista));
}
/*-------------------------------------------------------------------------*/
ResultadoComparacionPartido compararDatoPartido(Partido partido1, Partido partido2) {

    if (partido1.id > partido2.id) {
        return MAYOR_PARTIDO;
    }
    else if (partido1.id < partido2.id) {
        return MENOR_PARTIDO;
    }
    else {
        return IGUAL_PARTIDO;
    }
}
/*-------------------------------------------------------------------------*/
PtrNodoPartido localizarDato(ListaPartido &lista, Partido partido) {

   bool encontrado = false;
   Partido partidoCursor;
   PtrNodoPartido ptrCursor = primeroPartido(lista);

  /* recorre los nodos hasta llegar al último o hasta
     encontrar el nodo buscado */
  while ((ptrCursor != finListaPartido()) && (false== encontrado)) {

    /* obtiene el dato del nodo y lo compara */
    obtenerDato(lista,partidoCursor,ptrCursor);
```

```
199       if (compararDatoPartido(partidoCursor,partido) == IGUAL_PARTIDO)
200         encontrado = true;
201       else
202         ptrCursor = siguientePartido(lista,ptrCursor);
203     }
204
205     /* si no lo encontró devuelve fin */
206     if (! encontrado)
207       ptrCursor = finListaPartido();
208
209     return ptrCursor;
210   }
211   /*-------------------------------------------------------------------------*/
212   void eliminarDato(ListaPartido &lista, Partido partido) {
213
214     /* localiza el dato y luego lo elimina */
215     PtrNodoPartido ptrNodo = localizarDato(lista,partido);
216     if (ptrNodo != finListaPartido())
217       eliminarNodo(lista,ptrNodo);
218   }
219   /*-------------------------------------------------------------------------*/
220   /*************************************************************************/
```

Archivo Main.cpp

```cpp
#include <iostream>
#include "Equipo.h"
#include "ListaEquipo.h"
#include "ListaGrupo.h"
#include "ListaJugadores.h"
#include "ListaPartidos.h"
#include <stdlib.h>
#include <string>
#include <sstream>
#include <fstream>
using namespace std;
void bateriaJugadores();
void administrarPartidos();
Equipo validarEquipoEncontrado(ListaEquipo listaEquipo, Equipo equipo);
Partido validarPartidoEncontrado (ListaPartido listaPartido, Partido partido);
ListaEquipo cargarEquipos();
ListaGrupo cargarGrupos();
ListaPartido cargarPartidos();
void cargarJugadores(Equipo &equipo);
void imprimirListaJugadores(ListaJugador  &lista);
void imprimirListaPartidos(ListaPartido  &lista);
PtrNodoListaJugador traerJugador(ListaEquipo lista,int idEquipo,int idJugador);
PtrNodoListaJugador traerJugador(ListaJugador listaJugador,int idJugador);
PtrNodoListaEquipo traerEquipo(ListaEquipo listaEquipo,int idEquipo);
PtrNodoPartido traerNodoPartido(ListaPartido listaPartido,int idPartido);
PtrNodoGrupo traerGrupo(ListaGrupo listaGrupo,char idGrupo);
void guardarDatos(ListaEquipo listaEquipo,ListaGrupo listaGrupo,ListaPartido listaPartido);
void calcularOctavos(ListaEquipo listaEquipo,ListaGrupo listaGrupo,ListaPartido &listaPartido);
PtrNodoListaEquipo primero(Grupo grupo,ListaEquipo listaEquipo);
PtrNodoListaEquipo segundo(Grupo grupo,ListaEquipo listaEquipo);
void calcularCuartos(ListaPartido listaPartido);
int traerIdGanador(PtrNodoPartido ptrNodoPartido);
void calcularSemi(ListaPartido listaPartido);
void calcularFinal(ListaPartido listaPartido);
int main()
{
    int menu=0;
    int submenu2=0,submenu3=0;
    ListaEquipo listaEquipo=cargarEquipos();
    ListaGrupo listaGrupo=cargarGrupos();
    ListaPartido listaPartido;
    crearListaPartido(listaPartido);
    listaPartido=cargarPartidos();
    while(menu!=3){
     cout<<"1_Administrar partidos"<<endl;
     cout<<"2_Procesar reportes"<<endl;
     cout<<"3_Guardar y salir"<<endl;

     cin>>menu;

     switch(menu){

     case 1: //administrar partidos
            submenu2=0;
            while(submenu2!=4){
                 cout<<"1-Registrar inicio de partidos"<<endl;
                 cout<<"2_Registrar goles ocurridos en cada partido"<<endl;
                 cout<<"3_Registrar fin de un partido"<<endl;
                 cout<<"4_Atras"<<endl;
                 cin>>submenu2;

                 switch(submenu2){

             case 1:{
                    int auxId=0;
                    cout<<"Ingrese id del partido a iniciar"<<endl;
```

```
                cin>>auxId;
                Partido partidoAux ;
                partidoAux.id= auxId;


                if( localizarDato(listaPartido,partidoAux)==finListaPartido())
                  partidoAux=validarPartidoEncontrado(listaPartido,partidoAux);

                PtrNodoPartido ptrCursorAux=localizarDato(listaPartido,partidoAux);
                cout<<"Comenzo el partido con id: "<<ptrCursorAux->partido.id<<endl;
                if(auxId<49){
                ptrCursorAux->partido.golesL =  0;
                ptrCursorAux->partido.golesV = 0;
                ptrCursorAux->partido.idEquipoL = 0;
                ptrCursorAux->partido.idEquipoV = 0;}
                else{
                    ptrCursorAux->partido.golesL =  0;
                    ptrCursorAux->partido.golesV = 0;
                }


                break;
            }

            case 2:{
               int auxId=0,goles=0;
               int idE=0,idV=0,gol=0,golV=0,idJugador=0;
               Equipo equipoAuxiliar;
               Partido partidoAux;


               cout<<"ingrese id del partido a cargar los goles"<<endl;
               cin>>auxId;
               partidoAux.id=auxId;

               if( localizarDato(listaPartido,partidoAux)==finListaPartido())
                   partidoAux=validarPartidoEncontrado(listaPartido,partidoAux);

               PtrNodoPartido ptrCursor=localizarDato(listaPartido,partidoAux);

               cout<<"Id partido"<<ptrCursor->partido.id<<endl;
               cout<< "Ingrese Id del equipo local:"<<endl;
               cin>>idE;
               equipoAuxiliar.id = idE;

               if(localizarDato(listaEquipo,equipoAuxiliar)== finEquipo()){
                   equipoAuxiliar=validarEquipoEncontrado(listaEquipo,equipoAuxiliar);
               }


               setIdEquipoL(ptrCursor->partido,equipoAuxiliar.id);

                cout<< "Ingrese Id del equipo visitante:"<<endl;
                cin>>idV;
                equipoAuxiliar.id = idV;

                  if(localizarDato(listaEquipo,equipoAuxiliar)== finEquipo()){
                    equipoAuxiliar=validarEquipoEncontrado(listaEquipo,equipoAuxiliar);
               }

                setIdEquipoV(ptrCursor->partido,equipoAuxiliar.id);

                cout<<"Ingrese goles del equipo local:"<<endl;
                cin>>gol;
                goles=goles+gol;
                setGolesL(ptrCursor->partido,gol);
```

```
133
134                  while(gol>0){
135                     cout<<"Ingrese id del jugador del gol nro:"<<gol<<endl;
136                     cin>>idJugador;
137                     PtrNodoListaJugador ptrNodo = traerJugador(listaEquipo,idE,idJugador);
138                     ptrNodo->jugador.goles=ptrNodo->jugador.goles+1;
139                     gol=gol-1;
140                     }
141                  //
142                  idJugador=0;
143                  cout<<"Ingrese goles del equipo visitante:"<<endl;
144                  cin>>golV;
145                  goles=goles+golV;
146                  setGolesV(ptrCursor->partido,golV);
147                  while(golV>0){
148                     cout<<"Ingrese id del jugador del gol nro:"<<gol<<endl;
149                     cin>>idJugador;
150                     PtrNodoListaJugador ptrNodo =traerJugador(listaEquipo,idV,idJugador);
151                     ptrNodo->jugador.goles=ptrNodo->jugador.goles+1;
152                     golV=golV-1;
153                  }
154                  if(goles>10){
155                     cout<<"*****Convirtieron mas de 10 goles******"<<endl;
156
157                  }
158                  if((goles*100)%100){
159                     cout<<"MAS DE 100 GOLES"<<endl;
160                  }
161
162                   break;
163               }
164               case 3:{
165                       //Registrar fin de un partido
166                       int id;
167                       cout<<"Ingrese id del partido a finalizar"<<endl;
168                       cin>>id;
169                       cout<<"Id del partido:"<<id;
170                       PtrNodoPartido ptrNodoPartido = traerNodoPartido(listaPartido,id);
171                       PtrNodoListaEquipo ptrNodoEquipoL = traerEquipo(listaEquipo,ptrNodoPartido->partido.
idEquipoL);
172                       PtrNodoListaEquipo ptrNodoEquipoV = traerEquipo(listaEquipo,ptrNodoPartido->partido.
idEquipoV);
173
174                       setGolesAFavor(ptrNodoEquipoL->equipo,ptrNodoEquipoL->equipo.golesAFavor+
ptrNodoPartido->partido.golesL);
175                       setGolesEnContra(ptrNodoEquipoL->equipo,ptrNodoEquipoL->equipo.golesEnContra+
ptrNodoPartido->partido.golesV);
176
177                       setGolesAFavor(ptrNodoEquipoV->equipo,ptrNodoEquipoV->equipo.golesAFavor+
ptrNodoPartido->partido.golesV);
178                       setGolesEnContra(ptrNodoEquipoV->equipo,ptrNodoEquipoV->equipo.golesEnContra+
ptrNodoPartido->partido.golesL);
179
180                       if(ptrNodoPartido->partido.golesL > ptrNodoPartido->partido.golesV ){
181                          setPuntos(ptrNodoEquipoL->equipo,ptrNodoEquipoL->equipo.puntos+3);
182                       }
183                       else if(ptrNodoPartido->partido.golesL < ptrNodoPartido->partido.golesV){
184                          setPuntos(ptrNodoEquipoV->equipo,ptrNodoEquipoV->equipo.puntos+3);
185                       }
186                       else{
187                          setPuntos(ptrNodoEquipoL->equipo,ptrNodoEquipoL->equipo.puntos+1);
188                          setPuntos(ptrNodoEquipoV->equipo,ptrNodoEquipoV->equipo.puntos+1);
189                       }
190                       cout<<"Datos guardados"<<endl;
191                       calcularOctavos(listaEquipo,listaGrupo,listaPartido);
192                       calcularCuartos(listaPartido);
```

```
193                     calcularSemi(listaPartido);
194                     calcularFinal(listaPartido);
195
196
197
198
199
200
201                 }
202                case 4:break;
203                 }
204                 }
205                  break;
206     case 2: //reportes
207              submenu3=0;
208             while(submenu3!=3){
209                  cout<<"1_Orden de equipos por Grupo"<<endl;
210                  cout<<"2_Grupo de la muerte"<<endl;
211                  cout<<"3_Atras"<<endl;
212                  cin>>submenu3;
213
214                  switch(submenu3){
215
216              case 1:{
217                     int cantidadGrupos=0;
218                     int golesParciales=0;
219                     int golesTotales=0;
220
221                     PtrNodoGrupo cursor = primeroListaGrupo(listaGrupo);
222                     Grupo grupoAux;
223
224                     while (cursor != finGrupo()) {
225                     obtenerDato(listaGrupo, grupoAux, cursor);
226                     cout << grupoAux.nombre<<endl;
227                     cout <<traerEquipo(listaEquipo,grupoAux.idEquipo1)->equipo.nombre<<traerEquipo(
listaEquipo,grupoAux.idEquipo1)->equipo.golesAFavor<<traerEquipo(listaEquipo,grupoAux.idEquipo1)->equipo.puntos
<<endl;
228                     cout <<traerEquipo(listaEquipo,grupoAux.idEquipo2)->equipo.nombre<<traerEquipo(
listaEquipo,grupoAux.idEquipo2)->equipo.golesAFavor<<traerEquipo(listaEquipo,grupoAux.idEquipo2)->equipo.puntos
<<endl;
229                     cout <<traerEquipo(listaEquipo,grupoAux.idEquipo3)->equipo.nombre<<traerEquipo(
listaEquipo,grupoAux.idEquipo3)->equipo.golesAFavor<<traerEquipo(listaEquipo,grupoAux.idEquipo3)->equipo.puntos
<<endl;
230                     cout <<traerEquipo(listaEquipo,grupoAux.idEquipo4)->equipo.nombre<<traerEquipo(
listaEquipo,grupoAux.idEquipo4)->equipo.golesAFavor<<traerEquipo(listaEquipo,grupoAux.idEquipo4)->equipo.puntos
<<endl;
231                     cout << endl;
232                     cantidadGrupos=cantidadGrupos+1;
233                     golesParciales=traerEquipo(listaEquipo,grupoAux.idEquipo1)->equipo.golesAFavor+
traerEquipo(listaEquipo,grupoAux.idEquipo2)->equipo.golesAFavor+traerEquipo(listaEquipo,grupoAux.idEquipo3)->
equipo.golesAFavor+traerEquipo(listaEquipo,grupoAux.idEquipo4)->equipo.golesAFavor;
234                     cout<< "Cantidad Goles Parciales:"<<golesParciales<<endl;
235                     golesTotales=golesTotales+golesParciales;
236                     cursor = siguienteListaGrupo(listaGrupo, cursor);
237                      if(cantidadGrupos==8){
238                         cursor=finGrupo();
239                     }
240                     }
241                     cout<< "Cantidad GRUPOS:"<<cantidadGrupos<<endl;
242                     cout<< "Cantidad GOLES TOTALES:"<<golesTotales<<endl;
243                     cout << endl;
244
245              }break;
246
247
248                     //grupo de la muerte
```

```
249             case 2:{
250 
251                 int cantidadGrupos=0;
252                    int golesParciales=0;
253                    int golesParcialesMuerte=9999;
254                    string grupoMuerteNombre;
255 
256                    PtrNodoGrupo cursor = primeroListaGrupo(listaGrupo);
257                    Grupo grupoAux;
258                    while (cursor != finGrupo()) {
259                    obtenerDato(listaGrupo, grupoAux, cursor);
260 
261                    cantidadGrupos=cantidadGrupos+1;
262                    golesParciales=traerEquipo(listaEquipo,grupoAux.idEquipo1)->equipo.golesAFavor+
traerEquipo(listaEquipo,grupoAux.idEquipo2)->equipo.golesAFavor+traerEquipo(listaEquipo,grupoAux.idEquipo3)->
equipo.golesAFavor+traerEquipo(listaEquipo,grupoAux.idEquipo4)->equipo.golesAFavor;
263 
264                    if(golesParcialesMuerte==9999){
265                        golesParcialesMuerte=golesParciales;
266                        grupoMuerteNombre=grupoAux.nombre;
267                    }
268                    if(golesParcialesMuerte>=golesParciales){
269                        golesParcialesMuerte=golesParcialesMuerte;
270                        grupoMuerteNombre=grupoMuerteNombre;
271                    }
272                    if(golesParcialesMuerte<golesParciales){
273                        golesParcialesMuerte=golesParciales;
274                        grupoMuerteNombre=grupoAux.nombre;
275                    }
276 
277                    cursor = siguienteListaGrupo(listaGrupo, cursor);
278                     if(cantidadGrupos==8){
279                        cursor=finGrupo();
280                    }
281                    }
282                    cout<< "GRUPO MUERTE:"<<grupoMuerteNombre<<endl;
283                    cout<< "GOLES PARCIALES MUERTE:"<<golesParcialesMuerte<<endl;
284                    cout << endl;
285 
286 
287 
288 
289 
290 
291             }break;
292 
293 
294 
295 
296             case 3:break;
297 
298 
299 
300                 }
301                 }
302                  break;
303 
304 
305       case 3:
306                     calcularOctavos(listaEquipo,listaGrupo,listaPartido);
307                     calcularCuartos(listaPartido);
308                     calcularSemi(listaPartido);
309                     calcularFinal(listaPartido);
310             guardarDatos(listaEquipo,listaGrupo,listaPartido);
311          break;
312 
```

```
313       case 4:
314          bateriaJugadores();
315          break;
316       }
317      }
318      return 0;
319  }
320
321  ListaEquipo cargarEquipos(){
322  ListaEquipo listaEquipo;
323  crearListaEquipo(listaEquipo);
324  Equipo equipo;
325  crearEquipo(equipo);
326  ifstream archivo("equipos.txt"); // abrir el archivo en modo lectura//
327  int aux=0;
328  string linea;
329  bool fallos=false;
330  if(archivo.is_open()){
331      while(!archivo.eof()){        //mientras no sea el fin de archivo//
332              getline(archivo,linea,';'); // leo desde el comienzo has ";" y lo guardo en linea//
333              stringstream id(linea);  //Cargo en el stream "id", lo que esta en linea;
334              id>>aux;                  // lo paso a aux;
335                if(traerEquipo(listaEquipo,aux)!=finEquipo()){
336             cout<<"Informe de errores: Equipos"<<endl;
337             cout<<"El id"<< aux<< "ya esta cargado"<<endl;
338                fallos=true;
339          }
340              setId(equipo,aux);
341              getline(archivo,linea,';');
342              setNombre(equipo,linea);
343              getline(archivo,linea,';');
344              stringstream golA(linea);
345              golA>>aux;
346              setGolesAFavor(equipo,aux);
347              getline(archivo,linea,';');
348              stringstream golE(linea);
349              golE>>aux;
350              setGolesEnContra(equipo,aux);
351              getline(archivo,linea);
352              stringstream puntos(linea);
353              puntos>>aux;
354              setPuntos(equipo,aux);
355              cargarJugadores(equipo);
356              adicionarFinal(listaEquipo,equipo);
357              destructor(equipo);
358
359           }
360          archivo.seekg(0); //Me posiciono al principio del archivo
361  }
362  archivo.close();
363  if(fallos)
364     exit(1);
365  return listaEquipo;
366  }
367
368  ListaGrupo cargarGrupos(){
369  ListaGrupo listaGrupo;
370  crearListaGrupo(listaGrupo);
371  Grupo grupo;
372  crearGrupo(grupo);
373  ifstream archivo("grupos.txt");
374  int aux=0;
375  char aux1;
376  string linea;
377  bool fallos=false;
378  if(archivo.is_open()){
```

```
379
380     while(!archivo.eof()){
381          getline(archivo,linea,';');
382          aux1=linea[0];
383           if(traerGrupo(listaGrupo,aux1)!=finGrupo()){
384            cout<<"Informe de errores: Grupos"<<endl;
385            cout<<"El id"<< aux1<< "ya esta cargado"<<endl;
386               fallos=true;
387         }
388          setId(grupo,aux1);
389          getline(archivo,linea,';');
390          setNombre(grupo,linea);
391          getline(archivo,linea,';');
392          stringstream id1(linea);
393          id1>>aux;
394          setIdEquipo1(grupo,aux);
395          getline(archivo,linea,';');
396          stringstream id2(linea);
397          id2>>aux;
398          setIdEquipo2(grupo,aux);
399          getline(archivo,linea,';');
400          stringstream id3(linea);
401          id3>>aux;
402          setIdEquipo3(grupo,aux);
403          getline(archivo,linea);
404          stringstream id4(linea);
405          id4>>aux;
406          setIdEquipo4(grupo,aux);
407          adicionarFinal(listaGrupo,grupo);
408
409
410      }
411       archivo.seekg(0); //Me posiciono al principio del archivo
412 }
413 archivo.close();
414 if(fallos)
415     exit(1);
416 return listaGrupo;
417 }
418
419 void bateriaJugadores(){//bateria  de jugadores carga  todo en 0
420     int id =0;
421     ofstream ficheroSalida;
422
423     ficheroSalida.open ("jugadores.txt",ios::trunc);
424     ficheroSalida.close();
425
426     ficheroSalida.open ("jugadores.txt",ios::app);
427     for (int i=0; i<32; i++){
428         for (int u=0;u<23;u++){
429           id++;
430           ficheroSalida <<id<<";"<<"Equipo"<<i+1<< "JugadorNumero"<<u+1<<";"<<0<< ";"<<i+1 <<"\n";
431           //cout <<id<<";"<<"Equipo"<<i+1<< "JugadorNumero"<<u+1<<";"<<0<< ";"<<i+1 <<"\n";
432      }
433    }
434
435     ficheroSalida.close();
436
437 }
438 ListaPartido cargarPartidos(){//Levanta los partidos de los  txt y los pone en una lista
439 ListaPartido listaPartido;
440
441 crearListaPartido(listaPartido);
442 Partido partido;
443 crearPartido(partido);
444 ifstream archivo("partidos.txt");
```

```
int aux;
string linea;
bool fallos=false;
if(archivo.is_open()){
    while(!archivo.eof()){
        getline(archivo,linea,';');
        stringstream id(linea);
        id>>aux;
        if(traerNodoPartido(listaPartido,aux)!=finListaPartido()){
           cout<<"Informe de errores: Partidos"<<endl;
           cout<<"El id"<< aux<< "ya esta cargado"<<endl;
              fallos=true;
        }
        setId(partido,aux);

        getline(archivo,linea,';');
        stringstream el(linea);
        el>>aux;
        setIdEquipoL(partido,aux);
        getline(archivo,linea,';');
        stringstream ev(linea);
        ev>>aux;
        setIdEquipoV(partido,aux);
        getline(archivo,linea,';');
        stringstream gl(linea);
        gl>>aux;
        setGolesL(partido,aux);
        getline(archivo,linea);
        stringstream gv(linea);
        gv>>aux;
        setGolesV(partido,aux);
        adicionarFinal(listaPartido,partido);


    }


    archivo.seekg(0);

}
archivo.close();
if(fallos)
    exit(1);

return listaPartido;
}

void cargarJugadores(Equipo &equipo){
ListaJugador listaJugador;
crearListaJugador(listaJugador);
Jugador jugador;
crearJugador(jugador);
ifstream archivoJugadores("jugadores.txt");
int aux=0;
string linea;
bool fallos=false;
int idEquipo=0;
idEquipo=getId(equipo);
if(archivoJugadores.is_open()){

        while(!archivoJugadores.eof()){
        getline(archivoJugadores,linea,';');
        stringstream id(linea);
        id>>aux;
        if(traerJugador(listaJugador,aux)!=finJugador()){
           cout<<"Informe de errores: Jugadores"<<endl;
```

```
511             cout<<"El id "<< aux<< " ya esta cargado"<<endl;
512                fallos=true;
513         }
514         setId(jugador,aux);
515 
516         getline(archivoJugadores,linea,';');
517         setNombre(jugador,linea);
518 
519         getline(archivoJugadores,linea,';');
520         stringstream goles(linea);
521         goles>>aux;
522         setGoles(jugador,aux);
523 
524         getline(archivoJugadores,linea);
525         stringstream idE(linea);
526         idE>>aux;
527         if (aux >32 || aux <1){
528             cout<<"Informe de errores: Jugadores"<<endl;
529             cout<<"El id "<< aux<< " no corresponde a ningun equipo enlistado"<<endl;
530             fallos=true;
531         }
532         setIdEquipo(jugador,aux);
533         if(aux==idEquipo){
534             adicionarFinal(equipo.listaJugadores,jugador);
535             adicionarFinal(listaJugador,jugador);}
536 
537         }
538 
539 
540 
541 archivoJugadores.seekg(0);
542 
543 }
544 archivoJugadores.close();
545 if(fallos)
546     exit(1);
547 
548 }
549 void imprimirListaJugadores(ListaJugador  &lista){
550     PtrNodoListaJugador cursor = primeroJugador(lista);
551     Jugador jugador;
552 
553     while (cursor != finJugador()) {
554         obtenerDato(lista, jugador, cursor);
555         if(jugador.goles>0){
556         cout << jugador.nombre << endl;
557         cout<<jugador.goles<<endl;}
558         cursor = siguienteJugador(lista, cursor);
559     }
560 
561     cout << endl;
562 }
563 void imprimirListaPartidos(ListaPartido  &lista){
564     PtrNodoPartido cursor = primeroPartido(lista);
565     Partido partido;
566     cout<<"probando"<<endl;
567     while (cursor != finListaPartido()) {
568         obtenerDato(lista, partido, cursor);
569         cout << partido.id << endl;
570         cout << partido.idEquipoL << endl;
571         cout <<"-----"<<endl;
572         cursor = siguientePartido(lista, cursor);
573     }
574 
575     cout << endl;
576 }
```

```
577  PtrNodoListaJugador traerJugador(ListaEquipo lista,int idEquipo,int idJugador){
578
579  PtrNodoListaEquipo ptrNodo=traerEquipo(lista,idEquipo);
580  Jugador jugador;
581  crearJugador(jugador);
582  setId(jugador,idJugador);
583  PtrNodoListaJugador ptrNodoJugador = localizarDato(ptrNodo->equipo.listaJugadores,jugador);
584
585  return ptrNodoJugador;
586  }
587
588  PtrNodoListaEquipo traerEquipo(ListaEquipo listaEquipo,int idEquipo){
589  Equipo equipo;
590  crearEquipo(equipo);
591  setId(equipo,idEquipo);
592  PtrNodoListaEquipo ptrNodo= localizarDato(listaEquipo,equipo);
593
594  return ptrNodo;
595  }
596  PtrNodoPartido traerNodoPartido(ListaPartido listaPartido,int idPartido){
597  Partido partido;
598  crearPartido(partido);
599  setId(partido,idPartido);
600  PtrNodoPartido ptrNodo = localizarDato(listaPartido,partido);
601
602  return ptrNodo;
603  }
604
605
606  PtrNodoGrupo traerGrupo(ListaGrupo listaGrupo,char idGrupo){
607  Grupo grupo;
608  crearGrupo(grupo);
609  setId(grupo,idGrupo);
610  PtrNodoGrupo ptrNodo = localizarDato(listaGrupo,grupo);
611  return ptrNodo;
612  }
613
614
615  void guardarDatos(ListaEquipo listaEquipo,ListaGrupo listaGrupo,ListaPartido listaPartido){
616  ofstream archivoP("partidos.txt");
617  PtrNodoPartido ptrNodo= primeroPartido(listaPartido);
618  Partido partido;
619  int indice=0;
620  if(archivoP.is_open()){
621      while(ptrNodo!=finListaPartido() && indice!=64){
622          indice++;
623          obtenerDato(listaPartido,partido,ptrNodo);
624          archivoP<<partido.id;
625          archivoP<<";";
626          archivoP<<partido.idEquipoL;
627          archivoP<<";";
628          archivoP<<partido.idEquipoV;
629          archivoP<<";";
630          archivoP<<partido.golesL;
631          archivoP<<";";
632          archivoP<<partido.golesV;
633          if(ptrNodo->siguiente!=finListaPartido())
634          archivoP<<endl;
635
636          ptrNodo=siguientePartido(listaPartido,ptrNodo);
637
638
639      }
640      }
641  indice=0;
642  archivoP.close();
```

```
643  ofstream archivoG("grupos.txt");
644  PtrNodoGrupo ptrNodoGrupo=primeroListaGrupo(listaGrupo);
645  Grupo grupo;
646
647  if(archivoG.is_open()){
648      while(ptrNodoGrupo!=finGrupo() && indice!=8){
649          indice++;
650          obtenerDato(listaGrupo,grupo,ptrNodoGrupo);
651          archivoG<<ptrNodoGrupo->grupo.id;
652          archivoG<<";";
653          archivoG<<ptrNodoGrupo->grupo.nombre;
654          archivoG<<";";
655          archivoG<<ptrNodoGrupo->grupo.idEquipo1;
656          archivoG<<";";
657          archivoG<<ptrNodoGrupo->grupo.idEquipo2;
658          archivoG<<";";
659          archivoG<<ptrNodoGrupo->grupo.idEquipo3;
660          archivoG<<";";
661          archivoG<<ptrNodoGrupo->grupo.idEquipo4;
662          if(ptrNodoGrupo->siguiente != finGrupo())
663          archivoG<<endl;
664
665          ptrNodoGrupo=siguienteListaGrupo(listaGrupo,ptrNodoGrupo);
666
667      }
668
669  }
670  indice=0;
671  archivoG.close();
672  ofstream archivoE("equipos.txt");
673  PtrNodoListaEquipo ptrNodoEquipo = primeroEquipo(listaEquipo);
674  Equipo equipo;
675  ofstream archivoJ("jugadores.txt");
676  Jugador jugador;
677  if(archivoE.is_open() && archivoJ.is_open()){
678      while(ptrNodoEquipo!=finEquipo()){
679          archivoE<<ptrNodoEquipo->equipo.id;
680          archivoE<<";";
681          archivoE<<ptrNodoEquipo->equipo.nombre;
682          archivoE<<";";
683          archivoE<<ptrNodoEquipo->equipo.golesAFavor;
684          archivoE<<";";
685          archivoE<<ptrNodoEquipo->equipo.golesEnContra;
686          archivoE<<";";
687          archivoE<<ptrNodoEquipo->equipo.puntos;
688          PtrNodoListaJugador ptrNodoJugador=primeroJugador(ptrNodoEquipo->equipo.listaJugadores);
689          while(ptrNodoJugador!=finJugador() && indice!=736){
690              indice++;
691              archivoJ<<ptrNodoJugador->jugador.id;
692              archivoJ<<";";
693              archivoJ<<ptrNodoJugador->jugador.nombre;
694              archivoJ<<";";
695              archivoJ<<ptrNodoJugador->jugador.goles;
696              archivoJ<<";";
697              archivoJ<<ptrNodoJugador->jugador.idEquipo;
698              if(indice!=736)
699              archivoJ<<endl;
700
701              ptrNodoJugador=siguienteJugador(ptrNodoEquipo->equipo.listaJugadores,ptrNodoJugador);
702
703          }
704          if(ptrNodoEquipo->siguiente!=finEquipo()){
705          archivoE<<endl;}
706          ptrNodoEquipo=siguienteEquipo(listaEquipo,ptrNodoEquipo);
707      }
708  }
```

```
indice=0;
archivoJ.close();
archivoE.close();

cout<<"datos guardados"<<endl;
}
PtrNodoListaJugador traerJugador(ListaJugador listaJugador,int idJugador){
PtrNodoListaJugador ptrNodoJugador= primeroJugador(listaJugador);
Jugador jugador;
crearJugador(jugador);
setId(jugador,idJugador);
ptrNodoJugador=localizarDato(listaJugador,jugador);

return ptrNodoJugador;

}
void calcularOctavos(ListaEquipo listaEquipo,ListaGrupo listaGrupo,ListaPartido &listaPartido){
PtrNodoGrupo ptrNodoGrupoA=traerGrupo(listaGrupo,'A');
PtrNodoGrupo ptrNodoGrupoB=traerGrupo(listaGrupo,'B');
PtrNodoGrupo ptrNodoGrupoC=traerGrupo(listaGrupo,'C');
PtrNodoGrupo ptrNodoGrupoD=traerGrupo(listaGrupo,'D');
PtrNodoGrupo ptrNodoGrupoE=traerGrupo(listaGrupo,'E');
PtrNodoGrupo ptrNodoGrupoF=traerGrupo(listaGrupo,'F');
PtrNodoGrupo ptrNodoGrupoG=traerGrupo(listaGrupo,'G');
PtrNodoGrupo ptrNodoGrupoH=traerGrupo(listaGrupo,'H');

PtrNodoListaEquipo ptrPrimeroGA=primero(ptrNodoGrupoA->grupo,listaEquipo);
PtrNodoListaEquipo ptrSegundoGA=segundo(ptrNodoGrupoA->grupo,listaEquipo);

PtrNodoListaEquipo ptrPrimeroGB=primero(ptrNodoGrupoB->grupo,listaEquipo);
PtrNodoListaEquipo ptrSegundoGB=segundo(ptrNodoGrupoB->grupo,listaEquipo);

PtrNodoListaEquipo ptrPrimeroGC=primero(ptrNodoGrupoC->grupo,listaEquipo);
PtrNodoListaEquipo ptrSegundoGC=segundo(ptrNodoGrupoC->grupo,listaEquipo);

PtrNodoListaEquipo ptrPrimeroGD=primero(ptrNodoGrupoD->grupo,listaEquipo);
PtrNodoListaEquipo ptrSegundoGD=segundo(ptrNodoGrupoD->grupo,listaEquipo);

PtrNodoListaEquipo ptrPrimeroGE=primero(ptrNodoGrupoE->grupo,listaEquipo);
PtrNodoListaEquipo ptrSegundoGE=segundo(ptrNodoGrupoE->grupo,listaEquipo);

PtrNodoListaEquipo ptrPrimeroGF=primero(ptrNodoGrupoF->grupo,listaEquipo);
PtrNodoListaEquipo ptrSegundoGF=segundo(ptrNodoGrupoF->grupo,listaEquipo);

PtrNodoListaEquipo ptrPrimeroGG=primero(ptrNodoGrupoG->grupo,listaEquipo);
PtrNodoListaEquipo ptrSegundoGG=segundo(ptrNodoGrupoG->grupo,listaEquipo);

PtrNodoListaEquipo ptrPrimeroGH=primero(ptrNodoGrupoH->grupo,listaEquipo);
PtrNodoListaEquipo ptrSegundoGH=segundo(ptrNodoGrupoH->grupo,listaEquipo);

PtrNodoPartido ptrPartido= traerNodoPartido(listaPartido,49);
ptrPartido->partido.idEquipoL=ptrPrimeroGA->equipo.id;
ptrPartido->partido.idEquipoV=ptrSegundoGB->equipo.id;

ptrPartido=traerNodoPartido(listaPartido,50);
ptrPartido->partido.idEquipoL=ptrPrimeroGC->equipo.id;
ptrPartido->partido.idEquipoV=ptrSegundoGD->equipo.id;

ptrPartido=traerNodoPartido(listaPartido,51);
ptrPartido->partido.idEquipoL=ptrPrimeroGB->equipo.id;
ptrPartido->partido.idEquipoV=ptrSegundoGA->equipo.id;

ptrPartido=traerNodoPartido(listaPartido,52);
ptrPartido->partido.idEquipoL=ptrPrimeroGD->equipo.id;
ptrPartido->partido.idEquipoV=ptrSegundoGC->equipo.id;

```

```
775  ptrPartido=traerNodoPartido(listaPartido,53);
776  ptrPartido->partido.idEquipoL=ptrPrimeroGE->equipo.id;
777  ptrPartido->partido.idEquipoV=ptrSegundoGF->equipo.id;
778
779  ptrPartido=traerNodoPartido(listaPartido,54);
780  ptrPartido->partido.idEquipoL=ptrPrimeroGG->equipo.id;
781  ptrPartido->partido.idEquipoV=ptrSegundoGH->equipo.id;
782
783  ptrPartido=traerNodoPartido(listaPartido,55);
784  ptrPartido->partido.idEquipoL=ptrPrimeroGF->equipo.id;
785  ptrPartido->partido.idEquipoV=ptrSegundoGE->equipo.id;
786
787  ptrPartido=traerNodoPartido(listaPartido,56);
788  ptrPartido->partido.idEquipoL=ptrPrimeroGH->equipo.id;
789  ptrPartido->partido.idEquipoV=ptrSegundoGG->equipo.id;
790
791
792
793  }
794
795  PtrNodoListaEquipo primero(Grupo grupo,ListaEquipo listaEquipo){
796  PtrNodoListaEquipo nodo1= traerEquipo(listaEquipo,grupo.idEquipo1);
797  PtrNodoListaEquipo nodo2= traerEquipo(listaEquipo,grupo.idEquipo2);
798  PtrNodoListaEquipo nodo3= traerEquipo(listaEquipo,grupo.idEquipo3);
799  PtrNodoListaEquipo nodo4= traerEquipo(listaEquipo,grupo.idEquipo4);
800  PtrNodoListaEquipo primero=finEquipo();
801  ListaEquipo nuevaLista;
802  crearListaEquipo(nuevaLista);
803  adicionarAlPrincipio(nuevaLista,nodo1->equipo);
804  adicionarAlPrincipio(nuevaLista,nodo2->equipo);
805  adicionarAlPrincipio(nuevaLista,nodo3->equipo);
806  adicionarAlPrincipio(nuevaLista,nodo4->equipo);
807  reordenar(nuevaLista);
808  primero=primeroEquipo(nuevaLista);
809
810  return primero;
811
812  }
813
814  PtrNodoListaEquipo segundo(Grupo grupo,ListaEquipo listaEquipo){
815  PtrNodoListaEquipo nodo1= traerEquipo(listaEquipo,grupo.idEquipo1);
816  PtrNodoListaEquipo nodo2= traerEquipo(listaEquipo,grupo.idEquipo2);
817  PtrNodoListaEquipo nodo3= traerEquipo(listaEquipo,grupo.idEquipo3);
818  PtrNodoListaEquipo nodo4= traerEquipo(listaEquipo,grupo.idEquipo4);
819  PtrNodoListaEquipo segundo=finEquipo();
820  ListaEquipo nuevaLista;
821  crearListaEquipo(nuevaLista);
822  adicionarAlPrincipio(nuevaLista,nodo1->equipo);
823  adicionarAlPrincipio(nuevaLista,nodo2->equipo);
824  adicionarAlPrincipio(nuevaLista,nodo3->equipo);
825  adicionarAlPrincipio(nuevaLista,nodo4->equipo);
826  reordenar(nuevaLista);
827  segundo=primeroEquipo(nuevaLista)->siguiente;
828  return segundo;
829  }
830
831  void calcularCuartos(ListaPartido listaPartido){
832  int idL=0,idV=0;
833  PtrNodoPartido ptrNodoPartido=traerNodoPartido(listaPartido,49);
834  idL=traerIdGanador(ptrNodoPartido);
835  PtrNodoPartido ptrNodoPart=traerNodoPartido(listaPartido,50);
836  idV=traerIdGanador(ptrNodoPart);
837  PtrNodoPartido ptrProximo = traerNodoPartido(listaPartido,57);
838  ptrProximo->partido.idEquipoL=idL;
839  ptrProximo->partido.idEquipoV=idV;
840
```

```
841  ptrNodoPartido=traerNodoPartido(listaPartido,53);
842  idL=traerIdGanador(ptrNodoPartido);
843  ptrNodoPart=traerNodoPartido(listaPartido,54);
844  idV=traerIdGanador(ptrNodoPart);
845  ptrProximo=traerNodoPartido(listaPartido,58);
846  ptrProximo->partido.idEquipoL=idL;
847  ptrProximo->partido.idEquipoV=idV;
848
849  ptrNodoPartido=traerNodoPartido(listaPartido,51);
850  idL=traerIdGanador(ptrNodoPartido);
851  ptrNodoPart=traerNodoPartido(listaPartido,52);
852  idV=traerIdGanador(ptrNodoPart);
853  ptrProximo=traerNodoPartido(listaPartido,59);
854  ptrProximo->partido.idEquipoL=idL;
855  ptrProximo->partido.idEquipoV=idV;
856
857  ptrNodoPartido=traerNodoPartido(listaPartido,55);
858  idL=traerIdGanador(ptrNodoPartido);
859  ptrNodoPart=traerNodoPartido(listaPartido,56);
860  idV=traerIdGanador(ptrNodoPart);
861  ptrProximo=traerNodoPartido(listaPartido,60);
862  ptrProximo->partido.idEquipoL=idL;
863  ptrProximo->partido.idEquipoV=idV;
864
865
866
867  }
868  int traerIdGanador(PtrNodoPartido ptrNodoPartido){
869      int idGanador=-1;
870      if(ptrNodoPartido->partido.golesL>-1 && ptrNodoPartido->partido.golesV>-1 ){
871
872      if(ptrNodoPartido->partido.golesL > ptrNodoPartido->partido.golesV){
873         idGanador=ptrNodoPartido->partido.idEquipoL;
874      }
875      else{
876          idGanador=ptrNodoPartido->partido.idEquipoV;
877      }
878      }
879
880      return idGanador;
881
882  }
883  void calcularSemi(ListaPartido listaPartido){
884  int idL=0,idV=0;
885  PtrNodoPartido ptrNodoPartido=traerNodoPartido(listaPartido,57);
886  idL=traerIdGanador(ptrNodoPartido);
887  PtrNodoPartido ptrNodoPart=traerNodoPartido(listaPartido,58);
888  idV=traerIdGanador(ptrNodoPart);
889  PtrNodoPartido ptrProximo=traerNodoPartido(listaPartido,61);
890  ptrProximo->partido.idEquipoL=idL;
891  ptrProximo->partido.idEquipoV=idV;
892
893  ptrNodoPartido=traerNodoPartido(listaPartido,59);
894  idL=traerIdGanador(ptrNodoPartido);
895  ptrNodoPart=traerNodoPartido(listaPartido,60);
896  idV=traerIdGanador(ptrNodoPart);
897  ptrProximo=traerNodoPartido(listaPartido,61);
898  ptrProximo->partido.idEquipoL=idL;
899  ptrProximo->partido.idEquipoV=idV;
900
901
902
903  }
904
905  void calcularFinal(ListaPartido listaPartido){
906  int idL=0,idV=0;
```

```
907  PtrNodoPartido ptrNodoPartido=traerNodoPartido(listaPartido,61);
908  idL=traerIdGanador(ptrNodoPartido);
909
910  PtrNodoPartido ptrNodoPart = traerNodoPartido(listaPartido,62);
911  idV=traerIdGanador(ptrNodoPart);
912
913  PtrNodoPartido ptrProximo=traerNodoPartido(listaPartido,64);
914  ptrProximo->partido.idEquipoL=idL;
915  ptrProximo->partido.idEquipoV=idV;
916  PtrNodoPartido ptrTercerPuesto=traerNodoPartido(listaPartido,63);
917
918  if(ptrNodoPartido->partido.idEquipoL==idL){
919      ptrTercerPuesto->partido.idEquipoL=ptrNodoPartido->partido.idEquipoV;
920  }
921  else {
922          ptrTercerPuesto->partido.idEquipoL=ptrNodoPartido->partido.idEquipoL;
923     }
924  if(ptrNodoPart->partido.idEquipoV==idV){
925      ptrTercerPuesto->partido.idEquipoV=ptrNodoPart->partido.idEquipoL;
926  }
927  else{
928      ptrTercerPuesto->partido.idEquipoV=ptrNodoPart->partido.idEquipoV;
929
930  }
931
932
933  }
934
935  Partido validarPartidoEncontrado (ListaPartido listaPartido, Partido partido){
936      bool encontrado = false;
937      int auxId;
938      auxId = partido.id;
939      while(encontrado==false){
940
941          if( localizarDato(listaPartido,partido)==finListaPartido()){
942              cout<<"Informe de errores: Adminitrar Partido"<<endl;
943              cout<<"El id: |"<< auxId<< "|no existe en la lista de partidos enlistados"<<endl;
944              cout<<"Reintente por favor"<<endl;
945              cout<<"Ingrese id del partido a iniciar"<<endl;
946              cin>>auxId;
947              partido.id= auxId;
948              if( localizarDato(listaPartido,partido)!=finListaPartido())
949                  encontrado = true;
950
951              }else {encontrado = true;}
952      }
953      return partido;
954  }
955
956  Equipo validarEquipoEncontrado(ListaEquipo listaEquipo, Equipo equipo){
957      bool encontrado = false;
958      int auxId;
959      while (encontrado == false){
960           if( localizarDato(listaEquipo,equipo)==finEquipo()){
961              cout<<"Informe de errores: Adminitrar Partido"<<endl;
962              cout<<"El id: |"<< auxId<< "|no existe en la lista de equipos enlistados"<<endl;
963              cout<<"Reintente por favor ingresar el id"<<endl;
964
965              cin>>auxId;
966              equipo.id= auxId;
967              if( localizarDato(listaEquipo,equipo)!= finEquipo())
968                  encontrado = true;
969
970              }else {encontrado = true;}
971      }
972      return equipo;
```

```
973  }
```

Archivo Partido.cpp

```cpp
#include "partido.h"
#include <stdlib.h>
#include <string>
#include <sstream>
#include <fstream>
#ifndef NULL
#define NULL      0
#endif
using namespace std;
/*-------------------------------------------------------------------------*/
//                          IMPLEMENTACION DE PRIMITIVAS
/*-------------------------------------------------------------------------*/
void crearPartido(Partido &partido){
partido.id=0;
partido.idEquipoL=0;
partido.idEquipoV=0;
partido.golesL=0;
partido.golesV=0;
}
/*-------------------------------------------------------------------------*/
int getId(Partido &partido){
return partido.id;
}
/*-------------------------------------------------------------------------*/
int getIdEquipoL(Partido &partido){
return partido.idEquipoL;
}
/*-------------------------------------------------------------------------*/
int getIdEquipoV(Partido &partido){
return partido.idEquipoV;
}
/*-------------------------------------------------------------------------*/
int getGolesL(Partido &partido){
return partido.golesL;
}
/*-------------------------------------------------------------------------*/
int getGolesV(Partido &partido){
return partido.golesV;
}
/*-------------------------------------------------------------------------*/
void setId(Partido &partido,int id){
partido.id=id;
}
/*-------------------------------------------------------------------------*/
void setIdEquipoL(Partido &partido, int idEquipoL){
partido.idEquipoL=idEquipoL;
}
/*-------------------------------------------------------------------------*/
void setIdEquipoV(Partido &partido, int idEquipoV){
partido.idEquipoV=idEquipoV;
}
/*-------------------------------------------------------------------------*/
void setGolesL(Partido &partido, int golesL){
partido.golesL=golesL;
}
/*-------------------------------------------------------------------------*/
void setGolesV(Partido &partido, int golesV){
partido.golesV=golesV;
}
/*-------------------------------------------------------------------------*/
void destructor(Partido &partido){
partido.id=0;
partido.idEquipoL=0;
partido.idEquipoV=0;
partido.golesL=0;
partido.golesV=0;
```

```
67  }
68  /*---------------------------------------------------------------------------*/
69  /*****************************************************************************/
```

Archivo Equipo.cpp

```cpp
#include "Equipo.h"
#ifndef NULL
#define NULL      0
#endif
#include <cstring>
#include <string>
#include <iostream>
#include "ListaJugadores.h"
using namespace std;
/*-------------------------------------------------------------------------*/
//                          IMPLEMENTACION DE PRIMITIVAS
/*-------------------------------------------------------------------------*/
void crearEquipo(Equipo &equipo){
equipo.id=0;
equipo.nombre="-";
equipo.golesAFavor=0;
equipo.golesEnContra=0;
equipo.puntos=0;
crearListaJugador(equipo.listaJugadores);
}
/*-------------------------------------------------------------------------*/
int getId(Equipo equipo){
    return equipo.id;
}
/*-------------------------------------------------------------------------*/
string getNombre(Equipo equipo){
return equipo.nombre;
}
/*-------------------------------------------------------------------------*/
void setNombre(Equipo &equipo,string nombre){
equipo.nombre=nombre;
}
/*-------------------------------------------------------------------------*/
void setId(Equipo &equipo, int id){
equipo.id=id;
}
/*-------------------------------------------------------------------------*/
void setGolesAFavor(Equipo &equipo,int goles){
equipo.golesAFavor=goles;
}
/*-------------------------------------------------------------------------*/
int getGolesAFavor(Equipo equipo){
return equipo.golesAFavor;}
/*-------------------------------------------------------------------------*/
void setGolesEnContra(Equipo &equipo,int golesEnContra){
equipo.golesEnContra=golesEnContra;
}
/*-------------------------------------------------------------------------*/
int getGolesEnContra(Equipo equipo){
return equipo.golesEnContra;
}
/*-------------------------------------------------------------------------*/
void setPuntos(Equipo &equipo,int puntos){
equipo.puntos=puntos;
}
/*-------------------------------------------------------------------------*/
int getPuntos(Equipo equipo){
return equipo.puntos;
}
/*-------------------------------------------------------------------------*/
ListaJugador getLista(Equipo equipo){
return equipo.listaJugadores;}
/*-------------------------------------------------------------------------*/
void destructor(Equipo &equipo){
equipo.id=0;
equipo.nombre="-";
```

```
67  equipo.golesAFavor=0;
68  equipo.golesEnContra=0;
69  equipo.puntos=0;
70  crearListaJugador(equipo.listaJugadores);
71  }
72  /*----------------------------------------------------------------------------*/
73  /****************************************************************************/
```

Archivo Grupo.cpp

```cpp
#include "Grupo.h"
#ifndef NULL
#define NULL      0
#endif
#include<string>
#include <iostream>
/*-------------------------------------------------------------------------------*/
//                           IMPLEMENTACION DE PRIMITIVAS
/*-------------------------------------------------------------------------------*/
void crearGrupo(Grupo &grupo){
    grupo.id='-';
    grupo.nombre="-";
    grupo.idEquipo1=0;
    grupo.idEquipo2=0;
    grupo.idEquipo3=0;
    grupo.idEquipo4=0;
}
/*-------------------------------------------------------------------------------*/
void setId(Grupo &grupo,char id){
    grupo.id=id;
}
/*-------------------------------------------------------------------------------*/
char getId(Grupo grupo){
return grupo.id;
}
/*-------------------------------------------------------------------------------*/
void setNombre(Grupo &grupo,string nombre){
grupo.nombre=nombre;
}
/*-------------------------------------------------------------------------------*/
string getNombre(Grupo grupo){
return grupo.nombre;}
/*-------------------------------------------------------------------------------*/
void setIdEquipo1(Grupo &grupo,int id){
grupo.idEquipo1=id;
}
/*-------------------------------------------------------------------------------*/
int getIdEquipo1(Grupo grupo){
return grupo.idEquipo1;
}
/*-------------------------------------------------------------------------------*/
void setIdEquipo2(Grupo &grupo,int id){
grupo.idEquipo2=id;
}
/*-------------------------------------------------------------------------------*/
int getIdEquipo2(Grupo grupo){
return grupo.idEquipo2;
}
/*-------------------------------------------------------------------------------*/
void setIdEquipo3(Grupo &grupo,int id){
grupo.idEquipo3=id;
}
/*-------------------------------------------------------------------------------*/
int getIdEquipo3(Grupo grupo){
return grupo.idEquipo3;
}
/*-------------------------------------------------------------------------------*/
void setIdEquipo4(Grupo &grupo,int id){
grupo.idEquipo4=id;
}
/*-------------------------------------------------------------------------------*/
int getIdEquipo4(Grupo grupo){
return grupo.idEquipo4;
}
/*-------------------------------------------------------------------------------*/
void destructor(Grupo &grupo){
```

```
67      grupo.id='-';
68      grupo.nombre="-";
69      grupo.idEquipo1=0;
70      grupo.idEquipo2=0;
71      grupo.idEquipo3=0;
72      grupo.idEquipo4=0;
73  }
74  /*-------------------------------------------------------------------------------*/
75  /*********************************************************************************/
```