TRABAJO PRÁCTICO OBLIGATORIO FINAL

Programación II | 2018

Gatica Miguel Elías

18 DE FEBRERO DE 2021

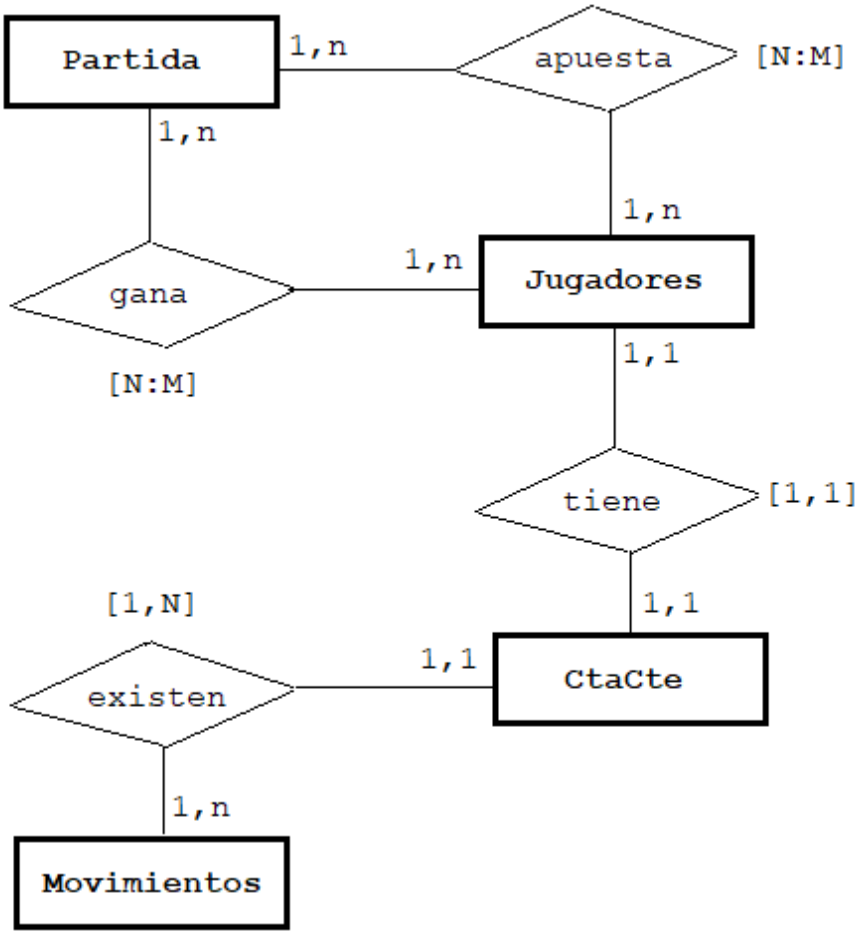
Profesor: Claudio Álvarez | Enc.Trab.Pcos: Raquel Bros

Indice

1 DER (Diagrama Entidad Relación) 1
2 Estructuras 2
2.1 Librería Auxiliar de Estructuras 2
2.2 TDA 3
2.2.1 Árbol AVL 3
2.2.2 Árbol Trinario AVL 26
2.2.3 Colas 48
2.2.4 Hash 52
2.2.4.1 Type Hash 55
2.2.5 Lista Simple Enlace 56
2.2.6 Lista Doble Enlace 64
2.2.7 Lista Parcial Doble Enlace 73
2.3 ME 82
2.3.1 Jugadores 82
2.3.1.1 Type Jugador 87
2.3.2 Cuenta Corriente 88
2.3.2.1 Type Almacén 100
2.3.3 Juego 101
2.3.3.1 Type Juego 110
2.3.4 Apuestas 111
2.3.4.1 Type Apuesta 118
2.3.5 Ganadores 119
2.3.5.1 Type Ganador 128
3 Proyecto 129
3.1 Librería Auxiliar Juego 129
3.2 Formularios 185
3.2.1 Formulario Login 185
3.2.2 Formulario Croupier 201
3.2.3 Formulario Jugadores 214
3.2.4 Formulario Juego 231
3.2.5 Formulario Apostar 327

3.2.6 Formulario Ficticios 339
3.2.7 Formulario Premios 350
3.2.8 Formulario Listados 353
3.2.9 Formulario Premio/Bloqueo 362

1 DER (Diagrama Entidad Relación)



2 Estructuras

2.1 Librería Auxiliar de Estructuras

```
unit Lib_Auxiliar;


interface
uses
  Sysutils;


  Const


    ApuestaMaxima = 1000;
    ApuestaMinima = 50;
    PosNula = -1;
    IniFicticios = 0;
    ClaveNula ='';
    Fict_IniNick = 'X_';


  type
    tClave          = string[15];
    tPos            = longint;
    tNivel          = 0.. maxint;
    tProfundidad    = 0.. maxint;
    tCantidad       = 0.. maxint;



Implementation


END.
```

2.2 TDA

2.2.1 Árbol AVL

```
unit LO_ArbolAVL;


{
Un árbol AVL es aquel que por cada nodo la altura del subárbol izquierdo menos la
altura del subárbol derecho difiera a lo sumo en 1

Antes de insertar hace un buscar, que sino encuentra la clave devuelvo posicion del
padre

Es por eso que al insertar el pos parametrizado corresponde a la posicion fisica del
padre y parado en el ve si corresponde que sea el hijo izquierdo o el hijo derecho

Una vez insertado el nodo, pregunto por el balanceo del árbol.

Se pregunta por el balanceo de cada nodo desde el nodo padre del recién insertado hasta
la raíz

Un nodo esta balanceado si su factor de equilibrio es -1, 1 o 0.., teniendo presenete
que el factor de equilibrio de un árbol binario es la diferencia en altura entre los
subárboles derecho e izquierdo.

Para ello la funcion 'factor de equilibrio' va a verificar que el nodo este
equilibrado. Si un nodo esta desequilibrado, se invoca a 'Caso Desquilibrio' donde
verifica que tipo de rotación se llevara a cabo para balancear dicho nodo

Si un nodo esta desequilibrado, se invoca a 'Caso Desquilibrio' donde verifica que tipo
de rotación se llevara a cabo para balancear dicho nodo.

Si el factor de equilibrio del nodo desbalanceado es positivo, está cargado a la
izquierda..., se revisa con hijo izquierdo

Si el factor de equilibrio del hijo izquierdo es positivo, se lleva a cabo una
RotacionSimpleDerecha sino una RotacionCompuestaDerecha


En el caso de que el factor de equilibrio del nodo desbalanceado es negativo, significa
que está cargado a la derecha...,

se revisa con hijo derecho. Si el factor de equilibrio del hijo derecho es positivo, se
hace una RotacionCompuestaIzquierda sino

una RotacionSimpleIzquierda

}


interface

USES

   SysUtils, Lib_Auxiliar;

TYPE
```

```
  tNodoIndice = record
                  clave:tClave;
                  padre, hi, hd:tPos;
                  nivel:integer;
                  PosEnDatos:tPos;
                end;

  tControlArbol= record
                  Raiz,Borrados: tPos;
                  Cantidad, NivelMax: TCantidad;
                  ficticio: tCantidad;
                end;

  tArchivoNodos= file of tNodoIndice;
  tArchivoControlArbol= file of tControlArbol;

  tArbol= record
                  D:tArchivoNodos;
                  C:tArchivoControlArbol;
                end;

  Procedure CrearMe_Arbol (var Me:tArbol;
                           _NombreArchivoControl,_NombreArchivoIndice:String;
                           _Directorio:String);
  Procedure AbrirMe_Arbol(var Me:tArbol);
  Procedure CerrarMe_Arbol(var Me:tArbol);
  Procedure DestruirMe_Arbol(var Me:tArbol);
  Function Arbol_Vacio(var Me: tArbol):Boolean;
  Procedure InsertarNodo_Arbol(var Me:tArbol; nodo:tNodoIndice;pos:tPos;
tipo:boolean);
  Procedure EliminarNodo_Arbol(var Me:tArbol; pos:tPos);
  Procedure ModificarNodo_Arbol(var Me: tArbol; RegInd:tNodoIndice;Pos:tPos);
  Function  BuscarNodo_Arbol (var Me:tArbol; clave:tClave;var pos:tPos):boolean;
  Procedure CapturarNodo_Arbol(var Me: tArbol; var RegInd:tNodoIndice; Pos:tPos);
```

```
  Function PosNula_Arbol(Var Me:tArbol):tPos;

  Function Raiz_Arbol(var Me: tArbol): tPos;

  Function Padre_Arbol(var Me: tArbol; var Reg: tNodoIndice): tPos;

  Function HijoDer_Arbol(var Me: tArbol; pos: tPos): tPos;

  Function HijoIzq_Arbol(var Me: tArbol; pos: tPos): tPos;

  {-------------------------(BALANCEO)-------------------------------------}

  Procedure RotacionSimpleIzquierda(var me: tArbol; PosNodo: tPos);

  Procedure RotacionSimpleDerecha(var me: tArbol; PosNodo: tPos);

  Procedure RotacionCompuestaDerecha(var me: tArbol; PosNodo: tPos);

  Procedure RotacionCompuestaIzquierda(var me: tArbol; PosNodo: tPos);

  Function ProfundidadNodo_Arbol(me: tArbol; raiz: tPos): integer;

  procedure ProfundidadTotal_Arbol(var me:tArbol; Raiz:tPos; var
Profundidad:tProfundidad);

  Procedure CasoDesequilibrio_Arbol(me: tArbol; PosNodo: tPos);

  Function FactorEquilibrio_Arbol(me: tArbol; PosNodo: tPos): integer;

  Procedure ActualizarNiveles_Arbol(var me: tArbol; Pos:TPos);

  procedure Balancear_Arbol(var me: tArbol; Pos:tpos);

  {------------------------------------------------------------------------}

  Function UltFicticio_Arbol(var Me: tArbol): tCantidad;

  Function CantidadNodos_Arbol(var Me: tArbol): tCantidad;

  Function ObtenerNivel_Arbol(var Me:tArbol; clave:tClave): tCantidad;


VAR

Me: tArbol;


Implementation

{-------------------------------------------------------------------------------}

Procedure CrearMe_Arbol (var Me:tArbol;
_NombreArchivoControl,_NombreArchivoIndice:String;_Directorio:String);

Var

  ioC,ioD:integer;

  Rc: tControlArbol;

begin

  //AssignFile reemplaza el procedimiento Assign que estaba disponible en las primeras
versiones de Delphi.

  assignFile (Me.C, _Directorio + _NombreArchivoControl);
```

```
  assignFile (Me.D, _Directorio + _NombreArchivoIndice);
  {$i-}
  reset(Me.C); ioC:=IoResult; //Si IoResult es cero, la operacion es exitosa
  reset(Me.D); ioD:=IoResult;

  if (IoC<>0) or (ioD<>0) then
  begin
    Rewrite(Me.C);
    ReWrite(Me.D);
    RC.Raiz:=Lib_Auxiliar.PosNula;
    RC.Borrados:=Lib_Auxiliar.PosNula;
    RC.ficticio:= Lib_Auxiliar.IniFicticios;
    Rc.Cantidad:= 0;
    Write(Me.C,RC);
  end;
  Close(Me.D);
  Close(Me.C);
  {$I+}
end;
{-------------------------------------------------------------------------------}
procedure AbrirMe_Arbol(var Me:tArbol);
begin
  Reset(Me.C);
  Reset(Me.D);
end;
{-------------------------------------------------------------------------------}
procedure CerrarMe_Arbol(var Me:tArbol);
begin
  Close(Me.D);
  Close(Me.C);
end;
{-------------------------------------------------------------------------------}
procedure DestruirMe_Arbol(var Me:tArbol);
begin
  Erase(Me.D);
```

```
  Erase(Me.C);
end;
{-------------------------------------------------------------------------------}
Function Arbol_Vacio(var Me: tArbol):Boolean;
var
  Rc:tControlArbol;
begin
  Seek(Me.C,0); Read(Me.c, Rc);
  Arbol_Vacio:=  (Rc.Raiz = Lib_Auxiliar.PosNula) ;
end;
{-------------------------------------------------------------------------------}
Procedure InsertarNodo_Arbol(var Me:tArbol; nodo:tNodoIndice; pos:tPos; tipo:boolean);
{Antes de insertar hace un buscar, que sino encuentra la clave devuelvo posicion del
padre. Es por eso que al insertar el pos parametrizado corresponde a la posicion fisica
del padre y parado en el ve si corresponde que sea el hijo izquierdo o el hijo derecho}
var
  posnueva:tPos;
  reg,rd:tNodoIndice;
  RC:tControlArbol;
begin
  seek(Me.C,0); read(Me.C,RC);

  if (RC.borrados = Lib_Auxiliar.PosNula) then //no hay borrados
  begin
    posnueva:=filesize(Me.D);
      end
  else //hay borrados
    begin
      Posnueva:= RC.Borrados;
      seek(Me.D,posnueva); read(Me.D,rD);
      RC.Borrados:= rD.hi;
        end;
   //Grabo al final ...

   if (RC.Raiz=Lib_Auxiliar.PosNula) then//inserto en árbol vacio
   begin
```

```
    RC.Raiz:= posnueva;
    nodo.padre:= Lib_Auxiliar.PosNula;
    nodo.hi:= Lib_Auxiliar.PosNula;
    nodo.hd:= Lib_Auxiliar.PosNula;
    nodo.nivel:= 1;
   end
   else //inserto como hoja

     begin
      seek(Me.D,pos); read(Me.D,reg);
      nodo.padre:= pos;
      nodo.hd:= Lib_Auxiliar.PosNula;
      nodo.hi:= Lib_Auxiliar.PosNula;
      nodo.nivel:= Reg.Nivel+1;
```

{Si la clave del nodo a insertar es menor a la clave del nodo padre, entonces la posicion nueva la tendra su hijo izquierdo (su hijo izquierdo estará al final físico),

Sino, si la clave del nodo a insertar es mayor a la clave del nodo padre, entonces la posicion nueva la tendra su hijo derecho (su hijo derecho estará al final físico)}

```
      if (nodo.clave < reg.clave) then
        reg.hi:=posnueva
      else
          reg.hd:=posnueva;
        //grabo nodo padre con nodo hijo actualizado
      seek(Me.d,pos); write(Me.d,reg);
  end;
  //Si la profuncididad del nodo es mayor a la prfundidad en el control actualizo.
  If nodo.nivel > RC.NivelMax then
    RC.NivelMax := nodo.nivel;

  //si tipo (jugador) es true es real sino ficticio
  if not (tipo) then
    RC.ficticio:= RC.ficticio+1;

  rc.cantidad:=rc.cantidad+1;
  //se graba el elemento nuevo insertado en el archivo
```

```
  seek(Me.d,posnueva); write(Me.d,nodo);
  //se actualiza el registro control
  seek(Me.c,0); write(Me.c,rc);

 //invoco al balanceo del árbol
  Balancear_Arbol(me, PosNueva);
end;
{--------------------------------------------------------------------------------}
Procedure EliminarNodo_Arbol(var Me:tArbol; pos:tPos);
var
  RD:tNodoIndice;
  RC:tControlArbol;
  rAux,RHD,RDP,RHI:tNodoIndice;
  posAux:tPos;
begin
      seek(Me.c,0); read(Me.c,rc);
      seek(Me.d,pos); read(Me.d,rd);   //Capturo el registro a eliminar
  //no tiene padre, no tiene hijos
      if (pos=rc.raiz) and (rd.hi=Lib_Auxiliar.PosNula) and
(rd.hd=Lib_Auxiliar.PosNula) then
        Rc.Raiz:=Lib_Auxiliar.PosNula
      else
      begin
        if (rd.hi=Lib_Auxiliar.PosNula) and (rd.hd=Lib_Auxiliar.PosNula) then
      //Tiene padre, no tiene hijos. Eliminar de una hoja
        begin
          Seek(Me.D,rd.padre);read(Me.d,rdp);
          If rdp.hi=pos then
            rdp.hi:=Lib_Auxiliar.PosNula
          else
            rdp.hd:=Lib_Auxiliar.PosNula;
          Seek(Me.d,rd.padre);write(Me.d,RDP);
      end
      else
        begin
```

```
    if (rd.hi <> Lib_Auxiliar.PosNula) and (rd.hd <> Lib_Auxiliar.PosNula) then
    //Caso general, tiene padre y tiene hijos...
    begin
      If pos<>RC.raiz then //...no es raiz
      begin
        Seek(Me.d,rd.padre);read(Me.d,rdp);
        If rdp.hi=pos then
          rdp.hi:=rd.hd
        else
          rdp.hd:=rd.hd;
        Seek(Me.d,RD.padre);Write(Me.d,rdp);
      end
      else
        RC.raiz:=rd.hd;

  Seek(Me.d,rd.hd);read(Me.d,RHD);
  RHD.padre:=RD.padre;
  posAux:=RD.hd;
  rAux:=RHD;
  While rAux.hi<>Lib_Auxiliar.PosNula do
  begin
    Seek(Me.d,posAux);read(Me.d,RAux);
    If raux.hi<>Lib_Auxiliar.PosNula then
      posaux:=rAux.hi;
  end;
  Seek(Me.d,RD.hi);Read(Me.d,RHI);
  RHI.padre:=posAux;
  RAux.hi:=RD.hi;
  Seek(Me.d,RD.hi);Write(Me.d,RHI);
  Seek(Me.d,posAux);Write(Me.d,raux);
  If posAux<>RD.hd then
  begin
   Seek(Me.d,RD.hd);Write(Me.d,RHD);
  end;
end
```

```
      else
        begin
          If pos<>RC.Raiz then
          begin
            Seek(Me.D,RD.padre);Read(Me.D,RDP);
            If RDP.hi=pos then
            begin
              If RD.hi=Lib_Auxiliar.PosNula then
              begin
                Rdp.hi:=RD.hd;
                Seek(Me.D,RD.hd);read(Me.d,RHD);
                RHD.padre:=RD.padre;
                Seek(Me.d,RD.hd);write(Me.d,RHD);
              end
              else
              begin
                Rdp.hi:=RD.hi;
                Seek(Me.D,RD.hi);read(Me.d,RHI);
                RHI.padre:=RD.padre;
                Seek(Me.d,RD.hi);write(Me.d,RHI);
              end
            end
            else
             If RD.hi=Lib_Auxiliar.PosNula then
              begin
                Rdp.hd:=RD.hd;
                Seek(Me.D,RD.hd);read(Me.d,RHD);
                RHD.padre:=RD.padre;
                Seek(Me.d,RD.hd);write(Me.d,RHD);
              end
              else
               begin
                Rdp.hd:=RD.hi;
                Seek(Me.D,RD.hi);read(Me.d,RHI);
                RHI.padre:=RD.padre;
```

```
                Seek(Me.d,RD.hi);write(Me.d,RHI);
               end;
            Seek(Me.d,rd.padre);write(Me.d,rdp);
          end
          else
           If RD.hi=Lib_Auxiliar.PosNula then
             RC.Raiz:=RD.hd
           else
             RC.raiz:=RD.hi;
        end;
     end;
   end;
   RD.hi:=RC.borrados;
   RD.hd:=RC.borrados;
   RC.borrados:=pos;
   RC.Cantidad:=RC.Cantidad-1;
   Seek(Me.c,0);write(Me.c,rc); //Grabo la cabecera de control
end;
{-------------------------------------------------------------------------------}
Procedure ModificarNodo_Arbol(var Me: tArbol; RegInd:tNodoIndice;Pos:tPos);
begin
  seek(Me.D,pos); write(Me.D,RegInd);
end;
{-------------------------------------------------------------------------------}
Function  BuscarNodo_Arbol (var Me:tArbol; clave:tClave;var pos:tPos):boolean;
 //El buscar sino encuentra la clave devuelvo la posicion del padre
var
  reg:tNodoIndice;
  RC:tControlArbol;
  encont:boolean;
  posPadre:tpos;
begin
  seek(Me.c,0); read(Me.c,rc);
  pos:=rc.raiz;
  encont:=false;
```

```
  posPadre:=Lib_Auxiliar.PosNula;


  while (not encont) and (pos<>Lib_Auxiliar.PosNula) do
  begin
    seek(Me.d,pos); read(Me.d,reg);
    if (clave=reg.clave) then
      encont:=true
    else
     begin
      if (reg.clave>clave) then
      begin
        posPadre:=pos;
        pos:=reg.hi;
      end
      else
       begin
         posPadre:=pos;
         pos:=reg.hd;
       end;
     end;
  end;
  if (not encont) then
    pos:=posPadre;
 BuscarNodo_Arbol:=encont;
end;
{-------------------------------------------------------------------------------}
Procedure CapturarNodo_Arbol(var Me: tArbol; var RegInd:tNodoIndice; Pos:tPos);
begin
  seek(Me.D,Pos); Read(Me.D,RegInd);
end;
{-------------------------------------------------------------------------------}
function PosNula_Arbol(Var Me:tArbol):tPos;
begin
  PosNula_Arbol:= Lib_Auxiliar.PosNula;
end;
```

```
{-------------------------------------------------------------------------------}
function Raiz_Arbol(var Me: tArbol): tPos;
var
  Rc: tControlArbol;
begin
  Seek(Me.c, 0);
  Read(Me.c, Rc);

  Raiz_Arbol:= Rc.Raiz;
end;
{-------------------------------------------------------------------------------}
Function Padre_Arbol(var Me: tArbol; var Reg: tNodoIndice): tPos;
begin
 Padre_Arbol:= Reg.padre;
end;
{-------------------------------------------------------------------------------}
function HijoDer_Arbol(var Me: tArbol; pos: tPos): tPos;
var
reg: tNodoIndice;
begin
  seek(me.D,pos); read(me.D,reg);
  HijoDer_Arbol:= reg.hd;
end;
{-------------------------------------------------------------------------------}
function HijoIzq_Arbol(var Me: tArbol; pos: tPos): tPos;
var
reg: tNodoIndice;
begin
  seek(me.D,pos); read(me.D,reg);
  HijoIzq_Arbol:= reg.hi;
end;
{-------------------------------------------------------------------------------}
//Rotacion en arbol tambien conocido como 'Rotacion Derecha-Derecha'
Procedure RotacionSimpleIzquierda (var me: tArbol; PosNodo: tPos);
Var
```

```
  NodoArriba,NodoAbajo,NodoAnterior:tNodoIndice;
  RC:tControlArbol;
  PosAux:tPos;
begin
  Seek (me.D,PosNodo); Read (me.D,NodoArriba);
  Seek (me.D,NodoArriba.hd); Read (me.D,NodoAbajo);
  Seek (me.C,0);Read(me.C,RC);

  //Cambio enlaces
  PosAux:=NodoAbajo.hi;
  NodoAbajo.padre:=NodoArriba.padre;
  NodoAbajo.hi:=PosNodo;
  NodoArriba.padre:=NodoArriba.hd;
  NodoArriba.hd := PosAux; ;

  //Si el desbalanceado no era raíz
  If NodoAbajo.padre <> Lib_Auxiliar.PosNula then
  begin
    //Me paro en el padre del hijo desbalanceado, y le pongo como hijo al que era su
nieto
    Seek (Me.D,NodoAbajo.padre); Read (Me.D,NodoAnterior);
    If NodoAnterior.hd = PosNodo then
      NodoAnterior.hd :=NodoArriba.padre
    else
      NodoAnterior.hi :=NodoArriba.padre;
    Seek (Me.D,NodoAbajo.padre); Write (Me.D,NodoAnterior);
  end;
  //Cambio niveles
  NodoAbajo.nivel:=NodoAbajo.nivel-1;
  NodoArriba.nivel := NodoArriba.nivel+1;

  //Verifico que el del problema no haya sido la raiz
  //Anteriormeente a NodoArriba.padre le da NodoAbajo.hdecho. Por lo tanto, pone como
nueva raiz
  //al hijo derecho del que estaba desbalanceado.
  If PosNodo=RC.Raiz then
```

```
    RC.raiz := NodoArriba.padre;


  Seek (Me.D,PosNodo); Write (Me.D,NodoArriba);
  Seek (Me.D,NodoArriba.padre); Write (Me.D,NodoAbajo);
  Seek (Me.C,0);Write(Me.C,RC);


  {Llamo a Actualizar Nivel para actualizar todos los nodos por debajo del padre del
nodo que estaba desbalanceado
  porque ese nodo sigue teniendo el nivel original}
  ActualizarNiveles_Arbol(ME,PosNodo);
end;
{-------------------------------------------------------------------------------}
//Rotacion en arbol tambien conocido como 'Rotacion Izquierda-Izquierda'
Procedure RotacionSimpleDerecha(var me: tArbol; PosNodo: tPos);
Var
  NodoArriba,NodoAbajo,NodoAnterior:tNodoIndice;
  RC:tControlArbol;
  PosAux:tPos;
begin
  Seek (me.D,PosNodo); Read (me.D,NodoArriba);
  Seek (me.D,NodoArriba.hi); Read (me.D,NodoAbajo);
  Seek (me.C,0);Read(me.C,RC);
  {Cambio enlaces}
  PosAux:=NodoAbajo.hd;
  NodoAbajo.padre:=NodoArriba.padre;
  NodoAbajo.hd:=PosNodo;
  NodoArriba.padre:=NodoArriba.hi;
  NodoArriba.hi := PosAux; ;


  If NodoAbajo.padre <> Lib_Auxiliar.PosNula then
  begin
    Seek (Me.D,NodoAbajo.padre); Read (Me.D,NodoAnterior);
    If NodoAnterior.hd = PosNodo then
      NodoAnterior.hd :=NodoArriba.padre
    else
```

```
      NodoAnterior.hi :=NodoArriba.padre;
    Seek (Me.D,NodoAbajo.padre); Write (Me.D,NodoAnterior);
  end;

  {Cambio el campo nivel}
  NodoAbajo.nivel:=NodoAbajo.nivel-1;
  NodoArriba.nivel := NodoArriba.nivel+1;

  {Verifico que el del drama no haya sido la raiz}
  If PosNodo=RC.Raiz then
    RC.raiz := NodoArriba.padre;

  Seek (Me.D,PosNodo); Write (Me.D,NodoArriba);
  Seek (Me.D,NodoArriba.padre); Write (Me.D,NodoAbajo);
  Seek (Me.C,0);Write(Me.C,RC);

  {Llamo a Actualizar Nivel para actualizar todos los nodos por debajo del padre del
nodo que estaba desbalanceado
  porque ese nodo sigue teniendo el nivel original}
  ActualizarNiveles_Arbol(ME,PosNodo);
end;
{--------------------------------------------------------------------------------}
//Rotacion en arbol tambien conocido como 'Rotacion Izquierda-Derecha'
Procedure RotacionCompuestaDerecha(var me: tArbol; PosNodo: tPos);
Var
  NodoArriba,NodoMedio,NodoAbajo,NodoAnterior:tNodoIndice;
  RC:tControlArbol;
  PosAux,PosMedio,PosAbajo:tPos;
begin
  Seek (Me.D,PosNodo); Read (Me.D,NodoArriba);
  Seek (Me.D,NodoArriba.hi); Read (Me.D,NodoMedio);
  Seek (Me.D,NodoMedio.hd); Read (Me.D,NodoAbajo);
  Seek (Me.C,0);Read(Me.C,RC);

  PosMedio:=NodoArriba.hi;
```

```
  PosAbajo:=NodoMedio.hd;
  PosAux:=NodoArriba.padre;

  NodoArriba.padre:=NodoMedio.hd;
  NodoArriba.hi:=NodoAbajo.hd;

  NodoMedio.padre:=NodoMedio.hd;
  NodoMedio.hd:= NodoAbajo.hi;

  NodoAbajo.padre:=PosAux;
  NodoAbajo.hd:=PosNodo;
  NodoAbajo.hi:=PosMedio;

  If NodoAbajo.padre <> Lib_Auxiliar.PosNula then
  begin
    Seek (Me.D,NodoAbajo.padre); Read (Me.D,NodoAnterior);
    If NodoAnterior.hd = PosNodo then
      NodoAnterior.hd :=PosAbajo
    else
      NodoAnterior.hi :=PosAbajo;
    Seek (Me.D,NodoAbajo.padre); Write (Me.D,NodoAnterior);
  end;
  {Cambio el campo nivel}
  NodoAbajo.nivel:=NodoAbajo.nivel-2;
  NodoArriba.nivel:= NodoArriba.nivel+1;

  {Verifico que el del drama no haya sido la raiz}
  If PosNodo=RC.Raiz then
    RC.raiz := NodoArriba.padre;

  Seek (Me.D,PosNodo); write (Me.D,NodoArriba);
  Seek (Me.D,PosMedio); write (Me.D,NodoMedio);
  Seek (Me.D,PosAbajo); write (Me.D,NodoAbajo);
  Seek (Me.C,0);Write (Me.C,RC);
```

```
  {Llamo a Actualizar Nivel para actualizar todos los nodos por debajo del padre del
nodo que estaba desbalanceado
  porque ese nodo sigue teniendo el nivel original}
  ActualizarNiveles_Arbol(ME,PosNodo);
end;
{--------------------------------------------------------------------------------}
//Rotacion en arbol tambien conocido como 'Rotacion Derecha-Izquierda'
Procedure RotacionCompuestaIzquierda(var me: tArbol; PosNodo: tPos);
Var
  NodoArriba,NodoMedio,NodoAbajo,NodoAnterior:tNodoIndice;
  RC:tControlArbol;
  PosAux,PosMedio,PosAbajo:tPos;
begin
  Seek (Me.D,PosNodo); Read (Me.D,NodoArriba);
  Seek (Me.D,NodoArriba.hd); Read (Me.D,NodoMedio);
  Seek (Me.D,NodoMedio.hi); Read (Me.D,NodoAbajo);
  Seek (Me.C,0);Read(Me.C,RC);

  PosMedio:=NodoArriba.hd;
  PosAbajo:=NodoMedio.hi;
  PosAux:=NodoArriba.padre;

  NodoArriba.padre:=NodoMedio.hi;
  NodoArriba.hd:=NodoAbajo.hi;

  NodoMedio.padre:=NodoMedio.hi;
  NodoMedio.hi:= NodoAbajo.hd;

  NodoAbajo.padre:=PosAux;
  NodoAbajo.hi:=PosNodo;
  NodoAbajo.hd:=PosMedio;

  If NodoAbajo.padre <> Lib_Auxiliar.PosNula then
  begin
    Seek (Me.D,NodoAbajo.padre); Read (Me.D,NodoAnterior);
```

```
    If NodoAnterior.hd = PosNodo then
      NodoAnterior.hd :=PosAbajo
    Else
      NodoAnterior.hi :=PosAbajo;
    Seek (Me.D,NodoAbajo.padre); Write (Me.D,NodoAnterior);
  end;
  {Cambio el campo nivel}
  NodoAbajo.nivel:=NodoAbajo.nivel-2;
  NodoArriba.nivel:= NodoArriba.nivel+1;
  {Verifico que el del drama no haya sido la raiz}
  If PosNodo=RC.Raiz then
    RC.raiz := NodoArriba.padre;

  Seek (Me.D,PosNodo); write (Me.D,NodoArriba);
  Seek (Me.D,PosMedio); write (Me.D,NodoMedio);
  Seek (Me.D,PosAbajo); write (Me.D,NodoAbajo);
  Seek (Me.C,0);Write (Me.C,RC);

  {Llamo a Actualizar Nivel para actualizar todos los nodos por debajo del padre del
nodo que estaba desbalanceado
  porque ese nodo sigue teniendo el nivel original}
  ActualizarNiveles_Arbol(ME,PosNodo);
end;
{-------------------------------------------------------------------------------}
Function ProfundidadNodo_Arbol(me: tArbol; raiz: tPos): integer;
var
 Profundidad:tProfundidad;
 Nodo:tNodoIndice;
Begin
  //Dada una posición, devuelve la profundidad (nivel) en el que se encuentra
  Profundidad := 0;
  CapturarNodo_Arbol(me, nodo, raiz);
  ProfundidadTotal_Arbol(me, raiz, Profundidad);
  result := (Profundidad - nodo.nivel) + 1;
End;
```

```
{-------------------------------------------------------------------------------}
procedure ProfundidadTotal_Arbol(var me:tArbol; Raiz:tPos; var
Profundidad:tProfundidad);
var
 Nodo: tNodoIndice;
 begin
   If Raiz <> Lib_Auxiliar.PosNula then
    begin
    // Primero recursivo tendiendo a la Izquierda
    ProfundidadTotal_Arbol(me, HijoIzq_Arbol(me, raiz), Profundidad);
    // Recursividad tendiendo a la Derecha.
    ProfundidadTotal_Arbol(me, HijoDer_Arbol(me, raiz), Profundidad);

    // Guardo en Nodo el nodo indice.
    CapturarNodo_Arbol(me, nodo, raiz);

    If Nodo.nivel > Profundidad then
      Profundidad := Nodo.nivel;
    end;
 end;
{-------------------------------------------------------------------------------}
Procedure CasoDesequilibrio_Arbol(me: tArbol; PosNodo: tPos);
var
  Nodo: tNodoIndice;
  begin
    CapturarNodo_Arbol(me, Nodo, PosNodo);

    //Si el factor de equilibrio es positivo, esta cargado a la izquierda..., reviso
con hijo izquierdo
    If FactorEquilibrio_Arbol(me, PosNodo) >= 0 then
    begin
      If FactorEquilibrio_Arbol(me, Nodo.hi) >= 0 then //Si el factor de equilibrio del
hijo izquierdo es positivo, hago una RotacionSimpleDerecha
        RotacionSimpleDerecha(me, PosNodo)
      else  //...si es negativo, hago una RotacionCompuestaDerecha
        RotacionCompuestaDerecha(me, PosNodo);
```

```
    end
    else //Si el factor de equilibrio es negativo, esta cargado a la derecha..., reviso
con hijo derecho
     begin
      If FactorEquilibrio_Arbol(me, Nodo.hd) >= 0 then  //Si el factor de equilibrio
del hijo derecho es positivo, hago una RotacionCompuestaIzquierda
        RotacionCompuestaIzquierda(me, PosNodo)
      else //...si es negativo, hago una RotacionSimpleIzquierda
        RotacionSimpleIzquierda(me, PosNodo);
     end;
  end;
{--------------------------------------------------------------------------------}
Function FactorEquilibrio_Arbol(me: tArbol; PosNodo: tPos): integer;
var
  Nodo: tNodoIndice;
  Ti, Td: integer;
begin
  If PosNodo = Lib_Auxiliar.PosNula then
    FactorEquilibrio_Arbol := 0
  else
    begin
     //Guardo en Nodo el nodo indice.
     CapturarNodo_Arbol(me, Nodo, PosNodo);

      // Calculo la profundidad de ambos hijos.
      If Nodo.hi <> Lib_Auxiliar.PosNula then
        Ti := ProfundidadNodo_Arbol(me, Nodo.hi)
      else
        Ti := 0;

      if Nodo.hd <> Lib_Auxiliar.PosNula then
        Td := ProfundidadNodo_Arbol(me, Nodo.hd)
      else
        Td := 0;

      //cada nodo del arbol equilibrado (balanceado) tiene balance igual a 1, -1 o 0
```

```
      FactorEquilibrio_Arbol := Ti - Td;
    end;
end;
{-------------------------------------------------------------------------------}
Procedure ActualizarNiveles_Arbol(var me: tArbol; Pos:TPos);
var
   Nodo,NodoPadre: tNodoIndice;
   PosPadre:TPos;
   Begin
    //Me paro en la posición para obtener la posición de su padre.
    seek(me.D, Pos);
    Read(me.D, Nodo);
    PosPadre:=Nodo.Padre;
    If PosPadre<>Lib_Auxiliar.Posnula Then
    Begin
    //Me paro en el padre y lo leo para saber su altura, la cual es correcta.
    Seek(me.D,PosPadre);
    Read(me.D,NodoPadre);
    //Al nodo actual le modifico la altura dandole la del padre más uno.
    Nodo.nivel:=NodoPadre.nivel+1;
    End
    ELse
      Nodo.nivel:=1;
    seek(me.D, Pos);
    write(me.D, Nodo);
    //El nodo ya tiene su altura correcta y llamo al procedimiento con la posición de
sus hijos.
    If Nodo.hi<>Lib_Auxiliar.PosNula Then
      ActualizarNiveles_Arbol(me, Nodo.hi);
    If Nodo.hd<>Lib_Auxiliar.PosNula Then
      ActualizarNiveles_Arbol(me, Nodo.hd);
   end;
```

```
{-------------------------------------------------------------------------------}

{

Balancear Arbol:

- Una vez insertado el nodo, pregunto por el balanceo del árbol.

- Se pregunta por el balanceo de cada nodo desde el nodo padre del recién insertado
hasta la raíz

- Un nodo esta balanceado si su factor de equilibrio es -1, 1 o 0.

- El 'factor de equilibrio' verifica que el nodo este equilibrado. Para ello la función
captura el nodo y calcula la profundidad de ambos hijos

El nodo esta equilibrado si la altura del subárbol izquierdo menos la altura del
subárbol derecho difiera a lo sumo en 1.

- Si un nodo esta desequilibrado, se invoca a 'Caso Desquilibrio' donde verifica que
tipo de rotación se llevara a cabo para balancear dicho nodo

}

procedure Balancear_Arbol(var me: tArbol; Pos:tpos);

var

  Nodo: tNodoIndice;

  result:Integer;

  begin

    CapturarNodo_Arbol(me,Nodo,pos);

    pos:= Nodo.Padre;


    while (pos<>Lib_Auxiliar.PosNula) do

    begin

      Result := FactorEquilibrio_Arbol(me, Pos);

      //si result es negativo, entonces nodo cargado a la izquierda

      If Result < 0 then

        Result := Result * (-1);

      //El nodo esta balanceado cuando las alturas de sus dos subarboles (izquierdo y
derecho) difiera¡en a lo sumo en 1

      //Si la diferencia de la altura de los hijo izquierdo y derecho es mayor a 1, el
arbol esta desbalanceado. Invoco a caso de desequilibrio

      If (Result > 1) then

        CasoDesequilibrio_Arbol(me,pos);
```

```
        CapturarNodo_Arbol(me,Nodo,pos);
        pos:= Nodo.Padre;
    end;
  end;
{-------------------------------------------------------------------------------}
function UltFicticio_Arbol(var Me: tArbol): tCantidad;
var
  Rc: tControlArbol;
begin
  Seek(Me.c, 0); Read(Me.c, Rc);
  UltFicticio_Arbol:= Rc.ficticio;
end;
{-------------------------------------------------------------------------------}
function CantidadNodos_Arbol(var Me: tArbol): tCantidad;
var
  Rc: tControlArbol;
begin
  Seek(Me.c, 0); Read(Me.c, Rc);
  CantidadNodos_Arbol:= Rc.Cantidad;
end;
{-------------------------------------------------------------------------------}
function ObtenerNivel_Arbol(var Me:tArbol; clave:tClave):tCantidad;
var
  pos:tPos;
  Nodo: tNodoIndice;
 begin
   if BuscarNodo_Arbol(Me, clave, pos) then
    begin
      seek(me.D,pos); read(me.D,Nodo);
      ObtenerNivel_Arbol:=Nodo.Nivel;
    end;
 end;
END.
```

2.2.2 Árbol Trinario AVL

```
unit LO_ArbolTrinario;

{

Con Hijo Medio se enganchan todas las jugadas (ganadoras) en las que intervino el
jugador, por ello hijo medio es un registro de control que apunta hacia una lista
parcial doble enlace donde se guardan estas jugadas ganadas por el jugador ganador.

Si el jugador ganador no esta en el arbol, se agrega como nodo en el arbol y se
insertar en la lista. Si ya se encuentra en el arbol se inserta solo en la lista y se
actualiza el registro de control

}

interface

Uses
  LO_ListaParcialDobleEnlace, Lib_Auxiliar;


TYPE
      tNodoIndice = record
            clave:tClave;
            padre, hi, hd:tPos;
            //Control de la lista parcial
            HijoMedio: LO_ListaParcialDobleEnlace.TipoRegControl;
            nivel: integer;
          end;


      tControlArbol = record
            Raiz,Borrados:TPos;
            Cantidad, NivelMax: TCantidad;
          end;


      tArchivoNodos=File of tNodoIndice;
      tArchivoControlArbol=File Of tControlArbol;


      tArbolTrinario = record
          D:tArchivoNodos;
          C:tArchivoControlArbol;
        end;
```

```
  procedure CrearMe_ArbolTri (var Me:tArbolTrinario;
                          _NombreArchivoControl,_NombreArchivoIndice:String;
                          _Directorio:String);
  Procedure AbrirMe_ArbolTri(var Me:tArbolTrinario);
  Procedure CerrarMe_ArbolTri(var Me:tArbolTrinario);
  Procedure DestruirMe_ArbolTri(var Me:tArbolTrinario);
  Function Arbol_Vacio(var Me: tArbolTrinario):Boolean;
  Procedure InsertarNodo_ArbolTri(var Me:tArbolTrinario; var
nodo:tNodoIndice;pos:tPos);
  Procedure EliminarNodo_ArbolTri(var Me:tArbolTrinario; pos:tPos);
  Procedure ModificarNodo_ArbolTri(var Me: tArbolTrinario;
RegInd:tNodoIndice;Pos:tPos);
  Function  BuscarNodo_ArbolTri (var Me:tArbolTrinario; clave:tClave;var
pos:tPos):boolean;
  Procedure CapturarNodo_ArbolTri(var Me: tArbolTrinario; var RegInd:tNodoIndice;
Pos:tPos);
  function PosNula_ArbolTri(Var Me:tArbolTrinario):tPos;
  function Raiz_ArbolTri(var Me: tArbolTrinario): tPos;
  Function Padre_ArbolTri(var Me: tArbolTrinario; var Reg: tNodoIndice): tPos;
  function HijoDer_ArbolTri(var Me: tArbolTrinario; pos: tPos): tPos;
  function HijoIzq_ArbolTri(var Me: tArbolTrinario; pos: tPos): tPos;
  {---------------------------(BALANCEO)---------------------------------------}
  Procedure RotacionSimpleIzquierda(var me: tArbolTrinario; PosNodo: tPos);
  Procedure RotacionSimpleDerecha(var me: tArbolTrinario; PosNodo: tPos);
  Procedure RotacionCompuestaDerecha(var me: tArbolTrinario; PosNodo: tPos);
  Procedure RotacionCompuestaIzquierda(var me: tArbolTrinario; PosNodo: tPos);
  Function ProfundidadNodo_ArbolTri(me: tArbolTrinario; raiz: tPos): integer;
  procedure ProfundidadTotal_ArbolTri(var me:tArbolTrinario; Raiz:tPos;  var
Profundidad:tProfundidad);
  Procedure CasoDesequilibrio_ArbolTri(me: tArbolTrinario; PosNodo: tPos);
  Function FactorEquilibrio_ArbolTri(me: tArbolTrinario; PosNodo: tPos): integer;
  Procedure ActualizarNiveles_ArbolTri(var me: tArbolTrinario; Pos:TPos);
  procedure Balancear_ArbolTri(var me: tArbolTrinario; Pos:tpos);
  {------------------------------------------------------------------------------}
  function CantidadNodos_Arbol(var Me: tArbolTrinario): tCantidad;
  function ObtenerNivel_ArbolTri(var Me: tArbolTrinario; clave:tClave):tCantidad;
```

```
VAR
  Me: tArbolTrinario;


implementation


{-------------------------------------------------------------------------------}
procedure CrearMe_ArbolTri (var Me:tArbolTrinario;
                        _NombreArchivoControl,_NombreArchivoIndice:String;
                        _Directorio:String);
Var
  ioC,ioI:integer;
  Rc: tControlArbol;


begin
  assignFile (Me.C, _Directorio + _NombreArchivoControl);
  assignFile (Me.D, _Directorio + _NombreArchivoIndice);
  {$i-}
  reset(Me.C); ioC:=IoResult; //Si IoResult es cero, la operacion es exitosa
  reset(Me.D); ioI:=IoResult;
  if (IoC<>0) or (ioI<>0) then
  begin
    Rewrite(Me.C);
    RC.Raiz:=Lib_Auxiliar.PosNula;
    RC.Borrados:=Lib_Auxiliar.PosNula;
    Rc.Cantidad:= 0;
    Write(Me.C,RC);
    ReWrite(Me.D);
  end;
  Close(Me.D);
  Close(Me.C);
  {$I+}
end;
{-------------------------------------------------------------------------------}
```

```
procedure AbrirMe_ArbolTri(var Me:tArbolTrinario);
begin
  Reset(Me.C);
  Reset(Me.D);
end;
{-------------------------------------------------------------------------------}
procedure CerrarMe_ArbolTri(var Me:tArbolTrinario);
begin
  Close(Me.D);
  Close(Me.C);
end;
{-------------------------------------------------------------------------------}
procedure DestruirMe_ArbolTri(var Me:tArbolTrinario);
begin
  Erase(Me.D);
  Erase(Me.C);
end;
{-------------------------------------------------------------------------------}
Function Arbol_Vacio(var Me: tArbolTrinario):Boolean;
var
  Rc:tControlArbol;
begin
  Seek(Me.C,0); Read(Me.c, Rc);
  Arbol_Vacio:=  (Rc.Raiz = Lib_Auxiliar.PosNula) ;
end;
{-------------------------------------------------------------------------------}
Procedure InsertarNodo_ArbolTri(var Me:tArbolTrinario; var
nodo:tNodoIndice;pos:tPos);//pos, posicion fisica de su padre
var
  posnueva:tPos;
  reg,rd:tNodoIndice;
  RC:tControlArbol;
  NodoMedio: boolean;
begin
  NodoMedio:= false;
```

```
  seek(Me.C,0); read(Me.C,RC);
  if RC.borrados=Lib_Auxiliar.PosNula then //No hay borrados
    posnueva:=filesize(Me.D)
  else //Hay borrados
   begin
     Posnueva:=RC.Borrados;
     seek(Me.D,posnueva); read(Me.D,rD);
     RC.Borrados:=rD.hi;
   end;

   if (RC.Raiz=Lib_Auxiliar.PosNula) then//Inserto En árbol vacio
   begin
     nodo.padre:=Lib_Auxiliar.PosNula;
     nodo.hi:=Lib_Auxiliar.PosNula;
     nodo.hd:=Lib_Auxiliar.PosNula;
     nodo.nivel:=1;
     RC.Raiz:=posnueva;
   end
   //Grabamos al final
   else //Inserto como hoja
     begin
      seek(Me.D,pos); read(Me.D,reg); //leo el registro del padre
      nodo.padre:= pos;
      nodo.hd:= Lib_Auxiliar.PosNula;
      nodo.hi:= Lib_Auxiliar.PosNula;
      nodo.nivel:=Reg.Nivel+1;

      //Si el nodo a insertar es la misma que la de su padre, entonces significa que
voy a insertar al medio
      if nodo.Clave = reg.Clave then
        NodoMedio:= true;

      if (nodo.clave<=reg.clave) then
        reg.hi:=posnueva
      else
```

```
        reg.hd:=posnueva;


       seek(Me.d,pos); write(Me.d,reg);
      end;


    if NodoMedio then //actualizo las poosiciones del nodo que se va a insertar al
medio
    begin
      posNueva:= pos;
      nodo.Hi:= reg.Hi;
      nodo.Hd:= reg.Hd;
      nodo.Padre:= reg.Padre;
      nodo.Nivel:= reg.Nivel;
      rc.Cantidad:= rc.Cantidad-1;
    end
    else  //Si no inserto al medio, entonces inicializo todas las posiciones
      begin
        LO_ListaParcialDobleEnlace.InicializarCabecera(nodo.HijoMedio);
        nodo.Hi:=Lib_Auxiliar.PosNula;
        nodo.Hd:=Lib_Auxiliar.PosNula;
      end;


   //Si la profuncididad del nodo es mayor a la prfundidad en el control actualizo.
   If nodo.nivel > RC.NivelMax then
     RC.NivelMax := nodo.nivel;
   rc.cantidad:=rc.cantidad+1;
   seek(Me.d,posnueva); write(Me.d,nodo); //se escribe el elemento en el archivo
   seek(Me.c,0); write(Me.c,rc);    // se actualiza el registro control


    //pregunto por el balanceo del árbol
   Balancear_ArbolTri(me, PosNueva);
end;
{--------------------------------------------------------------------------------}
```

```
Procedure EliminarNodo_ArbolTri(var Me:tArbolTrinario; pos:tPos);
var
  RD:tNodoIndice;
  RC:tControlArbol;
  rAux,RHD,RDP,RHI:tNodoIndice;
  posAux:tPos;
begin
  seek(Me.c,0); read(Me.c,rc);
  seek(Me.d,pos); read(Me.d,rd);   //Capturo el registro a eliminar

  //no tiene padre, no tiene hijos
  if (pos=rc.raiz) and (rd.hi=Lib_Auxiliar.PosNula) and (rd.hd=Lib_Auxiliar.PosNula)
then
    Rc.Raiz:=Lib_Auxiliar.PosNula
  else
    begin
      if (rd.hi=Lib_Auxiliar.PosNula) and (rd.hd=Lib_Auxiliar.PosNula) then
      //Tiene padre, no tiene hijos. Eliminar de una hoja
      begin
        Seek(Me.D,rd.padre);read(Me.d,rdp);
        If rdp.hi=pos then
          rdp.hi:=Lib_Auxiliar.PosNula
        else
          rdp.hd:=Lib_Auxiliar.PosNula;

        Seek(Me.d,rd.padre);write(Me.d,RDP);
      end
      else
      begin
        if (rd.hi <> Lib_Auxiliar.PosNula) and (rd.hd <> Lib_Auxiliar.PosNula) then
//Caso general, tiene padre y tiene hijos...
        begin
          If pos<>RC.raiz then //...no es raiz
          begin
            Seek(Me.d,rd.padre);read(Me.d,rdp);
            If rdp.hi=pos then
```

```
            rdp.hi:=rd.hd
          else
            rdp.hd:=rd.hd;

        Seek(Me.d,RD.padre);Write(Me.d,rdp);
      end
      else
        RC.raiz:=rd.hd;

      Seek(Me.d,rd.hd);read(Me.d,RHD);
      RHD.padre:=RD.padre;
      posAux:=RD.hd;
      rAux:=RHD;

      While rAux.hi<>Lib_Auxiliar.PosNula do
      begin
        Seek(Me.d,posAux);read(Me.d,RAux);
        If raux.hi<>Lib_Auxiliar.PosNula then
          posaux:=rAux.hi;
      end;

      Seek(Me.d,RD.hi);Read(Me.d,RHI);
      RHI.padre:=posAux;
      RAux.hi:=RD.hi;
      Seek(Me.d,RD.hi);Write(Me.d,RHI);
      Seek(Me.d,posAux);Write(Me.d,raux);
      If posAux<>RD.hd then
      begin
       Seek(Me.d,RD.hd);Write(Me.d,RHD);
      end;
    end
    else
      begin
        If pos<>RC.Raiz then
        begin
```

```
      Seek(Me.D,RD.padre);Read(Me.D,RDP);
      If RDP.hi=pos then
      begin
        If RD.hi=Lib_Auxiliar.PosNula then
        begin
          Rdp.hi:=RD.hd;
          Seek(Me.D,RD.hd);read(Me.d,RHD);
          RHD.padre:=RD.padre;
          Seek(Me.d,RD.hd);write(Me.d,RHD);
        end
        else
          begin
            Rdp.hi:=RD.hi;
            Seek(Me.D,RD.hi);read(Me.d,RHI);
            RHI.padre:=RD.padre;
            Seek(Me.d,RD.hi);write(Me.d,RHI);
          end
    end
    else
      If RD.hi=Lib_Auxiliar.PosNula then
       begin
          Rdp.hd:=RD.hd;
          Seek(Me.D,RD.hd);read(Me.d,RHD);
          RHD.padre:=RD.padre;
          Seek(Me.d,RD.hd);write(Me.d,RHD);
        end
        else
         begin
          Rdp.hd:=RD.hi;
          Seek(Me.D,RD.hi);read(Me.d,RHI);
          RHI.padre:=RD.padre;
          Seek(Me.d,RD.hi);write(Me.d,RHI);
         end;
  Seek(Me.d,rd.padre);write(Me.d,rdp);
end
```

```
        else
          If RD.hi=Lib_Auxiliar.PosNula then
            RC.Raiz:=RD.hd
          else
            RC.raiz:=RD.hi;
                  end;
            end;
      end;
  RD.hi:=RC.borrados;
  RD.hd:=RC.borrados;
      RC.borrados:=pos;
  RC.Cantidad:=RC.Cantidad-1;
  Seek(Me.c,0);write(Me.c,rc); //Grabo la cabecera de control
end;
{-------------------------------------------------------------------------------}
Procedure ModificarNodo_ArbolTri(var Me: tArbolTrinario; RegInd:tNodoIndice;Pos:tPos);
begin
  seek(Me.D,pos); write(Me.D,RegInd);
end;
{-------------------------------------------------------------------------------}
Function  BuscarNodo_ArbolTri (var Me:tArbolTrinario; clave:tClave;var
pos:tPos):boolean; //Buscar iterativo
var
  reg:tNodoIndice;
  RC:tControlArbol;
  encont:boolean;
  posPadre:tpos;
begin
  seek(Me.c,0); read(Me.c,rc);
  pos:=rc.raiz; encont:=false;
  posPadre:=Lib_Auxiliar.PosNula;
  while (not encont) and (pos<>Lib_Auxiliar.PosNula) do
  begin
    seek(Me.d,pos); read(Me.d,reg);
    if (clave=reg.clave) then
```

```
    begin
      encont:=true;
    end
    else
      begin
        if (reg.clave>clave) then
        begin
          posPadre:=pos;
          pos:=reg.hi;
        end
        else
         begin
           posPadre:=pos;
           pos:=reg.hd;
         end;
      end;
   end;
   if (not encont) then
   begin
     pos:=posPadre;
   end;
   BuscarNodo_ArbolTri:=encont;
end;
{-------------------------------------------------------------------------------------}
Procedure CapturarNodo_ArbolTri(var Me: tArbolTrinario; var RegInd:tNodoIndice;
Pos:tPos);
begin
  seek(Me.D,Pos); Read(Me.D,RegInd);
end;
{-------------------------------------------------------------------------------------}
function PosNula_ArbolTri(Var Me:tArbolTrinario):tPos;
begin
  PosNula_ArbolTri:= Lib_Auxiliar.PosNula;
end;
```

```
{-------------------------------------------------------------------------------}
function Raiz_ArbolTri(var Me: tArbolTrinario): tPos;
var
  Rc: tControlArbol;
begin
  Seek(Me.c, 0); Read(Me.c, Rc);
  Raiz_ArbolTri:= Rc.Raiz;
end;
{-------------------------------------------------------------------------------}
Function Padre_ArbolTri(var Me: tArbolTrinario; var Reg: tNodoIndice): tPos;
begin
 Padre_ArbolTri:= Reg.padre;
end;
{-------------------------------------------------------------------------------}
function HijoDer_ArbolTri(var Me: tArbolTrinario; pos: tPos): tPos;
var
  reg: tNodoIndice;
begin
  seek(me.D,pos); read(me.D,reg);
  HijoDer_ArbolTri:= reg.hd;
end;
{-------------------------------------------------------------------------------}
function HijoIzq_ArbolTri(var Me: tArbolTrinario; pos: tPos): tPos;
var
  reg: tNodoIndice;
begin
  seek(me.D,pos); read(me.D,reg);
  HijoIzq_ArbolTri:= reg.hi;
end;
{-------------------------------------------------------------------------------}
//Rotacion en arbol tambien conocido como 'Rotacion Derecha-Derecha'
Procedure RotacionSimpleIzquierda (var me: tArbolTrinario; PosNodo: tPos);
Var
  NodoArriba,NodoAbajo,NodoAnterior:tNodoIndice;
  RC:tControlArbol;
```

```
  PosAux:tPos;
begin
  Seek (me.D,PosNodo); Read (me.D,NodoArriba);
  Seek (me.D,NodoArriba.hd); Read (me.D,NodoAbajo);
  Seek (me.C,0);Read(me.C,RC);
  //Cambio enlaces
  PosAux:=NodoAbajo.hi;
  NodoAbajo.padre:=NodoArriba.padre;
  NodoAbajo.hi:=PosNodo;
  NodoArriba.padre:=NodoArriba.hd;
  NodoArriba.hd := PosAux; ;

  //Si el desbalanceado no era raíz
  If NodoAbajo.padre <> Lib_Auxiliar.PosNula then
  begin
    //Me paro en el padre del hijo desbalanceado, y le pongo como hijo al que era su
nieto
    Seek (Me.D,NodoAbajo.padre); Read (Me.D,NodoAnterior);
    If NodoAnterior.hd = PosNodo then
      NodoAnterior.hd :=NodoArriba.padre
    else
      NodoAnterior.hi :=NodoArriba.padre;
    seek (Me.D,NodoAbajo.padre); Write (Me.D,NodoAnterior);
  end;

  //Cambio niveles
  NodoAbajo.nivel:=NodoAbajo.nivel-1;
  NodoArriba.nivel := NodoArriba.nivel+1;

  //Verifico que el del problema no haya sido la raiz
  //Anteriormeente a NodoArriba.padre le da NodoAbajo.hdecho. Por lo tanto, pone como
nueva raiz
  //al hijo derecho del que estaba desbalanceado.
  If PosNodo=RC.Raiz then
    RC.raiz := NodoArriba.padre;
```

```
  Seek (Me.D,PosNodo); Write (Me.D,NodoArriba);
  Seek (Me.D,NodoArriba.padre); Write (Me.D,NodoAbajo);
  Seek (Me.C,0);Write(Me.C,RC);

  {Llamo a Actualizar Nivel para actualizar todos los nodos por debajo del padre del
nodo que estaba desbalanceado
  porque ese nodo sigue teniendo el nivel original}
  ActualizarNiveles_ArbolTri(ME,PosNodo);
end;
{--------------------------------------------------------------------------------}
//Rotacion en arbol tambien conocido como 'Rotacion Izquierda-Izquierda'
Procedure RotacionSimpleDerecha(var me: tArbolTrinario; PosNodo: tPos);
var
  NodoArriba,NodoAbajo,NodoAnterior:tNodoIndice;
  RC:tControlArbol;
  PosAux:tPos;
begin
  Seek (me.D,PosNodo); Read (me.D,NodoArriba);
  Seek (me.D,NodoArriba.hi); Read (me.D,NodoAbajo);
  Seek (me.C,0);Read(me.C,RC);

  {Cambio enlaces}
  PosAux:=NodoAbajo.hd;
  NodoAbajo.padre:=NodoArriba.padre;
  NodoAbajo.hd:=PosNodo;
  NodoArriba.padre:=NodoArriba.hi;
  NodoArriba.hi := PosAux; ;

  If NodoAbajo.padre <> Lib_Auxiliar.PosNula then
  begin
    Seek (Me.D,NodoAbajo.padre); Read (Me.D,NodoAnterior);
    If NodoAnterior.hd = PosNodo then
      NodoAnterior.hd :=NodoArriba.padre
    else
      NodoAnterior.hi :=NodoArriba.padre;
```

```
    Seek (Me.D,NodoAbajo.padre); Write (Me.D,NodoAnterior);
  end;

  {Cambio el campo nivel}
  NodoAbajo.nivel:= NodoAbajo.nivel-1;
  NodoArriba.nivel := NodoArriba.nivel+1;

  {Verifico que el del drama no haya sido la raiz}
  If PosNodo=RC.Raiz then
    RC.raiz := NodoArriba.padre;

  Seek (Me.D,PosNodo); Write (Me.D,NodoArriba);
  Seek (Me.D,NodoArriba.padre); Write (Me.D,NodoAbajo);
  Seek (Me.C,0);Write(Me.C,RC);

  {Llamo a Actualizar Nivel para actualizar todos los nodos por debajo del padre del
nodo que estaba desbalanceado
  porque ese nodo sigue teniendo el nivel original}
  ActualizarNiveles_ArbolTri(ME,PosNodo);
end;
{--------------------------------------------------------------------------------}
//Rotacion en arbol tambien conocido como 'Rotacion Izquierda-Derecha'
Procedure RotacionCompuestaDerecha(var me: tArbolTrinario; PosNodo: tPos);
Var
  NodoArriba,NodoMedio,NodoAbajo,NodoAnterior:tNodoIndice;
  RC:tControlArbol;
  PosAux,PosMedio,PosAbajo:tPos;
begin
  Seek (Me.D,PosNodo); Read (Me.D,NodoArriba);
  Seek (Me.D,NodoArriba.hi); Read (Me.D,NodoMedio);
  Seek (Me.D,NodoMedio.hd); Read (Me.D,NodoAbajo);
  Seek (Me.C,0);Read(Me.C,RC);

  PosMedio:=NodoArriba.hi;
  PosAbajo:=NodoMedio.hd;
```

```
  PosAux:=NodoArriba.padre;

  NodoArriba.padre:=NodoMedio.hd ;
  NodoArriba.hi:=NodoAbajo.hd;

  NodoMedio.padre:=NodoMedio.hd;
  NodoMedio.hd:= NodoAbajo.hi;

  NodoAbajo.padre:=PosAux;
  NodoAbajo.hd:=PosNodo;
  NodoAbajo.hi:=PosMedio;

  If NodoAbajo.padre <> Lib_Auxiliar.PosNula then
  begin
    Seek (Me.D,NodoAbajo.padre); Read (Me.D,NodoAnterior);
    If NodoAnterior.hd = PosNodo then
      NodoAnterior.hd :=PosAbajo
    else
      NodoAnterior.hi :=PosAbajo;
    Seek(Me.D,NodoAbajo.padre); Write (Me.D,NodoAnterior);
  end;
  {Cambio el campo nivel}
  NodoAbajo.nivel:=NodoAbajo.nivel-2;
  NodoArriba.nivel:= NodoArriba.nivel+1;

  {Verifico que el del drama no haya sido la raiz}
  If PosNodo=RC.Raiz then
    RC.raiz := NodoArriba.padre;

  Seek (Me.D,PosNodo); write (Me.D,NodoArriba);
  Seek (Me.D,PosMedio); write (Me.D,NodoMedio);
  Seek (Me.D,PosAbajo); write (Me.D,NodoAbajo);
  Seek (Me.C,0); Write(Me.C,RC);
```

```
  {Llamo a Actualizar Nivel para actualizar todos los nodos por debajo del padre del
nodo que estaba desbalanceado

  porque ese nodo sigue teniendo el nivel original}

  ActualizarNiveles_ArbolTri(ME,PosNodo);

end;

{-------------------------------------------------------------------------------}

//Rotacion en arbol tambien conocido como 'Rotacion Derecha-Izquierda'

Procedure RotacionCompuestaIzquierda(var me: tArbolTrinario; PosNodo: tPos);

Var

  NodoArriba,NodoMedio,NodoAbajo,NodoAnterior:tNodoIndice;

  RC:tControlArbol;

  PosAux,PosMedio,PosAbajo:tPos;

begin

  Seek (Me.D,PosNodo); Read (Me.D,NodoArriba);

  Seek (Me.D,NodoArriba.hd); Read (Me.D,NodoMedio);

  Seek (Me.D,NodoMedio.hi); Read (Me.D,NodoAbajo);

  Seek (Me.C,0);Read(Me.C,RC);


  PosMedio:=NodoArriba.hd;

  PosAbajo:=NodoMedio.hi;

  PosAux:=NodoArriba.padre;


  NodoArriba.padre:=NodoMedio.hi;

  NodoArriba.hd:=NodoAbajo.hi;


  NodoMedio.padre:=NodoMedio.hi;

  NodoMedio.hi:= NodoAbajo.hd;


  NodoAbajo.padre:=PosAux;

  NodoAbajo.hi:=PosNodo;

  NodoAbajo.hd:=PosMedio;


  If NodoAbajo.padre <> Lib_Auxiliar.PosNula then

  begin

    Seek (Me.D,NodoAbajo.padre); Read (Me.D,NodoAnterior);
```

```
    If NodoAnterior.hd = PosNodo then
      NodoAnterior.hd :=PosAbajo
    else
      NodoAnterior.hi :=PosAbajo;
    Seek (Me.D,NodoAbajo.padre); Write (Me.D,NodoAnterior);
  end;
  {Cambio el campo nivel}
  NodoAbajo.nivel:=NodoAbajo.nivel-2;
  NodoArriba.nivel:= NodoArriba.nivel+1;

  {Verifico que el del drama no haya sido la raiz}
  If PosNodo=RC.Raiz then
    RC.raiz := NodoArriba.padre;

  Seek (Me.D,PosNodo); write (Me.D,NodoArriba);
  Seek (Me.D,PosMedio); write (Me.D,NodoMedio);
  Seek (Me.D,PosAbajo); write (Me.D,NodoAbajo);
  Seek (Me.C,0);Write (Me.C,RC);

  {Llamo a Actualizar Nivel para actualizar todos los nodos por debajo del padre del
nodo que estaba desbalanceado
  porque ese nodo sigue teniendo el nivel original}
  ActualizarNiveles_ArbolTri(ME,PosNodo);
end;
{--------------------------------------------------------------------------------}
Function ProfundidadNodo_ArbolTri(me: tArbolTrinario; raiz: tPos): integer;
var
 Profundidad:tProfundidad;
 Nodo:tNodoIndice;
Begin
  //Dada una posición, devuelve la profundidad (nivel) en el que se encuentra}
  Profundidad := 0;
  CapturarNodo_ArbolTri(me, nodo, raiz);
  ProfundidadTotal_ArbolTri(me, raiz, Profundidad);
  result:= (Profundidad - nodo.nivel) + 1;
```

```
End;
{-------------------------------------------------------------------------------}
procedure ProfundidadTotal_ArbolTri(var me:tArbolTrinario; Raiz:tPos;
  var Profundidad:tProfundidad);
var
 Nodo: tNodoIndice;
 begin
   If Raiz <> Lib_Auxiliar.PosNula then
    begin
    // Primero recursivo tendiendo a la Izquierda
    ProfundidadTotal_ArbolTri(me, HijoIzq_ArbolTri(me, raiz), Profundidad);
    // Recursividad tendiendo a la Derecha.
    ProfundidadTotal_ArbolTri(me, HijoDer_ArbolTri(me, raiz), Profundidad);

    // Guardo en Nodo el nodo indice.
    CapturarNodo_ArbolTri(me, nodo, raiz);

    If Nodo.nivel > Profundidad then
      Profundidad := Nodo.nivel;
    end;
 end;
{-------------------------------------------------------------------------------}
Procedure CasoDesequilibrio_ArbolTri(me: tArbolTrinario; PosNodo: tPos);
var
  Nodo: tNodoIndice;
  begin
    CapturarNodo_ArbolTri(me, Nodo, PosNodo);

    If FactorEquilibrio_ArbolTri(me, PosNodo) >= 0 then
   //Si el factor de equilibrio es positivo, esta cargado a la izquierda..., reviso con
hijo izquierda
    begin
      If FactorEquilibrio_ArbolTri(me, Nodo.hi) >= 0 then //Si el factor de equilibrio
del hijo izq es positivo, hago una RotacionSimpleDerecha
        RotacionSimpleDerecha(me, PosNodo)
      else  //...si es negativo, hago una RotacionCompuestaDerecha
```

```
        RotacionCompuestaDerecha(me, PosNodo);
    end
    else
     begin
      If FactorEquilibrio_ArbolTri(me, Nodo.hd) >= 0 then  //Si el factor de equilibrio
del hijo der es positivo, hago una RotacionCompuestaIzquierda
       RotacionCompuestaIzquierda(me, PosNodo)
        else //...si es negativo, hago una RotacionSimpleIzquierda
         RotacionSimpleIzquierda(me, PosNodo);
     end;
  end;
{--------------------------------------------------------------------------------}
Function FactorEquilibrio_ArbolTri(me: tArbolTrinario; PosNodo: tPos): integer;
var
  Nodo: tNodoIndice;
  Ti, Td: integer;
begin
  If PosNodo = Lib_Auxiliar.PosNula then
    FactorEquilibrio_ArbolTri := 0
  Else
  begin
    // Guardo en Nodo el nodo indice.
   CapturarNodo_ArbolTri(me, Nodo, PosNodo);

    // Calculo la profundidad de ambos hijos.
    If Nodo.hi <> Lib_Auxiliar.PosNula then
      Ti := ProfundidadNodo_ArbolTri(me, Nodo.hi)
    Else
      Ti := 0;
    If Nodo.hd <> Lib_Auxiliar.PosNula then
      Td := ProfundidadNodo_ArbolTri(me, Nodo.hd)
    Else
      Td := 0;

    {Factor_Equilibrio = La altura del subarbol derecho menos la altura del subarbol
izquierdo para ese nodo
```

```
    dabe ser <=1 para estar equilibrado}
    FactorEquilibrio_ArbolTri := Ti - Td;
  end;
end;
{-------------------------------------------------------------------------------}
Procedure ActualizarNiveles_ArbolTri(var me: tArbolTrinario; Pos:TPos);
var
  Nodo,NodoPadre: tNodoIndice;
  PosPadre:TPos;
Begin
  //Me paro en la posición para obtener la posición de su padre.
  seek(me.D, Pos); Read(me.D, Nodo);
  PosPadre:=Nodo.Padre;
  If PosPadre<>Lib_Auxiliar.Posnula Then
  begin
  //Me paro en el padre y lo leo para saber su altura, la cual es correcta.
  Seek(me.D,PosPadre); Read(me.D,NodoPadre);
  //Al nodo actual le modifico la altura dandole la del padre más uno.
  //RI.Altura:=RIPadre.Altura+1;
  Nodo.nivel:=NodoPadre.nivel+1;
  end
  eLse
    Nodo.nivel:=1;

  seek(me.D, Pos); write(me.D, Nodo);
  //El nodo ya tiene su altura correcta y llamo al procedimiento con la posición de sus
hijos.
  If Nodo.hi<>Lib_Auxiliar.PosNula Then
    ActualizarNiveles_ArbolTri(me, Nodo.hi);
  If Nodo.hd<>Lib_Auxiliar.PosNula Then
    ActualizarNiveles_ArbolTri(me, Nodo.hd);
End;
{-------------------------------------------------------------------------------}
```

```
procedure Balancear_ArbolTri(var me: tArbolTrinario; Pos:tpos);
var
  Nodo: tNodoIndice;
  result:Integer;

begin
  CapturarNodo_ArbolTri(me,Nodo,pos);
  pos:=Nodo.Padre;

  while (pos<>Lib_Auxiliar.PosNula) do
  Begin
    Result := FactorEquilibrio_ArbolTri(me, Pos);
    If Result < 0 then
      Result := Result * (-1);

    If (Result > 1) then
      CasoDesequilibrio_ArbolTri(me,pos);

     CapturarNodo_ArbolTri(me,Nodo,pos);
     pos:=Nodo.Padre;
   End;
end;
{-------------------------------------------------------------------------------}
function CantidadNodos_Arbol(var Me: tArbolTrinario): tCantidad;
var
  Rc: tControlArbol;
begin
  Seek(Me.c, 0); Read(Me.c, Rc);
  CantidadNodos_Arbol:= Rc.Cantidad;
end;
{-------------------------------------------------------------------------------}
function ObtenerNivel_ArbolTri(var Me: tArbolTrinario; clave:tClave):tCantidad;
var
  pos:tPos;
  Nodo: tNodoIndice;
```

```
begin
 if BuscarNodo_ArbolTri(Me, clave, pos) then
  begin
    seek(me.D,pos); read(me.D,Nodo);
    ObtenerNivel_ArbolTri:= Nodo.Nivel;
  end;
end;


END.
```

2.2.3 Colas

```
unit LO_Colas;


{
Estructura lineal de tipo FIFO (First In First Out), es decir,
el primero en entrar a la cola, también el primero en salir.
Este TDA utiliza la libreria Libreria Operacional de "Simple Enlace", donde:
  - Se inserta siempre al final: Por eso Encolar le parametriza a la lista simple el
ultimo elemento
  - Se elimina siempre el primer elemento
  - Se captura siempre al frente: Por eso Frente captura de la lista el primer elemento
}


interface
  uses

  LO_ListaSimpleEnlace, Lib_Auxiliar;


type

 TipoRegDatos = LO_ListaSimpleEnlace.TipoRegDatos;
 TipoRegControl = LO_ListaSimpleEnlace.TipoRegControl;

 tCola = LO_ListaSimpleEnlace.tLsEnlace;
```

```
procedure CrearCola(var Me:tCola;NombreControl:string;NombreDatos:string; Ruta:String);
procedure AbrirCola(var Me:tCola);
procedure CerrarCola(var Me:tCola);
procedure DestruirCola(var ME:tCola);
function  ColaVacia(var Me:tCola):boolean;
procedure Frente(var Me:tCola; var reg:tiporegdatos);
procedure Decolar(var me:tCola);
procedure Encolar(var me:tCola; reg:tiporegdatos);
procedure InicializarRegBandera(var Reg:tiporegDatos);
function ClaveNula(var me:tCola): TClave;
function Buscar(var Me:tCola; Clave:TClave):boolean;

implementation

{-------------------------------------------------------------------------------}
procedure CrearCola(var Me:tCola;NombreControl:string;NombreDatos:string; Ruta:String);
begin
 //Creo estructura cola con el metodo de crear de la lista simple
 LO_ListaSimpleEnlace.CrearListaSimple(me,NombreControl,NombreDatos,ruta);
end;
{-------------------------------------------------------------------------------}
procedure AbrirCola(var Me:tCola);
begin
//Abro la cola invocando al procedimiento abrir de la lista simple
 LO_ListaSimpleEnlace.AbrirListaSimple(me);
end;
{-------------------------------------------------------------------------------}
procedure CerrarCola(var Me:tCola);
begin
//Cierro la cola invocando al procedimiento cerrar de la lista simple
  LO_ListaSimpleEnlace.CerrarListaSimple(me);
end;
{-------------------------------------------------------------------------------}
```

```
procedure DestruirCola(var ME:tCola);
begin
//Destruyo la cola invocando al procedimiento destruir de la lista simple
  LO_ListaSimpleEnlace.DestruirListaSimple(me);
end;
{-------------------------------------------------------------------------------}
function  ColaVacia(var Me:tCola):boolean;
 begin
 //Consulto si la cola esta vacia. Devuelve verdadero si la cola esta vacia
  ColaVacia:= LO_ListaSimpleEnlace.ListaSimpleVacia(me);
 end;
{-------------------------------------------------------------------------------}
procedure Frente(var Me:tCola; var reg:tiporegdatos);
 begin
 //Con Frente capturo siempre el primer elemento de la cola, por eso le paso la
posicion del primer elemento de la lista
  LO_ListaSimpleEnlace.CapturarInfo_LSimple(me,LO_ListaSimpleEnlace.Primero(me),reg);
 end;
{-------------------------------------------------------------------------------}
 procedure Decolar(var me:tCola);
 begin
 //Al decolar, le paso por parametrizo la posicion nula de la lista (-1) para que
elimine el primer elemento
  LO_ListaSimpleEnlace.EliminarInfo_LSimple(me,LO_ListaSimpleEnlace.Posnula(me));
 end;
{-------------------------------------------------------------------------------}
procedure Encolar(var me:tCola; reg:tiporegdatos);
 begin
  //Al encolar se inserta siempre al final, por eso le paso la ultima posicion de la
lista
  LO_ListaSimpleEnlace.InsertarInfo_LSimple(me,reg,LO_ListaSimpleEnlace.Ultimo(me));
 end;
{-------------------------------------------------------------------------------}
procedure InicializarRegBandera(var Reg:tiporegDatos);
 begin
  //Creo un registro nulo que servira de 'Bandera' para poder recorrer la cola
```

```
  Reg.Clave:=Lib_Auxiliar.ClaveNula;
  Reg.Siguiente:=Lib_Auxiliar.PosNula;
  Reg.PosEnDatos:= Lib_Auxiliar.PosNula;
 end;
{-------------------------------------------------------------------------------}
 function ClaveNula(var me:tCola): TClave;
 begin
 //Devuelve la clave nula de la estructura de lista simple
   claveNula:= LO_ListaSimpleEnlace.ClaveNula(ME);
 end;
{-------------------------------------------------------------------------------}
function Buscar(var Me:tCola; Clave:TClave):boolean;
var
 regBandera,reg:tiporegdatos;
 corte,enc:boolean;
begin
//Se encola un registro bandera para cortar recorrido cuando en el ciclo se topa con el
mismo. Se encontro clave si enc = true

 corte:=false;
 enc:=false;

 //Inicializo registro bandera
 InicializarRegBandera(regBandera);
 //Encolo el registro bandera
 Encolar(me,regBandera);

 while not corte do
  begin
    Frente(me,reg);
    Decolar(me);

    if clave = reg.Clave then
      enc:=true;
```

```
    if reg.Clave <> Lib_Auxiliar.ClaveNula then
      Encolar(me,reg)
    else
      corte:=true;
  end;

  Buscar:= enc;
end;

END.
```

2.2.4 Hash

```
unit LO_Hash;

{
'Tabla hash' contiene un archivo secuencial y fijo, donde se guardara en cada registro un registro control que apuntara a la lista de Cuentas Corrientes.
'Tabla hash' guarda en cada posicion el registro control de todas las cuentas corrientes de aquellos jugadores que coinciden en la misma posicion de la tabla hash, posicion dada por la funcion Hash
La funcion hash toma los 6 primeros caracteres del nick y los pasa a  mayuscula.
Se lleva a cabo la conversion de cada caracter a su ordinal.
Obtiene la suma de los tres primeros ordinales para luego restarlos con los proximos tres ordinales
El resultado de la funcion hash va a ir de un rango de minima posicion y maxima posicion declaradas como constantes
Minima posicion = 0
Maxima posicion = 126
}

interface

uses

Type_Hash, LO_ListaParcialDobleEnlace, Lib_Auxiliar, SysUtils,Math;
```

```
type

  tHash = file of LO_ListaParcialDobleEnlace.TipoRegControl; //Primero, Ultimo, Borrado


  Procedure Crear (var Me:tHash;Archivo: string; ruta:string);
  procedure Abrir (var Me: tHash);
  procedure Cerrar (var Me: tHash);
  procedure Destruir (var Me: tHash);
  procedure insertar(var me:tHash; reg:TipoRegControl;pos:tpos);
  procedure modificar(var me:tHash; reg:TipoRegControl;pos:tpos);
  procedure Capturar(var Me:tHash;pos:tPos; var Reg:TipoRegControl);
  function FuncionHash(Clave:tClaveHash): tPos;


implementation


{--------------------------------------------------------------------------------}
//Creo estructura Hash
Procedure Crear (var Me:tHash;Archivo: string; ruta:string);
var
  RegHash:TipoRegControl;
  FaltaDatos:Boolean;
  i:tPos;
begin
  AssignFile (me,ruta+Archivo);
   {$I-}
  Reset(me); FaltaDatos:=(ioResult<>0);
  If (FaltaDatos) then
  begin
    Rewrite(me);
    //Inicializo registro Hash para insertar en la estructura secuencial y fija
    //desde la mínima posicion hasta la máxima
    LO_ListaParcialDobleEnlace.InicializarCabecera(RegHash);


    For i:=_MinPosHash to _MaxPosHash do
    Begin
```

```
     seek(me,i);
     write(me,RegHash);
    end;
    Close(me);
  end;
  {$i+}
end;
{-------------------------------------------------------------------------}
function FuncionHash(Clave:tClaveHash): tPos;
var
  ValorCaracter, Resultado,i:integer;
  ClaveMayuscula: string;
  Begin
    //nick pasado a  mayuscula
    ClaveMayuscula:= UpperCase(clave);
    Resultado:=0;
    //recorrido 6 veces. Desde el primer caracter hasta el sexto
    for i:=1 to 6 do
    begin
    {excepcion: al ordinal de la letra Ñ se altera su valor para que no sea 241 y sea
91. De esta manera esta dentro de los rangos de minima y maxima posicion en tabla hash
previamente declaradas como constantes}
       if (ord(ClaveMayuscula[i])=241) then
        ValorCaracter:=91
       else
        ValorCaracter:=ord(ClaveMayuscula[i]);
      {suma de los tres primeros ordinales para luego restarlos con los proximos tres
ordinales}
      if (i<=3) then
       Resultado:=Resultado + ValorCaracter
       else
       Resultado:=Resultado - ValorCaracter;
    end;
    if Resultado < 0 then     //Si el resultado es negativo lo paso a positivo
     Resultado:= Resultado * (-1);
    FuncionHash:= Resultado;
```

```
  End;
{--------------------------------------------------------------------------------}
procedure Abrir (var Me: tHash);
begin
  reset(me);
end;
{--------------------------------------------------------------------------------}
procedure Cerrar (var Me: tHash);
begin
  close(me);
end;
{--------------------------------------------------------------------------------}
procedure Destruir (var Me: tHash);
begin
  erase(me);
end;
{--------------------------------------------------------------------------------}
 procedure insertar(var me:tHash; reg:TipoRegControl;pos:tpos);
 begin
   Seek(me,pos); write(me,reg);
 end;
{--------------------------------------------------------------------------------}
 procedure Capturar(var Me:tHash; pos:tPos; var Reg:TipoRegControl);
 begin
   Seek(me,pos); read(me,Reg);
 end;
{--------------------------------------------------------------------------------}
procedure modificar(var me:tHash; reg:TipoRegControl; pos:tpos);
begin
  Seek(Me,Pos); Write(Me,Reg);
end;
END.
```

2.2.4.1 Type Hash

```
unit Type_Hash;


Interface


Uses
  Lib_Auxiliar;


const
  _MinPosHash = 0;
  _MaxPosHash = 126; //Segun la funcion hash a utilizar que da maximo 129


type
 tClaveHash = String [6];
 tPosHash= _MinPosHash.._MaxPosHash;


implementation


END.
```

2.2.5 Lista Simple Enlace

```
unit LO_ListaSimpleEnlace;


interface


USES
  Lib_Auxiliar;


TYPE
     TipoRegDatos = record
                     Clave:Tclave;
                     Siguiente:TPos;
                     PosEnDatos: TPos;
                    end;
```

```
     TipoRegControl = record
                       Primero,Ultimo,Borrado :TPos;
                      end;

     TipoArchDatos = file of TipoRegDatos;
     TipoArchControl = file of TiporegControl;

     tLsEnlace = record
                  D: tipoArchDatos;
                  C: tipoArchControl;
                 end;

procedure CrearListaSimple(var Me:tLsEnlace;NombreControl:String;NombreDatos:String;
Ruta:String);
procedure AbrirListaSimple(var Me:tLsEnlace);
procedure CerrarListaSimple(var Me:tLsEnlace);
procedure DestruirListaSimple(var Me:tLsEnlace);
procedure InsertarInfo_LSimple(var Me:tLsEnlace; reg:TipoRegDatos; Pos:TPos);
procedure EliminarInfo_LSimple(var Me:tLsEnlace; pos:TPos);
procedure ModificarInfo_LSimple(var Me:tLsEnlace; pos:TPos; reg:tiporegdatos);
function BuscarInfo_LdSimple(var Me:tLsEnlace; Clave:Tclave; var Pos :TPos):boolean;
procedure CapturarInfo_LSimple(var ME:tLsEnlace; pos:TPos; var Reg:TipoRegDatos);
function Primero(var Me:tLsEnlace):TPos;
function Ultimo(var Me:tLsEnlace):TPos;
function Proximo(var Me:tLsEnlace; pos:TPos):TPos;
function ListaSimpleVacia(me:tLsEnlace):boolean;
function Posnula(var Me:tLsEnlace):TPos;
function ClaveNula(var Me:tLsEnlace):tClave;

implementation
{-------------------------------------------------------------------------}

procedure CrearListaSimple(var Me:tLsEnlace;NombreControl:String;NombreDatos:String;
Ruta:String);
Var
  ioD, ioC:integer;
```

```
  rc:TipoRegControl;
begin
//Vinculacion de las variable ME con los archivos fisicos
  assign(Me.C, Ruta + NombreControl);
  Assign(Me.D, Ruta + NombreDatos);
{$i-}
  //Si IoResult es cero, la operacion es exitosa
  Reset(Me.C); ioC:=IoResult;
  Reset(Me.D); ioD:=IoResult;

  if ((ioD<>0) or (ioC<>0)) then
  begin
    Rewrite(Me.D);
    Rewrite(Me.C);
    //Inicializo variables de control
    Rc.Primero:=Lib_Auxiliar.PosNula;
    Rc.Ultimo:=Lib_Auxiliar.PosNula;
    Rc.Borrado:=Lib_Auxiliar.Posnula;

    Write(Me.C,Rc);
    Close(Me.D);
    Close(Me.C);
    end;
 {$i+}
End;
{-------------------------------------------------------------------------------}
procedure AbrirListaSimple(var Me:tLsEnlace);
begin
  reset(me.D);
  reset(me.C);
end;

{-------------------------------------------------------------------------------}
procedure CerrarListaSimple(var Me:tLsEnlace);
begin
```

```
  close(me.D);
  close(me.C);
end;
{-------------------------------------------------------------------------------}
procedure DestruirListaSimple(var Me:tLsEnlace);
begin
  erase(me.D);
  erase(me.C);
end;
{-------------------------------------------------------------------------------}
procedure InsertarInfo_LSimple(var Me:tLsEnlace; reg:TipoRegDatos; Pos:tpos);
var
 rc:tiporegcontrol;
 regAnt:tiporegdatos;
 posNueva:tpos;
begin
   seek(me.C,0); read(me.C,rc); //Leo cabecera de control

   if rc.Borrado = Lib_Auxiliar.PosNula then  //No hay borrados
    posnueva:= filesize(me.D)
   else //hay borrados
    begin
      seek(me.D,rc.Borrado); read(me.D,regant);
      posnueva:= rc.Borrado; //PosNueva sera el primer elemento de la lista de borrados
      rc.Borrado:= regant.Siguiente;
    end;

   if rc.Primero = Lib_Auxiliar.PosNula then //Lista vacia
    begin
     rc.Primero:=posnueva;
     rc.Ultimo:= posnueva;
     reg.Siguiente:= Lib_Auxiliar.PosNula;
    end
     else
      if pos = Lib_Auxiliar.PosNula then //Inserto al principio
```

```
       begin
        reg.Siguiente:= rc.Primero;
        rc.Primero:= posnueva;
       end
        else
         if pos = rc.Ultimo then //Inserto al final
          begin
           seek(me.D,pos); read(me.D,regant);
           regant.Siguiente:= posnueva;
           reg.Siguiente:= Lib_Auxiliar.PosNula;
           rc.Ultimo:= posnueva;
           seek(me.D,pos); write(me.D,regant);
          end
           else //Inserto al medio
            begin
             seek(me.D,pos); read(me.D,regant);
             reg.Siguiente:= regant.Siguiente;
             regant.Siguiente:= posnueva;
             seek(me.D,pos); write(me.D,regant);
            end;

   //Grabo registro en la posNueva y cabecera de control actualizado
   seek(me.D,posnueva); write(me.D,reg);
   seek(me.C,0); write(me.C,rc);
end;
{--------------------------------------------------------------------------------}
procedure EliminarInfo_LSimple(var Me:tLsEnlace; pos:tpos);
var
 rc:tiporegcontrol;
 RegAnt,regaelim:tiporegdatos;
 posaux:tpos;

begin
  seek(me.C,0); read(me.C,rc);
```

```
  if pos = Lib_Auxiliar.PosNula then  //Elimino el primer elemento
  begin
    seek(me.D,rc.Primero); read(me.D,regaelim);
    posaux:= rc.Primero;
    rc.Primero:= regaelim.Siguiente;

    if rc.Primero = Lib_Auxiliar.PosNula then  //Ahora la lista esta vacia
    rc.Ultimo:= Lib_Auxiliar.PosNula;
  end
    else
      begin
        seek(me.D,pos); read(me.D,RegAnt);
        posaux:= RegAnt.Siguiente;
        seek(me.D,posaux); read(me.D,regaelim);

        if rc.Ultimo = RegAnt.Siguiente then
           rc.Ultimo := pos;

        RegAnt.Siguiente:= regaelim.Siguiente;
        regaelim.Siguiente:=Lib_Auxiliar.PosNula;
        seek(me.D,pos); write(me.D,RegAnt);
      end;

  regaelim.Siguiente:= rc.Borrado;
  rc.Borrado:=posaux;

  seek(me.D,posaux); write(me.D,regaelim);
  seek(me.C,0); write(me.C,rc);
end;
{--------------------------------------------------------------------------}


procedure ModificarInfo_LSimple(var Me:tLsEnlace; pos:tpos; reg:tiporegdatos);
 begin
  seek(Me.D,pos); write(Me.D,reg);
```

```
 end;
{--------------------------------------------------------------------------------}
function BuscarInfo_LdSimple(var Me:tLsEnlace; Clave:Tclave; var Pos :tPos):boolean;
var
 posaux:tPos;
 corte,encontrado:boolean;
 reg:tipoRegDatos;
 rc:tiporegControl;

 begin
  //Busco clave en lista, si no lo encuentra devuelve la posicion donde deberia estar
  corte:=false;
  encontrado:=false;

  seek(me.C,0); read(me.C,rc);
  posaux:= rc.Primero;
  pos:= Lib_Auxiliar.PosNula;

  while (posaux <> Lib_Auxiliar.PosNula) and not (Corte) and not (encontrado) do
   begin
      seek(me.D,posaux); read(me.D,reg);

      if reg.Clave = clave then
       encontrado:= true
        else
         if reg.Clave > Clave then
          corte:=true
         else
          begin
           pos:= posaux;
           posaux:= reg.Siguiente;
          end;
   end;
  BuscarInfo_LdSimple:= encontrado;
 end;
```

```
{--------------------------------------------------------------------------------}
procedure CapturarInfo_LSimple(var ME:tLsEnlace; pos:tPos; var Reg:TipoRegDatos);
 begin
   seek(me.D,pos); read(me.D,reg);
 end;
{--------------------------------------------------------------------------------}
function Primero(var Me:tLsEnlace):tPos;
var
 rc:tiporegcontrol;
begin
  seek(me.C,0); read(me.C,rc);
  Primero:= rc.Primero; //Posicion del primer elemento en la lista
end;
{--------------------------------------------------------------------------------}
function Ultimo(var Me:tLsEnlace):tPos;
var
 rc:tiporegcontrol;
begin
  seek(me.C,0); read(me.C,rc);
  Ultimo:= rc.Ultimo; //Posicion del ultimo elemento en la lista
end;
{--------------------------------------------------------------------------------}
function Proximo(var Me:tLsEnlace; pos:tPos):tPos;
var reg:tipoRegDatos;
begin
  seek(me.D,pos); read(me.D,reg);
  Proximo:= reg.Siguiente; //Posicion del proximo elemento en pos parametrizado
end;
{--------------------------------------------------------------------------------}



function ListaSimpleVacia(me:tLsEnlace):boolean;
var
 rc:tiporegcontrol;
 begin
```

```
   seek(me.C,0); read(me.C,rc);
   ListaSimpleVacia := (rc.Primero = Lib_Auxiliar.PosNula);
 end;
{--------------------------------------------------------------------------------}
function Posnula(var Me:tLsEnlace):TPos;
 begin
   Posnula:= Lib_Auxiliar.PosNula;
 end;
{--------------------------------------------------------------------------------}
function ClaveNula(var Me:tLsEnlace):tClave;
begin
  ClaveNula:= Lib_Auxiliar.ClaveNula;
end;

END.
```

2.2.6 Lista Doble Enlace

```
unit LO_ListaDobleEnlace;

interface
uses
  Lib_Auxiliar;

Type

      TipoRegDatos = Record
                      Ant,Sig:TPos;
                      Clave: TClave;
                      PosenDatos: TPos;
                     end;

      TipoRegControl= Record
                      Primero:TPos;
                      Ultimo:TPos;
```

```
                    Borrado: TPos;
                   end;

      TipoArchivoDatos = File Of TipoRegDatos;
      TipoArchivoControl = File of TipoRegControl;

      tListaDoble = Record
                  C:TipoArchivoControl;
                  D:TipoArchivoDatos;
                 end;

Procedure CrearLd (Var Me:tListaDoble; Nombre:String; Ruta:string);
procedure AbrirLd   (var Me: tListaDoble);
procedure CerrarLd  (var Me: tListaDoble);
procedure DestruirLista (var Me: tListaDoble);
function ListaVacia(me:tListaDoble):boolean;
Procedure Insertar (var Me:tListaDoble; Reg :TipoRegDatos;Pos:TPos);
procedure Eliminar(var Me:tListaDoble; Pos:TPos);
procedure Capturar(var Me:tListaDoble; Pos:TPos; var Reg:TipoRegDatos);
procedure Modificar(var Me:tListaDoble; Pos:TPos; Reg:TipoRegDatos);
function Primero(var Me:tListaDoble): TPos;
function Ultimo(var Me:tListaDoble): TPos;
function PosNula(var Me:tListaDoble): TPos;
function ClaveNula(var Me:tListaDoble): TClave;
Function Anterior(var Me:tListaDoble; Pos:TPos):TPos;
function Proximo(var Me:tListaDoble; Pos:TPos):TPos;
Function Buscar(var Me:tListaDoble;Clave:TClave;Var pos:TPos):Boolean;

Implementation

uses Math,SysUtils,Forms;

Var

Me:tListaDoble;
```

```
{-------------------------------------------------------------------------------}
Procedure CrearLd (Var Me:tListaDoble; Nombre:String; Ruta:string);
Var
  FaltaControl:Boolean;
  FaltaDatos:Boolean;
  rc:TipoRegControl;
Begin
//Vinculacion de la variable ME con los archivos fisicos
  AssignFile(Me.C,ruta+Nombre+'.CON');
  AssignFile(Me.D,ruta+Nombre+'.DAT');
{$i-}
  Reset(Me.C);
  FaltaControl:=(Ioresult<>0);
  Reset(Me.D);
  FaltaDatos:=(ioResult<>0);
  //Si alguno no existe, los creo nuevamente vacios
  If ((faltaControl) or (FaltaDatos)) then
  begin
    Rewrite(Me.C);
    Rc.Primero:=Lib_Auxiliar.PosNula;
    Rc.Ultimo:=Lib_Auxiliar.PosNula;
    Rc.Borrado:=Lib_Auxiliar.Posnula;
    Write(Me.C,Rc);
    Rewrite(Me.D);

    Close(Me.D);
    Close(Me.C);
  end;
{$i+}
end;

{-------------------------------------------------------------------------------}
procedure AbrirLD (var Me: tListaDoble);
begin
  //Abro los archivos
```

```
  reset (Me.D);
  reset (Me.C);
end;
{--------------------------------------------------------------------------------}
procedure CerrarLD (var Me: tListaDoble);
begin
  //Cierro los archivos
  Close (Me.D);
  Close (Me.C);
end;
{--------------------------------------------------------------------------------}
procedure DestruirLista (var Me: tListaDoble);
begin
  //Destruyo los archivos
  Erase (Me.D);
  Erase (Me.C);
end;
{--------------------------------------------------------------------------------}
function ListaVacia(me:tListaDoble):boolean;
var
 rc:tiporegcontrol;
begin
//Funcion que devuleve verdadero si es que que la lista esta vacia y falso si no es asi
 seek(me.C,0); read(me.C,rc);
 ListaVacia := (rc.Primero = Lib_Auxiliar.posnula);
end;
{--------------------------------------------------------------------------------}
Procedure Insertar(var Me: tListaDoble; Reg: TipoRegDatos; Pos: TPos);
// Se inserta a lo ultimo
var
  rc: TipoRegControl;
  Raux, RauxAnt, RegBorr: TipoRegDatos;
  PosNueva, posant: TPos;
begin
  { Dada una posición inserta un nuevo registro en la lista }
```

```
  seek(Me.C, 0); read(Me.C, rc);

  if rc.Borrado = Lib_Auxiliar.PosNula then // No hay registros borrados
    PosNueva:= FileSize(Me.D) // Uso el fin de archivo
  else
    begin
      // capturo el primer borrado
      seek(Me.D, rc.Borrado); Read(Me.D, RegBorr);
      PosNueva:= rc.Borrado;
      // el primer elemento de la lista de borrados pasa a ser siguiente
      rc.Borrado:= RegBorr.Sig;
    end;

  If (rc.Primero = Lib_Auxiliar.PosNula) Then // La lista esta vacia
  Begin
    rc.Primero:= PosNueva;
    rc.Ultimo:= PosNueva;
    Reg.Sig:= Lib_Auxiliar.PosNula;
    Reg.Ant:= Lib_Auxiliar.PosNula;
  end
  else If rc.Primero = Pos then // Inserto al principio
    Begin
      Reg.Sig:= rc.Primero;
      Reg.Ant:= Lib_Auxiliar.PosNula;

      seek(Me.D, rc.Primero); Read(Me.D, Raux);
      Raux.Ant:= PosNueva;
      seek(Me.D, rc.Primero); Write(Me.D, Raux);
      rc.Primero:= PosNueva;
    end
    else If (Pos = Lib_Auxiliar.PosNula) then // Inserto al final
      Begin
        seek(Me.D, rc.Ultimo); Read(Me.D, Raux);

        Raux.Sig:= PosNueva;
```

```
        Reg.Sig:= Lib_Auxiliar.PosNula;
        Reg.Ant:= rc.Ultimo;

        rc.Ultimo:= PosNueva;
        seek(Me.D, Reg.Ant); Write(Me.D, Raux);
      End
      Else // Inserto al medio
        Begin
      // Leo el registro que se encuentra en Pos
          seek(Me.D, Pos); Read(Me.D, Raux);
      // El nuevo registro tendra como siguiente al actual en "Pos"
          Reg.Sig:= Pos;
      // Capturo la posicion del anterior al registro de Raux
          posant:= Raux.Ant;
      //El nuevo registro (Reg) tendra como anterior al anterior del registro que
//"desplaza"
          Reg.Ant:= posant;
          // Leo el anterior a la posicion parametrizada "Pos" y capturo ese registro
          seek(Me.D, posant); Read(Me.D, RauxAnt);
          // Actualizo los enlances para que apunten al nuevo registro
          RauxAnt.Sig:= PosNueva;
          Raux.Ant:= PosNueva;
          //Actualizo el anterior con el nuevo enlace siguiente a Posnueva
          seek(Me.D, posant); Write(Me.D, RauxAnt);
          // Actualizo Raux con el nuevo enlaces anterior a PosNueva
          seek(Me.D, Pos); Write(Me.D, Raux);
        end;
  // Gravo los registros de control y el registro parametrizado con los enlaces
//actualizados
  seek(Me.D, PosNueva); Write(Me.D, Reg);
  seek(Me.C, 0); Write(Me.C, rc);
end;
{-------------------------------------------------------------------------------}
procedure Eliminar(var Me: tListaDoble; Pos: TPos);
var
  rc: TipoRegControl;
```

```
  Raux, RauxAdy, RauxAdy2: TipoRegDatos;
  PosNueva, posant: TPos;
begin
  seek(Me.C, 0); read(Me.C, rc);
  seek(Me.D, Pos); Read(Me.D, Raux);
  If (rc.Primero = Pos) and (rc.Ultimo = Pos) Then // Esta vacio1
  Begin
    rc.Primero:= Lib_Auxiliar.PosNula;
    rc.Ultimo:= Lib_Auxiliar.PosNula;
  end
  else If rc.Primero = Pos then // Elimino al principio1
    Begin
      seek(Me.D, rc.Primero); Read(Me.D, RauxAdy);
      RauxAdy.Ant:= Lib_Auxiliar.PosNula;
      seek(Me.D, rc.Primero); Write(Me.D, RauxAdy);
      rc.Primero:= Raux.Sig;
    end
    else
      If (Pos = rc.Ultimo) then // Elimino al final1
      Begin
        seek(Me.D, Raux.Ant); Read(Me.D, RauxAdy);
        RauxAdy.Sig:= Lib_Auxiliar.PosNula;

        rc.Ultimo:= Raux.Ant;
        seek(Me.D, Raux.Ant); Write(Me.D, RauxAdy);
      End
      Else // Elimino al medio
        Begin
          seek(Me.D, Raux.Sig); Read(Me.D, RauxAdy);
          seek(Me.D, Raux.Ant); Read(Me.D, RauxAdy2);

          RauxAdy2.Sig:= Raux.Sig;
          RauxAdy.Ant:= Raux.Ant;

          seek(Me.D, Raux.Sig); Write(Me.D, RauxAdy);
```

```
          seek(Me.D, Raux.Ant); Write(Me.D, RauxAdy2);
        end;
        Raux.Ant:= -1;

  // Grabo el archivo de control
  seek(Me.C, 0); Write(Me.C, rc);
end;
{-------------------------------------------------------------------------------}
procedure Capturar (var Me:tListaDoble; Pos:TPos; var Reg:TipoRegDatos);
begin
  Seek(Me.D,Pos); Read(Me.D,reg);
end;
{-------------------------------------------------------------------------------}
procedure Modificar(var Me:tListaDoble; Pos:TPos; Reg:TipoRegDatos);
var
  Rd:TipoRegDatos;
begin
//Dada una posicion cambio los enlaces hacia el nuevo registro y el viejo lo dejo
'suelto'
  Seek(Me.D,Pos); Read(Me.D,Rd);
  Reg.Ant:=Rd.Ant;
  Reg.Sig:=Rd.Sig;
  Seek(Me.D,Pos); Write(Me.D,Reg);
end;
{-------------------------------------------------------------------------------}
function Primero(var Me:tListaDoble): TPos;
var
  Rc:TipoRegControl;
begin
//Devuelve la posicion directa del primer elemento de la lista
   seek(Me.C,0); read(Me.C,Rc);
   Primero:=Rc.Primero;
end;
{-------------------------------------------------------------------------------}
function Ultimo(var Me:tListaDoble): TPos;
```

```
var
  Rc:TipoRegControl;
begin
//Devuelve la posicion de la ultima posicion
   seek(Me.C,0); read(Me.C,Rc);
   Ultimo:=Rc.Ultimo;
end;
{--------------------------------------------------------------------------------}
function PosNula(var Me:tListaDoble): TPos;
begin
//Posicion nula de la lista
  posNula:=Lib_Auxiliar.posnula;
end;
{--------------------------------------------------------------------------------}

function ClaveNula(var Me:tListaDoble): TClave;
begin
//Clave nula del TDA
  ClaveNula:=Lib_Auxiliar.ClaveNula;
end;
{--------------------------------------------------------------------------------}
function Proximo(var Me:tListaDoble; Pos:TPos):TPos;
var
Reg: TipoRegDatos;
begin
//Dada una posicion devuelve la posicion siguiente a ella
  Seek(Me.D,Pos); Read(Me.D,Reg);
  Proximo:=Reg.Sig;
end;

{--------------------------------------------------------------------------------}

Function Anterior(var Me:tListaDoble; Pos:TPos):TPos;
var
  Reg: TipoRegDatos;
```

```
begin
//Dada una posicion devuelve la posicion anterior a ella
  Seek(Me.D,Pos); Read(Me.D,Reg);
  Anterior:=Reg.Ant;
end;
{--------------------------------------------------------------------------------}
Function Buscar(var Me:tListaDoble;Clave:TClave;Var pos:TPos):Boolean;
Var
  Encontrado:Boolean;
  Reg:TipoRegDatos;
  rc:TipoRegControl;
begin
  //Recorre la lista y dad una clave devulve si la encontro o no. Si no la encuentra
devuelve la posicion donde deveria estar
  Seek(Me.C,0); Read(Me.C,Rc);
  Pos:=Rc.Primero;
  Encontrado:=False;
  While ((Not Encontrado) And (Pos<>Lib_Auxiliar.posNula)) Do
    Begin
      Seek(Me.D,Pos); Read(Me.D,Reg);
      IF ((reg.Clave=Clave)) Then
        Encontrado:=true
      Else
        Begin
          Pos:=Reg.Sig;
        end;
    end;
  Buscar:=Encontrado;
end;


END.
```

2.2.7 Lista Parcial Doble Enlace

```
unit LO_ListaParcialDobleEnlace;
```

```
{
  Esta Libreria Operacional cuenta solo con un archivo de datos,
  donde el control (primero, ultimo borrado) de dicho archivo es parametrizado
}
interface
USES
  Lib_Auxiliar, SysUtils;

TYPE
    TipoRegControl= Record
                    Primero:TPos;
                    Ultimo:TPos;
                    Borrado: TPos;
                  end;

    TipoRegDatos = Record
                    Ant,Sig:TPos;
                    Clave: TClave;
                    PosEnDatos:tPos;
                  end;

    TipoArchivoDatos = File Of TipoRegDatos;
    TipoArchivoControl = File Of TipoRegControl;

    //'Lista doble parcial' cuenta solo con un archivo de datos
    tLdParcial = Record
                    D:TipoArchivoDatos;
                  End;

Procedure CrearLdParcial (var Me:tLdParcial; Nombre:String; Ruta:String);
procedure AbrirLdParcial (var Me: tLdParcial);
procedure CerrarLdParcial (var Me: tLdParcial);
procedure DestruirLdParcial (var Me: tLdParcial);
Procedure InsertarInfo_LdParcial (var Me:tLdParcial; Reg :TipoRegDatos; Pos:TPos; var
RegControl:tipoRegControl);
```

```
procedure EliminarInfo_LdParcial (var Me:tLdParcial; Pos:TPos; var
RegControl:TipoRegControl);

procedure ModificarInfo_LdParcial (var Me:tLdParcial; Pos:TPos; Reg:TipoRegDatos);

Function BuscarInfo_LdParcial(var Me:tLdParcial;Clave:TClave;Var pos:TPos;var
RegControl:TipoRegControl):Boolean;

procedure CapturarInfo_LdParcial (var Me:tLdParcial; Pos:TPos; var Reg:TipoRegDatos);

function Primero (var Me:tLdParcial; RegControl:TipoRegControl): TPos;

function Ultimo (var Me:tLdParcial; RegControl:TipoRegControl): TPos;

Function Anterior (var Me:tLdParcial; Pos:TPos):TPos;

function Proximo (var Me:tLdParcial; Pos:TPos):TPos;

function LdParcialVacia (var Me:tLdParcial; RegControl:TipoRegControl):boolean;

function PosNula (var Me:tLdParcial): TPos;

function ClaveNula (var Me:tLdParcial): TClave;

procedure InicializarCabecera (var regControl: TipoRegControl);


Implementation

{-------------------------------------------------------------------------------}

Procedure CrearLdParcial (var Me:tLdParcial; Nombre:String; Ruta:String);

Var

  FaltaDatos:Boolean;

Begin

  AssignFile(Me.D,Ruta+Nombre);

  AssignFile(Me.D,Ruta+Nombre);

{$i-}

  Reset(Me.D); FaltaDatos:=(ioResult<>0);

  if (FaltaDatos) then

  begin

    Rewrite(Me.D);

    Close(Me.D);

  end;

{$i+}

end;

{-------------------------------------------------------------------------------}

Procedure AbrirLdParcial (var Me: tLdParcial);

begin

  reset (Me.D);
```

```
end;
{--------------------------------------------------------------------------------}
Procedure CerrarLDParcial (var Me: tLdParcial);
begin
  Close (Me.D);
end;
{--------------------------------------------------------------------------------}
Procedure DestruirLdParcial (var Me: tLdParcial);
begin
  Erase (Me.D);
end;
{--------------------------------------------------------------------------------}
Procedure InsertarInfo_LdParcial (var Me:tLdParcial; Reg :TipoRegDatos; Pos:TPos; var
RegControl:tipoRegControl); //Se inserta a lo ultimo
var
RegAux,RegAntAux,RegBorrado:TipoRegDatos;
PosNueva,posAnt:TPos;
begin
  if RegControl.Borrado = Lib_Auxiliar.Posnula then //No hay borrados
   PosNueva:=FileSize(Me.D)
  else
   begin //si hay borrados
     seek(Me.D,RegControl.Borrado); Read(Me.D,RegBorrado);
    //PosNueva sera el primer elemento de la lista de borrados
     PosNueva:= RegControl.Borrado;
     RegControl.Borrado:= RegBorrado.Sig;
   end;

  If (RegControl.Primero=Lib_Auxiliar.posnula) Then //Lista vacia
  begin
    RegControl.Primero:=PosNueva;
    RegControl.Ultimo:=PosNueva;
    Reg.Sig:=Lib_Auxiliar.PosNula;
    Reg.Ant:=Lib_Auxiliar.PosNula;
  end
```

```
else
 If RegControl.Primero=pos then //Inserto al principio
 Begin
   Reg.Sig:=RegControl.Primero;
   Reg.Ant:=Lib_Auxiliar.PosNula;
   seek(Me.D,RegControl.Primero); read(Me.D,RegAux);
   RegAux.Ant:=PosNueva;
   seek(Me.D,RegControl.Primero); write(Me.D,RegAux);
   RegControl.Primero:=PosNueva;
 end
  else
    If(Pos=Lib_Auxiliar.posnula) then //Inserto al final
    Begin
       seek(Me.D,RegControl.Ultimo); read(Me.D,RegAux);
       RegAux.Sig:=PosNueva;
       Reg.Sig:=Lib_Auxiliar.PosNula;
       Reg.Ant:=RegControl.Ultimo;
       RegControl.Ultimo:=PosNueva;
       seek(Me.D,Reg.Ant); write(Me.D,RegAux);
    end
    else //Inserto al medio
     begin
       Seek(Me.D,Pos); Read(Me.D,RegAux); //Leo reg en Pos
      //El nuevo registro tendra como siguiente al actual en que se encuentra en Pos
       Reg.Sig:=pos;
       PosAnt:=RegAux.ant;
       Reg.Ant:=PosAnt;
      //Leo el anterior a reg que se encuentra en Pos
       Seek(Me.D, PosAnt); Read(Me.D, RegAntAux);
       //Actualizo los enlances para que apunten al nuevo registro
       RegAntAux.Sig:=PosNueva;
       RegAux.Ant:=PosNueva;
       //Actualizo el anterior con el nuevo enlace siguiente a Posnueva
       Seek(Me.D,PosAnt); Write(Me.D,RegAntAux);
       //Actualizo Raux con el nuevo enlaces anterior a PosNueva
```

```
          Seek(Me.D,Pos); Write(Me.D,RegAux);
        end;


   //Grabo registro en la posNueva
    Seek(Me.D,Posnueva); Write(Me.D,Reg);
end;
{--------------------------------------------------------------------------------}
Procedure EliminarInfo_LdParcial (var Me:tLdParcial; Pos:TPos; var
RegControl:TipoRegControl);
var
  RegAux,RegAuxSig,RegAuxAnt:TipoRegDatos;
  PosNueva,posant:TPos;
begin
  seek(Me.D,Pos); read(Me.D,RegAux);

  If (RegControl.Primero=pos) and (RegControl.Ultimo=Pos) Then //Lista vacia
      Begin
        RegControl.Primero:=Lib_Auxiliar.Posnula;
        RegControl.Ultimo:=Lib_Auxiliar.Posnula;
      end
      else
        If RegControl.Primero=pos then //Se elimina al principio
          Begin
            seek(Me.D,RegAux.sig); read(Me.D,RegAuxSig);
            RegAuxSig.Ant:=Lib_Auxiliar.Posnula;
            seek(Me.D,RegAuxSig.sig); write(Me.D,RegAuxSig);
            RegControl.Primero:=RegAux.Sig;
          end
          else
            If(Pos=RegControl.Ultimo) then //Se elimina al final
              Begin
                seek(Me.D,RegAux.Ant); read(Me.D,RegAuxAnt);
                RegAuxAnt.Sig:=Lib_Auxiliar.Posnula;
                RegControl.Ultimo:=RegAux.Ant;
                seek(Me.D,RegAux.Ant); write(Me.D,RegAuxAnt);
```

```
              end
              else //Se elimina al medio
                Begin
                  seek(Me.D,RegAux.Sig); read(Me.D,RegAuxSig);
                  seek(Me.D,RegAux.Ant); read(Me.D,RegAuxAnt);
                  RegAuxAnt.Sig:=RegAux.Sig;
                  RegAuxSig.Ant:=RegAux.Ant;
                  seek(Me.D,RegAux.Sig); write(Me.D,RegAuxSig);
                  seek(Me.D,RegAux.Ant); write(Me.D,RegAuxAnt);
                end;
  RegAux.Ant:=-1;
  //Pila de borrados
  RegAux.Ant:= Lib_Auxiliar.Posnula;
  RegAux.Sig:= RegControl.Borrado;
  RegControl.Borrado:= Pos;
  //Grabo registro borrado
  seek(Me.D,pos); write(Me.D,RegAux);
end;
{-------------------------------------------------------------------------------}
Procedure ModificarInfo_LdParcial (var Me:tLdParcial; Pos:TPos; Reg:TipoRegDatos);
begin
  seek(Me.D,Pos); write(Me.D,Reg);
end;
{-------------------------------------------------------------------------------}
Function BuscarInfo_LdParcial (var Me:tLdParcial;Clave:TClave;Var pos:TPos;var
RegControl:TipoRegControl):Boolean;
Var
  Enc, corte:Boolean;
  Reg:TipoRegDatos;
  rc:TipoRegControl;

Begin
  Pos:=RegControl.Primero;
  Enc:=False;
  corte:=false;
```

```
  //Busco clave en lista, si no lo encuentra devuelve la posicion donde deberia estar
  While ((Not Enc) And (Pos<>Lib_Auxiliar.posNula) and not corte) Do
  Begin
    seek(Me.D,Pos); read(Me.D,Reg);
    if (reg.Clave=Clave) Then
      Enc:=true
    else
     begin
       if Clave < reg.Clave then
         corte:=true
       else
         Pos:=Reg.Sig;
     end;
  end;
   BuscarInfo_LdParcial:=Enc;
end;
{-------------------------------------------------------------------------------------}
Procedure CapturarInfo_LdParcial (var Me:tLdParcial; Pos:TPos; var Reg:TipoRegDatos);
begin
  seek(Me.D,Pos); read(Me.D,reg);
end;
{-------------------------------------------------------------------------------------}
Function Primero (var Me:tLdParcial; RegControl:TipoRegControl): TPos;
begin
   Primero:=RegControl.Primero;
end;
{-------------------------------------------------------------------------------------}
Function Ultimo (var Me:tLdParcial; RegControl:TipoRegControl): TPos;
begin
   Ultimo:=RegControl.Ultimo;
end;
{-------------------------------------------------------------------------------------}
Function Anterior (var Me:tLdParcial; Pos:TPos):TPos;
var
  Reg: TipoRegDatos;
```

```
begin
  seek(Me.D,Pos); read(Me.D,Reg);
  Anterior:=Reg.Ant;
end;
{-------------------------------------------------------------------------------}
Function Proximo (var Me:tLdParcial; Pos:TPos):TPos;
var
  Reg: TipoRegDatos;
begin
  seek(Me.D,Pos); read(Me.D,Reg);
  Proximo:=Reg.Sig;
end;
{-------------------------------------------------------------------------------}
Function LdParcialVacia (var Me:tLdParcial; RegControl:TipoRegControl):boolean;
begin
  LdParcialVacia := (RegControl.Primero = Lib_Auxiliar.Posnula);
end;
{-------------------------------------------------------------------------------}
Function PosNula (var Me:tLdParcial): TPos;
begin
  posNula:=Lib_Auxiliar.posnula;
end;
{-------------------------------------------------------------------------------}
Function ClaveNula (var Me:tLdParcial): TClave;
begin
  ClaveNula:=Lib_Auxiliar.ClaveNula;
end;
{-------------------------------------------------------------------------------}
Procedure InicializarCabecera (var regControl: TipoRegControl);
begin
  regcontrol.Primero:= -1;
  regcontrol.Ultimo:= -1;
  regcontrol.Borrado:= -1;
end;
END.
```

2.3 ME

2.3.1 Jugadores

```
unit ME_JUGADORES;


{
Consiga:


Tiene por objeto registrar a todos los jugadores de RuleTrucha
 El ME_JUGADORES contendrá los siguientes archivos:


 1a) El archivo JUGADORES.DAT, que se encuentran organizados a través de un  índice de Árbol AVL,
    que contiene los datos del jugador


 2a) El archivo JUGADORES.CON, cabecera de Indice


 3a) El archivo JUGADORES.NTX, Indice
 }


interface


uses
  Lib_Auxiliar, LO_ArbolAVL, Type_JUGADOR,
  System.Math,System.SysUtils,Forms,Dialogs;


CONST


  _Directorio = 'Archivos\';
  _NombreArchivoDatos = 'JUGADORES.DAT';
  _NombreArchivoControl = 'JUGADORES.CON';
  _NombreArchivoIndice = 'JUGADORES.NTX';


TYPE
```

```
      ME_JUGADOR = Record
              D: Type_JUGADOR.tArchDatos; //Archivo datos
              I: LO_ArbolAVL.tArbol; //Archivo indice de datos
            End;

procedure CrearME_Jugadores (var Me: ME_JUGADOR);
Procedure AbrirME_Jugadores(var Me:ME_JUGADOR);
Procedure CerrarME_Jugadores(var Me:ME_JUGADOR);
procedure DestruirME_Jugadores(var Me:ME_JUGADOR);
Function MeVacio_Jugadores(var Me: ME_JUGADOR):Boolean;
Procedure InsertarInfoME_Jugadores(var Me: ME_JUGADOR;Reg:tRegDatos;Pos:tPos);
Procedure EliminarInfoME_Jugadores(var Me: ME_JUGADOR; Pos:tPos);
Procedure ModificarInfoME_Jugadores(var Me: ME_JUGADOR; Reg:tRegDatos;Pos:tPos);
Procedure CapturarInfoME_Jugadores(var Me: ME_JUGADOR; var Reg:tRegDatos; Pos:tPos);
Function BuscarInfoME_Jugadores(Var Me:ME_JUGADOR; Var Pos:tPos; Clave:TClave):Boolean;
function PosNula_Jugadores(Var Me:ME_JUGADOR):tPos;
function ClaveNula_Jugadores(Var Me:ME_JUGADOR):tClave;
function Raiz(var Me: ME_JUGADOR): tPos;
function ProximoIzq(var Me: ME_JUGADOR; pos: tPos): tPos;
function ProximoDer(var Me: ME_JUGADOR; pos: tPos): tPos;
function UltFicticio (var Me: ME_JUGADOR): tCantidad;
function Cantidad_Jugadores(var Me: ME_JUGADOR): tCantidad;
function ObtenerNivel_enArbol(var Me:ME_JUGADOR; clave:tClave): tCantidad;
procedure Asignar(var Me:ME_JUGADOR);

implementation
{-------------------------------------------------------------------------------}
procedure CrearME_Jugadores (var Me: ME_JUGADOR);
Var
  ioD:integer;
begin
  assignFile (Me.D, _Directorio + _NombreArchivoDatos);
{$i-}
  reset(Me.D); ioD:=IoResult; //Si IoResult es cero, la operacion es exitosa
```

```
  if (IoD<>0)then
    Rewrite(Me.D); Close(Me.D);
{$I+}

LO_ArbolAVL.CrearMe_Arbol(me.I,_NombreArchivoControl,_NombreArchivoIndice,_Directorio);
end;
{-------------------------------------------------------------------------------}
Procedure AbrirME_Jugadores(var Me:ME_JUGADOR);
begin
  Asignar(Me);
  Reset(Me.D);
  LO_ArbolAVL.AbrirMe_Arbol(me.I);
end;
{-------------------------------------------------------------------------------}
Procedure CerrarME_Jugadores(var Me:ME_JUGADOR);
begin
  Close(Me.D);
  LO_ArbolAVL.CerrarMe_Arbol(me.i);
end;
{-------------------------------------------------------------------------------}
procedure DestruirME_Jugadores(var Me:ME_JUGADOR);
begin
  Erase(Me.D);
  LO_ArbolAVL.DestruirMe_Arbol(me.i);
end;
{-------------------------------------------------------------------------------}
Function MeVacio_Jugadores(var Me: ME_JUGADOR):Boolean;
begin
  MeVacio_Jugadores:= LO_ArbolAVL.Arbol_Vacio(me.i);
end;

{-------------------------------------------------------------------------------}
Procedure InsertarInfoME_Jugadores(var Me: ME_JUGADOR;Reg:tRegDatos;Pos:tPos);
Var
  Aux,RegPadre:tRegDatos;
```

```
  Rc:tControlArbol;
  PosenDatos:tPos;
  RegME: LO_ArbolAVL.tNodoIndice;
Begin
  PosenDatos:= filesize(Me.D);
  reg.Nick:=  UpperCase(Reg.Nick);
  RegME.PosenDatos:= PosenDatos;
  RegME.Clave:= Reg.Nick;
  LO_ArbolAVL.InsertarNodo_Arbol(Me.I,RegME,Pos, reg.TipoJugador);
  Seek (Me.D,RegME.PosEnDatos); Write(Me.D,Reg);
end;
{-------------------------------------------------------------------------------}
Procedure EliminarInfoME_Jugadores(var Me: ME_JUGADOR; Pos:tPos);
begin
    //Elimina de la estructura el registro que esta en la posicion parametrizada
    LO_ArbolAVL.EliminarNodo_Arbol(me.I,pos);
end;
{-------------------------------------------------------------------------------}
Procedure ModificarInfoME_Jugadores(var Me: ME_JUGADOR; Reg:tRegDatos; Pos:tPos);
var
  RE: LO_ArbolAVL.tNodoIndice;
  regAlm:tRegDatos;
begin
//Se le pasa la posicion de la estructrura, trae el registro y sobreescribe el
//registro en el almacen de datos
  Seek(Me.I.D,pos); Read(Me.I.D,RE);
  Seek(Me.D,RE.PosEnDatos); Write(Me.D, Reg);
end;
{-------------------------------------------------------------------------------}


Procedure CapturarInfoME_Jugadores(var Me: ME_JUGADOR; var Reg:tRegDatos; Pos:tPos);
var
  regIndice: LO_ArbolAVL.tNodoIndice;
begin
```

```
 //Se le pasa la posicion en arbol, trae el registro y con posEnDatos trae ese registro
  LO_ArbolAVL.CapturarNodo_Arbol(ME.I, regIndice, pos);
  Seek(Me.D, regIndice.posendatos);Read(Me.D, Reg);
end;
{-------------------------------------------------------------------------------}
Function BuscarInfoME_Jugadores(Var Me:ME_JUGADOR; Var Pos:tPos; Clave:TClave):Boolean;
Begin
//Busco en la arbol, si la clave esta devuelve la posicion donde se encuentra, sino
//devuelve la posicion donde deberia estar
   BuscarInfoME_Jugadores:= LO_ArbolAVL.BuscarNodo_Arbol(ME.i,clave, pos)
End;
{-------------------------------------------------------------------------------}
function PosNula_Jugadores(Var Me:ME_JUGADOR):tPos;
begin
  PosNula_Jugadores:= LO_ArbolAVL.PosNula_Arbol(me.I);
end;
{-------------------------------------------------------------------------------}
function ClaveNula_Jugadores(Var Me:ME_JUGADOR):tClave;
begin
  ClaveNula_Jugadores:= _clave_nula_archivo;
end;
{-------------------------------------------------------------------------------}
function Raiz(var Me: ME_JUGADOR): tPos;
begin
  Raiz:= LO_ArbolAVL.Raiz_Arbol(me.I);
end;
{-------------------------------------------------------------------------------}
function ProximoIzq(var Me: ME_JUGADOR; pos: tPos): tPos;
begin
  ProximoIzq:= LO_ArbolAVL.HijoIzq_Arbol(ME.I,pos);
end;
{-------------------------------------------------------------------------------}
function ProximoDer(var Me: ME_JUGADOR; pos: tPos): tPos;
begin
  ProximoDer:= LO_ArbolAVL.HijoDer_Arbol(ME.I,pos);
```

```
end;
{-------------------------------------------------------------------------------}
function UltFicticio (var Me: ME_JUGADOR): tCantidad;
begin
  UltFicticio:= LO_ArbolAVL.UltFicticio_Arbol(me.I);
end;
{-------------------------------------------------------------------------------}
function Cantidad_Jugadores(var Me: ME_JUGADOR): tCantidad;

begin
  Cantidad_Jugadores:= LO_ArbolAVL.CantidadNodos_Arbol(me.I);
end;
{-------------------------------------------------------------------------------}
function ObtenerNivel_enArbol(var Me:ME_JUGADOR; clave:tClave): tCantidad;

begin
  ObtenerNivel_enArbol:= LO_ArbolAVL.ObtenerNivel_Arbol(Me.i, clave);
end;
{-------------------------------------------------------------------------------}
procedure Asignar(var Me:ME_JUGADOR);

begin
  AssignFile(me.D, _Directorio + _NombreArchivoDatos);
  AssignFile(me.I.C, _Directorio + _NombreArchivoControl);
  AssignFile(me.I.D, _Directorio + _NombreArchivoIndice);
end;


END.
```

2.3.1.1 Type Jugador

```
unit Type_JUGADOR;
```

```
interface
USES
  Sysutils, dateutils;

Const
  _clave_nula_archivo='00000000';

TYPE
  tClave = string[15];
  tString = string [40];
  tContrasenia = string[15];
  tFecha = TDateTime;
// true = jugador real ... false = jugador ficticio
  tTipoJugador = boolean;
// true = jugador conectado ... false = jugador no conectado
  tEstadoJugador = boolean;
  tBloqueado = boolean; //true = bloqueado ... false = no bloqueado

  tRegDatos = Record
      Nick: tclave;
      Contrasenia: tContrasenia;
      Nombre, Apellido: tString;
      Alta: tFecha;
      UltimaConexion: tFecha;
      TipoJugador: tTipoJugador;
      Estado: tEstadoJugador;
      Bloqueado: tbloqueado;
   end;
  tArchDatos = File Of tRegDatos;
implementation
END.
```

2.3.2 Cuenta Corriente

```
unit ME_CTACTE;
```

```
{

ME Cuenta Corriente utiliza:


(1) Un archivo de datos donde se almacenan todos los movimientos de cada cuenta
corriente

(2) Una lista doble parcial que hará de indice sobre archivo almacen

(3) Una tabla secuencial dinámica que hará de control sobre la lista parcial anterior
mencionada

(4) Una lista doble parcial de cuentas corrientes, que almacena todas las cuentas
corrientes existentes y que apunta a la tabla secuencial anterior mencionada.

Las cuentas corrientes son de los jugadores cuyos nicks coinciden con el mismo hash

(5) Un archivo fijo (tabla hash) que tiene control sobre la lista cuentas corrientes
anterior mencionada. En este archivo se guardara en cada posicion un registro control
que apuntara a la lista de Cuentas Corrientes.

Cada posicion del archivo contiene un registro control que corresponde a una lista de
cuentas corrientes de aquellos jugadores que coinciden en la misma posicion de la tabla
hash, posicion dada por la funcion Hash

}

Interface


uses

 SysUtils,

 Type_Hash, Type_ALMACEN, LO_Hash, LO_ListaParcialDobleEnlace, Lib_Auxiliar;


Const

  _Directorio = 'Archivos\';


  _NombreArch_TablaHash = 'CUENTA.DAT';

  _NombreArch_LDCtaCte = 'LISTACUENTAS.CTE';

  _NombreArch_TablaIndMov = 'CONTROLINDICE.COX';

  _NombreArch_LDIndMov = 'INDICEALMACEN.NTX';

  _NombreArch_Almacen = 'ALMACEN.DAT';


type


  tipoEstructura = record

            tablaHash: LO_Hash.tHash; //(5)

            ListaCtasCtes: LO_ListaParcialDobleEnlace.tLdParcial; //(4)
```

```
            end;


  tipoDatos = record
              tablaControlIndice:LO_ListaParcialDobleEnlace.TipoArchivoControl; //(3)
              ListaIndiceAlmacen:LO_ListaParcialDobleEnlace.tLdParcial; //(2)
              Almacen:Type_ALMACEN.tipoArchAlmacen; //(1)
            end;


  ME_CUENTA = record
              Estructura: tipoEstructura;
              Datos: tipoDatos;
            end;


  Procedure CrearMe_CtaCte (Var Me:ME_CUENTA);
  procedure AbrirMe_CtaCte (var Me: ME_CUENTA);
  procedure CerrarMe_CtaCte (var Me: ME_CUENTA);
  procedure DestruirMe_CtaCte (var Me: ME_CUENTA);
  Procedure InsertarCtaCte (var Me:ME_CUENTA; Clave:tClaveHash; var PosEnDatos:tPos);
  Function SaldoAcumulado (var me:ME_CUENTA; Clave: tclave): timporte;
  Procedure CapturarCtaCte(var Me:ME_CUENTA; Clave:tClaveHash; var Reg:tipoRegDatos);
  procedure CapturarAlmacen (var Me:ME_CUENTA; Pos:TPos; var Reg:TipoRegAlmacen);
  Procedure InsertarIndiceAlmacen(var Me:ME_CUENTA; Clave:tClaveHash; PosAlmacen:tPos);
  procedure CapturarTablaIndiceMov (var Me:ME_CUENTA; Pos:TPos; var
Reg:TipoRegControl);
  procedure GrabarTablaIndiceMov (var Me:ME_CUENTA; Pos:TPos; Reg:TipoRegControl);
  Procedure InsertarAlmacen (var Me:ME_CUENTA; var RegAlmacen:tipoRegAlmacen);
  function PosNula (var Me:ME_CUENTA): TPos;
  function ProximoEnLdParcial (var Me:ME_CUENTA; Pos:TPos):TPos;
  Function PrimeroLdParcial(var Me:ME_CUENTA; Clave:tClaveHash):tPos;
  procedure CapturarEnAlmacen (var Me:ME_CUENTA; Pos:TPos; var Reg:TipoRegAlmacen);
  procedure Asignar(var Me:ME_CUENTA);


implementation


{-------------------------------------------------------------------------------}
```

```
Procedure CrearMe_CtaCte (Var Me:ME_CUENTA);
Var
  FaltaAlmacen, FaltaIndice:Boolean;
Begin
  LO_Hash.Crear(Me.Estructura.tablaHash,_NombreArch_TablaHash,_Directorio);
  LO_ListaParcialDobleEnlace.CrearLdParcial
  (me.Estructura.ListaCtasCtes,_NombreArch_LDCtaCte,_Directorio);

  AssignFile(Me.Datos.tablaControlIndice,_Directorio +
  _NombreArch_TablaIndMov);//control indiceAlmacen
 {$i-}
  Reset(Me.Datos.tablaControlIndice);
  FaltaAlmacen:=(Ioresult<>0);
  If (FaltaAlmacen)  then
  begin
    Rewrite(Me.Datos.tablaControlIndice);
    close(me.Datos.tablaControlIndice);
  end;
 LO_ListaParcialDobleEnlace.CrearLdParcial
(me.Datos.ListaIndiceAlmacen,_NombreArch_LDIndMov,_Directorio);

    AssignFile(Me.Datos.Almacen,_Directorio + _NombreArch_Almacen);//datos almacen
    {$i-}
    Reset(Me.Datos.Almacen);
    FaltaAlmacen:=(Ioresult<>0);
    If (FaltaAlmacen)  then
    begin
      Rewrite(Me.Datos.Almacen);
      close(me.Datos.Almacen);
    end;
end;

{-------------------------------------------------------------------------------}
```

```
procedure AbrirMe_CtaCte (var Me: ME_CUENTA);
begin
  Asignar(Me);
  LO_Hash.Abrir(Me.Estructura.tablaHash);
  LO_ListaParcialDobleEnlace.AbrirLDParcial(Me.Estructura.ListaCtasCtes);
  reset(me.Datos.tablaControlIndice);
  LO_ListaParcialDobleEnlace.AbrirLDParcial(me.Datos.ListaIndiceAlmacen);
  reset(me.Datos.Almacen);
end;

{-------------------------------------------------------------------------------}

procedure CerrarMe_CtaCte (var Me: ME_CUENTA);
begin
  LO_Hash.Cerrar(Me.Estructura.tablaHash);
  LO_ListaParcialDobleEnlace.CerrarLDParcial(Me.Estructura.ListaCtasCtes);
  close(me.Datos.tablaControlIndice);
  LO_ListaParcialDobleEnlace.CerrarLDParcial(me.Datos.ListaIndiceAlmacen);
  close(me.Datos.Almacen);
end;

{-------------------------------------------------------------------------------}

procedure DestruirMe_CtaCte  (var Me: ME_CUENTA);
begin
  LO_Hash.Destruir(Me.Estructura.tablaHash);
  LO_ListaParcialDobleEnlace.DestruirLdParcial(Me.Estructura.ListaCtasCtes);
  erase(me.Datos.tablaControlIndice);
  LO_ListaParcialDobleEnlace.DestruirLdParcial(me.Datos.ListaIndiceAlmacen);
  erase(me.Datos.Almacen);
end;

{-------------------------------------------------------------------------------}

Procedure InsertarCtaCte (var Me:ME_CUENTA; Clave:tClaveHash; var PosEnDatos:tPos);
```

```
var
  RegHashControl:LO_ListaParcialDobleEnlace.TipoRegControl;
  RegCtaCte:LO_ListaParcialDobleEnlace.TipoRegDatos;
  posEnCtaCte, posClaveHash, pos, posLista, posUltimo:TPos;
  buscar:boolean;
  RegControlIndice:tipoRegControl;
begin
  RegCtaCte.Clave:=Clave;
  posClaveHash:=LO_Hash.FuncionHash(Clave);//Otengo posicion de la tabla Hash
  //Obtengo contenido de la tabla Hash con la posicion parametrizada
  LO_Hash.Capturar(me.Estructura.tablaHash,posClaveHash,RegHashControl);
  pos:=RegHashControl.primero;

  if (pos = Lib_Auxiliar.PosNula) then  //Si no existe el equipo
  begin
    LO_ListaParcialDobleEnlace.BuscarInfo_LdParcial
   (me.Estructura.ListaCtasCtes,Clave,PosLista, RegHashControl);
    {Antes de insertar RegCtaCte, voy a buscar su posEnTablaControlIndice (vengo de
    insertar un movimiento}
    posUltimo:=Filesize(me.Datos.tablaControlIndice);
    RegCtaCte.PosEnDatos:=posUltimo;
    PosEnDatos:=posUltimo;

    LO_ListaParcialDobleEnlace.InsertarInfo_LdParcial
    (me.Estructura.ListaCtasCtes,RegCtaCte,PosLista,RegHashControl);
    LO_Hash.modificar(me.Estructura.tablaHash, RegHashControl,
    posClaveHash);//actualizo cabecera tabla hash

    LO_ListaParcialDobleEnlace.InicializarCabecera(regControlIndice);
    GrabarTablaIndiceMov(me,posEnDatos,regControlIndice);
  end
  else //si no existe la CuentaCorriente...
   begin
     if not (LO_ListaParcialDobleEnlace.BuscarInfo_LdParcial
     (me.Estructura.ListaCtasCtes,Clave,PosLista, RegHashControl)) then
```

```
     begin
       buscar:=LO_ListaParcialDobleEnlace.BuscarInfo_LdParcial
       (me.Estructura.ListaCtasCtes,Clave,PosLista, RegHashControl);

       //Antes de insertar RegCtaCte ...
       posUltimo:=Filesize(me.Datos.tablaControlIndice);
       RegCtaCte.PosEnDatos:=posUltimo;
       PosEnDatos:=posUltimo;
       //...grabo registro en tabla indice de movimientos
       LO_ListaParcialDobleEnlace.InicializarCabecera(regControlIndice);
       GrabarTablaIndiceMov(me,posEnDatos,regControlIndice);

       LO_ListaParcialDobleEnlace.InsertarInfo_LdParcial
       (me.Estructura.ListaCtasCtes,RegCtaCte,PosLista,RegHashControl);
       LO_Hash.modificar(me.Estructura.tablaHash, RegHashControl, posClaveHash);
       //ACTUALIZO CABECERA TABLA HASH
     end
      else
       begin
         LO_ListaParcialDobleEnlace.CapturarInfo_LdParcial
         (me.Estructura.ListaCtasCtes,PosLista,RegCtaCte);
         PosEnDatos:=RegCtaCte.posEnDatos;
       end;
   end;
end;

{-------------------------------------------------------------------------------}

Procedure InsertarAlmacen (var Me:ME_CUENTA; var RegAlmacen:tipoRegAlmacen);
var
  posAlmacen:tPos;
begin
  posAlmacen:=fileSize(me.Datos.Almacen);
  Seek(me.Datos.Almacen, PosAlmacen);
  write(me.Datos.Almacen, RegAlmacen);
```

```
  InsertarIndiceAlmacen(me, RegAlmacen.Nick, PosAlmacen);
end;

{-------------------------------------------------------------------------------}

Procedure InsertarIndiceAlmacen(var Me:ME_CUENTA; Clave:tClaveHash; PosAlmacen:tPos);
var
  posEnDatos:tPos;
  PosLista:tPos;
  regCI, RegControlIndice:tipoRegControl;
  regIndiceAlmacen:tipoRegDatos;
begin
  //Invoco InsertarCtaCte por si no existe la cuenta. De todas formar me trae en
PosEnDatos que le paso por referencia la posicion que apunta a la 'tabla indice
movimientos'
  InsertarCtaCte(me, clave, posEnDatos);
  //Me paro en la tabla tablaControlIndice, y capturo ese registroControl
  CapturarTablaIndiceMov(me,posEnDatos,regControlIndice);
  //Preparo registro indice almacen antes de insertar
  regIndiceAlmacen.posEnDatos:=PosAlmacen;
  regIndiceAlmacen.Clave:=Clave;



  //Parametrizo en Buscar el registro control indice que esta en la tabla Indice
  //Movimientos en esa posicion
  LO_ListaParcialDobleEnlace.BuscarInfo_LdParcial
  (me.Datos.ListaIndiceAlmacen,Clave,PosLista, RegControlIndice);

  LO_ListaParcialDobleEnlace.InsertarInfo_LdParcial
  (me.Datos.ListaIndiceAlmacen,regIndiceAlmacen,PosLista,RegControlIndice);

  //Actualizo cabecera tablba control indice
  GrabarTablaIndiceMov(me,posEnDatos,regControlIndice);
end;
```

```
{-------------------------------------------------------------------------------}
Procedure CapturarCtaCte(var Me:ME_CUENTA; Clave:tClaveHash; var Reg:tipoRegDatos);
var
  RegHashControl:LO_ListaParcialDobleEnlace.TipoRegControl;
  posClaveHash, PosLista:TPos;
begin
  posClaveHash:=LO_Hash.FuncionHash(Clave);//Otengo posicion de la tabla Hash
  //capturo el registro control de la tabla hash
  LO_Hash.Capturar(me.Estructura.tablaHash,posClaveHash,RegHashControl);
  //Busco en 'lista doble parcial' parametrizando el registro control y capturo
  //registro
  LO_ListaParcialDobleEnlace.BuscarInfo_LdParcial
  (me.Estructura.ListaCtasCtes,Clave,PosLista, RegHashControl);
  LO_ListaParcialDobleEnlace.CapturarInfo_LdParcial
  (me.Estructura.ListaCtasCtes,PosLista,Reg);
end;
{-------------------------------------------------------------------------------}
procedure CapturarAlmacen (var Me:ME_CUENTA; Pos:TPos; var Reg:TipoRegAlmacen);
begin
  seek(me.Datos.Almacen,Pos);
  read(me.Datos.Almacen,Reg);
end;

{-------------------------------------------------------------------------------}

//capturar registro en tabla indice de movimientos
procedure CapturarTablaIndiceMov (var Me:ME_CUENTA; Pos:TPos; var Reg:TipoRegControl);
begin
  seek(me.Datos.tablaControlIndice,pos);
  read(me.Datos.tablaControlIndice,Reg);
end;

{-------------------------------------------------------------------------------}

//Grabo registro en tabla indice de movimientos
```

```
procedure GrabarTablaIndiceMov (var Me:ME_CUENTA; Pos:TPos; Reg:TipoRegControl);
var
  RegIndMov:tipoRegControl;
begin
  seek(me.Datos.tablaControlIndice,pos);
  write(me.Datos.tablaControlIndice,Reg);
end;

{-------------------------------------------------------------------------------}

Function SaldoAcumulado  (var me:ME_CUENTA; Clave: tclave): timporte;
var
  debeacumulado, haberacumulado: timporte;
  regCtaCte,RegIndMovimiento:tipoRegDatos;
  pos,PosLista,i:Tpos;
  ClaveMayuscula:tClaveHash;
  regTablaIndice:tipoRegControl;
  RegAlm:TipoRegAlmacen;
begin
  ClaveMayuscula:= UpperCase(clave);
  CapturarCtaCte(me,ClaveMayuscula,regCtaCte);
  CapturarTablaIndiceMov(me,regCtaCte.PosEnDatos,regTablaIndice);

  i:=LO_ListaParcialDobleEnlace.Primero(me.Datos.ListaIndiceAlmacen, regTablaIndice);
  debeAcumulado:= 0;
  haberAcumulado:= 0;

  while (i<>Lib_Auxiliar.PosNula) do
  begin
    LO_ListaParcialDobleEnlace.CapturarInfo_LdParcial
    (me.Datos.ListaIndiceAlmacen,i,RegIndMovimiento);
    CapturarAlmacen(me,RegIndMovimiento.PosEnDatos,RegAlm);
    debeAcumulado:= debeAcumulado + RegAlm.Debe;
    haberAcumulado:= haberAcumulado + RegAlm.haber;
```

```
    i:=LO_ListaParcialDobleEnlace.Proximo(me.Datos.ListaIndiceAlmacen,i);
  end;
  SaldoAcumulado:= debeAcumulado-haberAcumulado;
end;

{-------------------------------------------------------------------------------}

Procedure Capturar_InfoAlmacen(var Me:ME_CUENTA; Clave:tClaveHash; var
RegAlm:TipoRegAlmacen);
var
  regCtaCte,RegIndMovimiento:tipoRegDatos;
  ClaveMayuscula:tClaveHash;
  regTablaIndice:tipoRegControl;
  posLista:tPos;
begin
  ClaveMayuscula:= UpperCase(clave);
  CapturarCtaCte(me,ClaveMayuscula,regCtaCte);
  CapturarTablaIndiceMov(me,regCtaCte.PosEnDatos,regTablaIndice);
  LO_ListaParcialDobleEnlace.CapturarInfo_LdParcial
  (me.Datos.ListaIndiceAlmacen,posLista,RegIndMovimiento);
  CapturarAlmacen(me,RegIndMovimiento.PosEnDatos,RegAlm);
end;

{-------------------------------------------------------------------------------}

Function PrimeroLdParcial(var Me:ME_CUENTA; Clave:tClaveHash):tPos;
var
  regCtaCte,RegIndMovimiento:tipoRegDatos;
  pos,PosLista,i:Tpos;
  ClaveMayuscula:tClaveHash;
  regTablaIndice:tipoRegControl;
  RegAlm:TipoRegAlmacen;
begin
  ClaveMayuscula:= UpperCase(clave);
  CapturarCtaCte(me,ClaveMayuscula,regCtaCte);
```

```
  CapturarTablaIndiceMov(me,regCtaCte.PosEnDatos,regTablaIndice);


  PrimeroLdParcial:=LO_ListaParcialDobleEnlace.Primero (me.Datos.ListaIndiceAlmacen,

  regTablaIndice);

end;


{--------------------------------------------------------------------------------}


procedure CapturarEnAlmacen (var Me:ME_CUENTA; Pos:TPos; var Reg:TipoRegAlmacen);

var

  RegIndMovimiento:tipoRegDatos;

begin

  LO_ListaParcialDobleEnlace.CapturarInfo_LdParcial
(me.Datos.ListaIndiceAlmacen,pos,RegIndMovimiento);

  seek(me.Datos.Almacen,RegIndMovimiento.PosEnDatos);

  read(me.Datos.Almacen,Reg);

end;


{--------------------------------------------------------------------------------}

function ProximoEnLdParcial (var Me:ME_CUENTA; Pos:TPos):TPos;

begin

   ProximoEnLdParcial:= LO_ListaParcialDobleEnlace.Proximo

   (me.Datos.ListaIndiceAlmacen,pos);

end;


{--------------------------------------------------------------------------------}


function PosNula (var Me:ME_CUENTA): TPos;

begin

  PosNula:= Lib_Auxiliar.PosNula;

end;


{--------------------------------------------------------------------------------}


procedure Asignar(var Me:ME_CUENTA);
```

```
begin
  AssignFile(me.Estructura.tablaHash, _Directorio + _NombreArch_TablaHash);
  AssignFile(me.Estructura.ListaCtasCtes.D, _Directorio + _NombreArch_LDCtaCte);
  AssignFile(Me.Datos.tablaControlIndice, _Directorio + _NombreArch_TablaIndMov);
  AssignFile(me.Datos.ListaIndiceAlmacen.D, _Directorio + _NombreArch_LDIndMov);
  AssignFile(Me.Datos.Almacen, _Directorio + _NombreArch_Almacen);
end;
END.
```

2.3.2.1 Type Almacén

```
unit Type_ALMACEN;

interface
USES
  Sysutils, dateutils;

Const
  //movimientos ctacte
  _creacionCtaCte = 'Creacion de cuenta';
  _bajaCtaCte = 'Baja de la cuenta';
  _CompraFicha = 'Compra de fichas';
  _PremioJuego = 'Premio por juego nro. ';
  _RegaloCredito = 'Regalo de credito';
  _Apuesta = 'Apuesta juego N° ';

type
   tClave = string[15];
   tFecha = TDateTime;
   tConcepto = string[50];
   tImporte = 0..maxint;

   tipoRegAlmacen = record
        Nick: Type_ALMACEN.tClave;
        FechaHora: Type_ALMACEN.tFecha;
```

```
        Concepto: Type_ALMACEN.tConcepto;
        Debe, haber, saldo: Type_ALMACEN.tImporte;
       end;

   //Archivo donde se almacenan los datos de la cuenta corriente
   tipoArchAlmacen = file of TipoRegAlmacen;

   //Tipo de conceptos
   tipoConceptos = record
      const Creacion_Cuenta = _creacionCtaCte;
      const Baja_cuenta  = _bajaCtaCte;
      const Compra_Fichas = _CompraFicha;
      const Premio_Juego    = _PremioJuego;
      const Credito_Regalado = _RegaloCredito;
      const Numero_Apuesta = _Apuesta;
    end;

   implementation
END.
```

2.3.3 Juego

```
unit ME_JUEGO;
```

```
{

Este ME consta con un archivo de datos denominado RULETA.DAT organizado a través de un

TDA Cola, donde se almacena los datos de cada jugada (NroJugada, FechaHora, Estado, Bolilla)

También dispone de un archivo de control denominado RULETA.CON, donde se tiene control sobre el juego (maxima apuesta y minima apuesta)

Este ME al utilizar un TDA de colas, se inserta siempre al final, se elimina el primer elemento, y se captura al frente

Ya sea para UltimaJugada (que cicla hasta encontrar la ultima jugada) o ModificoUltimo(que modifica el ultimo registro con los datos actuales de la jugada parametrizada) se insertar un registroBandera que permite ciclar y funcionar de corte sin cambiar el orden de la cola

}


interface


Uses
  LO_Colas, LO_ListaSimpleEnlace, Type_JUEGO, Lib_Auxiliar, Dialogs, SysUtils;



Const
  _Directorio = 'Archivos\';
  _NombreArchDatos = 'RULETA.DAT';
  _NombreArchControl = 'RULETA.CON';
  _NombreArchColasDatos = 'COLASRULETA.DAT';
  _NombreArchColasControl = 'COLASRULETA.CON';


Type


      ME_RULETA = Record
                     E: LO_Colas.tCola;
                    D: Type_JUEGO.TipoArchivoDatos;
                     C: Type_JUEGO.tipoArchControlJuego;  //Control sobre el juego
                  End;



  Procedure CrearME (Var Me:ME_RULETA);
```

```
  procedure AbrirME (var Me:ME_RULETA);
  procedure CerrarME (var Me: ME_RULETA);
  procedure DestruirME (var Me: ME_RULETA);
  function MEVacio(var Me: ME_RULETA): Boolean;
  Procedure Insertar (Var Me: ME_RULETA; Reg: TipoRegDatos);
  Procedure Frente(Var Me: ME_RULETA; var regDat: TipoRegDatos );
  Procedure Decolar(Var Me: ME_RULETA );
  function Presente(var ME: ME_RULETA; reg: TipoRegDatos): Boolean ;
  procedure RegistroNulo(var Nulo:TipoRegDatos);
  function ClaveNula(var Me: ME_RULETA):TClave;
  function ApuestaMaxima(var Me: ME_RULETA):tImporte;
  function ApuestaMinima(var Me: ME_RULETA):tImporte;
  Procedure UltimaJugada(var Me:ME_RULETA; var reg:type_JUEGO.TipoRegDatos);
  procedure ModificoUltimo(Me: ME_RULETA; reg:type_JUEGO.TipoRegDatos);
  procedure Asignar(var Me:ME_RULETA);



implementation

{-------------------------------------------------------------------------------}

Procedure CrearME (Var Me:ME_RULETA);
Var
      FaltaControl:Boolean;
      FaltaDatos:Boolean;
  rc:TipoRegControlJuego;
Begin
 AssignFile(Me.D,_Directorio+_NombreArchDatos);
  {$i-}
 Reset(Me.D);
 FaltaDatos:=(Ioresult<>0);

  If (FaltaDatos) then
  begin
   Rewrite(Me.D); Close(Me.D);
```

```
  end;

  AssignFile(Me.C,_Directorio+_NombreArchControl);
      {$i-}
      Reset(Me.C);
      FaltaControl:=(Ioresult<>0);
      If (FaltaControl) then
    begin
      Rewrite(Me.C);
       //Se incializan las variables de control
      Rc.MaximaApuesta:= Type_JUEGO.MaximaApuesta;
      Rc.MinimaApuesta := Type_JUEGO.MinimaApuesta;

      seek(Me.C,0); Write(Me.C,rc);

              Close(Me.C);
      Close(Me.D);
    end;

 LO_Colas.CrearCola(Me.E, _NombreArchColasControl, _NombreArchColasDatos, _Directorio);
End;

{-------------------------------------------------------------------------------}

procedure AbrirME (var Me:ME_RULETA);
var
  rc: TipoRegControlJuego;
begin
  Asignar(Me);
  reset(Me.D);
  reset(Me.C);
  LO_Colas.AbrirCola(Me.E);
end;
```

```
{--------------------------------------------------------------------------}

procedure CerrarME (var Me:ME_RULETA);
begin
  Close(Me.D);
  Close(Me.C);
  LO_Colas.CerrarCola(Me.E);
end;

{--------------------------------------------------------------------------}

procedure DestruirME (var Me:ME_RULETA);
begin
  Erase(Me.D);
  Erase(Me.C);
  LO_Colas.DestruirCola(Me.E);
end;

{--------------------------------------------------------------------------}

function Presente(var ME: ME_RULETA; reg: TipoRegDatos): Boolean ;
begin
//Busca si la jugada pasada en el reg esta en la cola
  presente:=  LO_Colas.Buscar(me.E, reg.NroJugada);
end;

{--------------------------------------------------------------------------}

function MEVacio(var Me:ME_RULETA): Boolean;
begin
  MEVacio:= LO_Colas.ColaVacia(Me.E);
end;

{--------------------------------------------------------------------------}
```

```
Procedure Insertar (Var Me: ME_RULETA; Reg: TipoRegDatos);
var
  RegET: LO_Colas.TipoRegDatos;
  RegControl: TipoRegControl;
begin
  //Se inserta al final del archivo
  RegET.PosEnDatos:= FileSize(Me.D);
  RegET.Clave:= (Reg.NroJugada);
  LO_Colas.Encolar(Me.E,RegET);
  Seek (Me.D,RegET.PosEnDatos); Write(Me.D,Reg);
end;
{-------------------------------------------------------------------------------}

Procedure Frente(Var Me:ME_RULETA; var regDat: TipoRegDatos );
var
RegAux: LO_Colas.TipoRegDatos;
Begin
//Tomo el primer valor de la estructura
   LO_Colas.Frente(me.E,RegAux);
   seek(me.D,RegAux.PosEnDatos);
   read(me.D,regDat);
End;

{-------------------------------------------------------------------------------}

Procedure Decolar(Var Me: ME_RULETA );
Begin
//Elimino el primer valor de la estructura
   LO_Colas.Decolar(me.E);
End;

{-------------------------------------------------------------------------------}
//Inicializo el registro nulo que lo utilizo de bandera en la busqueda
procedure RegistroNulo(var Nulo:TipoRegDatos);
begin
```

```
  nulo.NroJugada:= Lib_Auxiliar.clavenula;

  nulo.Estado:= -1;
  nulo.bolilla:= -1;
end;

{-------------------------------------------------------------------------------}

function ClaveNula(var Me:ME_RULETA):TClave;

begin

  //Devuelve la clave nula
  ClaveNula:=LO_Colas.ClaveNula(Me.E);
end;

{-------------------------------------------------------------------------------}

function ApuestaMaxima(var Me:ME_RULETA):tImporte;
var
  RegControl: tiporegcontrolJuego;
begin

  //Devuelve el valor del importe de la apuesta maxima

  seek(me.C, 0); Read(me.C, RegControl);

  ApuestaMaxima:= regcontrol.MaximaApuesta;
end;

{-------------------------------------------------------------------------------}

function ApuestaMinima(var Me:ME_RULETA):tImporte;
var
RegControl: tiporegcontroljuego;
begin
```

```
  //Devuelve el valor del importe de la apuesta minima

  seek(Me.C,0); Read(Me.C, RegControl);

  ApuestaMinima:= regcontrol.MinimaApuesta;
end;

{-------------------------------------------------------------------------------}

//Devuelve el reg de la ultima jugada
Procedure UltimaJugada(var Me:ME_RULETA; var reg:type_JUEGO.TipoRegDatos);
var
regAux: TipoRegDatos;
regnulo: TipoRegDatos;
begin

  ME_JUEGO.RegistroNulo(regnulo);

  ME_JUEGO.Insertar(me, regnulo);
  ME_JUEGO.Frente(me,regAux);
  ME_JUEGO.Decolar(me);

  //Mientras no me encuentre con el RegBandera, que tiene como clave -1,
  //voy sobreescribiendo reg que pase por referencia hasta que el ciclo se encuentra de
frente con el RegBandera y lo decole
  //Es asi que los valores que se guardaran en reg seran del ultimo registro de la cola
  while regAux.NroJugada<>ME_JUEGO.ClaveNula(me) do
  begin
    reg.NroJugada:= regAux.NroJugada;
    reg.FechaHora:= regAux.FechaHora;
    reg.Estado:=  regAux.Estado;
    reg.Bolilla:= regAux.Bolilla;

    ME_JUEGO.Insertar(me, regAux);
    ME_JUEGO.Frente(me,regAux);
    ME_JUEGO.Decolar(me);
```

```
  end;
end;


{--------------------------------------------------------------------------------}
//Modifico el ultimo registro con los datos actuales de la jugada que le parametrizo en
reg
procedure ModificoUltimo(Me: ME_RULETA; reg:type_JUEGO.TipoRegDatos);
var
regAux, regauxUltimo: TipoRegDatos;
regMod:TipoRegDatos;
regNulo: TipoRegDatos;
begin
if (Presente(me,reg)) then//el registro que le paso esta en la cola
    begin
      RegistroNulo(regnulo);
      Insertar(me, regnulo);
      Frente(me,regaux);
      Decolar(me);

      while regaux.NroJugada<>ClaveNula(me) do//me encontre el registro bandera
      begin
        if (regaux.NroJugada = reg.NroJugada) then
        begin
          regaux.FechaHora:= reg.FechaHora;
          regaux.Estado:= reg.Estado;
          regaux.Bolilla:= reg.Bolilla;

          Insertar(me,regaux);
          Frente(me,regaux);
          Decolar(me);
        end
        else
        begin //no es el registro que busco
          Insertar(me,regaux);
          Frente(me,regaux);
```

```
          Decolar(me);
        end;
      end;

    end;
end;
{-------------------------------------------------------------------------------}

procedure Asignar(var Me:ME_RULETA);
begin
  AssignFile(Me.D, _Directorio + _NombreArchDatos);
  AssignFile(Me.C, _Directorio + _NombreArchControl);
  AssignFile(Me.E.D, _Directorio + _NombreArchColasDatos);
  AssignFile(Me.E.C, _Directorio + _NombreArchColasControl);
end;

END.
```

2.3.3.1 Type Juego

```
unit Type_JUEGO;

interface
Const
    MaximaApuesta = 1000; //es el importe máximo de una apuesta. Por defecto 1000
pesos.
    MinimaApuesta = 50; //es el importe mínimo de una apuesta. Por defecto 50 pesos.

Type
     tImporte = 0..maxint;
     tClave = string[6];
     tFecha = TDateTime;
     tEstados = -1..4; {Juego no creado(-1)/ Juego creado(0)/ Hagan sus apuestas(1)/ no
va mas(2)/ bolilla tirada (3)/ premios repartidos y fin de juego(4)};
     tBolilla = -1..36; //identifica a la bolilla que ha salido. Se inicializa en -1
(no ha salido la bolilla)
```

```
     TipoRegControlJuego = record
                     MaximaApuesta, MinimaApuesta: tImporte;
                   end;

    tipoArchControlJuego = File of TipoRegControlJuego;

    TipoRegDatos = Record
              NroJugada: TClave;
              FechaHora: tFecha;
              Estado: tEstados;
              Bolilla: tBolilla;
            end;

    TipoArchivoDatos = File Of TipoRegDatos;

implementation

END.
```

2.3.4 Apuestas

```
unit ME_APUESTAS;

{El ME APUESTAS guarda todas las apuestas efectuadas en el juego, tanto de jugadores
reales como ficticios.
Utiliza un TDA de lista doble}

interface

Uses
  Type_APUESTA, LO_ListaDobleEnlace, Lib_Auxiliar, Dialogs;

Const
  _Directorio = 'Archivos\';
```

```
  _NombreArchDatos = 'APUESTAS.DAT';
  _NombreArchApuestasLDDatos = 'APUESTASLD.DAT';
  _NombreArchApuestasLDControl = 'APUESTASLD.CON';



Type
    ME_APUESTA = Record
           E:LO_ListaDobleEnlace.tListaDoble;
           D:Type_APUESTA.TipoArchDatos;
         End;


Procedure CrearME (Var Me:ME_APUESTA);
procedure AbrirME_Apuestas   (var Me: ME_APUESTA);
procedure CerrarME_Apuestas  (var Me: ME_APUESTA);
procedure DestruirME_Apuestas (var Me: ME_APUESTA);
function MEVacio_Apuestas(me:ME_APUESTA):boolean;
Procedure InsertarInfoME_Apuestas (var Me:ME_APUESTA;
Reg:Type_APUESTA.TipoRegDatos;Pos:tPos);
procedure EliminarInfoME_Apuestas (var Me:ME_APUESTA; Pos:tPos);
procedure CapturarInfoME_Apuestas (var Me:ME_APUESTA; Pos:tPos; var
Reg:Type_APUESTA.TipoRegDatos);
procedure ModificarInfoME_Apuestas (var Me:ME_APUESTA; Pos:tPos;
Reg:Type_APUESTA.TipoRegDatos);
function PosNula_Apuestas (var Me:ME_APUESTA): tPos;
function ClaveNula_Apuestas(var Me:ME_APUESTA): tClave;
Function BuscarInfoME_Apuestas(var Me:ME_APUESTA;Clave:tClave;Var pos:tPos):Boolean;
function Primero (var Me:ME_APUESTA): tPos;
function Ultimo  (var Me:ME_APUESTA): tPos;
Function Anterior(var Me:ME_APUESTA; Pos:tPos):tPos;
function Proximo (var Me:ME_APUESTA; Pos:tPos):tPos;
function SaldoApuestasPartida (var Me:ME_APUESTA; Clave: tclave; Jugada: tClave):
tImporte;
procedure Asignar(var Me:ME_APUESTA);
```

```
Implementation


Procedure CrearME (Var Me:ME_APUESTA);
Var
      FaltaControl:Boolean;
      FaltaDatos:Boolean;
  rc:TipoRegControl;
Begin
 AssignFile(Me.D,_Directorio+'APUESTAS.DAT');
  {$i-}
 Reset(Me.D);
 FaltaDatos:=(ioResult<>0);
 If (FaltaDatos) then
  begin
   Rewrite(Me.D); Close(Me.D);
  end;

 LO_ListaDobleEnlace.CrearLd(Me.E, 'APUESTASLD',_Directorio);

End;

{-------------------------------------------------------------------------------}

procedure AbrirME_Apuestas (var Me: ME_APUESTA);
begin
  Asignar(Me);
  reset (Me.D);
  LO_ListaDobleEnlace.AbrirLd(me.E);
end;

{-------------------------------------------------------------------------------}

procedure CerrarME_Apuestas  (var Me: ME_APUESTA);
begin
  Close (Me.D);
```

```
  LO_ListaDobleEnlace.CerrarLd(me.E)
end;

{--------------------------------------------------------------------------------}



procedure DestruirME_Apuestas (var Me: ME_APUESTA);
begin
  Erase (Me.D);
  LO_ListaDobleEnlace.DestruirLista(me.E);
end;

{--------------------------------------------------------------------------------}

function MEVacio_Apuestas(me:ME_APUESTA):boolean;
begin
 MEVacio_Apuestas:=  LO_ListaDobleEnlace.ListaVacia(me.E);
end;

{--------------------------------------------------------------------------------}

Procedure InsertarInfoME_Apuestas (var Me:ME_APUESTA;
Reg:Type_APUESTA.TipoRegDatos;Pos:tPos);
var
  RegEst: LO_ListaDobleEnlace.TipoRegDatos;
  Raux,RauxAnt,RauxSig:TipoRegDatos;
  PosenDatos:TPos;
begin
  //obtengo final del archivo almacen donde se guardarà la apuesta
  PosenDatos:= filesize(Me.D);

  //se completa registro para insertar en la lista doble con  pos apuntando al registro
  //en el almacen
  RegEst.PosenDatos:= PosenDatos;
```

```
  RegEst.Clave:= Reg.NroJugada;


  //inserto en lista doble
  LO_ListaDobleEnlace.Insertar(Me.E,RegEst,Pos);



  //grabo en almacen
  Seek (Me.D,RegEst.PosEnDatos); Write(Me.D,Reg);
end;

{-------------------------------------------------------------------------------}

procedure EliminarInfoME_Apuestas (var Me:ME_APUESTA; Pos:tPos);
begin
  LO_ListaDobleEnlace.Eliminar(me.E,pos);
end;

{-------------------------------------------------------------------------------}

procedure CapturarInfoME_Apuestas(var Me:ME_APUESTA; Pos:tPos; var
Reg:Type_APUESTA.TipoRegDatos);
var
  RE: LO_ListaDobleEnlace.TipoRegDatos;
begin
  //Accediendo a la lista en la posicion parametrizada, obtengo la posicion en datos
  //(posicion que corresponde al archivo almacen)
  LO_ListaDobleEnlace.Capturar(ME.E,Pos,RE);
  Seek(Me.D,RE.PosEnDatos); Read(Me.D,Reg);
end;

{-------------------------------------------------------------------------------}

procedure ModificarInfoME_Apuestas(var Me:ME_APUESTA; Pos:tPos;
Reg:Type_APUESTA.TipoRegDatos);
var
  RE:LO_ListaDobleEnlace.TipoRegDatos;
```

```
begin
{Cuando se quiere modificar se pide la posicion de la estructura, se lee el registro de
la estructura que trae la posicion en datos del alamcen con la informacion y ese
registro sera modificado}
  Seek(Me.E.D,pos);
  Read(Me.E.D,RE);

  Seek(Me.D,RE.PosEnDatos);
  Write(Me.D,Reg);
end;

{-------------------------------------------------------------------------------}

function Primero(var Me:ME_APUESTA):TPos;
var
Rc:TipoRegControl;
begin
  //Obtengo la posicion del primero que se encuentra en la estructura
  Primero:=LO_ListaDobleEnlace.primero(me.e)
end;

{-------------------------------------------------------------------------------}

function Ultimo (var Me:ME_APUESTA):TPos;
var
Rc:TipoRegControl;
begin
  //Obtengo la posicion del ultimo que se encuentra en la estructura
  Ultimo:=LO_ListaDobleEnlace.Ultimo(me.e);
end;
{-------------------------------------------------------------------------------}
function PosNula_Apuestas (var Me:ME_APUESTA): tPos;
begin
  //Obtengo la posicion nula de la estructura
  PosNula_Apuestas:=LO_ListaDobleEnlace.PosNula(me.e)
end;
```

```
{-------------------------------------------------------------------------------}
function ClaveNula_Apuestas(var Me:ME_APUESTA): tClave;
begin //Obtengo la clave nula de la estructura
  ClaveNula_Apuestas:=LO_ListaDobleEnlace.ClaveNula(me.e);
end;

{-------------------------------------------------------------------------------}

function Proximo(var Me:ME_APUESTA;Pos:TPos):TPos;
begin
  //Dada una posicion, obtengo la posicion del proximo que se encuentra en la
estructura
  Proximo:=LO_ListaDobleEnlace.Proximo(me.e, Pos);
end;
{-------------------------------------------------------------------------------}

Function Anterior(var Me:ME_APUESTA;Pos:TPos):TPos;
begin
  //Dada una posicion, obtengo la posicion del anterior que se encuentra en la
estructura
  Anterior:=LO_ListaDobleEnlace.Anterior(me.e, Pos);
end;

{-------------------------------------------------------------------------------}

Function BuscarInfoME_Apuestas(var Me:ME_APUESTA;Clave:tClave;Var pos:tPos):Boolean;
{Con la clave parametrizada se busca en la lista doble, si se encuentra devuelve la
posicion sino la encuentra devuelve la posicion donde deberia estar esa clave}
Begin
  BuscarInfoME_Apuestas:= LO_ListaDobleEnlace.Buscar(me.E,clave,pos);
end;

{-------------------------------------------------------------------------------}

//Dada una clave y un numero de jugada, obtengo el saldo total apostado por ese jugador
//en esa jugada
```

```
function SaldoApuestasPartida (var Me:ME_APUESTA; Clave: tclave; Jugada: tClave):
tImporte;

var

i: tpos;

Reg: Type_APUESTA.TipoRegDatos;

Importe: tImporte;

begin

  Importe:= 0;

  i:= Primero(me);

  while i<>PosNula_Apuestas(me) do

  begin

    CapturarInfoME_Apuestas(me,i,Reg);


    if (reg.NroJugada= Jugada) and (reg.Nick=Clave) then

    begin

      Importe:=Importe+reg.Importe;

    end;

    i:= Proximo(me, i);

  end;


 SaldoApuestasPartida:= Importe;

end;


{-------------------------------------------------------------------------------}

procedure Asignar(var Me:ME_APUESTA);

begin

  AssignFile(Me.D,_Directorio+_NombreArchDatos);

  AssignFile(Me.E.D,_Directorio+ _NombreArchApuestasLDDatos);

  AssignFile(Me.E.C,_Directorio+ _NombreArchApuestasLDControl);

end;


END.
```

2.3.4.1 Type Apuesta

```
unit Type_APUESTA;


interface
USES
  Sysutils;
CONST
  //Nomencladores
 Nom_Rojonegro = 'RN';
 Nom_ParImpar= 'PI';
 Nom_PasaFalta= 'PF';
 Nom_Docena= 'DO';
 Nom_Columna= 'CO';
 Nom_Pleno= 'XX';


TYPE
     tClaveApuesta = string[15];
     tNomenclador = string[2];
     tValor = string[2];
     tImporte = 0..maxint;


      TipoRegDatos = Record
              NroJugada:tClaveApuesta;
              Nick:tClaveApuesta;
              Nomenclador: tNomenclador;
              Valor: tValor;
              Importe: tImporte;
            end;
  TipoArchDatos = File Of TipoRegDatos;
implementation
END.
```

2.3.5 Ganadores

```
unit ME_GANADORES;

{

En este método se recolectan los ganadores de cada juego y las jugadas ganadas de cada
jugador

Cada nuevo ganador se inserta en el arbol

Con Hijo Medio se enganchan todas las jugadas (ganadoras) en las que intervino el
jugador, por ello hijo medio es un registro de control que apunta hacia una lista
parcial doble enlace donde se guardan estas jugadas ganadas por el jugador ganador.

Si el jugador ganador no esta en el arbol, se agrega como nodo en el arbol y se
insertar en la lista

Si ya se encuentra en el arbol se inserta solo en la lista y se actualiza el registro
de control

}

interface


Uses
  Lib_Auxiliar, LO_ArbolTrinario, LO_ListaParcialDobleEnlace, Type_GANADOR;


CONST
  _Directorio = 'Archivos\';
  _NombreArchivoDatos = 'GANADORES.DAT';
  _NombreArchControl_ArbolTrinario = 'GANADORES.CON';
  _NombreArchIndice_ArbolTrinario = 'GANADORES.NTX';
  _NombreListaIndiceDatos = 'INDICEGANADORES.IND';


TYPE
      regNodo = LO_ArbolTrinario.tNodoIndice;


      ME_GANADOR = Record
          Almacen: Type_GANADOR.TipoArchivoDatos;  //datos
          //lista indice sobre archivo almacen de datos
          ListaIndiceAlmacen: LO_ListaParcialDobleEnlace.tLdParcial;
          ArbolTri: LO_ArbolTrinario.tArbolTrinario; //estructura arbol
      end;
```

```
  procedure CrearME_Ganadores (var Me: ME_GANADOR);

  Procedure AbrirME_Ganadores(var Me:ME_GANADOR);

  Procedure CerrarME_Ganadores(var Me:ME_GANADOR);

  Procedure DestruirME_Ganadores(var Me:ME_GANADOR);

  Function MeVacio_Ganadores(var Me: ME_GANADOR):Boolean;

  Procedure InsertarInfoME_Ganadores(var Me: ME_GANADOR;
RegAlm:Type_GANADOR.TipoRegDatos; Clave:tClave);

  Procedure CapturarInfoME_Ganadores(var Me: ME_GANADOR; var
nodo:LO_ArbolTrinario.tNodoIndice; Pos:TPos);

  Function BuscarInfoME_Ganadores(Var Me:ME_GANADOR; Var Pos:TPos;
Clave:TClave):Boolean;

  function PosNula_Ganadores(Var Me:ME_GANADOR):TPos;

  function ProximoIzq (var Me:ME_GANADOR; pos: TPos): TPos;

  function ProximoDer (var Me:ME_GANADOR; pos: TPos): TPos;

  function Raiz(var Me: ME_GANADOR): TPos;

  function PrimerJugada_Ganador(var Me:ME_GANADOR;
nodo:LO_ArbolTrinario.tNodoIndice):TPos;

  function SiguienteJugada_Ganador(var Me:ME_GANADOR; pos:TPos): TPos;

  procedure CapturarJugada_Ganador(var Me:ME_GANADOR; pos:TPos; var
reg:Type_GANADOR.TipoRegDatos);

  function Cantidad_Ganadores(var Me: ME_GANADOR): tCantidad;

  function ObtenerNivel_enArbolTri(var Me:ME_GANADOR; clave:tClave): tCantidad;

  procedure Asignar(var Me:ME_GANADOR);


IMPLEMENTATION


{-------------------------------------------------------------------------------}


procedure CrearME_Ganadores (var Me: ME_GANADOR);

var

  ioD:integer;

Begin

  assignFile (Me.Almacen, _Directorio + _NombreArchivoDatos);

  {$i-}

  reset(Me.Almacen); ioD:=IoResult; //Si IoResult es cero, la operacion es exitosa

  if (IoD<>0)then

    Rewrite(Me.Almacen);
```

```
    Close(Me.Almacen);
  {$I+}
LO_ListaParcialDobleEnlace.CrearLdParcial(Me.ListaIndiceAlmacen,_NombreListaIndiceDatos
,_Directorio);

LO_ArbolTrinario.CrearMe_ArbolTri(Me.ArbolTri,_NombreArchControl_ArbolTrinario,_NombreA
rchIndice_ArbolTrinario,_Directorio);
end;

{-------------------------------------------------------------------------------}

Procedure AbrirME_Ganadores(var Me:ME_GANADOR);
Begin
  Asignar(Me);
  reset(Me.Almacen);
  LO_ListaParcialDobleEnlace.AbrirLdParcial(Me.ListaIndiceAlmacen);
  LO_ArbolTrinario.AbrirMe_ArbolTri(Me.ArbolTri);
end;

{-------------------------------------------------------------------------------}

Procedure CerrarME_Ganadores(var Me:ME_GANADOR);
Begin
  close(Me.Almacen);
  LO_ListaParcialDobleEnlace.CerrarLdParcial(Me.ListaIndiceAlmacen);
  LO_ArbolTrinario.CerrarMe_ArbolTri(Me.ArbolTri);
end;

{-------------------------------------------------------------------------------}

Procedure DestruirME_Ganadores(var Me:ME_GANADOR);
Begin
  erase(Me.Almacen);
  LO_ListaParcialDobleEnlace.DestruirLdParcial(Me.ListaIndiceAlmacen);
  LO_ArbolTrinario.DestruirMe_ArbolTri(Me.ArbolTri);
end;
```

```
{-------------------------------------------------------------------------------}




Function MeVacio_Ganadores(var Me: ME_GANADOR):Boolean;

begin

   MeVacio_Ganadores:= LO_ArbolTrinario.Arbol_Vacio(me.arbolTri);

end;


{-------------------------------------------------------------------------------}


Procedure InsertarInfoME_Ganadores(var Me: ME_GANADOR;
RegAlm:Type_GANADOR.TipoRegDatos; Clave:tClave);

var

   PosNodo, PosLista, PosFinalAlm: TPos;

   nodoAux: LO_ArbolTrinario.tNodoIndice;

   regIndAlm: LO_ListaParcialDobleEnlace.TipoRegDatos;

   RegDatos:Type_GANADOR.TipoRegDatos;

   nodo:LO_ArbolTrinario.tNodoIndice;

Begin

  //Busco en arbol trinario

  if (LO_ArbolTrinario.BuscarNodo_ArbolTri(Me.ArbolTri,clave,PosNodo)) then //Si existe
//el nodo entonces...

      begin

        //Caputuro nodo en arbol

        LO_ArbolTrinario.CapturarNodo_ArbolTri(Me.ArbolTri,nodoAux,PosNodo);


        //A PosFinalAlm le doy el final del archivo de almacen de datos

        PosFinalAlm:=FileSize(Me.Almacen);

        regIndAlm.PosEnDatos:= PosFinalAlm;

        regIndAlm.Clave:=clave;


        //Busco la posicion en la lista indice datos e inserto en la lista Indice datos
//pasandole por parametro el regControl que es el campo HijoMedio
```

```
        LO_ListaParcialDobleEnlace.BuscarInfo_LdParcial(Me.ListaIndiceAlmacen,Clave,
PosLista, nodoAux.HijoMedio);

        LO_ListaParcialDobleEnlace.InsertarInfo_LdParcial(Me.ListaIndiceAlmacen,
regIndAlm, PosLista, nodoAux.HijoMedio);



        //Inserto al final del archivo almacen.

        Seek (me.Almacen,PosFinalAlm);

        Write(me.Almacen,RegAlm);


        //Busco la posicion nuevamente del arbol y lo modifico con campo HijoMedio
//actualizado

        LO_ArbolTrinario.BuscarNodo_ArbolTri(Me.ArbolTri,nodoAux.clave,PosNodo);

        LO_ArbolTrinario.ModificarNodo_ArbolTri(Me.ArbolTri,nodoAux,PosNodo);

      end

      else //No existe el nodo, entonces...

        begin

            nodo.clave:=Clave;

            //Inicializo la cabecera del hijo del medio

            LO_ListaParcialDobleEnlace.InicializarCabecera(nodo.HijoMedio);

           //No existe en el árbol el nodo lo inserto

            LO_ArbolTrinario.InsertarNodo_ArbolTri(Me.ArbolTri,nodo,PosNodo);


            //A PosFinalAlm le doy el final del archivo de almacen de datos

            PosFinalAlm:=FileSize(Me.Almacen);

            regIndAlm.PosEnDatos:= PosFinalAlm;

            regIndAlm.Clave:=clave;


          //Busco la posicion en la lista indice datos e inserto en la lista Indice
//datos pasandole por parametro el regControl que es el campo HijoMedio

            LO_ListaParcialDobleEnlace.BuscarInfo_LdParcial

            (Me.ListaIndiceAlmacen,Clave, PosLista, nodo.HijoMedio);

            LO_ListaParcialDobleEnlace.InsertarInfo_LdParcial (Me.ListaIndiceAlmacen,

            regIndAlm, PosLista, nodo.HijoMedio);


            //Inserto al final del archivo almacen.
```

```
            Seek (me.Almacen,PosFinalAlm);
            Write(me.Almacen,RegAlm);

            //Busco la posicion nuevamente del arbol y lo modifico con campo HijoMedio
//actualizado
            LO_ArbolTrinario.BuscarNodo_ArbolTri(Me.ArbolTri,nodo.clave,PosNodo);
            LO_ArbolTrinario.ModificarNodo_ArbolTri(Me.ArbolTri,nodo,PosNodo);
        end;
end;

{-------------------------------------------------------------------------------}

Procedure CapturarInfoME_Ganadores(var Me: ME_GANADOR; var
nodo:LO_ArbolTrinario.tNodoIndice; Pos:TPos);
begin
   LO_ArbolTrinario.CapturarNodo_ArbolTri(me.ArbolTri, nodo, pos);
end;

{-------------------------------------------------------------------------------}

Function BuscarInfoME_Ganadores(Var Me:ME_GANADOR; Var Pos:TPos; Clave:TClave):Boolean;
begin
 BuscarInfoME_Ganadores:=LO_ArbolTrinario.BuscarNodo_ArbolTri(me.ArbolTri,clave,pos);
end;

{-------------------------------------------------------------------------------}

function PosNula_Ganadores(var Me:ME_GANADOR):TPos;
begin
  PosNula_Ganadores:= LO_ArbolTrinario.PosNula_ArbolTri(me.arbolTri);
end;

{-------------------------------------------------------------------------------}

function Raiz (var Me: ME_GANADOR): TPos;
begin
```

```
  Raiz:= LO_ArbolTrinario.Raiz_ArbolTri(me.arbolTri);
end;

{-------------------------------------------------------------------------------}



function PrimerJugada_Ganador(var Me:ME_GANADOR;
nodo:LO_ArbolTrinario.tNodoIndice):TPos;
var
  pos: tpos;
begin
  LO_ArbolTrinario.BuscarNodo_ArbolTri(ME.ArbolTri, nodo.clave, pos);
  LO_ArbolTrinario.CapturarNodo_ArbolTri(ME.ArbolTri, nodo, pos);


  PrimerJugada_Ganador:= LO_ListaParcialDobleEnlace.Primero(ME.ListaIndiceAlmacen,
nodo.HijoMedio);
end;

{-------------------------------------------------------------------------------}


function SiguienteJugada_Ganador(var Me:ME_GANADOR; pos:TPos): TPos;
begin
SiguienteJugada_Ganador:=LO_ListaParcialDobleEnlace.Proximo(ME.ListaIndiceAlmacen,pos);
end;

{-------------------------------------------------------------------------------}


procedure CapturarJugada_Ganador(var Me:ME_GANADOR; pos:TPos; var
reg:Type_GANADOR.TipoRegDatos);
begin
 Seek(Me.Almacen, pos);
 Read(Me.Almacen, reg);
end;

{-------------------------------------------------------------------------------}
```

```
function ProximoIzq (var Me:ME_GANADOR; pos: TPos): TPos;
begin
  ProximoIzq:= LO_ArbolTrinario.HijoIzq_ArbolTri(Me.ArbolTri, pos);
end;

{-------------------------------------------------------------------------------}
function ProximoDer (var Me:ME_GANADOR; pos: TPos): TPos;
begin
  ProximoDer:= LO_ArbolTrinario.HijoDer_ArbolTri(Me.ArbolTri, pos);
end;

{-------------------------------------------------------------------------------}

function Cantidad_Ganadores(var Me: ME_GANADOR): tCantidad;

begin
  Cantidad_Ganadores:= LO_ArbolTrinario.CantidadNodos_Arbol(me.ArbolTri);
end;

{-------------------------------------------------------------------------------}

function ObtenerNivel_enArbolTri(var Me:ME_GANADOR; clave:tClave): tCantidad;
begin
  ObtenerNivel_enArbolTri:= LO_ArbolTrinario.ObtenerNivel_ArbolTri(Me.ArbolTri, clave);
end;

{-------------------------------------------------------------------------------}

procedure Asignar(var Me:ME_GANADOR);
begin
  AssignFile(Me.Almacen, _Directorio + _NombreArchivoDatos);
  AssignFile(me.ListaIndiceAlmacen.D, _Directorio + _NombreListaIndiceDatos);
  AssignFile(Me.ArbolTri.D, _Directorio + _NombreArchIndice_ArbolTrinario);
  AssignFile(Me.ArbolTri.C, _Directorio + _NombreArchControl_ArbolTrinario);
end;
```

```
END.
```

2.3.5.1 Type Ganador

```
unit Type_GANADOR;
interface
USES
  Sysutils;

CONST
//Valor del premio segun el nomenclador
  Prem_Cero = 0.5;
  Prem_Rojonegro = 1;
  Prem_ParImpar= 1;
  Prem_PasaFalta= 1;
  Prem_Docena= 2;
  Prem_Columna= 2;
  Prem_Pleno= 35;

TYPE
  tNomenclador = string[2];
  tNumero = -1..36;
  tImporte = 0..maxint;
  tNumeroJugada = 0.. maxint;

  TipoRegDatos = Record
              Nomenclador:tNomenclador;
              Numero: tNumero; //Solo si el nomenclador es pleno (XX) el resto -1
              Importe: tImporte;
              NroJugada: tNumeroJugada;
          end;
```

```
  TipoArchivoDatos = File Of TipoRegDatos;


implementation

END.
```

3 Proyecto

3.1 Librería Auxiliar Juego

```
unit Lib_AuxJuego;


interface

uses

  ME_JUGADORES,

  Type_JUGADOR,

  ME_CTACTE,

  Type_ALMACEN,

  ME_JUEGO,

  Type_JUEGO,

  ME_APUESTAS,

  Type_APUESTA,

  ME_GANADORES,

  Type_GANADOR,

  Lib_Auxiliar,

  SysUtils, Vcl.Dialogs, Winapi.Windows, Winapi.Messages, System.Variants,
System.Classes, Vcl.Graphics,

  Vcl.Controls, Vcl.Forms,  Vcl.ExtCtrls, Vcl.StdCtrls, Vcl.Buttons, Vcl.Grids, Math;


//Consultas sobre rango de valores a la que puede llegar a pertenecer la bolilla
ganadora. Si se encuentra devuelve true

function IsIn_PrimeraColumna(BolillaGanadora: tBolilla): boolean;

function IsIn_SegundaColumna(BolillaGanadora: tBolilla): boolean;

function IsIn_TerceraColumna(BolillaGanadora: tBolilla): boolean;

function IsIn_PrimerDocena(BolillaGanadora: tbolilla): boolean;

function IsIn_SegundaDocena(BolillaGanadora: tbolilla): boolean;

function IsIn_TercerDocena(BolillaGanadora: tbolilla): boolean;
```

```
function IsIn_PASA(BolillaGanadora: tbolilla): boolean;

function IsIn_FALTA(BolillaGanadora: tbolilla): boolean;

function IsIn_PAR(BolillaGanadora: tbolilla): boolean;

function IsIn_IMPAR(BolillaGanadora: tbolilla): boolean;

function IsIn_ROJOS(BolillaGanadora: tbolilla): boolean;

function IsIn_NEGROS(BolillaGanadora: tbolilla): boolean;


//Procedimientos sobre ficticios

Procedure Insertar_JugadoresFicticios(var Me: ME_JUGADORES.ME_JUGADOR; cant:tCantidad);

procedure Activar_JugadoresFicticios(var Me: ME_JUGADORES.ME_JUGADOR;
cantFict_Activar:tCantidad);

procedure Desactivar_JugadoresFicticios(RaizJugadores: Lib_Auxiliar.tPos);

procedure Cant_FicticiosActivos(RaizJugadores: Lib_Auxiliar.tPos; var cant:tCantidad);

procedure Apuestas_JugadoresFicticios(RaizJugadores: Lib_Auxiliar.tPos; var
cantFict_Activos:tCantidad);




//Generales

procedure Ini_Sesion (var Me: ME_JUGADORES.ME_JUGADOR; var regJugador:
Type_JUGADOR.tRegDatos);

procedure Fin_Sesion (var Me: ME_JUGADORES.ME_JUGADOR; var regJugador:
Type_JUGADOR.tRegDatos);

procedure repartirPremio_xJugada (RaizJugadores: Lib_Auxiliar.tPos; nroJugada:
Type_JUEGO.tClave);

Function ExistenApuestas_enEsteJuego(Num_Jugada: Type_JUEGO.tClave):boolean;

Procedure CantGanadores_EnUnaJugada(RaizGanadores: Lib_Auxiliar.tPos; jugada:
Type_JUEGO.TClave; var cantGanadores: tCantidad);

Procedure CantGanadores_xNomenclador(RaizGanadores: Lib_Auxiliar.tPos; nomenclador:
tNomenclador; var cantGanadores: tCantidad);

Procedure MontoEnApuestaXPartida (NroJugada: Type_JUEGO.TClave; var montoTotalApuestas:
tImporte);

Procedure MontoPremiosXPartida(RaizGanadores: Lib_Auxiliar.tPos; NroJugada:
Type_JUEGO.TClave; var montoTotalPremios: tImporte);

Procedure Cant_RealesYFicticios_xPartida(RaizJugadores: Lib_Auxiliar.tPos; NroJugada:
Type_JUEGO.TClave; var cantReales: tCantidad; var cantFicticios: tCantidad);

procedure ReordenarMasPremiados (var grid: TStringGrid; num:integer; cantFilas:
integer);

Procedure MovimientoHaberApostadores();

procedure recuperarRegPartida (Nro_Jugada: Type_JUEGO.tClave; var regBuscado:
Type_JUEGO.TipoRegDatos);
```

```
Procedure InsertoPremioUnGanador(regGanador: Type_GANADOR.TipoRegDatos; var regApuesta:
Type_APUESTA.TipoRegDatos; tipoGanador: tNomenclador; Importe: tImporte);

Procedure MovimientoApuestaUnJugador(nick: tClave; regApuesta:
Type_APUESTA.TipoRegDatos; Importe: tImporte);

procedure ReordenarGridPorFecha (var grid: TStringGrid; cantFilas: integer);

procedure Cant_JugadoresEnLinea(RaizJugadores: Lib_Auxiliar.tPos; var cant:tCantidad);

Procedure CantidadJugadoresNuncaGanaron(RaizJugadores: Lib_Auxiliar.tPos; var cant:
tCantidad);

Procedure Premios_AcumuladosXJugador(RaizGanadores: Lib_Auxiliar.tPos;
Nick:Type_JUGADOR.tClave; var cantPremios: tCantidad; var Monto: tImporte);

Procedure Premios_AcumuladosXJugadorUnaJugada(RaizGanadores: Lib_Auxiliar.tPos;
Nick:Type_JUGADOR.tClave; jugada: Type_JUEGO.TClave; var Monto: tImporte);

Function JugadorTieneApuestasEnPartida (NroJugada: Type_JUEGO.TClave; nick:
Type_JUGADOR.tClave): boolean;

Procedure JugadorEsUnGanandor(pos:tPos; Nick: tClave; var esGanador: boolean);

Procedure EliminarApuestasUnJugador(Nick: tClave);

Procedure Cant_JugadoresDesbloqueados(RaizJugadores: Lib_Auxiliar.tPos; var
cantDesbloq: Lib_Auxiliar.tCantidad);

Procedure Cant_JugadoresActivos(RaizJugadores: Lib_Auxiliar.tPos; var cantActivos:
Lib_Auxiliar.tCantidad);


//Listados

Procedure ListadoGeneral_JugadoresEnGrid(RaizJugadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);

Procedure Listado_JugadoresActivos(RaizJugadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);

Procedure ListadoArbolGanadores(RaizGanadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);

Procedure ListadoArbolJugadores(RaizJugadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);

procedure ListadoPartidas (var grid: TStringGrid);

Procedure ListadoJugadoresNuncaGanaron(RaizJugadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);

procedure ListadoCtaCte_deUnJugador(posListaParcial: tPos; Nick: tclave; var grid:
TStringGrid);

Procedure ListadoPremiados_Nomenclador(RaizGanadores: Lib_Auxiliar.tPos; nomenclador:
tNomenclador; var grid: TStringGrid);

Procedure ListadoMasPremiados(RaizGanadores: Lib_Auxiliar.tPos; var importe:tImporte;
var grid: TStringGrid; var cantFilas: integer);

Procedure Listado_GeneralApuestas(var grid: TStringGrid; Nro_Jugada:
Type_JUEGO.tClave);
```

```
Procedure Listado_ApuestasUnJugador(var grid: TStringGrid; Nro_Jugada:
Type_JUEGO.tClave; nick: Type_JUGADOR.TClave);

Procedure Listado_Premios(RaizGanadores: Lib_Auxiliar.tPos; jugada: Type_JUEGO.TClave;
var grid: TStringGrid);


//grid

function Mensaje_Confirmacion(Msje: String): boolean;

procedure limpiarGrid(grid: TStringGrid);

Procedure EncabezadoDetalladoJugadores (grid: TStringGrid);

Procedure EncabezadoArbolJugadores(grid: TStringGrid);

Procedure EncabezadoListadoPartidas(grid: TStringGrid);

Procedure EncabezadoJugadoresMasGanaron(grid: TStringGrid);

Procedure EncabezadoGanadoresXPremio(grid: TStringGrid);

Procedure EncabezadoFiltroCtaCte(grid: TStringGrid);

Procedure EncabezadosComunApuestas(grid: TStringGrid);

Procedure EncabezadosComunJugadores(grid: TStringGrid);

Procedure EncabezadoGridPremios(grid: TStringGrid);

Procedure EncabezadosComunFicticios(grid: TStringGrid);


IMPLEMENTATION


uses
  Form_Login;


{-------------------------------------------------------------------------------}
//Inserto cantidad parametrizada de jugadores ficticios
Procedure Insertar_JugadoresFicticios(var Me: ME_JUGADORES.ME_JUGADOR; cant:tCantidad);
var
 numFicticio: Lib_Auxiliar.tCantidad;
 posJugador: tpos;
 regJugador: type_JUGADOR.tRegDatos;
 RegCtaCte: Type_ALMACEN.tipoRegAlmacen;
 i: Integer;
 largo:integer;
 cadena:string;
 pad:char;
```

```
begin
pad:='0';
  for I := 1 to cant do
  begin
   //Cuando el croupier genera un nuevo jugador ficticio, se les asigna el nick
   //que surja de la combinación "X_" + str (<ficticio>) completado con ceros la parte
no significativa.
   //Por ejemplo: X_0001, X_0002, etc.
    numFicticio:= UltFicticio(Me) + 1;
    regJugador.Nick:= Lib_Auxiliar.Fict_IniNick + intTostr(numFicticio).PadLeft(4,pad);
    regJugador.Nombre:='N_Ficticio';
    regJugador.Apellido:='A_Ficticio';
    regJugador.Contrasenia:='';
    regJugador.Alta:= Now();
    regJugador.UltimaConexion:= Now();
    regJugador.TipoJugador:= false;
    regJugador.Estado:= false;
    regJugador.Bloqueado:= false;
    BuscarInfoME_Jugadores(Me,posJugador,regJugador.Nick);
    InsertarInfoME_Jugadores(Me,regJugador,posJugador);

    //Cargo datos de ctacte a ficticios
    RegCtaCte.Nick:= regJugador.Nick;
    RegCtaCte.FechaHora:= Now();
    RegCtaCte.Concepto:= Type_ALMACEN.tipoConceptos.Creacion_Cuenta;
    RegCtaCte.Debe:= ME_JUEGO.ApuestaMinima(Form_Login.ME_JUEGORULETA)*10;
    RegCtaCte.haber:= 0;
    RegCtaCte.saldo:= ME_JUEGO.ApuestaMinima(Form_Login.ME_JUEGORULETA)*10;
    //inserto cuenta corriente de ficticios
    ME_CTACTE.InsertarAlmacen(Form_Login.ME_CUENTACORRIENTE,RegCtaCte);
  end;
end;
{-------------------------------------------------------------------------------}
```

```
procedure Activar_JugadoresFicticios(var Me: ME_JUGADORES.ME_JUGADOR;
cantFict_Activar:tCantidad);

var

  RD:Type_JUGADOR.tRegDatos;

  CantFictio, esteFicticio:tCantidad;

  pad:char;

  estado:string;

begin

  Randomize;

  if cantFict_Activar>0 then

  begin

    //Nick random ficticio con el campo guardo en RC de ultimoFicticio

    pad:='0';

    CantFictio:= ME_JUGADORES.UltFicticio(ME_JUGADOR);

    esteFicticio:= random(CantFictio) + 1;

    RD.Nick:= Lib_Auxiliar.Fict_IniNick + intTostr(esteFicticio).PadLeft(4,pad);


    ME_JUGADORES.BuscarInfoME_Jugadores(Form_Login.ME_JUGADOR,posJugador,RD.Nick);

    ME_JUGADORES.CapturarInfoME_Jugadores(Form_Login.ME_JUGADOR,RD,posJugador);


    if (not RD.Estado)then

    begin

      RD.Nick:= RD.Nick;

      RD.Contrasenia:= RD.Contrasenia;

      RD.Nombre:= RD.Nombre;

      RD.Apellido:= RD.Apellido;

      RD.Alta:= RD.Alta;

      RD.UltimaConexion:= Now(); // Última conexión actual

      RD.Bloqueado:= RD.Bloqueado;

      RD.Estado:= true; // activo jugador ficticio


      ME_JUGADORES.ModificarInfoME_Jugadores(Form_Login.ME_JUGADOR, RD, posJugador);


      Activar_JugadoresFicticios(Form_Login.ME_JUGADOR, cantFict_Activar - 1);

    end
```

```
    else
      Activar_JugadoresFicticios(Form_Login.ME_JUGADOR, cantFict_Activar);
  end;
end;

{-------------------------------------------------------------------------------}

procedure Desactivar_JugadoresFicticios(RaizJugadores: Lib_Auxiliar.tPos);
var
  RD:Type_JUGADOR.tRegDatos;
begin
 if (RaizJugadores<>Lib_Auxiliar.PosNula)then
 begin
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);
    if ((not RD.TipoJugador) and (RD.Estado)) then //ficticio y activo
    begin
      RD.Nick:= RD.Nick;
      RD.Contrasenia:= RD.Contrasenia;
      RD.Nombre:= RD.Nombre;
      RD.Apellido:= RD.Apellido;
      RD.Alta:= RD.Alta;
      RD.UltimaConexion:= Now(); // Última conexión actual
      RD.Bloqueado:= RD.Bloqueado;
      RD.Estado:= false; // desactivo jugador ficticio

      ME_JUGADORES.ModificarInfoME_Jugadores(Form_Login.ME_JUGADOR, RD, RaizJugadores);
    end;

  Desactivar_JugadoresFicticios(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores));
  Desactivar_JugadoresFicticios(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores));
 end;
end;

{-------------------------------------------------------------------------------}
```

```
procedure Cant_FicticiosActivos(RaizJugadores: Lib_Auxiliar.tPos; var cant:tCantidad);
var
  RD:Type_JUGADOR.tRegDatos;
begin
 if (RaizJugadores<>Lib_Auxiliar.PosNula)then
 begin
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);
    if ((not RD.TipoJugador) and (RD.Estado)) then //ficticio y activo
      cant:= cant + 1;

  Cant_FicticiosActivos(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores), cant);
  Cant_FicticiosActivos(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores), cant);
 end;
end;

{-------------------------------------------------------------------------------}

procedure Cant_JugadoresEnLinea(RaizJugadores: Lib_Auxiliar.tPos; var cant:tCantidad);
var
  RD:Type_JUGADOR.tRegDatos;
begin
 if (RaizJugadores<>Lib_Auxiliar.PosNula)then
 begin
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);

    if ((RD.TipoJugador) and (RD.Estado) and (RD.Nick<>'ADMINISTRADOR')) then //real,
activo. Administrador no suma
      cant:= cant + 1;

  Cant_JugadoresEnLinea(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores), cant);
  Cant_JugadoresEnLinea(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores), cant);
 end;
end;

{-------------------------------------------------------------------------------}
```

```
procedure Apuestas_JugadoresFicticios(RaizJugadores: Lib_Auxiliar.tPos; var
cantFict_Activos:tCantidad);
var
  regJF, RD:Type_JUGADOR.tRegDatos; //registro Jugador Ficticio
  regAF: Type_APUESTA.TipoRegDatos; //registro Apuesta Ficticio
  regJuego: Type_JUEGO.TipoRegDatos;
  rNomenclador, rValor: integer; //nomenclador random, valor random
  posAlmacenFict: tPos;
  regAlmacenFict: Type_ALMACEN.tipoRegAlmacen;
  saldoFict, importeFict: tImporte;
  importeValido: boolean;
begin
 if (RaizJugadores<>Lib_Auxiliar.PosNula) then
 begin
  if (cantFict_Activos > 0) then
  begin
    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);

    if ((not RD.TipoJugador) and (RD.Estado) and (not RD.Bloqueado)) then //ficticio,
activo, no bloqueado
    begin
      Randomize;
      ME_JUEGO.UltimaJugada(ME_JUEGORULETA, regJuego);
      regAF.NroJugada:= regJuego.NroJugada;
      regAF.Nick:= RD.Nick;

      rNomenclador:= random(6) + 1;

      CASE (rNomenclador) of
      1:begin // Pleno
        Randomize;
          regAF.Nomenclador:= type_APUESTA.Nom_Pleno;
          regAF.Valor:= inttostr(random(36));
        end;
      2:begin // Columna
```

```
    Randomize;
      regAF.Nomenclador:= type_APUESTA.Nom_Columna;
      rValor:= random(3) + 1;
      case (rValor) of
      1:regAF.Valor:= '1';
      2:regAF.Valor:= '2';
      3:regAF.Valor:= '3';
      end; //case rValor
    end;
  3:begin // Docena
    Randomize;
      regAF.Nomenclador:= type_APUESTA.Nom_Docena;
      rValor:= random(3) + 1;
      case (rValor) of
      1:regAF.Valor:= '1';
      2:regAF.Valor:= '2';
      3:regAF.Valor:= '3';
      end; //case rValor
    end;
  4:begin // Pasa o Falta
    Randomize;
      regAF.Nomenclador:= type_APUESTA.Nom_PasaFalta;
      rValor:= random(2) + 1;
      if (rValor = 1) then
        regAF.Valor:= 'P'
      else
        regAF.Valor:= 'F';
    end;
  5:begin // Par o Impar
    Randomize;
      regAF.Nomenclador:= type_APUESTA.Nom_ParImpar;
      rValor:= random(2) + 1;
      if (rValor = 1) then
        regAF.Valor:= 'P'
      else
```

```
            regAF.Valor:= 'I';
        end;
      6:begin //Rojo o Negro
        Randomize;
          regAF.Nomenclador:= type_APUESTA.Nom_Rojonegro;
          rValor:= random(2) + 1;
          if (rValor = 1) then
            regAF.Valor:= 'R'
          else
            regAF.Valor:= 'N';
        end;
      END;//case

      saldoFict:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE, RD.Nick);

      //Si saldo de ficticio es mayor que la apuesta minina
      if (saldoFict >= ME_JUEGO.ApuestaMinima(ME_JUEGORULETA)) then
      begin
        Randomize;
      //busco un importe que sea un rango aleatorio que este entre la apuesta minima y
la maxima permitida
        importeFict :=
RANDOMRANGE(ME_JUEGO.ApuestaMinima(ME_JUEGORULETA),ME_JUEGO.ApuestaMaxima(ME_JUEGORULET
A));

        importeValido:= false;
        while (not importeValido) do
        begin
          //El IMPORTE A APOSTAR del ficticio es valida si es menor o igual a su saldo,
mayor o igual a la apuesta minima, menor o igual a la apueta maxima.
          if ((importeFict <= saldoFict) and (importeFict >=
ME_JUEGO.ApuestaMinima(ME_JUEGORULETA)) and (importeFict <=
ME_JUEGO.ApuestaMaxima(ME_JUEGORULETA))) then
            importeValido:= true
          else
            //si el importe a apostar no es valido, busco de nuevo un rango que este
entre la apuesta minima y la maxima permitida
```

```
            importeFict :=
RANDOMRANGE(ME_JUEGO.ApuestaMinima(ME_JUEGORULETA),ME_JUEGO.ApuestaMaxima(ME_JUEGORULET
A));

        end;//while


        regAF.Importe:= importeFict;

        ME_APUESTAS.InsertarInfoME_Apuestas(form_login.ME_APUESTA, regAF,
ME_APUESTAS.PosNula_Apuestas(ME_APUESTA));


        cantFict_Activos:= cantFict_Activos-1;

      end;//If

    end;

  end;


  Apuestas_JugadoresFicticios(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores),
cantFict_Activos);

  Apuestas_JugadoresFicticios(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores),
cantFict_Activos);

 end;

end;


{-------------------------------------------------------------------------------}


procedure repartirPremio_xJugada (RaizJugadores: Lib_Auxiliar.tPos; nroJugada:
Type_JUEGO.tClave);

var

  i:tPos;

  regJugador, RD:Type_JUGADOR.tRegDatos;

  regJuego: Type_JUEGO.TipoRegDatos;

  regApuesta: Type_APUESTA.TipoRegDatos;

  regGanador: Type_GANADOR.TipoRegDatos;

  regCtaCte: Type_ALMACEN.tipoRegAlmacen;

  importe: tImporte;


begin

  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, regJuego);
```

```
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin
    //capturo jugador para verificar si tiene alguna apuesta
    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, RaizJugadores);

    i:= ME_APUESTAS.Primero(ME_APUESTA);

    while (i<>ME_APUESTAS.PosNula_Apuestas(ME_APUESTA)) do
    begin
      //capturo apuesta del jugador
      ME_APUESTAS.CapturarInfoME_Apuestas(ME_APUESTA,i, regApuesta);

       //si la apuesta apuesta es de la misma jugada y mismo jugador
      if ((regApuesta.NroJugada = regJuego.NroJugada) and (regApuesta.Nick =
regJugador.Nick)) then
      begin //busco si coinciden sus apuestas con el resultado de la jugada
          if (regApuesta.Nomenclador = Type_APUESTA.Nom_Pleno) then //PLENO
          begin
            if (regApuesta.Valor = intTostr(regJuego.Bolilla)) then
            begin
              //Calculo el importe que gano el apostador en dicha jugada
              Importe:= regApuesta.Importe * type_GANADOR.Prem_Pleno;
              //Invoco a PremioAunGanador que completa el registro de ganador y lo
inserta
              InsertoPremioUnGanador(regGanador, regApuesta, Type_APUESTA.Nom_Pleno,
Importe);

              //Inserto movimiento de premio de un jugador en CtaCte
              MovimientoApuestaUnJugador(regJugador.Nick, regApuesta, Importe);
            end;
          end;

          if (regApuesta.Nomenclador = Type_APUESTA.Nom_Columna) then // COLUMNA
          begin
            if (((regApuesta.Valor = '1') and (IsIn_PrimeraColumna(regJuego.Bolilla)))
or ((regApuesta.Valor = '2') and (IsIn_SegundaColumna(regJuego.Bolilla))) or
((regApuesta.Valor = '3') and (IsIn_TerceraColumna(regJuego.Bolilla)))) then
```

```
            begin
              //Calculo el importe que gano el apostador en dicha jugada
              Importe:= regApuesta.Importe * type_GANADOR.Prem_Pleno;
              //Invoco a PremioAunGanador que completa el registro de ganador y lo
inserta
              InsertoPremioUnGanador(regGanador, regApuesta, Type_APUESTA.Nom_Columna,
Importe);

                //Inserto movimiento de premio de un jugador en CtaCte
               MovimientoApuestaUnJugador(regJugador.Nick, regApuesta, Importe);
            end;
          end;

          if (regApuesta.Nomenclador = Type_APUESTA.Nom_Docena) then // DOCENA
          begin
            if (((regApuesta.Valor = '1') and (IsIn_PrimerDocena(regJuego.Bolilla))) or
((regApuesta.Valor = '2') and (IsIn_SegundaDocena(regJuego.Bolilla))) or
((regApuesta.Valor = '3') and (IsIn_TercerDocena(regJuego.Bolilla)))) then
            begin
              //Calculo el importe que gano el apostador en dicha jugada
              Importe:= regApuesta.Importe * type_GANADOR.Prem_Pleno;
              //Invoco a PremioAunGanador que completa el registro de ganador y lo
inserta
              InsertoPremioUnGanador(regGanador, regApuesta, Type_APUESTA.Nom_Docena,
Importe);

                 //Inserto movimiento de premio de un jugador en CtaCte
               MovimientoApuestaUnJugador(regJugador.Nick, regApuesta, Importe);
            end;
          end;

          if (regApuesta.Nomenclador = Type_APUESTA.Nom_PasaFalta) then // PASA-FALTA
          begin
            if (((regApuesta.Valor = 'P') and (IsIn_PASA(regJuego.Bolilla))) or
((regApuesta.Valor = 'F') and (IsIn_FALTA(regJuego.Bolilla)))) then
            begin
              //Calculo el importe que gano el apostador en dicha jugada
              Importe:= regApuesta.Importe * type_GANADOR.Prem_Pleno;
```

```
              //Invoco a PremioAunGanador que completa el registro de ganador y lo
inserta
              InsertoPremioUnGanador(regGanador, regApuesta,
Type_APUESTA.Nom_PasaFalta, Importe);


                //Inserto movimiento de premio de un jugador en CtaCte
              MovimientoApuestaUnJugador(regJugador.Nick, regApuesta, Importe);
            end;
          end;


          if (regApuesta.Nomenclador = Type_APUESTA.Nom_ParImpar) then // PAR-IMPAR
          begin
            if (((regApuesta.Valor = 'P') and (IsIn_PAR(regJuego.Bolilla))) or
((regApuesta.Valor = 'I') and (IsIn_IMPAR(regJuego.Bolilla)))) then
            begin
               //Calculo el importe que gano el apostador en dicha jugada
              Importe:= regApuesta.Importe * type_GANADOR.Prem_Pleno;
              //Invoco a PremioAunGanador que completa el registro de ganador y lo
inserta
              InsertoPremioUnGanador(regGanador, regApuesta, Type_APUESTA.Nom_ParImpar,
Importe);


              //Inserto movimiento de premio de un jugador en CtaCte
              MovimientoApuestaUnJugador(regJugador.Nick, regApuesta, Importe);
            end;
          end;


          if (regApuesta.Nomenclador = Type_APUESTA.Nom_Rojonegro) then // ROJO-NEGRO
          begin
            if (((regApuesta.Valor = 'R') and (IsIn_ROJOS(regJuego.Bolilla))) or
((regApuesta.Valor = 'N') and (IsIn_NEGROS(regJuego.Bolilla)))) then
            begin
               //Calculo el importe que gano el apostador en dicha jugada
              Importe:= regApuesta.Importe * type_GANADOR.Prem_Pleno;
              //Invoco a PremioAunGanador que completa el registro de ganador y lo
inserta
              InsertoPremioUnGanador(regGanador, regApuesta,
Type_APUESTA.Nom_Rojonegro, Importe);
```

```
            //Inserto movimiento de premio de un jugador en CtaCte
            MovimientoApuestaUnJugador(regJugador.Nick, regApuesta, Importe);
          end;
        end;

        //Si la bola cae sobre el cero los jugadores que hicieron apuestas sencillas
(rojo/negro, par/impar, pasa/falta)
        //recuperan la mitad de su apuesta y pierden la otra mitad.
        if (regJuego.Bolilla = 0) and ((regApuesta.Nomenclador =
Type_APUESTA.Nom_Columna) or
        (regApuesta.Nomenclador = Type_APUESTA.Nom_Docena) or
        (regApuesta.Nomenclador = Type_APUESTA.Nom_PasaFalta) or
        (regApuesta.Nomenclador = Type_APUESTA.Nom_ParImpar) or
        (regApuesta.Nomenclador = Type_APUESTA.Nom_Rojonegro)) then
        begin
          //completo registro de ganador
          regGanador.Nomenclador:= regApuesta.Nomenclador;
          regGanador.Numero:= strToint('0');
          regGanador.NroJugada:= strToint(regApuesta.NroJugada);
          regGanador.Importe:= round(regApuesta.Importe*type_GANADOR.Prem_Cero);
          //Inserto ganador
          ME_GANADORES.InsertarInfoME_Ganadores(ME_GANADOR,regGanador,
regApuesta.Nick);

          //Completo registro movimiento en CtaCte
          regCtaCte.Nick:= regJugador.Nick;
          regCtaCte.FechaHora:= Now();
          regCtaCte.Concepto:= Type_ALMACEN._PremioJuego + regApuesta.NroJugada;
          regCtaCte.Debe:= regGanador.Importe;
          regCtaCte.haber:= 0;
          regCtaCte.saldo:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,
regApuesta.Nick) + regGanador.Importe;
          //Inserto movimiento en CtaCte
          ME_CTACTE.InsertarAlmacen(ME_CUENTACORRIENTE, regCtaCte);
        end;
```

```
      end;


      i:= ME_APUESTAS.Proximo(ME_APUESTA, i)

    end;//while


    repartirPremio_xJugada(ME_JUGADORES.ProximoIzq(ME_JUGADOR, RaizJugadores),
nroJugada);

    repartirPremio_xJugada(ME_JUGADORES.ProximoDer(ME_JUGADOR, RaizJugadores),
nroJugada);

  end;

end;


{-------------------------------------------------------------------------------}


function IsIn_PrimeraColumna(BolillaGanadora: tBolilla): boolean;

begin

  IsIn_PrimeraColumna:= (BolillaGanadora = 1) or  (BolillaGanadora = 4) or
(BolillaGanadora = 7)

      or (BolillaGanadora = 10) or (BolillaGanadora = 13) or (BolillaGanadora = 16) or
(BolillaGanadora = 19)

      or (BolillaGanadora = 22)  or (BolillaGanadora = 25) or (BolillaGanadora = 28) or
(BolillaGanadora = 31) or (BolillaGanadora = 34);

end;


{-------------------------------------------------------------------------------}


function IsIn_SegundaColumna(BolillaGanadora: tBolilla): boolean;

begin

  IsIn_SegundaColumna:= (BolillaGanadora = 2) or  (BolillaGanadora = 5) or
(BolillaGanadora = 8)

      or (BolillaGanadora = 11) or (BolillaGanadora = 14) or (BolillaGanadora = 17) or
(BolillaGanadora = 20)

      or (BolillaGanadora = 23)  or (BolillaGanadora = 26) or (BolillaGanadora = 29) or
(BolillaGanadora = 32) or (BolillaGanadora = 35);

end;


{-------------------------------------------------------------------------------}
```

```
function IsIn_TerceraColumna(BolillaGanadora: tBolilla): boolean;

begin

  IsIn_TerceraColumna:= (BolillaGanadora = 3) or  (BolillaGanadora = 6) or
(BolillaGanadora = 9)

      or (BolillaGanadora = 12) or (BolillaGanadora = 15) or (BolillaGanadora = 18) or
(BolillaGanadora = 21)

      or (BolillaGanadora = 24)  or (BolillaGanadora = 27) or (BolillaGanadora = 30) or
(BolillaGanadora = 33) or (BolillaGanadora = 36);

end;


{-------------------------------------------------------------------------------}


function IsIn_PrimerDocena(BolillaGanadora: tbolilla): boolean;

begin

  IsIn_PrimerDocena:= (BolillaGanadora>=1) and (BolillaGanadora<=12);

end;


{-------------------------------------------------------------------------------}


function IsIn_SegundaDocena(BolillaGanadora: tbolilla): boolean;

begin

  IsIn_SegundaDocena:= (BolillaGanadora>=13) and (BolillaGanadora<=24);

end;


{-------------------------------------------------------------------------------}


function IsIn_TercerDocena(BolillaGanadora: tbolilla): boolean;

begin

  IsIn_TercerDocena:= (BolillaGanadora>=25) and (BolillaGanadora<=38);

end;


{-------------------------------------------------------------------------------}



function IsIn_PASA(BolillaGanadora: tbolilla): boolean;

begin
```

```
  IsIn_PASA:= (BolillaGanadora>0) and (BolillaGanadora<=18);
end;

{-------------------------------------------------------------------------------}

function IsIn_FALTA(BolillaGanadora: tbolilla): boolean;
begin
  IsIn_FALTA:= (BolillaGanadora>18) and (BolillaGanadora<=38);
end;

{-------------------------------------------------------------------------------}

function IsIn_PAR(BolillaGanadora: tbolilla): boolean;
begin
  IsIn_PAR:= (BolillaGanadora = 2) or  (BolillaGanadora = 4) or (BolillaGanadora = 6)
      or (BolillaGanadora = 8) or (BolillaGanadora = 10) or (BolillaGanadora = 12) or
(BolillaGanadora = 14)
      or (BolillaGanadora = 16)  or (BolillaGanadora = 18) or (BolillaGanadora = 20) or
(BolillaGanadora = 22) or (BolillaGanadora = 24)
      or (BolillaGanadora = 26)  or (BolillaGanadora = 28) or (BolillaGanadora = 30) or
(BolillaGanadora = 32) or (BolillaGanadora = 34)
      or (BolillaGanadora = 36);
end;

{-------------------------------------------------------------------------------}

function IsIn_IMPAR(BolillaGanadora: tbolilla): boolean;
begin
  IsIn_IMPAR:= (BolillaGanadora = 1) or  (BolillaGanadora = 3) or (BolillaGanadora = 5)
      or (BolillaGanadora = 7) or (BolillaGanadora = 9) or (BolillaGanadora = 11) or
(BolillaGanadora = 13)
      or (BolillaGanadora = 15)  or (BolillaGanadora = 17) or (BolillaGanadora = 19) or
(BolillaGanadora = 21) or (BolillaGanadora = 23)
      or (BolillaGanadora = 25)  or (BolillaGanadora = 27) or (BolillaGanadora = 29) or
(BolillaGanadora = 31) or (BolillaGanadora = 33)
      or (BolillaGanadora = 35);
end;
```

```
{-------------------------------------------------------------------------------}


//1,3,5,7,9,12,14,16,18,19,21,23,25,27,30,32,34,36 (rojos)


function IsIn_ROJOS(BolillaGanadora: tbolilla): boolean;

begin

  IsIn_ROJOS:= (BolillaGanadora = 1) or  (BolillaGanadora = 3) or (BolillaGanadora = 5)

      or (BolillaGanadora = 7) or (BolillaGanadora = 9) or (BolillaGanadora = 12) or
(BolillaGanadora = 14)

      or (BolillaGanadora = 16)  or (BolillaGanadora = 18) or (BolillaGanadora = 19) or
(BolillaGanadora = 21) or (BolillaGanadora = 23)

      or (BolillaGanadora = 25)  or (BolillaGanadora = 27) or (BolillaGanadora = 30) or
(BolillaGanadora = 32) or (BolillaGanadora = 34)

      or (BolillaGanadora = 36);

end;


{-------------------------------------------------------------------------------}


//2,4,6,8,10,11,13,15,17,20,22,24,26,28,29,31,33,35 (negros)


function IsIn_NEGROS(BolillaGanadora: tbolilla): boolean;

begin

  IsIn_NEGROS:= (BolillaGanadora = 2) or  (BolillaGanadora = 4) or (BolillaGanadora =
6)

      or (BolillaGanadora = 8) or (BolillaGanadora = 10) or (BolillaGanadora = 11) or
(BolillaGanadora = 13)

      or (BolillaGanadora = 15)  or (BolillaGanadora = 17) or (BolillaGanadora = 20) or
(BolillaGanadora = 22) or (BolillaGanadora = 24)

      or (BolillaGanadora = 26)  or (BolillaGanadora = 28) or (BolillaGanadora = 29) or
(BolillaGanadora = 31) or (BolillaGanadora = 33)

      or (BolillaGanadora = 35);

end;


{-------------------------------------------------------------------------------}


Function ExistenApuestas_enEsteJuego(Num_Jugada: Type_JUEGO.tClave):boolean;

var
```

```
  i:tPos;
  encontre:boolean;
begin
  encontre:= false;
  i:= ME_APUESTAS.Primero(ME_APUESTA);
        while ((i<>ME_APUESTAS.PosNula_Apuestas(ME_APUESTA)) and (not encontre)) do
        begin
          ME_APUESTAS.CapturarInfoME_Apuestas(ME_APUESTA,i,regApuesta);

          if (regApuesta.NroJugada = Num_Jugada) then
          begin
            encontre:= true;
            ExistenApuestas_enEsteJuego:= true;
          end
          else
            i:= ME_APUESTAS.Proximo(ME_APUESTA,i);
        END; //while
end;

{-------------------------------------------------------------------------------}

Procedure CantGanadores_EnUnaJugada(RaizGanadores: Lib_Auxiliar.tPos; jugada:
Type_JUEGO.TClave; var cantGanadores: tCantidad);
var
RegNodoGanador: regNodo;
RegAlmacen: type_GANADOR.TipoRegDatos;
i: tPos;
begin
  if (RaizGanadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, RaizGanadores);

    i:= ME_GANADORES.PrimerJugada_Ganador(ME_GANADOR, RegNodoGanador);

    while i<> ME_GANADORES.PosNula_Ganadores(ME_GANADOR) do
```

```
    begin
      ME_GANADORES.CapturarJugada_Ganador(ME_GANADOR, i, RegAlmacen);

      if (intTostr(RegAlmacen.NroJugada) = jugada)  then
        cantGanadores:= cantGanadores + 1;

      i:= ME_GANADORES.SiguienteJugada_Ganador(ME_GANADOR, i);
    end; //while

    CantGanadores_EnUnaJugada(ME_GANADORES.ProximoIzq(ME_GANADOR, RaizGanadores),
jugada, cantGanadores);
    CantGanadores_EnUnaJugada(ME_GANADORES.ProximoDer(ME_GANADOR, RaizGanadores),
jugada, cantGanadores);
  end;
end;

{-------------------------------------------------------------------------------}

Procedure CantGanadores_xNomenclador(RaizGanadores: Lib_Auxiliar.tPos; nomenclador:
tNomenclador; var cantGanadores: tCantidad);
var
RegNodoGanador: regNodo;
RegAlmacen: type_GANADOR.TipoRegDatos;
i: tPos;
begin
  if (RaizGanadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, RaizGanadores);

    i:= ME_GANADORES.PrimerJugada_Ganador(ME_GANADOR, RegNodoGanador);

    while i<> ME_GANADORES.PosNula_Ganadores(ME_GANADOR) do
    begin
      ME_GANADORES.CapturarJugada_Ganador(ME_GANADOR, i, RegAlmacen);

      if (RegAlmacen.Nomenclador = nomenclador)  then
```

```
        cantGanadores:= cantGanadores + 1;


      i:= ME_GANADORES.SiguienteJugada_Ganador(ME_GANADOR, i);
    end; //while


    CantGanadores_xNomenclador(ME_GANADORES.ProximoIzq(ME_GANADOR, RaizGanadores),
nomenclador, cantGanadores);
    CantGanadores_xNomenclador(ME_GANADORES.ProximoDer(ME_GANADOR, RaizGanadores),
nomenclador, cantGanadores);
  end;
end;


{-------------------------------------------------------------------------------}


 procedure MontoEnApuestaXPartida (NroJugada: Type_JUEGO.TClave; var
montoTotalApuestas: tImporte);
 var
 i: tpos;
 regApuesta: Type_APUESTA.TipoRegDatos;
 begin
  montoTotalApuestas:= 0;


    i:= ME_APUESTAS.Primero(ME_APUESTA);


     while i<>ME_APUESTAS.PosNula_Apuestas(ME_APUESTA) do
     begin
       ME_APUESTAS.CapturarInfoME_Apuestas(ME_APUESTA,i,regApuesta);


       if (regapuesta.NroJugada= nrojugada) then //pregunto si es la misma jugada que
le paso
          montoTotalApuestas:= montoTotalApuestas + regapuesta.Importe;  //voy juntando
el dinero de la partida


      i:= ME_APUESTAS.Proximo(ME_APUESTA,i);
     end;
 end;
```

```
{-------------------------------------------------------------------------------}


procedure MontoPremiosXPartida(RaizGanadores: Lib_Auxiliar.tPos; NroJugada:
Type_JUEGO.TClave; var montoTotalPremios: tImporte);

var

RegNodoGanador: regNodo;

RegAlmacen: type_GANADOR.TipoRegDatos;

i: tPos;

begin

  if (RaizGanadores<>Lib_Auxiliar.PosNula) then

  begin

    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, RaizGanadores);


    i:= ME_GANADORES.PrimerJugada_Ganador(ME_GANADOR, RegNodoGanador);


    while i<> ME_GANADORES.PosNula_Ganadores(ME_GANADOR) do

    begin

      ME_GANADORES.CapturarJugada_Ganador(ME_GANADOR, i, RegAlmacen);


      if (intTostr(RegAlmacen.NroJugada) = NroJugada)  then

        montoTotalPremios:= montoTotalPremios + RegAlmacen.Importe;


      i:= ME_GANADORES.SiguienteJugada_Ganador(ME_GANADOR, i);

    end; //while


    MontoPremiosXPartida(ME_GANADORES.ProximoIzq(ME_GANADOR, RaizGanadores), NroJugada,
montoTotalPremios);

    MontoPremiosXPartida(ME_GANADORES.ProximoDer(ME_GANADOR, RaizGanadores), NroJugada,
montoTotalPremios);

  end;

end;


{-------------------------------------------------------------------------------}


Procedure Cant_RealesYFicticios_xPartida(RaizJugadores: Lib_Auxiliar.tPos; NroJugada:
Type_JUEGO.TClave; var cantReales: tCantidad; var cantFicticios: tCantidad);
```

```
var
RD:Type_JUGADOR.tRegDatos;
i: tPos;
enc: boolean;
begin
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);

    enc:= false;
    i:= ME_APUESTAS.Primero(ME_APUESTA);
    while i<>ME_APUESTAS.PosNula_Apuestas(ME_aPUESTA) do
    begin
       ME_APUESTAS.CapturarInfoME_Apuestas(ME_aPUESTA,i,regApuesta);

       if (regApuesta.NroJugada = NroJugada) and (regApuesta.Nick = RD.Nick) and
not(enc) then
       begin
        if RD.TipoJugador then
          cantReales:= cantReales + 1
        else
          cantFicticios:= cantFicticios + 1;

       enc:= true;
       end;
      i:= ME_APUESTAS.Proximo(ME_aPUESTA,i);
     end; //while

    Cant_RealesYFicticios_xPartida(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores),
NroJugada, cantReales, cantFicticios);
    Cant_RealesYFicticios_xPartida(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores),
NroJugada, cantReales, cantFicticios);
  end;
end;
```

```
{-------------------------------------------------------------------------------}

Procedure JugadorEsUnGanandor(pos:tPos; Nick: tClave; var esGanador: boolean);
var
RegNodoGanador: regNodo;
RegAlmacen: type_GANADOR.TipoRegDatos;
i: tPos;
begin
  if (pos<>Lib_Auxiliar.PosNula) then
  begin
    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, pos);
    if (RegNodoGanador.clave = Nick) then
      esGanador:=true;

    JugadorEsUnGanandor(ME_GANADORES.ProximoIzq(ME_GANADOR, pos), Nick, esGanador);
    JugadorEsUnGanandor(ME_GANADORES.ProximoDer(ME_GANADOR, pos), Nick, esGanador);
  end;
end;

{-------------------------------------------------------------------------------}

procedure ReordenarMasPremiados (var grid: TStringGrid; num:integer; cantFilas:
integer);
var
i, j, saldoA, saldoB: integer;
nickA, nickB: Lib_Auxiliar.tClave;
begin
   for I := 1 to cantFilas-1 do
       for j := 1 to cantFilas-1 do
       begin
         SaldoA:= strtoint(grid.Cells[1,i]);
         SaldoB:= strtoint(grid.Cells[1,j]);
         NickA:=  grid.Cells [0,i];
         NickB:=  grid.Cells [0,j];
           if (saldoA>saldoB) then
```

```
            begin
               grid.Cells[1,i]:=inttostr(saldoB);
               grid.Cells[0,i]:=nickB;

               grid.Cells[1,j]:=inttostr(saldoA);
               grid.Cells[0,j]:=nickA;
            end;
        end;

    Grid.RowCount:= num + 1;
end;



{-------------------------------------------------------------------------------}

Procedure MovimientoHaberApostadores();
var
RegApuesta: type_APUESTA.TipoRegDatos;
RegCtaCte: Type_ALMACEN.tipoRegAlmacen;
i: tpos;
begin
  //Recorro el ME_APUESTAS capturando las apuestas de la ultima jugada e inserto
movimientos del jugador con el importe apostado en su haber
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, regJuegoRuleta);
  i:= ME_APUESTAS.Primero(ME_APUESTA);

    while i<> ME_APUESTAS.PosNula_Apuestas(ME_APUESTA) do
    begin

      ME_APUESTAS.CapturarInfoME_Apuestas(ME_APUESTA,i, regApuesta);

      if (regApuesta.NroJugada = regJuegoRuleta.NroJugada) then
      begin
        RegCtaCte.Nick:= Uppercase(regApuesta.Nick);
        RegCtaCte.FechaHora:= now();
```

```
        RegCtaCte.Concepto:= Type_ALMACEN.tipoConceptos.Numero_Apuesta +
regJuegoRuleta.NroJugada;

        RegCtaCte.Debe:=  0;

        RegCtaCte.haber:= regApuesta.Importe;

        RegCtaCte.saldo:= (ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,
Uppercase(regApuesta.Nick)) - regApuesta.Importe);

        ME_CTACTE.InsertarAlmacen(ME_CUENTACORRIENTE, RegCtaCte);

      end;


      i:= ME_APUESTAS.Proximo(ME_APUESTA,i);

    end;

end;


{-------------------------------------------------------------------------------}


Procedure EliminarApuestasUnJugador(Nick: tClave);

var

RegApuesta: type_APUESTA.TipoRegDatos;

RegCtaCte: Type_ALMACEN.tipoRegAlmacen;

i: tpos;

begin

  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, regJuegoRuleta);

  i:= ME_APUESTAS.Primero(ME_APUESTA);


    while i<> ME_APUESTAS.PosNula_Apuestas(ME_APUESTA) do

    begin


      ME_APUESTAS.CapturarInfoME_Apuestas(ME_APUESTA,i, regApuesta);


      if (regApuesta.Nick = Nick) then

      begin

        if (regApuesta.NroJugada = regJuegoRuleta.NroJugada) then

          ME_APUESTAS.EliminarInfoME_Apuestas(ME_APUESTA,i);

      end;


      i:= ME_APUESTAS.Proximo(ME_APUESTA,i);
```

```
    end;
end;

{-------------------------------------------------------------------------------}

Procedure Cant_JugadoresDesbloqueados(RaizJugadores: Lib_Auxiliar.tPos; var
cantDesbloq: Lib_Auxiliar.tCantidad);
var
  RD:Type_JUGADOR.tRegDatos;
begin
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);

    if (RD.Nick <> 'ADMINISTRADOR') then
    begin
      if not (RD.Bloqueado) then
        cantDesbloq:= cantDesbloq + 1;
    end;
  Cant_JugadoresDesbloqueados(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores),
cantDesbloq);
  Cant_JugadoresDesbloqueados(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores),
cantDesbloq);
  end;
end;

{-------------------------------------------------------------------------------}

Procedure Cant_JugadoresActivos(RaizJugadores: Lib_Auxiliar.tPos; var cantActivos:
Lib_Auxiliar.tCantidad);
var
RD:Type_JUGADOR.tRegDatos;
begin
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);
```

```
    if (RD.Nick <> 'ADMINISTRADOR') and (RD.Estado) then //si no es el administrador y
esta activo...
      cantActivos:= cantActivos + 1;

  Cant_JugadoresActivos(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores),
cantActivos);
  Cant_JugadoresActivos(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores),
cantActivos);
  end;
end;

{-------------------------------------------------------------------------------}

Procedure ListadoGeneral_JugadoresEnGrid(RaizJugadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);
var
RD:Type_JUGADOR.tRegDatos;
begin
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);
    if (RD.Nick<>'ADMINISTRADOR') then
    begin
      grid.RowCount:= grid.RowCount + 1;

      grid.Cells[0, grid.RowCount-1] := RD.Nombre;

      grid.Cells[1, grid.RowCount-1] := RD.Apellido;

      grid.Cells[2, grid.RowCount-1] := RD.Nick;

      if RD.TipoJugador then
        grid.Cells[3, grid.RowCount-1] :='Real'
      else
          grid.Cells[3, grid.RowCount-1] :='Ficticio';
```

```
       if (RD.Bloqueado) then
        grid.Cells[4, grid.RowCount-1] :='Bloqueado'
      else
        grid.Cells[4, grid.RowCount-1] :='Desbloqueado';

      if (RD.Estado) then
        grid.Cells[5, grid.RowCount-1] :='SI'
      else
        grid.Cells[5, grid.RowCount-1] :='NO';

      grid.Cells[6, grid.RowCount-1] := datetimetostr(RD.Alta);
      grid.Cells[7, grid.RowCount-1] := datetimetostr(RD.UltimaConexion);

      grid.Cells[8, grid.RowCount-1]:=
inttostr(ME_CTACTE.SaldoAcumulado(me_cuentacorriente,RD.Nick));

    end;
    ListadoGeneral_JugadoresEnGrid(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores),
grid);
    ListadoGeneral_JugadoresEnGrid(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores),
grid);
  end;
end;

{-------------------------------------------------------------------------------}

Procedure Listado_JugadoresActivos(RaizJugadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);
var
RD:Type_JUGADOR.tRegDatos;

begin
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);
```

```
    if (RD.Nick <> 'ADMINISTRADOR') then
    begin
        if (RD.Estado) then
        begin
          Grid.RowCount:= Grid.RowCount + 1;

          Grid.Cells[0, Grid.RowCount-1] := RD.Nombre;

          Grid.Cells[1, Grid.RowCount-1] := RD.Apellido;

          Grid.Cells[2, Grid.RowCount-1] := RD.Nick;

          if (RD.TipoJugador) then
            Grid.Cells[3, Grid.RowCount-1] :='Real'
          else
            Grid.Cells[3, Grid.RowCount-1] :='Ficticio';

          if (RD.Bloqueado) then
            Grid.Cells[4, Grid.RowCount-1] :='BLOQUEADO'
          else
            Grid.Cells[4, Grid.RowCount-1] :='DESBLOQUEADO';

          if (RD.Estado) then
            Grid.Cells[5, Grid.RowCount-1] :='SI'
          else
            Grid.Cells[5, Grid.RowCount-1] :='NO';

          Grid.Cells[6, Grid.RowCount-1] := datetimetostr(RD.Alta);

          Grid.Cells[7, Grid.RowCount-1] := datetimetostr(RD.UltimaConexion);

          grid.Cells[8, grid.RowCount-1]:=
inttostr(ME_CTACTE.SaldoAcumulado(me_cuentacorriente,RD.Nick));
        end;
```

```
    end;
    Listado_JugadoresActivos(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores), grid);
    Listado_JugadoresActivos(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores), grid);
  end;
end;


{-------------------------------------------------------------------------------}
Procedure ListadoArbolGanadores(RaizGanadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);
var
RegNodoGanador: regNodo;
RegAlmacen: type_GANADOR.TipoRegDatos;
i: tPos;
begin
  if (RaizGanadores<>Lib_Auxiliar.PosNula) then
  begin
    ListadoArbolGanadores(ME_GANADORES.ProximoIzq(ME_GANADOR,RaizGanadores), grid);
    ListadoArbolGanadores(ME_GANADORES.ProximoDer(ME_GANADOR,RaizGanadores), grid);


    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, RaizGanadores);
    Grid.RowCount:= Grid.RowCount + 1;


    grid.Cells[0, grid.RowCount-1] :=
intTostr(ME_GANADORES.ObtenerNivel_enArbolTri(ME_GANADOR, RegNodoGanador.clave));
    grid.Cells[1, grid.RowCount-1] := RegNodoGanador.clave;


  end;
end;


{-------------------------------------------------------------------------------}


Procedure ListadoArbolJugadores(RaizJugadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);
var
RD:Type_JUGADOR.tRegDatos;
begin
```

```
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin
    ListadoArbolJugadores(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores), grid);
    ListadoArbolJugadores(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores), grid);

    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);
    grid.RowCount:= grid.RowCount + 1;

    grid.Cells[0, grid.RowCount-1] :=
intTostr(ME_JUGADORES.ObtenerNivel_enArbol(ME_JUGADOR, RD.nick));
    grid.Cells[1, grid.RowCount-1] := RD.nick;
  end;
end;

{-------------------------------------------------------------------------------}

procedure ListadoPartidas (var grid: TStringGrid);
var
regPartida: Type_JUEGO.TipoRegDatos;
posPartida: tpos;
i: tpos;
regnulo: Type_JUEGO.TipoRegDatos;

cantReales: tCantidad;
cantFicticios: tCantidad;
montoTotalApuestas: tImporte;
montoTotalPremios: tImporte;
begin
  ME_JUEGO.RegistroNulo(regnulo);
  ME_JUEGO.Insertar(ME_JUEGORULETA, regnulo);
  ME_JUEGO.Frente(ME_JUEGORULETA,regPartida);
  ME_JUEGO.Decolar(ME_JUEGORULETA);

  while regpartida.NroJugada<>ME_JUEGO.ClaveNula(ME_JUEGORULETA) do
  begin
```

```
    montoTotalPremios:= 0;

    cantReales:= 0;

    cantFicticios:= 0;

    grid.RowCount:= grid.RowCount + 1;


    MontoEnApuestaXPartida(regpartida.NroJugada, montoTotalApuestas);

    MontoPremiosXPartida(ME_GANADORES.Raiz(ME_GANADOR), regpartida.NroJugada ,
montoTotalPremios);

    Cant_RealesYFicticios_xPartida(ME_JUGADORES.Raiz(ME_JUGADOR), regpartida.NroJugada,
cantReales, cantFicticios);


    grid.Cells[0, grid.RowCount-1]:= regpartida.nrojugada;

    grid.Cells[1, grid.RowCount-1]:= datetimetostr(regPartida.FechaHora);;

    grid.Cells[2, grid.RowCount-1]:= inttostr(cantReales);

    grid.Cells[3, grid.RowCount-1]:= inttostr(cantFicticios);

    grid.Cells[4, grid.RowCount-1]:= inttostr(montoTotalApuestas);

    grid.Cells[5, grid.RowCount-1]:= inttostr(montoTotalPremios);

    grid.Cells[6, grid.RowCount-1]:= inttostr(regPartida.Bolilla);



    ME_JUEGO.Insertar(ME_JUEGORULETA, regPartida);

    ME_JUEGO.Frente(ME_JUEGORULETA,regPartida);

    ME_JUEGO.Decolar(ME_JUEGORULETA);

  end;

end;


{-------------------------------------------------------------------------------}



Procedure ListadoJugadoresNuncaGanaron(RaizJugadores: Lib_Auxiliar.tPos; var grid:
TStringGrid);

var

RD:Type_JUGADOR.tRegDatos;

esGanador: boolean;

begin

  esGanador:= false;
```

```
if (RaizJugadores<>Lib_Auxiliar.PosNula) then
begin
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);
  if (RD.Nick<>'ADMINISTRADOR') then
  begin
    JugadorEsUnGanandor(ME_GANADORES.Raiz(ME_GANADOR), RD.Nick, esGanador);
    if not(esGanador) then
    begin
      grid.RowCount:= grid.RowCount + 1;

      grid.Cells[0, grid.RowCount-1] := RD.Nombre;

      grid.Cells[1, grid.RowCount-1] := RD.Apellido;

      grid.Cells[2, grid.RowCount-1] := RD.Nick;

      if RD.TipoJugador then
        grid.Cells[3, grid.RowCount-1] :='Real'
      else
          grid.Cells[3, grid.RowCount-1] :='Ficticio';

       if (RD.Bloqueado) then
        grid.Cells[4, grid.RowCount-1] :='Bloqueado'
      else
        grid.Cells[4, grid.RowCount-1] :='Desbloqueado';

      if (RD.Estado) then
        grid.Cells[5, grid.RowCount-1] :='SI'
      else
        grid.Cells[5, grid.RowCount-1] :='NO';

      grid.Cells[6, grid.RowCount-1] := datetimetostr(RD.Alta);
      grid.Cells[7, grid.RowCount-1] := datetimetostr(RD.UltimaConexion);
```

```
        grid.Cells[8, grid.RowCount-1]:=
inttostr(ME_CTACTE.SaldoAcumulado(me_cuentacorriente,RD.Nick));


      end;

    end;

    ListadoJugadoresNuncaGanaron(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores),
grid);

    ListadoJugadoresNuncaGanaron(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores),
grid);

  end;

end;




{-------------------------------------------------------------------------------}

//Procedimiento recursivo donde muestra la cuenta corriente de un jugador de forma
ordenada por fecha y hora

procedure ListadoCtaCte_deUnJugador(posListaParcial: tPos; Nick: tclave; var grid:
TStringGrid);

var

i: tpos;

regAlm: Type_ALMACEN.tipoRegAlmacen;

begin

  i:= posListaParcial;


  If i<>ME_CTACTE.PosNula(ME_CUENTACORRIENTE) then

  begin

    ListadoCtaCte_deUnJugador(ME_CTACTE.ProximoEnLdParcial(ME_CUENTACORRIENTE,i), Nick,
grid);


    ME_CTACTE.CapturarEnAlmacen(ME_CUENTACORRIENTE,i,regAlm);


    if (regAlm.Nick = Nick) then

    begin

      grid.RowCount:= grid.RowCount + 1;


      grid.Cells[0, grid.RowCount-1]:= regAlm.Nick;

      grid.Cells[1, grid.RowCount-1]:= datetimetostr(regAlm.FechaHora);
```

```
      grid.Cells[2, grid.RowCount-1]:= regAlm.Concepto;
      grid.Cells[3, grid.RowCount-1]:= inttostr(regAlm.Debe);
      grid.Cells[4, grid.RowCount-1]:= inttostr(regAlm.haber);
      grid.Cells[5, grid.RowCount-1]:= inttostr(regAlm.saldo);
    end;
  end;
end;

{-------------------------------------------------------------------------------}

procedure ReordenarGridPorFecha (var grid: TStringGrid; cantFilas: integer);
var
i, j, cant: integer;
saldo1, saldo2, debe1, debe2, haber1, haber2: integer;
nick1, nick2: Lib_Auxiliar.tClave;
concepto1, concepto2: string;
FechaHora1, FechaHora2: tDateTime;
begin
  showmessage(intTostr(cantFilas));

   for I := 1 to cantFilas - 1 do
   begin
       for j := 1 to cantFilas - 1 do
       begin
        Saldo1:= strtoint(grid.Cells[5,i]); Saldo2:= strtoint(grid.Cells[5,j]);
        Haber1:= strtoint(grid.Cells[4,i]); Haber2:= strtoint(grid.Cells[4,j]);
        Debe1:= strtoint(grid.Cells[3,i]); Debe2:= strtoint(grid.Cells[3,j]);
        Concepto1:= (grid.Cells[2,i]); Concepto2:=(grid.Cells[2,j]);
        FechaHora1:= strTodatetime(grid.Cells[1,i]); FechaHora2:=
strTodatetime(grid.Cells[1,j]);
        Nick1:= grid.Cells [0,i]; Nick2:= grid.Cells [0,j];

        //Intercambio celdas
        grid.Cells[5,i]:= intTostr(Saldo2);
        grid.Cells[5,j]:= intTostr(Saldo1);
```

```
        grid.Cells[4,i]:= intTostr(Haber2);
        grid.Cells[4,j]:= intTostr(Haber1);

        grid.Cells[3,i]:= intTostr(Debe2);
        grid.Cells[3,j]:= intTostr(Debe1);

        grid.Cells[2,i]:= Concepto2;
        grid.Cells[2,j]:= Concepto1;

        grid.Cells[1,i]:= datetimetostr(FechaHora2);
        grid.Cells[1,j]:= datetimetostr(FechaHora1);

        grid.Cells[0,i]:= nick2;
        grid.Cells[0,j]:= nick1;

       end;
   end;

  grid.RowCount:= cantFilas + 1;
end;

{-------------------------------------------------------------------------------}

Procedure ListadoPremiados_Nomenclador(RaizGanadores: Lib_Auxiliar.tPos; nomenclador:
tNomenclador; var grid: TStringGrid);
var
RegNodoGanador: regNodo;
RegAlmacen: type_GANADOR.TipoRegDatos;
i: tPos;
begin
  if (RaizGanadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, RaizGanadores);
```

```
    i:= ME_GANADORES.PrimerJugada_Ganador(ME_GANADOR, RegNodoGanador);


    while i<> ME_GANADORES.PosNula_Ganadores(ME_GANADOR) do
    begin
      ME_GANADORES.CapturarJugada_Ganador(ME_GANADOR, i, RegAlmacen);


      if (RegAlmacen.Nomenclador = nomenclador)  then
      begin
        Grid.RowCount:= Grid.RowCount + 1;


        Grid.Cells[0, Grid.RowCount-1] := RegNodoGanador.clave;
        Grid.Cells[1, Grid.RowCount-1] := intTostr(RegAlmacen.NroJugada);
        Grid.Cells[2, Grid.RowCount-1] := RegAlmacen.Nomenclador;
        Grid.Cells[3, Grid.RowCount-1] := intTostr(RegAlmacen.Numero);
        Grid.Cells[4, Grid.RowCount-1] := intTostr(RegAlmacen.Importe);
      end;
      i:= ME_GANADORES.SiguienteJugada_Ganador(ME_GANADOR, i);
    end; //while


    ListadoPremiados_Nomenclador(ME_GANADORES.ProximoIzq(ME_GANADOR, RaizGanadores),
nomenclador, grid);
    ListadoPremiados_Nomenclador(ME_GANADORES.ProximoDer(ME_GANADOR, RaizGanadores),
nomenclador, grid);
  end;
end;


{-------------------------------------------------------------------------------}


Procedure ListadoMasPremiados(RaizGanadores: Lib_Auxiliar.tPos; var importe:tImporte;
var grid: TStringGrid; var cantFilas: integer);
var
RegNodoGanador: regNodo;
RegAlmacen: type_GANADOR.TipoRegDatos;
i: tPos;
begin
  if (RaizGanadores<>Lib_Auxiliar.PosNula) then
```

```
  begin
    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, RaizGanadores);

    i:= ME_GANADORES.PrimerJugada_Ganador(ME_GANADOR, RegNodoGanador);

    while i<> ME_GANADORES.PosNula_Ganadores(ME_GANADOR) do
    begin
      ME_GANADORES.CapturarJugada_Ganador(ME_GANADOR, i, RegAlmacen);
      importe:= importe + RegAlmacen.Importe;
      i:= ME_GANADORES.SiguienteJugada_Ganador(ME_GANADOR, i);
    end; //while

    cantFilas:= cantFilas + 1;
    Grid.RowCount:= Grid.RowCount + 1;
    Grid.Cells[0, Grid.RowCount-1] := RegNodoGanador.clave;
    Grid.Cells[1, Grid.RowCount-1] := intTostr(importe);

    ListadoMasPremiados(ME_GANADORES.ProximoIzq(ME_GANADOR, RaizGanadores), importe,
grid, cantFilas);
    ListadoMasPremiados(ME_GANADORES.ProximoDer(ME_GANADOR, RaizGanadores), importe,
grid, cantFilas);
  end;
end;

{-------------------------------------------------------------------------------}

Procedure Listado_GeneralApuestas(var grid: TStringGrid; Nro_Jugada:
Type_JUEGO.tClave);
var
  i:tPos;
begin
  i:= ME_APUESTAS.Primero(ME_APUESTA);

        while i<>ME_APUESTAS.PosNula_Apuestas(ME_APUESTA) do
        begin
          ME_APUESTAS.CapturarInfoME_Apuestas(ME_APUESTA,i,regApuesta);
```

```
          if (regApuesta.NroJugada = Nro_Jugada) then
            begin
              grid.RowCount:= grid.RowCount + 1;
              {Agrego renglon}
              grid.Cells[0, grid.RowCount-1] := regApuesta.Nick;
              grid.Cells[1, grid.RowCount-1] := regApuesta.Nomenclador;
              grid.Cells[2, grid.RowCount-1] := regApuesta.Valor;
              grid.Cells[3, grid.RowCount-1] := intTostr(regApuesta.Importe);
            end;

         i:= ME_APUESTAS.Proximo(ME_APUESTA,i);
        END; //while
end;

{-------------------------------------------------------------------------------}

Procedure Listado_ApuestasUnJugador(var grid: TStringGrid; Nro_Jugada:
Type_JUEGO.tClave; nick: Type_JUGADOR.TClave);
var
  i:tPos;
begin
i:= ME_APUESTAS.Primero(ME_APUESTA);

  while i<>ME_APUESTAS.PosNula_Apuestas(ME_APUESTA) do
  begin
    ME_APUESTAS.CapturarInfoME_Apuestas(ME_APUESTA,i,regApuesta);
    if (regApuesta.NroJugada = Nro_Jugada) then
    begin
      if (regApuesta.Nick = nick) then
      begin
        grid.RowCount:= grid.RowCount + 1;
        {Agrego renglon}
        grid.Cells[0, grid.RowCount-1] := regApuesta.Nick;
        grid.Cells[1, grid.RowCount-1] := regApuesta.Nomenclador;
```

```
        grid.Cells[2, grid.RowCount-1] := regApuesta.Valor;
        grid.Cells[3, grid.RowCount-1] := intTostr(regApuesta.Importe);
      end;
    end;

   i:= ME_APUESTAS.Proximo(ME_APUESTA,i);
  END; //while
end;

{-------------------------------------------------------------------------------}

Procedure Listado_Premios(RaizGanadores: Lib_Auxiliar.tPos; jugada: Type_JUEGO.TClave;
var grid: TStringGrid);
var
RegNodoGanador: regNodo;
RegAlmacen: type_GANADOR.TipoRegDatos;
i: tPos;
begin
  if (RaizGanadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, RaizGanadores);

    i:= ME_GANADORES.PrimerJugada_Ganador(ME_GANADOR, RegNodoGanador);

    while i<> ME_GANADORES.PosNula_Ganadores(ME_GANADOR) do
    begin
      ME_GANADORES.CapturarJugada_Ganador(ME_GANADOR, i, RegAlmacen);

      if (intTostr(RegAlmacen.NroJugada) = jugada)  then
      begin
        grid.RowCount:= grid.RowCount + 1;
        grid.Cells[0, grid.RowCount-1] := RegNodoGanador.clave;
        grid.Cells[1, grid.RowCount-1] := RegAlmacen.Nomenclador;
        grid.Cells[2, grid.RowCount-1] := intTostr(RegAlmacen.Numero);
        grid.Cells[3, grid.RowCount-1] := intTostr(RegAlmacen.Importe);
```

```
      end;
      i:= ME_GANADORES.SiguienteJugada_Ganador(ME_GANADOR, i);
    end; //while

    Listado_Premios(ME_GANADORES.ProximoIzq(ME_GANADOR, RaizGanadores), jugada, grid);
    Listado_Premios(ME_GANADORES.ProximoDer(ME_GANADOR, RaizGanadores), jugada, grid);
  end;
end;

{-------------------------------------------------------------------------------}

procedure Ini_Sesion (var Me: ME_JUGADORES.ME_JUGADOR; var regJugador:
Type_JUGADOR.tRegDatos);
var
  pos:tPos;
begin
  regJugador.UltimaConexion:= NOW();
  regJugador.Estado:= true;

  ME_JUGADORES.BuscarInfoME_Jugadores(Me, pos, regJugador.Nick);
  ME_JUGADORES.ModificarInfoME_Jugadores(Me, regJugador,pos);
end;

{-------------------------------------------------------------------------------}

procedure Fin_Sesion (var Me: ME_JUGADORES.ME_JUGADOR; var regJugador:
Type_JUGADOR.tRegDatos);
var
  pos:tPos;
begin
  regJugador.UltimaConexion:= NOW();
  regJugador.Estado:= false;

  ME_JUGADORES.BuscarInfoME_Jugadores(Me, pos, regJugador.Nick);
  ME_JUGADORES.ModificarInfoME_Jugadores(Me, regJugador,pos);
end;
```

```
{-------------------------------------------------------------------------------}


procedure recuperarRegPartida (Nro_Jugada: Type_JUEGO.tClave; var regBuscado:
Type_JUEGO.TipoRegDatos);

var

  regnulo, regPartida: Type_JUEGO.TipoRegDatos;

  corte: boolean;

begin

  ME_JUEGO.RegistroNulo(regnulo);

  ME_JUEGO.Insertar(ME_JUEGORULETA, regnulo);

  ME_JUEGO.Frente(ME_JUEGORULETA,regPartida);

  ME_JUEGO.Decolar(ME_JUEGORULETA);


  //Ciclo hasta que se encuentra con el regBandera

  while regPartida.NroJugada<>ME_JUEGO.ClaveNula(ME_JUEGORULETA) do

  begin

    //Si encontre el regustro de la jugada que busco, lo guardo en el registro
parametrizado por referencia

    if Nro_Jugada = regPartida.NroJugada then

      regBuscado:= regPartida;


    ME_JUEGO.Insertar(ME_JUEGORULETA, regPartida);

    ME_JUEGO.Frente(ME_JUEGORULETA,regPartida);

    ME_JUEGO.Decolar(ME_JUEGORULETA);

  end;

end;


{-------------------------------------------------------------------------------}


Procedure CantidadJugadoresNuncaGanaron(RaizJugadores: Lib_Auxiliar.tPos; var cant:
tCantidad);

var

RD:Type_JUGADOR.tRegDatos;

esGanador: boolean;
```

```
begin
  esGanador:= false;
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);
    if (RD.Nick<>'ADMINISTRADOR') then
    begin
      JugadorEsUnGanandor(ME_GANADORES.Raiz(ME_GANADOR), RD.Nick, esGanador);
      if not(esGanador) then
        cant:= cant + 1;
    end;
    CantidadJugadoresNuncaGanaron(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores),
cant);
    CantidadJugadoresNuncaGanaron(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores),
cant);
  end;
end;

{-------------------------------------------------------------------------------}

Procedure InsertoPremioUnGanador(regGanador: Type_GANADOR.TipoRegDatos; var regApuesta:
Type_APUESTA.TipoRegDatos; tipoGanador: tNomenclador; Importe: tImporte);
begin
  if (tipoGanador <> type_APUESTA.Nom_Pleno) then
    regApuesta.Valor:= '-1';

  regGanador.Nomenclador:= regApuesta.Nomenclador;
  regGanador.Numero:= strToint(regApuesta.Valor); //
  regGanador.NroJugada:= strToint(regApuesta.NroJugada);
  regGanador.Importe:= Importe; //
  //Inserto ganador
  ME_GANADORES.InsertarInfoME_Ganadores(ME_GANADOR,regGanador, regApuesta.Nick);
end;

{-------------------------------------------------------------------------------}
```

```
Procedure MovimientoApuestaUnJugador(nick: tClave; regApuesta:
Type_APUESTA.TipoRegDatos; Importe: tImporte);

begin

  //Completo registro movimiento en CtaCte

  regCtaCte.Nick:= nick;

  regCtaCte.FechaHora:= Now();

  regCtaCte.Concepto:= Type_ALMACEN._PremioJuego + regApuesta.NroJugada; //

  regCtaCte.Debe:= Importe;

  regCtaCte.haber:= 0;

  regCtaCte.saldo:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE, regApuesta.Nick) +
Importe;

  //Inserto movimiento en CtaCte

  ME_CTACTE.InsertarAlmacen(ME_CUENTACORRIENTE, regCtaCte);

end;


{-------------------------------------------------------------------------------}


Procedure Premios_AcumuladosXJugador(RaizGanadores: Lib_Auxiliar.tPos;
Nick:Type_JUGADOR.tClave; var cantPremios: tCantidad; var Monto: tImporte);

var

RegNodoGanador: regNodo;

RegAlmacen: type_GANADOR.TipoRegDatos;

i: tPos;

begin

  if (RaizGanadores<>Lib_Auxiliar.PosNula) then

  begin

    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, RaizGanadores);


    if (RegNodoGanador.clave = Nick) then

    begin

       i:= ME_GANADORES.PrimerJugada_Ganador(ME_GANADOR, RegNodoGanador);

        while i<> ME_GANADORES.PosNula_Ganadores(ME_GANADOR) do

        begin

          ME_GANADORES.CapturarJugada_Ganador(ME_GANADOR, i, RegAlmacen);

          Monto:= Monto + RegAlmacen.Importe;

          cantPremios:= cantPremios + 1;
```

```
          i:= ME_GANADORES.SiguienteJugada_Ganador(ME_GANADOR, i);
        end; //while
    end;

    Premios_AcumuladosXJugador(ME_GANADORES.ProximoIzq(ME_GANADOR, RaizGanadores),
Nick, cantPremios, Monto);
    Premios_AcumuladosXJugador(ME_GANADORES.ProximoDer(ME_GANADOR, RaizGanadores),
Nick, cantPremios, Monto);
  end;
end;

{-------------------------------------------------------------------------------}

Procedure Premios_AcumuladosXJugadorUnaJugada(RaizGanadores: Lib_Auxiliar.tPos;
Nick:Type_JUGADOR.tClave; jugada: Type_JUEGO.TClave; var Monto: tImporte);
var
RegNodoGanador: regNodo;
RegAlmacen: type_GANADOR.TipoRegDatos;
i: tPos;
begin
  if (RaizGanadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_GANADORES.CapturarInfoME_Ganadores(ME_GANADOR, RegNodoGanador, RaizGanadores);

    if (RegNodoGanador.clave = Nick) then
    begin
       i:= ME_GANADORES.PrimerJugada_Ganador(ME_GANADOR, RegNodoGanador);
        while i<> ME_GANADORES.PosNula_Ganadores(ME_GANADOR) do
        begin
          ME_GANADORES.CapturarJugada_Ganador(ME_GANADOR, i, RegAlmacen);
          if (intTostr(RegAlmacen.NroJugada) = jugada)  then
            Monto:= Monto + RegAlmacen.Importe;

          i:= ME_GANADORES.SiguienteJugada_Ganador(ME_GANADOR, i);
        end; //while
```

```
    end;


    Premios_AcumuladosXJugadorUnaJugada(ME_GANADORES.ProximoIzq(ME_GANADOR,
RaizGanadores), Nick, jugada, Monto);

    Premios_AcumuladosXJugadorUnaJugada(ME_GANADORES.ProximoDer(ME_GANADOR,
RaizGanadores), Nick, jugada, Monto);

  end;

end;


{-------------------------------------------------------------------------------}


 Function JugadorTieneApuestasEnPartida (NroJugada: Type_JUEGO.TClave; nick:
Type_JUGADOR.tClave): boolean;

 var

 i: tpos;

 regApuesta: Type_APUESTA.TipoRegDatos;

 begin

    i:= ME_APUESTAS.Primero(ME_APUESTA);


     while i<>ME_APUESTAS.PosNula_Apuestas(ME_APUESTA) do

     begin

       ME_APUESTAS.CapturarInfoME_Apuestas(ME_APUESTA,i,regApuesta);


       if (regapuesta.NroJugada= nrojugada) then //pregunto si es la misma jugada que
le paso

       begin

         if (regapuesta.Nick= nick) then //pregunto si es el jugador que le paso

          JugadorTieneApuestasEnPartida:= true;

       end;


      i:= ME_APUESTAS.Proximo(ME_APUESTA,i);

     end;

 end;


{-------------------------------------------------------------------------------}
```

```
function Mensaje_Confirmacion(Msje: String): boolean;
begin
  if messagedlg(Msje, mtConfirmation, [mbOk, mbCancel], 0) = mrOk then
    Mensaje_Confirmacion := true
  else
    Mensaje_Confirmacion := false;
end;

{-------------------------------------------------------------------------------}

procedure limpiarGrid(grid: TStringGrid);
var
  i:integer;
begin
  with grid  do
    for i := 0 to RowCount - 1 do
      Rows[i].Clear;
end;

{-------------------------------------------------------------------------------}

Procedure EncabezadoDetalladoJugadores(grid: TStringGrid);
Begin
  with grid do
  Begin
  //cambio el total de columnas
    grid.ColCount:= 9;
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[2] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[3] := Canvas.TextWidth('xxxxxxxxxxx');
    ColWidths[4] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[5] := Canvas.TextWidth('xxxxxxxxx');
    ColWidths[6] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxxxxx');
```

```
    ColWidths[7] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxxxxxx');
    ColWidths[8] := Canvas.TextWidth('xxxxxxxxxxxx');
    Cells[0, 0] := 'NOMBRE';
    Cells[1, 0] := 'APELLIDO';
    Cells[2, 0] := 'NICK';
    Cells[3, 0] := 'TIPO';
    Cells[4, 0] := 'ESTADO';
    Cells[5, 0] := 'ACTIVO';
    Cells[6, 0] := 'FECHA ALTA';
    Cells[7, 0] := 'ULT. CONEXIÓN';
    Cells[8, 0] := 'SALDO';
  End;
End;

{-------------------------------------------------------------------------------}

Procedure EncabezadoArbolJugadores(grid: TStringGrid);
Begin
  with grid do
  Begin
  //cambio el total de columnas
    grid.ColCount:= 2;
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxxxxxxx');
    Cells[0, 0] := 'NIVEL';
    Cells[1, 0] := 'NICK';
  End;
End;

{-------------------------------------------------------------------------------}

Procedure EncabezadoListadoPartidas(grid: TStringGrid);
Begin
  with grid do
```

```
  Begin
  //cambio el total de columnas
    grid.ColCount:= 7;
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxxxx');
    ColWidths[2] := Canvas.TextWidth('xxxxxxxxxxxxxxxxx');
    ColWidths[3] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxx');
    ColWidths[4] := Canvas.TextWidth('xxxxxxxxxxxxxxxxx');
    ColWidths[5] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxxxx');
    ColWidths[6] := Canvas.TextWidth('xxxxxxxxxxxxxxxxx');
    Cells[0, 0] := 'NRO.PARTIDA';
    Cells[1, 0] := 'FECHA-HORA';
    Cells[2, 0] := 'TOTAL J.REALES';
    Cells[3, 0] := 'TOTAL J.FICTICIOS';
    Cells[4, 0] := '$ APOSTADO';
    Cells[5, 0] := '$ PREMIOS PAGADOS';
    Cells[6, 0] := 'NRO. BOLILLA';
  End;
End;

{-------------------------------------------------------------------------------}

Procedure EncabezadoJugadoresMasGanaron(grid: TStringGrid);
Begin
  with grid do
  Begin
  //cambio el total de columnas
    grid.ColCount:= 2;
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxx');
    Cells[0, 0] := 'NICK';
    Cells[1, 0] := 'SALDO';
  End;
```

```
End;

{-------------------------------------------------------------------------------}

Procedure EncabezadoGanadoresXPremio(grid: TStringGrid);
Begin
  with grid do
  Begin
  //cambio el total de columnas
    grid.ColCount:= 5;
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[2] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[3] := Canvas.TextWidth('xxxxxxxxxxxx');
    ColWidths[4] := Canvas.TextWidth('xxxxxxxxxxxxxx');
    Cells[0, 0] := 'NICK';
    Cells[1, 0] := 'JUGADA';
    Cells[2, 0] := 'NOMENCLADOR';
    Cells[3, 0] := 'NUMERO';
    Cells[4, 0] := 'PREMIO';
  End;
End;

{-------------------------------------------------------------------------------}

Procedure EncabezadoFiltroCtaCte(grid: TStringGrid);
Begin
  with grid do
  Begin
  //cambio el total de columnas
    grid.ColCount:= 6;
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxx');
```

```
    ColWidths[2] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxxxxxxxxxxx');
    ColWidths[3] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[4] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[5] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    Cells[0, 0] := 'NICK';
    Cells[1, 0] := 'FECHA/HR';
    Cells[2, 0] := 'CONCEPTO';
    Cells[3, 0] := 'DEBE';
    Cells[4, 0] := 'HABER';
    Cells[5, 0] := 'SALDO';
  End;
End;

{-------------------------------------------------------------------------------}

Procedure EncabezadosComunApuestas(grid: TStringGrid);
begin
  with grid do
  Begin
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxxxxxx');
    ColWidths[2] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[3] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    Cells[0, 0] := 'NICK';
    Cells[1, 0] := 'NOMENCLADOR';
    Cells[2, 0] := 'VALOR';
    Cells[3, 0] := 'IMPORTE';
  End;
end;

{-------------------------------------------------------------------------------}

Procedure EncabezadosComunJugadores(grid: TStringGrid);
Begin
```

```
  with grid do
  Begin
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[2] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[3] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[4] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[5] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxxxx');
    Cells[0, 0] := 'NICK';
    Cells[1, 0] := 'NOMBRE J.';
    Cells[2, 0] := 'APELLIDO J.';
    Cells[3, 0] := 'TIPO';
    Cells[4, 0] := 'BLOQUEADO';
    Cells[5, 0] := 'ULT. CONEXIÓN';
  End;
End;

{-------------------------------------------------------------------------------}

Procedure EncabezadoGridPremios(grid: TStringGrid);
begin
  with grid do
  Begin
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[2] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    ColWidths[3] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxx');
    Cells[0, 0] := 'JUGADOR';
    Cells[1, 0] := 'NOMENCLADOR';
    Cells[2, 0] := 'NUMERO';
    Cells[3, 0] := 'PREMIO';
  End;
end;
```

```
{------------------------------------------------------------------------------}


Procedure EncabezadosComunFicticios(grid: TStringGrid);
Begin
  with grid do
  Begin
  // Título de las columnas
    ColWidths[0] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[1] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[2] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[3] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[4] := Canvas.TextWidth('xxxxxxxxxxxxxxxx');
    ColWidths[5] := Canvas.TextWidth('xxxxxxxxxx');
    ColWidths[6] := Canvas.TextWidth('xxxxxxxxxxxxxxxxxxxxxxxxxxxx');
    Cells[0, 0] := 'NICK';
    Cells[1, 0] := 'NOMBRE J.';
    Cells[2, 0] := 'APELLIDO J.';
    Cells[3, 0] := 'TIPO';
    Cells[4, 0] := 'ESTADO';
    Cells[5, 0] := 'SALDO';
    Cells[6, 0] := 'ALTA';
  End;
End;


END.
```

3.2 Formularios

3.2.1 Formulario Login

```
unit Form_Login;

interface

uses
  ME_JUGADORES,
  Type_JUGADOR,
  ME_CTACTE,
  Type_ALMACEN,
  ME_JUEGO,
  Type_JUEGO,
  ME_APUESTAS,
  Type_APUESTA,
  ME_GANADORES,
  Type_GANADOR,
  Lib_Auxiliar,
  Lib_AuxJuego,
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Buttons, StdCtrls, ExtCtrls, jpeg, Vcl.Menus;

type
  TF_Login = class(TForm)
    Panel1: TPanel;
    PanelGeneral: TPanel;
    Label6: TLabel;
    PanelBotones1: TPanel;
```

```
    BtnLogin: TButton;
    BtnAltas: TButton;
    PanelFormulario: TPanel;
    Lab_apellido: TLabel;
    Lab_nombre: TLabel;
    Lab_contrasenia: TLabel;
    Lab_nick: TLabel;
    Txt_Nombre: TEdit;
    Txt_Contrasenia: TEdit;
    Txt_NICK: TEdit;
    BtnOk: TBitBtn;
    BtnCancelar: TBitBtn;
    Txt_Apellido: TEdit;
    Btn_HideShowContrasenia: TBitBtn;
    Menu_Login: TMainMenu;
    SALIR1: TMenuItem;
    Function CamposAltaOk():Boolean;
    Function  NickValido():Boolean;
    Function CamposLoginOk():Boolean;
    procedure Limpiar_Formulario();
    procedure show_apellidoNombre();
    procedure Pantalla_InicioLogin();
    procedure hide_apellidoNombre();
    procedure BtnLoginClick(Sender: TObject);
    procedure BtnOkClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BtnAltasClick(Sender: TObject);
    procedure BtnCancelarClick(Sender: TObject);
    procedure BtnEraseClick(Sender: TObject);
    procedure Txt_ContraseniaKeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure Txt_NombreKeyUp(Sender: TObject; var Key: Word;
      Shift: TShiftState);
    procedure Txt_NombreKeyPress(Sender: TObject; var Key: Char);
    procedure Txt_ApellidoKeyPress(Sender: TObject; var Key: Char);
```

```
    procedure FormShow(Sender: TObject);
    procedure Btn_HideShowContraseniaClick(Sender: TObject);
    procedure Txt_NICKKeyPress(Sender: TObject; var Key: Char);
    procedure Txt_ContraseniaKeyPress(Sender: TObject; var Key: Char);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure SALIR1Click(Sender: TObject);



  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Login: TF_Login;
  Opcion: char;
  AdminLog : boolean; //si es el administrador quien se loguea es true
  NickLogueado : Type_JUGADOR.tClave;

  ME_JUGADOR:  ME_JUGADORES.ME_Jugador;
  regJugador: Type_JUGADOR.tRegDatos;
  posjugador: Lib_Auxiliar.tpos;

  ME_CUENTACORRIENTE: ME_CTACTE.ME_CUENTA;
  RegCtaCte: Type_ALMACEN.tipoRegAlmacen;
  posAlmacen: Lib_Auxiliar.tpos;
  saldoAcumulado: tImporte;

  ME_JUEGORULETA: ME_JUEGO.ME_RULETA;
  RegJuegoRuleta: Type_JUEGO.TipoRegDatos;

  ME_APUESTA: ME_APUESTAS.ME_APUESTA;
  RegApuesta: Type_APUESTA.TipoRegDatos;
```

```
  ME_GANADOR: ME_GANADORES.ME_GANADOR;
  RegGanador: Type_GANADOR.TipoRegDatos;




implementation

uses
  Form_Jugadores, Form_Croupier, Form_Juego;



{$R *.dfm}

{-------------------------------------------------------------------------------}

procedure TF_Login.Btn_HideShowContraseniaClick(Sender: TObject);
begin
  if (Txt_Contrasenia.PasswordChar = #0) then
  begin
    Txt_Contrasenia.PasswordChar:='*';
    Btn_HideShowContrasenia.Glyph.LoadFromFile(Lib_Auxiliar.rutaIconoOjoAbierto);
  end
  else
  begin
    Txt_Contrasenia.PasswordChar:= #0;
    Btn_HideShowContrasenia.Glyph.LoadFromFile(Lib_Auxiliar.rutaIconoOjoCerrado);
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Login.BtnAltasClick(Sender: TObject);
```

```
begin
  Opcion:='B';
  BtnLogin.Enabled:=false;
  panelformulario.Enabled:= true;
  show_ApellidoNombre;
  Txt_NICK.SetFocus;
  Limpiar_Formulario;
end;

{-------------------------------------------------------------------------------}

procedure TF_Login.BtnCancelarClick(Sender: TObject);
begin
  Btn_HideShowContrasenia.hide;
  hide_apellidoNombre;
  BtnLogin.Enabled:=true;
  BtnAltas.Enabled:=true;
  Limpiar_Formulario;
end;

{-------------------------------------------------------------------------------}

procedure TF_Login.BtnLoginClick(Sender: TObject);
begin
  Opcion:='A';
  BtnAltas.Enabled:=false;
  panelformulario.Enabled:= true;
  Txt_NICK.Enabled:= true;
  Txt_Contrasenia.Enabled:= true;
  Txt_Nombre.Enabled:= true;
  Txt_Apellido.Enabled:= true;
  Txt_NICK.SetFocus;
  Limpiar_Formulario;
  hide_apellidoNombre;
```

```
  Txt_NICK.Text:='ADMINISTRADOR';
  Txt_Contrasenia.Text:='MONDONGO';
  PanelFormulario.Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Login.BtnOkClick(Sender: TObject);
begin
  Btn_HideShowContrasenia.hide;
  NickLogueado:= Txt_NICK.Text;

case Opcion of
  'A': begin //LOGIN
      if (CamposLoginOk) then
      begin
        if not
(ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,PosJugador,NickLogueado)) then
        begin //Nick incorrecto
          MessageDlg('Nick incorrecto',mtError , [mbOK], 0);
          Txt_NICK.Clear;
          Txt_CONTRASENIA.Clear;
          Txt_NICK.SetFocus;
        end
        else
          begin
            //capturo registro para comparar nick con su contraseña
            ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,regJugador,PosJugador);

            if not (Txt_CONTRASENIA.Text = regJugador.Contrasenia) then
            begin //contraseña incorrecta
              MessageDlg('La contraseña no coincide con el Nick ingresado',mtError ,
[mbOK], 0);
              Txt_CONTRASENIA.Clear;
              Txt_CONTRASENIA.SetFocus;
            end
```

```
            else //Usuario logueado correctamente
              begin
              Form_Login.F_Login.hide;
                //Consulto si el jugador ya esta en el juego. Si esta activo, esta en
el juego
                if not (regJugador.Estado) then
                begin //jugador no esta en el juego. Pregunto si es el crupier
                   if (regJugador.Nick = 'ADMINISTRADOR') then
                   begin //ADMINISTRADOR LOGUEADO...
                     AdminLog:=true;
                     Limpiar_Formulario;
                     Form_Croupier.F_Croupier.Show;
                   end
                   else  //USUARIO COMUN LOGUEADO...
                     begin
                      AdminLog:=false;
                      Form_Jugadores.F_Jugadores.Show;

                      //busco saldo acumulado del jugador logueado
                      saldoAcumulado:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,
regJugador.Nick);
                     end;
                     Lib_AuxJuego.Ini_Sesion(ME_JUGADOR, regJugador); //Inicio sesion,
ya sea administrador o usuario común
                end
                else //jugador ya esta en el juego
                begin
                   MessageDlg('El usuario ya esta en el juego',mtInformation , [mbOK],
0);
                   Limpiar_Formulario;
                   Txt_NICK.SetFocus;
                end;
              end; //fin usuario logueado correctamente
          end; //fin existe nick
      end //fin campos completados correctamente
      else
```

```
        MessageDlg('Debe completar todos los campos.',mtError , [mbOK], 0);
  end;
  'B': begin //ALTA
      if (CamposAltaOk) then
      begin
        if (NickValido) then
        begin
          if (ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,PosJugador, NickLogueado))
then
          begin //Nick en uso
            MessageDlg('Nick ya esta en uso. Intente con otro',mtConfirmation , [mbOK],
0);
            Txt_NICK.Clear;
            Txt_NICK.SetFocus;
          end
          else
          begin  //Inserto juegador
            regJugador.Nick:= NickLogueado;
            regJugador.Contrasenia:= Txt_Contrasenia.Text;
            regJugador.Apellido:= Txt_Apellido.Text;
            regJugador.Nombre:= Txt_Nombre.Text;
            regJugador.Alta:= now();
            regJugador.UltimaConexion:= now();
            regJugador.Bloqueado:= false;
            regJugador.Estado:= false;
            regJugador.TipoJugador:= true;
            ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR, PosJugador,
regJugador.Nick);
            ME_JUGADORES.InsertarInfoME_Jugadores(ME_JUGADOR,regJugador,PosJugador);

            //Cargo datos de ctacte
            RegCtaCte.Nick:=regJugador.Nick;
            RegCtaCte.FechaHora:= Now();
            RegCtaCte.Concepto:= Type_ALMACEN.tipoConceptos.Creacion_Cuenta;
            RegCtaCte.Debe:= ME_JUEGO.ApuestaMinima(ME_JUEGORULETA)*15; //50*15 = 750
            RegCtaCte.haber:= 0;
```

```
            RegCtaCte.saldo:= RegCtaCte.Debe;//al crearse la cuenta se carga 15 veces
la apuesta minima y se refleja en el saldo

            //inserto cuenta corriente

            ME_CTACTE.InsertarAlmacen(ME_CUENTACORRIENTE,RegCtaCte);


            MessageDlg('El usuario ha sido registrado!',mtConfirmation , [mbOK], 0);

            Limpiar_Formulario;

            hide_apellidoNombre;


            PanelFormulario.Enabled:=false;

            BtnLogin.Enabled:=true;

            BtnAltas.Enabled:=true;

            BtnLogin.SetFocus;

          end;

        end//NickValido

        else

        begin

          MessageDlg('NICK INVÁLIDO!' + #13 + #13 +'Comienzo del Nick con "' +
Lib_Auxiliar.Fict_IniNick + '"' + #13 + 'está reservado solo para los
ficticios!',mtWarning , [mbOK], 0);

          Txt_Nick.Clear;

          Txt_Nick.SetFocus;

        end;

      end //CamposAltaOk

      else

        MessageDlg('Debe completar todos los campos.',mtError , [mbOK], 0);

  end;//fin opcion B

end;//case

end;


{-------------------------------------------------------------------------}


procedure TF_Login.BtnEraseClick(Sender: TObject);

begin

  ME_JUGADORES.CerrarME_Jugadores(ME_JUGADOR);

  ME_JUGADORES.DestruirME_Jugadores(ME_JUGADOR);
```

```
  ME_CTACTE.CerrarMe_CtaCte(ME_CUENTACORRIENTE);
  ME_CTACTE.DestruirMe_CtaCte(ME_CUENTACORRIENTE);

  ME_JUEGO.CerrarME(ME_JUEGORULETA);
  ME_JUEGO.DestruirME(ME_JUEGORULETA);

  ME_APUESTAS.CerrarME_Apuestas(ME_APUESTA);
  ME_APUESTAS.DestruirME_Apuestas(ME_APUESTA);

  ME_GANADORES.CerrarME_Ganadores(ME_GANADOR);
  ME_GANADORES.DestruirME_Ganadores(ME_GANADOR);

end;

{-------------------------------------------------------------------------------}

Function  TF_Login.CamposAltaOk():Boolean;
begin
  CamposAltaOk:= (Txt_Nombre.Text<>'')and(Txt_Contrasenia.Text<>'')and
(Txt_NICK.Text<>'') and
               (Txt_Apellido.Text<>'');
end;

{-------------------------------------------------------------------------------}

Function  TF_Login.NickValido():Boolean;
var
  comienzoNick: string[2];
begin
  comienzoNick:= copy(Txt_NICK.Text, 1, 2);
  NickValido:= (comienzoNick <> Lib_Auxiliar.Fict_IniNick);
end;

{-------------------------------------------------------------------------------}
```

```
Function  TF_Login.CamposLoginOk():Boolean;
begin
  CamposLoginOk:= (Txt_NICK.Text<>'') and (Txt_Contrasenia.Text<>'');
end;

{-------------------------------------------------------------------------------}


procedure TF_Login.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  ME_JUGADORES.CerrarME_Jugadores(ME_JUGADOR);
  ME_CTACTE.CerrarMe_CtaCte(ME_CUENTACORRIENTE);
  ME_JUEGO.CerrarME(ME_JUEGORULETA);
  ME_APUESTAS.CerrarME_Apuestas(ME_APUESTA);
  ME_GANADORES.CerrarME_Ganadores(ME_GANADOR);
end;

procedure TF_Login.FormCreate(Sender: TObject);
var
  regAuxJuego: TYPE_JUEGO.TipoRegDatos;
begin
  ME_JUGADORES.CrearME_Jugadores(ME_JUGADOR);
  ME_JUGADORES.AbrirME_Jugadores(ME_JUGADOR);

  ME_CTACTE.CrearMe_CtaCte(ME_CUENTACORRIENTE);
  ME_CTACTE.AbrirMe_CtaCte(ME_CUENTACORRIENTE);

  ME_JUEGO.CrearME(ME_JUEGORULETA);
  ME_JUEGO.AbrirME(ME_JUEGORULETA);

  ME_APUESTAS.CrearME(ME_APUESTA);
  ME_APUESTAS.AbrirME_Apuestas(ME_APUESTA);

  ME_GANADORES.CrearME_Ganadores(ME_GANADOR);
```

```
  ME_GANADORES.AbrirME_Ganadores(ME_GANADOR);

  if ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) then
  begin
    regJugador.Nick:='ADMINISTRADOR';
    regJugador.Contrasenia:= 'MONDONGO';
    regJugador.Nombre:='CRUPIER';
    regJugador.Apellido:='CRUPIER';
    regJugador.Alta:=NOW();
    regJugador.UltimaConexion:=NOW();
    regJugador.Estado:=false;
    regJugador.TipoJugador:=true;

    ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,regJugador.Nick);
    ME_JUGADORES.InsertarInfoME_Jugadores(ME_JUGADOR,regJugador,PosJugador);
  end;

  //operaciones generales
  AdminLog:=false;
  hide_apellidoNombre;
  self.Position := poScreenCenter;
  self.WindowState := wsMaximized;
end;


{-------------------------------------------------------------------------------}

procedure TF_Login.FormShow(Sender: TObject);
begin
  Pantalla_InicioLogin();

  ME_JUGADORES.AbrirME_Jugadores(ME_JUGADOR);
  ME_CTACTE.AbrirMe_CtaCte(ME_CUENTACORRIENTE);
  ME_JUEGO.AbrirME(ME_JUEGORULETA);
  ME_APUESTAS.AbrirME_Apuestas(ME_APUESTA);
```

```
  ME_GANADORES.AbrirME_Ganadores(ME_GANADOR);


end;

{-------------------------------------------------------------------------------}

procedure TF_Login.Limpiar_Formulario();
begin
  Txt_NICK.Clear;
  Txt_Contrasenia.Clear;
  Txt_Nombre.Clear;
  Txt_Apellido.Clear;
end;
{-------------------------------------------------------------------------------}

procedure TF_Login.show_apellidoNombre();
begin
  Lab_nombre.Visible:=true;
  Lab_apellido.Visible:=true;
  Txt_Nombre.Visible:=true;
  Txt_Apellido.Visible:=true;
  Lab_nombre.Show;
  Lab_apellido.Show;
  Txt_Nombre.Show;
  Txt_Apellido.Show;
end;

{-----------------------------------------------------------------------}

procedure TF_Login.Txt_ApellidoKeyPress(Sender: TObject; var Key: Char);
const
  CARAC_HABILITADOS = ['a'..'z', 'A'..'Z', #0..#27];//solo letras y tecla borrar
begin
  if not (Key IN CARAC_HABILITADOS) then
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
```

```
end;


{-------------------------------------------------------------------------}


procedure TF_Login.Txt_ContraseniaKeyPress(Sender: TObject; var Key: Char);

const

  CARAC_HABILITADOS = ['a'..'z', 'A'..'Z', '0'..'9', #0..#27];//solo letras, numero y
borrar

begin

  if not (Key IN CARAC_HABILITADOS) then

    Key:=#0; //la tecla vale null si presiona caracteres no habilitados

end;


procedure TF_Login.Txt_ContraseniaKeyUp(Sender: TObject; var Key: Word;

  Shift: TShiftState);

begin

  if (length(Txt_NICK.Text)<6) then

  begin

    MessageDlg('Nick debe tener al menos 6 caracteres',mtConfirmation , [mbOK], 0);

    Txt_CONTRASENIA.Clear;

    Txt_NICK.Clear;

    Txt_NICK.SetFocus;

  end

  else

    Btn_HideShowContrasenia.show;

end;



{-------------------------------------------------------------------------}

procedure TF_Login.Txt_NICKKeyPress(Sender: TObject; var Key: Char);

const

  CARAC_HABILITADOS = ['a'..'z', 'A'..'Z', '0'..'9', #95, #46, #0..#27];//solo letras,
numeros, signo punto, signo guion bajo y tecla borrar

begin

  if not (Key IN CARAC_HABILITADOS) then

    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
```

```
end;

{-------------------------------------------------------------------------}

procedure TF_Login.Txt_NombreKeyPress(Sender: TObject; var Key: Char);
const
  CARAC_HABILITADOS = ['a'..'z', 'A'..'Z', #0..#27];//solo letras y tecla borrar
begin
  if not (Key IN CARAC_HABILITADOS) then
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
end;

{-------------------------------------------------------------------------}

Procedure TF_Login.Txt_NombreKeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  if (length(Txt_Contrasenia.Text)<4) then
  begin
    MessageDlg('La contraseña debe tener 4 caracteres como minimo',mtConfirmation ,
[mbOK], 0);
    Txt_Contrasenia.Clear;
    Txt_Contrasenia.SetFocus;
  end;
end;

{-------------------------------------------------------------------------}

procedure TF_Login.hide_apellidoNombre();
begin
  Lab_nombre.Visible:=false;
  Lab_apellido.Visible:=false;
  Txt_Nombre.Visible:=false;
  Txt_Apellido.Visible:=false;
  Lab_nombre.Hide;
```

```
  Lab_apellido.Hide;
  Txt_Nombre.Hide;
  Txt_Apellido.Hide;
end;

{-------------------------------------------------------------------------}

procedure TF_Login.Pantalla_InicioLogin();
begin
  Form_Juego.F_Juego.Hide;
  Btn_HideShowContrasenia.Hide;
  BtnLogin.Enabled:= true;
  BtnAltas.Enabled:= true;
  PanelFormulario.Enabled:= false;
  Txt_NICK.Clear;
  Txt_Contrasenia.Clear;
  Txt_Nombre.Clear;
  Txt_Apellido.Clear;
end;

{-------------------------------------------------------------------------}

procedure TF_Login.SALIR1Click(Sender: TObject);
begin
  F_Login.Close;
end;
END.
```

3.2.2 Formulario Croupier

```
unit Form_Croupier;


interface


uses
  ME_JUGADORES,
  Type_JUGADOR,
  ME_CTACTE,
  Type_ALMACEN,
  ME_JUEGO,
  Type_JUEGO,
  ME_APUESTAS,
  Type_APUESTA,
  ME_GANADORES,
  Type_GANADOR,
  Lib_Auxiliar,
  Lib_AuxJuego,
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Menus, Vcl.StdCtrls, Vcl.ExtCtrls,
  Vcl.Buttons, Vcl.Imaging.jpeg;


type
  TF_Croupier = class(TForm)
    MainMenu1: TMainMenu;
    Juego1: TMenuItem;
    Listados1: TMenuItem;
    Bloquear1: TMenuItem;
```

```
    Activos1: TMenuItem;
    Jugarpartida1: TMenuItem;
    Listadograljugadores1: TMenuItem;
    Ganadoresdeterminadopremio1: TMenuItem;
    Jugadoresmasdineroganaron1: TMenuItem;
    Jugadoresquenuncaganaron1: TMenuItem;
    Listagraldepartidas1: TMenuItem;
    HerramientasdelME1: TMenuItem;
    Mostrararboljugadores1: TMenuItem;
    Mostrararbolganadores1: TMenuItem;
    Cerrarsesion1: TMenuItem;
    Crearficticios1: TMenuItem;
    Panel_CroupierGral: TPanel;
    Panel_CrearFicticios: TPanel;
    Label1: TLabel;
    Btn_crearFicticios: TButton;
    Bit_closeCrearFicticios: TBitBtn;
    Edit_cantFicticios: TEdit;
    Label2: TLabel;
    Fichaunjugador1: TMenuItem;
    Panel_EncabezadoCroupier: TPanel;
    Panel2: TPanel;
    Label3: TLabel;
    Panel15: TPanel;
    Label4: TLabel;
    Lbl_nickCroupier: TLabel;
    Panel16: TPanel;
    Label5: TLabel;
    Lbl_fechaCroupier: TLabel;
    Label6: TLabel;
    Lbl_horaCroupier: TLabel;
    Img_fondoCroupier: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Cerrarsesion1Click(Sender: TObject);
    procedure Jugarpartida1Click(Sender: TObject);
```

```
    procedure FormShow(Sender: TObject);
    procedure Crearficticios1Click(Sender: TObject);
    procedure Edit_cantFicticiosKeyPress(Sender: TObject; var Key: Char);
    procedure Btn_crearFicticiosClick(Sender: TObject);
    procedure Bloquear1Click(Sender: TObject);
    procedure Listadograljugadores1Click(Sender: TObject);
    procedure Ganadoresdeterminadopremio1Click(Sender: TObject);
    procedure Jugadoresmasdineroganaron1Click(Sender: TObject);
    procedure Jugadoresquenuncaganaron1Click(Sender: TObject);
    procedure CtaCtedeunjugador1Click(Sender: TObject);
    procedure Listagraldepartidas1Click(Sender: TObject);
    procedure Mostrararboljugadores1Click(Sender: TObject);
    procedure Mostrararbolganadores1Click(Sender: TObject);
    procedure Activos1Click(Sender: TObject);
    procedure Bit_closeCrearFicticiosClick(Sender: TObject);
    procedure Fichaunjugador1Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Croupier: TF_Croupier;

implementation

uses
   Form_Login, Form_Juego, Form_PanelCaseButtons, Form_Listados;

{$R *.dfm}

{-------------------------------------------------------------------------------}
```

```
procedure TF_Croupier.Activos1Click(Sender: TObject);
var
  Cant: tCantidad; // cantidad jugadores en general
  Cant_Activos: tCantidad; // cantidad de jugadores activos
begin
  Panel_CrearFicticios.Hide;
  Form_Listados.opListado:= 9;

  cant:= 0;
  Cant_Activos:= 0;

  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);
  Lib_AuxJuego.Cant_JugadoresActivos(ME_JUGADORES.Raiz(ME_JUGADOR), Cant_Activos);

   if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cant>1)) then //si cant es
>1 entonces al menos hay un jugador que no es el administrador
   begin
        //Si hay jugadores activos. Podria usar una funcion booleana. Pero la cantidad
de activos la voy a necesitar para mostrar
      if (Cant_Activos>=1) then
        F_Listados.Show
      else
        MessageDlg('No hay jugadores activos!',mtWarning, [mbOK], 0);
    end
    else
      MessageDlg('No existen jugadores!!',mtWarning, [mbOK], 0);
end;

{-------------------------------------------------------------------------------}

procedure TF_Croupier.Bit_closeCrearFicticiosClick(Sender: TObject);
begin
   Panel_CrearFicticios.Hide;
   Edit_cantFicticios.Clear;
end;
```

```
{-------------------------------------------------------------------------------}


procedure TF_Croupier.Bloquear1Click(Sender: TObject);
var
  Cant: tCantidad; // cantidad jugadores en general
begin
  Panel_CrearFicticios.Hide;
  Cant:= 0;
  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);

  if (not(ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR)) and (cant>1)) then //si cant es
>1 entonces al menos hay un jugador que no es el administrador
    begin
      F_PanelCaseButtons.MostrarEnGridJugadores();
    end
    else
      begin
        MessageDlg('No existen jugadores!!',mtWarning, [mbOK], 0);
        Form_Croupier.F_Croupier.Panel_CroupierGral.Enabled:= true;
      end;

  F_PanelCaseButtons.Panel_LockUnlock.Enabled:= true;
  F_PanelCaseButtons.Panel_LockUnlock.show;

  F_PanelCaseButtons.Panel_RegalarCredito.Enabled:= false;
  F_PanelCaseButtons.Panel_RegalarCredito.Hide;
  F_PanelCaseButtons.Lbl_regalarCred.Hide;
end;

{-------------------------------------------------------------------------------}


procedure TF_Croupier.Btn_crearFicticiosClick(Sender: TObject);
begin
  if(Edit_cantFicticios.Text <> '')then
  begin
```

```
    if Lib_Auxiliar.Mensaje_Confirmacion('¿Esta seguro que quiere crear '+
Edit_cantFicticios.Text + ' ficticios?') then

    begin

      Lib_AuxJuego.Insertar_JugadoresFicticios(ME_JUGADOR,
strToint(Edit_cantFicticios.Text));

      Edit_cantFicticios.Clear;

      Panel_CrearFicticios.Hide;

      MessageDlg(Edit_cantFicticios.Text + ' jugadores ficticios han sido creado con
éxito.',mtConfirmation , [mbOK], 0);

    end

    else

    begin

      Edit_cantFicticios.SetFocus;

      Edit_cantFicticios.Clear;

    end;

  end

  else

    begin

      MessageDlg('Debe ingresar cantidad a crear.',mtWarning , [mbOK], 0);

      Edit_cantFicticios.SetFocus;

    end;

end;


{-------------------------------------------------------------------------------}


procedure TF_Croupier.Cerrarsesion1Click(Sender: TObject);

begin

  if (Lib_Auxiliar.Mensaje_Confirmacion('¿Esta seguro que quiere cerrar sesión?')) then

  begin

    {CerrarSesion}

    Lib_AuxJuego.Fin_Sesion(ME_JUGADOR, regJugador);

    F_Croupier.Hide;

    Form_Login.F_Login.show;

  end;

end;
```

```
{-------------------------------------------------------------------------------}


procedure TF_Croupier.Crearficticios1Click(Sender: TObject);

begin

  Panel_CrearFicticios.Show;

  Edit_cantFicticios.SetFocus;

end;


{-------------------------------------------------------------------------------}


procedure TF_Croupier.CtaCtedeunjugador1Click(Sender: TObject);

var

  cant:tCantidad;

begin

  cant:=0;

//-   Cuenta corriente de un Jugador

  Form_Listados.opListado:= 5;


  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);


  if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cant>1)) then //si cant es >1
entonces al menos hay un jugador que no es el administrador

    Form_Listados.F_Listados.Show

  else

    MessageDlg('No existen jugadores!!',mtWarning , [mbOK], 0);

end;


{-------------------------------------------------------------------------------}


procedure TF_Croupier.Edit_cantFicticiosKeyPress(Sender: TObject;

  var Key: Char);

const

  CARAC_HABILITADOS = ['0'..'9', #0..#27];//solo letras y tecla borrar

begin

  if not (Key IN CARAC_HABILITADOS) then
```

```
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
end;

{-------------------------------------------------------------------------------}



procedure TF_Croupier.Fichaunjugador1Click(Sender: TObject);
var
  cant:tCantidad;
begin
  Panel_CrearFicticios.Hide;
  cant:=0;
//-   Ficha de un Jugador
  Form_Listados.opListado:= 5;


  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);


  if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cant>1)) then //si cant es >1
entonces al menos hay un jugador que no es el administrador
    Form_Listados.F_Listados.Show
  else
    MessageDlg('No existen jugadores!!',mtWarning , [mbOK], 0);
end;


{-------------------------------------------------------------------------------}


procedure TF_Croupier.FormCreate(Sender: TObject);
begin
  self.Position := poScreenCenter;
  self.WindowState := wsMaximized;
end;


{-------------------------------------------------------------------------------}


procedure TF_Croupier.FormShow(Sender: TObject);
```

```
begin
  Panel_CrearFicticios.Hide;
  Lbl_nickCroupier.Caption:= NickLogueado;
  F_Juego.ActualizarFechaHora(Lbl_fechaCroupier, Lbl_horaCroupier); //Actualizo fecha y
hora a mostrar
end;


{-------------------------------------------------------------------------------}


procedure TF_Croupier.Ganadoresdeterminadopremio1Click(Sender: TObject);
var
  cantJ:tCantidad;
begin
  Panel_CrearFicticios.Hide;
  cantJ:=0;
//-   Listado ganadores determinado premio
  Form_Listados.opListado:= 2;


  cantJ:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);


  if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cantJ>1)) then //si cant es
>1 entonces al menos hay un jugador que no es el administrador
  begin
    if not(ME_JUEGO.MEVacio(ME_JUEGORULETA)) then
    begin
      if not ME_GANADORES.MeVacio_Ganadores(ME_GANADOR) then
        Form_Listados.F_Listados.Show
      else
        MessageDlg('No existen ganadores!!',mtWarning , [mbOK], 0);
    end
    else
      MessageDlg('No se ha efectuado ninguna jugada aún!!',mtWarning , [mbOK], 0);
  end
  else
    MessageDlg('No existen jugadores!!',mtWarning , [mbOK], 0);
end;
```

```
{-------------------------------------------------------------------------------}

procedure TF_Croupier.Jugadoresmasdineroganaron1Click(Sender: TObject);
var
  cantJ: tCantidad; //cantidad jugadores
begin
  Panel_CrearFicticios.Hide;
  cantJ:=0;
//-   Listado de los jugadores que mas dinero ganaron
  Form_Listados.opListado:= 3;

  cantJ:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);

  if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cantJ>1)) then //si cant es
>1 entonces al menos hay un jugador que no es el administrador
  begin
    if not(ME_JUEGO.MEVacio(ME_JUEGORULETA)) then
    begin
      if not ME_GANADORES.MeVacio_Ganadores(ME_GANADOR) then
        Form_Listados.F_Listados.Show
      else
        MessageDlg('No existen ganadores!!',mtWarning , [mbOK], 0);
    end
    else
      MessageDlg('No se ha efectuado ninguna jugada aún!!',mtWarning , [mbOK], 0);
  end
  else
    MessageDlg('No existen jugadores!!',mtWarning , [mbOK], 0);
end;

{-------------------------------------------------------------------------------}

procedure TF_Croupier.Jugadoresquenuncaganaron1Click(Sender: TObject);
var
```

```
  cantJ:tCantidad;//cantidad jugadores
  cantJNG: tCantidad; //cantidad jugadores que nunca ganaron
begin
  Panel_CrearFicticios.Hide;
  cantJ:= 0;
  cantJNG:= 0;
//-   Listado de los jugadores que nunca ganaron apuestas.
  Form_Listados.opListado:= 4;

  cantJ:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);
  Lib_AuxJuego.CantidadJugadoresNuncaGanaron(ME_JUGADORES.Raiz(ME_JUGADOR), cantJNG);

  if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cantJ>1)) then //si cant es
>1 entonces al menos hay un jugador que no es el administrador
  begin
    if not(ME_JUEGO.MEVacio(ME_JUEGORULETA)) then
    begin
      if (cantJNG>0) then
        Form_Listados.F_Listados.Show
      else
        MessageDlg('No existen jugadores que no hayan perdido alguna
apuesta!!',mtWarning , [mbOK], 0);
    end
    else
      MessageDlg('No se ha efectuado ninguna jugada aún!!',mtWarning , [mbOK], 0);
  end
  else
    MessageDlg('No existen jugadores!!',mtWarning , [mbOK], 0);
end;

{-------------------------------------------------------------------------------}

procedure TF_Croupier.Jugarpartida1Click(Sender: TObject);
begin
  //Msj de advertencia para que no se intente salir del juego si el mismo no esta en
estado 4 (fin de juego y repartido los premios)
```

```
  MessageDlg('TENGA PRESENTE QUE: ' + #13 + #13 + ' Una vez seleccionado "CREAR PARTIDA
NUEVA" solo podrá salir del juego cuando el mismo finalice',mtInformation , [mbOK], 0);

  F_Croupier.Hide;

  F_Juego.Show;

  F_Juego.pantallaIniJuegoCrupier();

end;


{-------------------------------------------------------------------------------}


procedure TF_Croupier.Listadograljugadores1Click(Sender: TObject);

var

  cant:tCantidad;

begin

  Panel_CrearFicticios.Hide;

  cant:=0;

//Listado General de Usuarios (jugadores). Debe incluir nombre, Nick, foto, clave,
estado,  etc. de cada uno (línea por línea).

  Form_Listados.opListado:= 1;


  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);


  if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cant>1)) then //si cant es >1
entonces al menos hay un jugador que no es el administrador

    Form_Listados.F_Listados.Show

  else

    MessageDlg('No existen jugadores!!',mtWarning , [mbOK], 0);

end;


{-------------------------------------------------------------------------------}


procedure TF_Croupier.Listagraldepartidas1Click(Sender: TObject);

var

  cant:tCantidad;

begin

  Panel_CrearFicticios.Hide;

  cant:=0;
```

```
//-   Listado general de las partidas
  Form_Listados.opListado:= 6;

  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);

  if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cant>1)) then //si cant es >1
entonces al menos hay un jugador que no es el administrador
  begin
    if not(ME_JUEGO.MEVacio(ME_JUEGORULETA)) then
      Form_Listados.F_Listados.Show
    else
      MessageDlg('No se ha efectuado ninguna jugada aún!!',mtWarning , [mbOK], 0);
  end
  else
    MessageDlg('No existen jugadores!!',mtWarning , [mbOK], 0);
end;

{-------------------------------------------------------------------------------}

procedure TF_Croupier.Mostrararbolganadores1Click(Sender: TObject);
begin
//Voy a mostrar arbol de ganadores indicando el nick y nivel del mismo
  Panel_CrearFicticios.Hide;
  Form_Listados.opListado:= 8;

  if (not ME_GANADORES.MeVacio_Ganadores(ME_GANADOR)) then //si cant es >1 entonces al
menos hay un jugador que no es el administrador
    Form_Listados.F_Listados.Show
  else
    MessageDlg('No existen ganadores!!',mtWarning , [mbOK], 0);
end;

{-------------------------------------------------------------------------------}

procedure TF_Croupier.Mostrararboljugadores1Click(Sender: TObject);
var
```

```
  cant:tCantidad;
begin
  Panel_CrearFicticios.Hide;
  cant:=0;
//Voy a mostrar de jugadores indicando el nick y nivel del mismo
  Form_Listados.opListado:= 7;

  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);

  if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cant>1)) then //si cant es >1
entonces al menos hay un jugador que no es el administrador
    Form_Listados.F_Listados.Show
  else
    MessageDlg('No existen jugadores!!',mtWarning , [mbOK], 0);
end;
END.
```

3.2.3 Formulario Jugadores

```
unit Form_Jugadores;

interface

uses
  ME_JUGADORES,
  Type_JUGADOR,
  ME_CTACTE,
  Type_ALMACEN,
  ME_JUEGO,
  Type_JUEGO,
  ME_APUESTAS,
  Type_APUESTA,
  ME_GANADORES,
  Type_GANADOR,
```

```
  Lib_Auxiliar,
  Lib_AuxJuego,
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Grids, Vcl.Buttons,
  Vcl.ExtCtrls, Vcl.Menus, Vcl.Imaging.jpeg;


type
  TF_Jugadores = class(TForm)
    Menu_Jugadores: TMainMenu;
    Entraraljuego1: TMenuItem;
    Comprarcredito1: TMenuItem;
    Jugadoresactivos1: TMenuItem;
    Modificardatos1: TMenuItem;
    Cambiarcontrasea1: TMenuItem;
    SALIR1: TMenuItem;
    Panel1: TPanel;
    Panel_BajaModificacion: TPanel;
    caption_panelGral: TLabel;
    Panel_Botones1: TPanel;
    BtnModificacion: TButton;
    BtnBajas: TButton;
    Panel_Formulario: TPanel;
    Lab_apellido: TLabel;
    Lab_nombre: TLabel;
    Lab_nick: TLabel;
    Txt_Nombre: TEdit;
    Txt_NICK: TEdit;
    BtnOk: TBitBtn;
    BtnCancelar: TBitBtn;
    Txt_Apellido: TEdit;
    Btn_closePnelBM: TBitBtn;
    panel_CambioContrasenia: TPanel;
    btnCambiarContraseña: TBitBtn;
    Bit_ShowHide1: TBitBtn;
```

```
    Bit_ShowHide2: TBitBtn;
    Btn_closePnelAlterContrasenia: TBitBtn;
    panel_Compracredito: TPanel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    btnRecargar: TBitBtn;
    Txt_Saldo: TEdit;
    Btn_closePnelCredito: TBitBtn;
    lbEdit_Recarga: TEdit;
    Label4: TLabel;
    lbEdit_Contraseña: TEdit;
    Label5: TLabel;
    lbEdit_ConfirmContraseñaConf: TEdit;
    Label6: TLabel;
    Panel_EncabezadoUser: TPanel;
    Panel2: TPanel;
    Label7: TLabel;
    Panel15: TPanel;
    Label8: TLabel;
    Lbl_nickUser: TLabel;
    Panel16: TPanel;
    Label9: TLabel;
    Lbl_fechaUser: TLabel;
    Label10: TLabel;
    Lbl_horaUser: TLabel;
    Img_fondoJugadores: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Pantalla_InicioJugadores();
    procedure FormShow(Sender: TObject);
    procedure SALIR1Click(Sender: TObject);
    procedure Jugadoresactivos1Click(Sender: TObject);
    procedure Entraraljuego1Click(Sender: TObject);
    procedure BtnModificacionClick(Sender: TObject);
    procedure BtnOkClick(Sender: TObject);
```

```
    procedure BtnBajasClick(Sender: TObject);
    procedure Modificardatos1Click(Sender: TObject);
    procedure BtnCancelarClick(Sender: TObject);
    procedure Txt_NombreKeyPress(Sender: TObject; var Key: Char);
    procedure Txt_ApellidoKeyPress(Sender: TObject; var Key: Char);
    procedure Cambiarcontrasea1Click(Sender: TObject);
    procedure lbEdit_ContraseñaKeyPress(Sender: TObject; var Key: Char);
    procedure lbEdit_ConfirmContraseñaConfKeyPress(Sender: TObject;
      var Key: Char);
    procedure Bit_ShowHide1Click(Sender: TObject);
    procedure Bit_ShowHide2Click(Sender: TObject);
    procedure btnCambiarContraseñaClick(Sender: TObject);
    procedure Comprarcredito1Click(Sender: TObject);
    procedure Btn_closePnelBMClick(Sender: TObject);
    procedure Btn_closePnelAlterContraseniaClick(Sender: TObject);
    procedure Btn_closePnelCreditoClick(Sender: TObject);
    procedure btnRecargarClick(Sender: TObject);
    procedure lbEdit_RecargaKeyPress(Sender: TObject; var Key: Char);



  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Jugadores: TF_Jugadores;

implementation

uses
  Form_login, Form_juego, Form_Listados;
```

```
VAR
  Opcion: char;

{$R *.dfm}



{-------------------------------------------------------------------------------}

procedure TF_Jugadores.FormCreate(Sender: TObject);
begin
  self.Position := poScreenCenter;
  self.WindowState := wsMaximized;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.FormShow(Sender: TObject);
begin
  F_Jugadores.Pantalla_InicioJugadores();
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.Entraraljuego1Click(Sender: TObject);
begin
  ME_JUEGO.RegistroNulo(regJuegoRuleta);
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, regJuegoRuleta);

  //busco y capturo jugador para verificar si esta bloqueado y a su vez para activarlo
si ingresa al juego
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR, posJugador, RegJugador.Nick);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  //Solo podra entrar a la partida si el estado es igual 0 (Juego creado)
```

```
  if (regJuegoRuleta.Estado = 0) then
  begin
    //Podra entrar al juego sino esta bloquado
    if not(regJugador.Bloqueado) then
    begin
      F_Jugadores.Hide;
      //---------------
      regJugador.Nick:= regJugador.Nick;
      regJugador.Contrasenia:= regJugador.Contrasenia;
      regJugador.Nombre:= regJugador.Nombre;
      regJugador.Apellido:= regJugador.Apellido;
      regJugador.Alta:= regJugador.Alta;
      regJugador.UltimaConexion:= Now();
      regJugador.Bloqueado:= regJugador.Bloqueado;
      regJugador.Estado:= true; //se conecta al juego, lo activo

      ME_JUGADORES.ModificarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);
      //---------------
      F_Juego.Show;
      F_Juego.refrezcarPantallaNormalUser();
    end
    else
      MessageDlg('USTED ESTA BLOQUEADO: No puede ingresar a ningun juego hasta que sea
desbloqueado!',mtWarning , [mbOK], 0);
  end
  else
      MessageDlg('No puede ingresar aún. El juego no ha sido creado o está en
curso!',mtWarning , [mbOK], 0);
end;

{-------------------------------------------------------------------------------}



procedure TF_Jugadores.Jugadoresactivos1Click(Sender: TObject);
var
  Cant: tCantidad; // cantidad jugadores en general
```

```
  Cant_Activos: tCantidad; // cantidad de jugadores activos
begin
  Form_Listados.opListado:= 9;

  cant:= 0;
  Cant_Activos:= 0;

  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);
  Lib_AuxJuego.Cant_JugadoresActivos(ME_JUGADORES.Raiz(ME_JUGADOR), Cant_Activos);

   if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cant>1)) then //si cant es
>1 entonces al menos hay un jugador que no es el administrador
   begin
        //Si hay jugadores activos
      if (Cant_Activos>=1) then
        F_Listados.Show
      else
        MessageDlg('No hay jugadores activos!',mtWarning, [mbOK], 0);
    end
    else
      MessageDlg('No existen jugadores!!',mtWarning, [mbOK], 0);
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.lbEdit_ConfirmContraseñaConfKeyPress(Sender: TObject;
  var Key: Char);
const
  CARAC_HABILITADOS = ['a'..'z', 'A'..'Z', '0'..'9', #0..#27];//solo letras, numero y
borrar
begin
  if not (Key IN CARAC_HABILITADOS) then
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
end;

{-------------------------------------------------------------------------------}
```

```
procedure TF_Jugadores.lbEdit_RecargaKeyPress(Sender: TObject; var Key: Char);
const
  CARAC_HABILITADOS = ['0'..'9', #0..#27];//solo numero y tecla borrar
begin
  if not (Key IN CARAC_HABILITADOS) then
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.Modificardatos1Click(Sender: TObject);
var
  i: integer;
begin
  //Cuando usuario va a modificar sus datos, como lo que se muestra es un panel y no un
formulario, debo dehabilitar el menu principal
  for i:=0 to menu_Jugadores.Items.Count-1 do
    menu_Jugadores.Items[I].Enabled:= false;

  panel_Compracredito.Hide;
  panel_CambioContrasenia.Hide;
  Panel_BajaModificacion.Show;
  Panel_Formulario.Enabled:= false;
  BtnModificacion.Enabled:= true;
  BtnBajas.Enabled:= true;

  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,PosJugador,F_Jugadores.Txt_NICK.Text);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,regJugador,PosJugador);

  F_Jugadores.Txt_Nombre.Text:= regJugador.Nombre;
  F_Jugadores.Txt_Apellido.Text:= regJugador.Apellido;
end;
```

```
{-------------------------------------------------------------------------------}

procedure TF_Jugadores.Bit_ShowHide1Click(Sender: TObject);
begin
  if (lbEdit_Contraseña.PasswordChar = #0) then
  begin
    lbEdit_Contraseña.PasswordChar:='*';
    Bit_ShowHide1.Glyph.LoadFromFile(Lib_Auxiliar.rutaIconoOjoAbierto);
  end
  else
  begin
    lbEdit_Contraseña.PasswordChar:= #0;
    Bit_ShowHide1.Glyph.LoadFromFile(Lib_Auxiliar.rutaIconoOjoCerrado);
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.Bit_ShowHide2Click(Sender: TObject);
begin
  if (lbEdit_ConfirmContraseñaConf.PasswordChar = #0) then
  begin
    lbEdit_ConfirmContraseñaConf.PasswordChar:='*';
    Bit_ShowHide2.Glyph.LoadFromFile(Lib_Auxiliar.rutaIconoOjoAbierto);
  end
  else
  begin
    lbEdit_ConfirmContraseñaConf.PasswordChar:= #0;
    Bit_ShowHide2.Glyph.LoadFromFile(Lib_Auxiliar.rutaIconoOjoCerrado);
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.BtnBajasClick(Sender: TObject);
```

```
begin
    if Lib_Auxiliar.Mensaje_Confirmacion('MENSAJE DE CONFIRMACIÓN: ' + #13 + #13 +
'¿Seguro que quiere darse de baja?') then
    begin
      //Cargo datos de ctacte
      RegCtaCte.Nick:= F_Jugadores.Txt_Nick.Text;
      RegCtaCte.FechaHora:= Now();
      RegCtaCte.Concepto:= Type_ALMACEN.tipoConceptos.Baja_cuenta;
      RegCtaCte.Debe:= 0;
      RegCtaCte.haber:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE, RegCtaCte.Nick);
      RegCtaCte.saldo:= 0;

      //inserto cuenta corriente
      ME_CTACTE.InsertarAlmacen(ME_CUENTACORRIENTE,RegCtaCte);
      ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,
F_Jugadores.Txt_NICK.Text);
      ME_JUGADORES.EliminarInfoME_Jugadores(ME_JUGADOR, posJugador);
      Showmessage('HA SIDO DADA DE BAJA: ' + #13 + #13 + 'Será redirigido a la página
principal');

      F_Jugadores.Close;
      F_login.Show;
    end
    else
      Pantalla_InicioJugadores();
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.btnCambiarContraseñaClick(Sender: TObject);
begin
//usuario va a cambiar su contrasenia
 if ((lbEdit_Contraseña.Text <> '') and (lbEdit_ConfirmContraseñaConf.Text <> '')) then
 begin
   if (lbEdit_Contraseña.Text = lbEdit_ConfirmContraseñaConf.Text) then
   begin
```

```
      if Lib_Auxiliar.Mensaje_Confirmacion('MENSAJE DE CONFIRMACIÓN: ' + #13 + #13 +
'¿Seguro que cambiar su contraseña?') then

      begin

        ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,
F_Jugadores.Txt_NICK.Text);

        ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

        regJugador.Contrasenia:= lbEdit_Contraseña.Text;

        ME_JUGADORES.ModificarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

        Pantalla_InicioJugadores();

        messagedlg('MODICICACIÓN CON ÉXITO: ' + #13 + #13 + 'Su contraseña ha sido
modificada con éxito!', mtInformation, [mbOk], 0);

      end;

   end

   else

    begin

      MessageDlg('Las contraseñas deben ser iguales!',mtError , [mbOK], 0);

      lbEdit_Contraseña.Clear;

      lbEdit_ConfirmContraseñaConf.Clear;

      lbEdit_Contraseña.SetFocus;

    end;

 end

 else

  begin

    MessageDlg('Debe completar los dos campos!!',mtError , [mbOK], 0);

    if (lbEdit_Contraseña.Text = '') then

      lbEdit_Contraseña.SetFocus

    else

      lbEdit_ConfirmContraseñaConf.SetFocus

  end;

end;


{-------------------------------------------------------------------------------}


procedure TF_Jugadores.BtnCancelarClick(Sender: TObject);

begin

  BtnModificacion.Enabled:=true;
```

```
  BtnBajas.Enabled:=true;
  Panel_Formulario.Enabled:= false;

  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,PosJugador,F_Jugadores.Txt_NICK.Text);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,regJugador,PosJugador);

  F_Jugadores.Txt_Nombre.Text:= regJugador.Nombre;
  F_Jugadores.Txt_Apellido.Text:= regJugador.Apellido;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.BtnModificacionClick(Sender: TObject);
begin
  Panel_Formulario.Enabled:= true;
  BtnBajas.Enabled:= false;
  F_Jugadores.Txt_Nombre.SetFocus;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.BtnOkClick(Sender: TObject);
begin
//usuario comun modifica sus datos
  if ((F_Jugadores.Txt_Nombre.Text <> '') and (F_Jugadores.Txt_Apellido.Text <> ''))
then
  begin
      if Lib_Auxiliar.Mensaje_Confirmacion('MENSAJE DE CONFIRMACIÓN: ' + #13 + #13 +
'¿Seguro que quiere modificar sus datos?') then
      begin
        ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,
F_Jugadores.Txt_NICK.Text);
        ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);
        regJugador.Nick:= regJugador.Nick;
        regJugador.Nombre:= F_Jugadores.Txt_Nombre.Text;
        regJugador.Apellido:= F_Jugadores.Txt_Apellido.Text;
```

```
        regJugador.TipoJugador:= regJugador.TipoJugador;
        regJugador.Estado:= regJugador.Estado;
        ME_JUGADORES.ModificarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);
        Pantalla_InicioJugadores();
        messagedlg('MODICICACIÓN CON ÉXITO: ' + #13 + #13 + 'Sus datos han sido
modificados con éxito!', mtInformation, [mbOk], 0);
      end
      else
        Pantalla_InicioJugadores();
  end
  else
    begin
      MessageDlg('Debe completar los dos campos!!',mtError , [mbOK], 0);
      if (F_Jugadores.Txt_Nombre.Text = '') then
        F_Jugadores.Txt_Nombre.SetFocus
      else
        F_Jugadores.Txt_Apellido.SetFocus
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.btnRecargarClick(Sender: TObject);
begin
if (lbEdit_Recarga.Text<>'') then
    begin
      if (strtoint(lbEdit_Recarga.Text) >= ME_JUEGO.ApuestaMinima(ME_JUEGORULETA)) and
(strtoint(lbEdit_Recarga.Text) <= ME_JUEGO.ApuestaMaxima(ME_JUEGORULETA)) then
      begin
        if Lib_Auxiliar.Mensaje_Confirmacion('¿Está seguro que quiere comprar $'+
lbEdit_Recarga.Text +' en créditos?') then
        begin
          regCtaCte.Nick:= RegJugador.Nick;
          regCtaCte.FechaHora:= Now();
          regCtaCte.Concepto:= Type_ALMACEN.tipoConceptos.Compra_Fichas;
          regCtaCte.Debe:= strToint(lbEdit_Recarga.Text);
```

```
          regCtaCte.haber:= 0;
          regCtaCte.saldo:= regCtaCte.Debe +
ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE, RegJugador.Nick);
          ME_CTACTE.InsertarAlmacen(ME_CUENTACORRIENTE, RegCtaCte);
          lbEdit_Recarga.Clear;
          Pantalla_InicioJugadores();
          messagedlg('Compra relizada con éxito!', mtInformation, [mbOk], 0);
        end;
        lbEdit_Recarga.Clear;
        Pantalla_InicioJugadores();
      end
      else
        begin
          MessageDlg('El crédito minimo debe ser de $ '
+intTostr(ME_JUEGO.ApuestaMinima(ME_JUEGORULETA))+
                ', y el crédito máximo debe ser de $ '
+intTostr(ME_JUEGO.ApuestaMaxima(ME_JUEGORULETA)),mtWarning, [mbOK], 0);
          lbEdit_Recarga.Clear;
          lbEdit_Recarga.SetFocus;
        end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.Btn_closePnelAlterContraseniaClick(Sender: TObject);
var
 i: integer;
begin
  panel_CambioContrasenia.Hide;

  for i:=0 to menu_Jugadores.Items.Count-1 do
    menu_Jugadores.Items[I].Enabled:= true;
end;

{-------------------------------------------------------------------------------}
```

```
procedure TF_Jugadores.Btn_closePnelBMClick(Sender: TObject);
var
 i:integer;
begin
  Panel_BajaModificacion.Hide;
  for i:=0 to menu_Jugadores.Items.Count-1 do
    menu_Jugadores.Items[I].Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.Btn_closePnelCreditoClick(Sender: TObject);
var
 i: integer;
begin
  panel_Compracredito.Hide;
  lbEdit_Recarga.Clear;

  for i:=0 to menu_Jugadores.Items.Count-1 do
    menu_Jugadores.Items[I].Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.Cambiarcontrasea1Click(Sender: TObject);
var
 i: integer;
begin
  panel_Compracredito.Hide;
  Panel_BajaModificacion.Hide;
  panel_CambioContrasenia.Show;

  for i:=0 to menu_Jugadores.Items.Count-1 do
    menu_Jugadores.Items[I].Enabled:= false;
```

```
  lbEdit_Contraseña.Clear;
  lbEdit_ConfirmContraseñaConf.Clear;
  lbEdit_Contraseña.SetFocus;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.Comprarcredito1Click(Sender: TObject);
var
 i: integer;
begin
  panel_CambioContrasenia.Hide;
  Panel_BajaModificacion.Hide;
  panel_Compracredito.Show;

  for i:=0 to menu_Jugadores.Items.Count-1 do
    menu_Jugadores.Items[I].Enabled:= false;

  F_Jugadores.Txt_Saldo.Text:=intTostr(ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,
RegJugador.Nick));
  lbEdit_Recarga.SetFocus;
end;

{-------------------------------------------------------------------------------}



procedure TF_Jugadores.Pantalla_InicioJugadores();
var
 i: integer;


begin
  Lbl_nickUser.Caption:= NickLogueado;
  F_Juego.ActualizarFechaHora(Lbl_fechaUser, Lbl_horaUser); //Actualizo fecha y hora a
mostrar
```

```
  F_Jugadores.panel_Compracredito.Hide;
  F_Jugadores.panel_CambioContrasenia.Hide;
  F_Jugadores.Panel_BajaModificacion.Hide;

  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,PosJugador,RegJugador.Nick);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,regJugador,PosJugador);

  F_Jugadores.Txt_NICK.Text:= RegJugador.Nick;
  F_Jugadores.Txt_Nombre.Text:= regJugador.Nombre;
  F_Jugadores.Txt_Apellido.Text:= regJugador.Apellido;

  for i:=0 to menu_Jugadores.Items.Count-1 do
    menu_Jugadores.Items[I].Enabled:= true;

end;

{-------------------------------------------------------------------------------}
procedure TF_Jugadores.SALIR1Click(Sender: TObject);
begin
  if (Lib_Auxiliar.Mensaje_Confirmacion('¿Esta seguro que quiere salir?')) then
  begin
    Lib_AuxJuego.Fin_Sesion(ME_JUGADOR, regJugador);
    F_Jugadores.Close;
    F_Login.Show;
    F_Login.Limpiar_Formulario();
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Jugadores.Txt_ApellidoKeyPress(Sender: TObject; var Key: Char);
const
  CARAC_HABILITADOS = ['a'..'z', 'A'..'Z', #0..#27];//solo letras y tecla borrar
begin
  if not (Key IN CARAC_HABILITADOS) then
```

```
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
end;

procedure TF_Jugadores.Txt_NombreKeyPress(Sender: TObject; var Key: Char);
const
  CARAC_HABILITADOS = ['a'..'z', 'A'..'Z', #0..#27];//solo letras y tecla borrar
begin
  if not (Key IN CARAC_HABILITADOS) then
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
end;

procedure TF_Jugadores.lbEdit_ContraseñaKeyPress(Sender: TObject;
  var Key: Char);
const
  CARAC_HABILITADOS = ['a'..'z', 'A'..'Z', '0'..'9', #0..#27];//solo letras, numero y
borrar
begin
  if not (Key IN CARAC_HABILITADOS) then
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
end;
END.
```

3.2.4 Formulario Juego

```
unit Form_Juego;

interface

uses
  ME_JUGADORES,
  Type_JUGADOR,
  ME_CTACTE,
  Type_ALMACEN,
  ME_JUEGO,
  Type_JUEGO,
```

```
  ME_APUESTAS,
  Type_APUESTA,
  ME_GANADORES,
  Type_GANADOR,
  Lib_Auxiliar,
  Lib_AuxJuego,
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls, Vcl.Buttons,
  Vcl.Imaging.pngimage, Vcl.ToolWin, Vcl.ComCtrls, Vcl.Imaging.jpeg, Vcl.Menus,
  Vcl.Imaging.GIFImg, Vcl.Grids;


type
  TF_Juego = class(TForm)
    Splitter1: TSplitter;
    Panel_JuegoGral: TPanel;
    Panel_EncabezadoJuego: TPanel;
    Panel2: TPanel;
    Label1: TLabel;
    Lbl_nroPartida: TLabel;
    Panel15: TPanel;
    Label4: TLabel;
    Lbl_usuarioActual: TLabel;
    Panel16: TPanel;
    Label2: TLabel;
    Lbl_fechaActual: TLabel;
    Label3: TLabel;
    Lbl_horaActual: TLabel;
    Panel_Crupier: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    Panel6: TPanel;
    Panel8: TPanel;
    Panel9: TPanel;
    Panel10: TPanel;
```

```
    Panel7: TPanel;
    Panel11: TPanel;
    Lbl_bolillaASalir: TLabel;
    Lbl_nroBolillaG: TLabel;
    Panel12: TPanel;
    Panel_NameTablero: TPanel;
    Label9: TLabel;
    menuFormJuego: TMainMenu;
    SALIR1: TMenuItem;
    Panel_Jugadores: TPanel;
    Panel18: TPanel;
    Label11: TLabel;
    Lbl_estadoJuego: TLabel;
    Label10: TLabel;
    Panel19: TPanel;
    Label12: TLabel;
    Lbl_saldoAcumulado: TLabel;
    Label14: TLabel;
    Panel20: TPanel;
    Label13: TLabel;
    Label15: TLabel;
    Lbl_ApuestaMax: TLabel;
    Panel21: TPanel;
    Label16: TLabel;
    Label17: TLabel;
    Lbl_ApuestaMin: TLabel;
    Panel23: TPanel;
    Label19: TLabel;
    Spbtn_haganApuestas: TSpeedButton;
    Spbtn_noVaMas: TSpeedButton;
    Spbtn_tirarBolilla: TSpeedButton;
    Spbtn_obsequiarCred: TSpeedButton;
    Spbtn_activarFicticios: TSpeedButton;
    SpeedButton12: TSpeedButton;
    Spbtn_CrearPartida: TSpeedButton;
```

```
    Lbl_totalFicticios: TLabel;
    Lbl_cantFicticios: TLabel;
    Listarapuestas1: TMenuItem;
    Premiados1: TMenuItem;
    Spbtn_repartirPremios: TBitBtn;
    Spbtn_trampa: TSpeedButton;
    Panel13: TPanel;
    Spbtn_refresh: TSpeedButton;
    Spbtn_bloqDesbloq: TSpeedButton;
    Panel14: TPanel;
    Label5: TLabel;
    Label6: TLabel;
    Lbl_CantEnLinea: TLabel;
    Lbl_CantFictActivados: TLabel;
    panel_tablero: TPanel;
    tabla_Image0: TImage;
    tabla_Image6: TImage;
    tabla_Image15: TImage;
    tabla_Image21: TImage;
    tabla_Image12: TImage;
    tabla_Image18: TImage;
    tabla_Image9: TImage;
    tabla_Image24: TImage;
    tabla_Image27: TImage;
    tabla_Image30: TImage;
    tabla_Image33: TImage;
    tabla_Image36: TImage;
    tabla_ImageTerceraCol: TImage;
    tabla_Image2: TImage;
    tabla_Image5: TImage;
    tabla_Image8: TImage;
    tabla_Image11: TImage;
    tabla_Image14: TImage;
    tabla_Image17: TImage;
    tabla_Image20: TImage;
```

```
    tabla_Image23: TImage;
    tabla_Image26: TImage;
    tabla_Image29: TImage;
    tabla_Image32: TImage;
    tabla_Image35: TImage;
    tabla_Image4: TImage;
    tabla_Image7: TImage;
    tabla_Image10: TImage;
    tabla_Image13: TImage;
    tabla_Image16: TImage;
    tabla_Image19: TImage;
    tabla_Image22: TImage;
    tabla_Image25: TImage;
    tabla_Image28: TImage;
    tabla_Image31: TImage;
    tabla_Image34: TImage;
    tabla_ImageSegundaCol: TImage;
    tabla_ImagePrimeraCol: TImage;
    tabla_ImagePrimeros12: TImage;
    tabla_ImageSegundos12: TImage;
    tabla_ImageTerceros12: TImage;
    tabla_ImageRojo: TImage;
    tabla_ImageNegro: TImage;
    tabla_ImageImpares: TImage;
    tabla_Image_19a36: TImage;
    tabla_Image3: TImage;
    tabla_Image1: TImage;
    tabla_Image_1a18: TImage;
    tabla_ImagePares: TImage;
    Panel_Winner: TPanel;
    Timer_showWinner1: TTimer;
    Timer_showWinner2: TTimer;
    Timer_showWinner3: TTimer;
    Timer_showWinner4: TTimer;
    Timer_showLoser1: TTimer;
```

```
    Timer_showLoser2: TTimer;
    Timer_showLoser3: TTimer;
    Timer_showLoser4: TTimer;
    Timer_showWinner5: TTimer;
    Timer_showWinner6: TTimer;
    Timer_showLoser5: TTimer;
    Timer_showLoser6: TTimer;
    Panel_ShowWinner: TPanel;
    Label7: TLabel;
    Label18: TLabel;
    Lbl_montoWinner: TLabel;
    Label20: TLabel;
    Lbl_bolillaGWinner: TLabel;
    gifWinner: TImage;
    Btn_closePnelWinner: TBitBtn;
    Lbl_waiting: TLabel;
    Panel_Loser: TPanel;
    Label8: TLabel;
    Lbl_bolillaGLoser: TLabel;
    Label22: TLabel;
    Btn_closePnelLoser: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure AutoSize_ImagenesTablero();
    procedure pantallaIniJuegoCrupier();
    procedure pantallaIniJuegoJugadores();
    procedure ListarPremiados(Nro_Jugada: Type_JUEGO.tClave);
    procedure ListandoApuestas();
    procedure Spbtn_bloqDesbloqClick(Sender: TObject);
    procedure Spbtn_obsequiarCredClick(Sender: TObject);
    procedure SALIR1Click(Sender: TObject);
    procedure
TableroLegalSeleccionado(nomenclador:Type_APUESTA.tNomenclador;valor:Type_APUESTA.tNome
nclador; nameOp:string);
    procedure TableroTrampaSeleccionado(num_bolilla:Type_JUEGO.tBolilla);
    procedure refrezcarPantallaNormalUser();
```

```
    procedure ActualizarFechaHora(var Lbl_fecha:TLabel; var Lbl_hora: TLabel);
    procedure tabla_Image3Click(Sender: TObject);
    procedure Spbtn_cancelarSelectTabClick(Sender: TObject);
    procedure tabla_Image1Click(Sender: TObject);
    procedure tabla_Image2Click(Sender: TObject);
    procedure tabla_Image4Click(Sender: TObject);
    procedure tabla_Image5Click(Sender: TObject);
    procedure tabla_Image6Click(Sender: TObject);
    procedure tabla_Image7Click(Sender: TObject);
    procedure tabla_Image8Click(Sender: TObject);
    procedure tabla_Image9Click(Sender: TObject);
    procedure tabla_Image10Click(Sender: TObject);
    procedure tabla_Image11Click(Sender: TObject);
    procedure tabla_Image12Click(Sender: TObject);
    procedure tabla_Image13Click(Sender: TObject);
    procedure tabla_Image14Click(Sender: TObject);
    procedure tabla_Image15Click(Sender: TObject);
    procedure tabla_Image16Click(Sender: TObject);
    procedure tabla_Image17Click(Sender: TObject);
    procedure tabla_Image18Click(Sender: TObject);
    procedure tabla_Image19Click(Sender: TObject);
    procedure tabla_Image20Click(Sender: TObject);
    procedure tabla_Image21Click(Sender: TObject);
    procedure tabla_Image22Click(Sender: TObject);
    procedure tabla_Image23Click(Sender: TObject);
    procedure tabla_Image24Click(Sender: TObject);
    procedure tabla_Image25Click(Sender: TObject);
    procedure tabla_Image26Click(Sender: TObject);
    procedure tabla_Image27Click(Sender: TObject);
    procedure tabla_Image28Click(Sender: TObject);
    procedure tabla_Image29Click(Sender: TObject);
    procedure tabla_Image30Click(Sender: TObject);
    procedure tabla_Image31Click(Sender: TObject);
    procedure tabla_Image32Click(Sender: TObject);
    procedure tabla_Image33Click(Sender: TObject);
```

```
    procedure tabla_Image34Click(Sender: TObject);
    procedure tabla_Image35Click(Sender: TObject);
    procedure tabla_Image36Click(Sender: TObject);
    procedure tabla_Image0Click(Sender: TObject);
    procedure tabla_ImageTerceraColClick(Sender: TObject);
    procedure tabla_ImageSegundaColClick(Sender: TObject);
    procedure tabla_ImagePrimeraColClick(Sender: TObject);
    procedure tabla_ImagePrimeros12Click(Sender: TObject);
    procedure tabla_ImageSegundos12Click(Sender: TObject);
    procedure tabla_ImageTerceros12Click(Sender: TObject);
    procedure tabla_Image_1a18Click(Sender: TObject);
    procedure tabla_ImageParesClick(Sender: TObject);
    procedure tabla_ImageImparesClick(Sender: TObject);
    procedure tabla_Image_19a36Click(Sender: TObject);
    procedure tabla_ImageRojoClick(Sender: TObject);
    procedure tabla_ImageNegroClick(Sender: TObject);
    procedure Spbtn_activarFicticiosClick(Sender: TObject);
    procedure Spbtn_CrearPartidaClick(Sender: TObject);
    procedure SpeedButton12Click(Sender: TObject);
    procedure Spbtn_haganApuestasClick(Sender: TObject);
    procedure Spbtn_noVaMasClick(Sender: TObject);
    procedure Spbtn_tirarBolillaClick(Sender: TObject);
    procedure Spbtn_trampaClick(Sender: TObject);
    procedure Spbtn_repartirPremiosClick(Sender: TObject);
    procedure ActiveBtnTrampa();
    procedure DesactiveBtnTrampa();
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Listarapuestas1Click(Sender: TObject);
    //procedure SoloMostrarApuestas();
    procedure Premiados1Click(Sender: TObject);
    procedure SalirDelJuego();
    procedure Spbtn_refreshClick(Sender: TObject);
    procedure Timer_showWinner1Timer(Sender: TObject);
    procedure Timer_showWinner2Timer(Sender: TObject);
    procedure Timer_showWinner3Timer(Sender: TObject);
```

```
    procedure Timer_showWinner4Timer(Sender: TObject);
    procedure Timer_showLoser1Timer(Sender: TObject);
    procedure Timer_showLoser2Timer(Sender: TObject);
    procedure Timer_showLoser3Timer(Sender: TObject);
    procedure Timer_showLoser4Timer(Sender: TObject);
    procedure Btn_closePnelWinnerClick(Sender: TObject);
    procedure Btn_closePnelLoserClick(Sender: TObject);
    procedure Timer_showWinner5Timer(Sender: TObject);
    procedure Timer_showWinner6Timer(Sender: TObject);
    procedure Timer_showLoser5Timer(Sender: TObject);
    procedure Timer_showLoser6Timer(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Juego: TF_Juego;
  Nomenclador: Type_APUESTA.tNomenclador;
  totalFicticios: Lib_Auxiliar.tCantidad;
  apuestaMax, apuestaMin: Type_JUEGO.tImporte;
  btnActiveNow:string;
  BtnTrampaIsActive: boolean;
  nameOp:string;
  valor: string;
  EstadoJuego: integer;
  SoloListar: boolean;
  ShowWinner: boolean;


implementation

uses
```

```
  Form_login, Form_Jugadores, Form_Croupier, Form_PanelCaseButtons, Form_Apostar,
Form_PanelFicticios, Form_Premios;


{$R *.dfm}


{-------------------------------------------------------------------------------}


procedure TF_Juego.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  F_PanelFicticios.Close;
  F_PanelCaseButtons.Close;
end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.FormCreate(Sender: TObject);
begin
  apuestaMax:= ME_JUEGO.ApuestaMaxima(ME_JUEGORULETA);
  apuestaMin:= ME_JUEGO.ApuestaMinima(ME_JUEGORULETA);


  Spbtn_bloqDesbloq.Caption:= '[ BLOQUEAR/' +#13+ 'DESBLOQUEAR ]';


  //generalidades de formulario
  self.Position := poScreenCenter;
  self.WindowState := wsMaximized;
end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.FormShow(Sender: TObject);
var
  i:integer;
begin
  Panel_Winner.Visible:= false;
  Panel_Loser.Visible:= false;
```

```
  Panel_ShowWinner.Visible:= false;
  //Si es croupier
  if (Form_login.AdminLog) then
    pantallaIniJuegoCrupier()
  else // si es jugador comun
    pantallaIniJuegoJugadores();
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Listarapuestas1Click(Sender: TObject);
  var
  exist: boolean;
  Nro_Jugada: Type_JUEGO.tClave;
begin
  SoloListar:= true;
  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar
  Nro_Jugada:= Lbl_nroPartida.Caption;

  if not (ME_APUESTAS.MEVacio_Apuestas(ME_APUESTA)) then
  begin
    exist:= Lib_AuxJuego.ExistenApuestas_enEsteJuego(Nro_Jugada);

    if (exist) then
    begin
      F_Apostar.Show;
      F_Apostar.Lbl_ListaDeApuestas.Hide;
      F_Apostar.Lbl_SelectJugador.hide;
      F_Apostar.Lbl_ListaJugadores.Show;

      F_Apostar.Grid_ListaJugadores.Hide;
      F_Apostar.Panel_DatosApostar.hide;

      F_Apostar.Grid_ListaApuestas.Show;
```

```
      Lib_Auxiliar.limpiarGrid(F_Apostar.Grid_ListaApuestas);
      Lib_Auxiliar.EncabezadosComunApuestas(F_Apostar.Grid_ListaApuestas);
      F_Apostar.Grid_ListaApuestas.RowCount:=1;
      Lib_AuxJuego.Listado_GeneralApuestas(F_Apostar.Grid_ListaApuestas, Nro_Jugada);
      F_Apostar.Grid_ListaApuestas.FixedRows:=1;
    end
    else
      MessageDlg('No hay apuestas efectuadas en este juego!!',mtWarning, [mbOK], 0);
  end
  else
    MessageDlg('No hay apuestas efectuadas!!',mtWarning, [mbOK], 0);
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Premiados1Click(Sender: TObject);
var
  cantGanadores: tCantidad;
  Nro_Jugada: Type_JUEGO.tClave;
begin
  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar
  Nro_Jugada:= Lbl_nroPartida.Caption;
  cantGanadores:= 0;

  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, regJuegoRuleta);

  if (regJuegoRuleta.NroJugada = Nro_Jugada) then
  begin //Soy el croupier
      if (regJuegoRuleta.Estado = 4) then // (4) premios repartidos y fin de juego
      begin
        if not (ME_GANADORES.MeVacio_Ganadores(ME_GANADOR)) then
        begin
          Lib_AuxJuego.CantGanadores_EnUnaJugada(ME_GANADORES.Raiz(ME_GANADOR),
Nro_Jugada, cantGanadores);
          if (cantGanadores>0) then
```

```
            listarPremiados(Nro_Jugada)
          else
            MessageDlg('NO HAY GANADORES!' + #13 + #13 +' No hay ganadores en la jugada
Nro: ' +RegJuegoRuleta.NroJugada,mtInformation, [mbOK], 0);
        end
        else
          MessageDlg('NO HAY GANADORES!' + #13 + #13 +' No hay ganadores en la jugada
Nro: ' +RegJuegoRuleta.NroJugada,mtInformation, [mbOK], 0);
      end
      else
        MessageDlg('EL JUEGO ESTA EN CURSO!' + #13 + #13 +' Una vez finalizado la
partida podra listar los ganadores.',mtInformation, [mbOK], 0);
  end
  else //Soy jugador comun. El juego habia finalizado
    begin
        if not (ME_GANADORES.MeVacio_Ganadores(ME_GANADOR)) then
        begin
          Lib_AuxJuego.CantGanadores_EnUnaJugada(ME_GANADORES.Raiz(ME_GANADOR),
Nro_Jugada, cantGanadores);
          if (cantGanadores>0) then
            listarPremiados(Nro_Jugada)
          else
            MessageDlg('NO HAY GANADORES!' + #13 + #13 +' No hay ganadores en la jugada
Nro: ' +Nro_Jugada,mtInformation, [mbOK], 0);
        end
        else
          MessageDlg('NO HAY GANADORES!' + #13 + #13 +' No hay ganadores en la jugada
Nro: ' +Nro_Jugada,mtInformation, [mbOK], 0);
    end;
end;
{-------------------------------------------------------------------------------}


procedure TF_Juego.SALIR1Click(Sender: TObject);
var
  Nro_Jugada: Type_JUEGO.tClave;
begin
  Nro_Jugada:= Lbl_nroPartida.Caption;
```

```
  ME_JUEGO.RegistroNulo(regJuegoRuleta);
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, regJuegoRuleta);


    //Si el numero de jugada de la pantalla es distito a la ultima jugada significa que
el jugador no esta jugando la partida actual. La partida que juga ya culmino. Puede
salir
  if (Nro_Jugada <> regJuegoRuleta.NroJugada) then
      RegJuegoRuleta.estado:= 4;


  if (RegJuegoRuleta.Estado = 4) then
  begin
      Form_Juego.F_Juego.Close;


      if AdminLog then
        Form_Croupier.F_Croupier.Show
      else
        Form_Jugadores.F_Jugadores.Show;
  end
  else
    MessageDlg('No puede salir del juego hasta que el mismo haya finalizado!',mtWarning
, [mbOK], 0)
end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.ActiveBtnTrampa();
begin
  panel_tablero.Enabled:= true;
  tabla_ImagePrimeros12.Hide;
  tabla_ImageSegundos12.Hide;
  tabla_ImageTerceros12.Hide;
  tabla_ImageTerceraCol.Hide;
  tabla_ImageSegundaCol.Hide;
  tabla_ImagePrimeraCol.Hide;
  tabla_Image_1a18.Hide;
```

```
  tabla_ImagePares.Hide;
  tabla_ImageRojo.Hide;
  tabla_ImageNegro.Hide;
  tabla_ImageImpares.Hide;
  tabla_Image_19a36.Hide;
  panel_tablero.Enabled:=true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.DesactiveBtnTrampa();
begin
  panel_tablero.Enabled:= false;
  tabla_ImagePrimeros12.show;
  tabla_ImageSegundos12.show;
  tabla_ImageTerceros12.show;
  tabla_ImageTerceraCol.show;
  tabla_ImageSegundaCol.show;
  tabla_ImagePrimeraCol.show;
  tabla_Image_1a18.show;
  tabla_ImagePares.show;
  tabla_ImageRojo.show;
  tabla_ImageNegro.show;
  tabla_ImageImpares.show;
  tabla_Image_19a36.show;
  panel_tablero.Enabled:=false;
end;


{-------------------------------------------------------------------------------}

procedure TF_Juego.Spbtn_cancelarSelectTabClick(Sender: TObject);
begin
  AutoSize_ImagenesTablero();
```

```
  panel_tablero.Enabled:= true;
  F_Apostar.Edit_nomenclador.Clear;
  F_Apostar.Edit_valor.Clear;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.SpeedButton12Click(Sender: TObject);
var
  Nro_Jugada: Type_JUEGO.tClave;
  regBuscado: Type_JUEGO.TipoRegDatos;
begin
  Nro_Jugada:= Lbl_nroPartida.Caption;
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);

  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR, posJugador,
Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  if (regJugador.Bloqueado) then
  begin
    MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
    SalirDelJuego();
  end
  else
    begin
    //Si el numero de jugada de la pantalla es distito a la ultima jugada significa que
el jugador no esta jugando la partida actual
      if (Nro_Jugada <> RegJuegoRuleta.NroJugada) then
      begin
        //.. me fijo en que estado esta la jugada actual. Si el estado es igual a 0
(patida creada), lo invito a unirse a la nueva partida
        if (RegJuegoRuleta.Estado = 0) then
        begin
          if Lib_Auxiliar.Mensaje_Confirmacion('PARTIDA NRO. '
+RegJuegoRuleta.NroJugada+ ' HA SIDO CREADA:' +#13+#13+'¿Quiere unirse a la nueva
partida?') then
```

```
          begin
            pantallaIniJuegoJugadores;
            refrezcarPantallaNormalUser();
          end;
        end
        else
          MessageDlg('SE ESTA JUGANDO PARTIDA NRO. ' +RegJuegoRuleta.NroJugada+
':'+#13+#13+'Deberá esperar hasta que una nueva partida comience!',mtInformation ,
[mbOK], 0)
      end
      else
        refrezcarPantallaNormalUser();
    end;
end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.Spbtn_CrearPartidaClick(Sender: TObject);
var
  i:integer;
  cantFictActivos, cantEnLinea: tCantidad;
  nroJugada: Type_JUEGO.tClave;
begin
 cantFictActivos:= 0;
 cantEnLinea:= 0;
 ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar
 Lbl_nroBolillaG.Caption:= '--';
 Spbtn_CrearPartida.Enabled:= false;
 Spbtn_haganApuestas.Enabled:= true;
 Spbtn_activarFicticios.Enabled:= true;
 Spbtn_obsequiarCred.Enabled:= true;


 //Muestroa la cantidad de ficticios creados al ingresar al juego buscando el ultimo
ficticio
 totalFicticios:= ME_JUGADORES.UltFicticio(ME_JUGADOR);
```

```
 Lbl_cantFicticios.Caption:= intTostr(totalFicticios);


  if ME_JUEGO.MEVacio(ME_JUEGORULETA) then
  begin //El ME_Juego esta vacio, inserto la primera jugada
    RegJuegoRuleta.NroJugada:=inttostr(1);
    RegJuegoRuleta.FechaHora:= Now();
    RegJuegoRuleta.Estado:= 0;
    RegJuegoRuleta.Bolilla:= Lib_Auxiliar.PosNula;
    ME_JUEGO.Insertar(ME_JUEGORULETA, RegJuegoRuleta);


    Lbl_cantFicticios.Show;
    Lbl_estadoJuego.Caption:='"Juego Creado"';
  end
  else
    begin //El ME_Juego no esta vacio. Solo sera posible si el estado es = 4 (premios
repartidos y fin de juego)
      ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);
      if (RegJuegoRuleta.Estado = 4) then
      begin
        //creo un nuevo RegJuegoRuleta y lo inserto
        RegJuegoRuleta.NroJugada:= intTostr(strToint(RegJuegoRuleta.NroJugada)+1);
        RegJuegoRuleta.FechaHora:= Now();
        RegJuegoRuleta.Estado:= 0;
        RegJuegoRuleta.Bolilla:= Lib_Auxiliar.PosNula;
        ME_JUEGO.Insertar(ME_JUEGORULETA, RegJuegoRuleta);


        Lbl_estadoJuego.Caption:='"Juego Creado"';
      end;
    end;


  //Muestro numero de partida luego de haber iniciado la partida
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);
  Lbl_nroPartida.Caption:= RegJuegoRuleta.NroJugada;


  //Muestro la cantidad de ficticios activas. Al comienzo siempre sera cero, porque el
croupier debera activarlos
```

```
  Lib_AuxJuego.Cant_FicticiosActivos(ME_JUGADORES.Raiz(ME_JUGADOR), cantFictActivos);
  F_Juego.Lbl_CantFictActivados.Caption:= intTostr(cantFictActivos);

  //Muestro la cantidad de jugadores que se unen a la partida y estaran en linea
(activos) una vez creada la partida
  Lib_AuxJuego.Cant_JugadoresEnLinea(ME_JUGADORES.Raiz(ME_JUGADOR), cantEnLinea);
  F_Juego.Lbl_cantEnLinea.Caption:= intTostr(cantEnLinea); //menos el croupier
end;

{-------------------------------------------------------------------------------}


procedure TF_Juego.Spbtn_haganApuestasClick(Sender: TObject);
var
  cantActivos: tCantidad;
begin
  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar
  cantActivos:= 0;

  Spbtn_haganApuestas.Enabled:=false;
  Spbtn_noVaMas.Enabled:=true;
  //habilito tablero para que lleven a cabo sus apuestas tanto crupier como usuario
comun
  panel_tablero.Enabled:= true;

  ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);
      if (RegJuegoRuleta.Estado = 0) then
      begin
        RegJuegoRuleta.NroJugada:= RegJuegoRuleta.NroJugada;
        RegJuegoRuleta.FechaHora:= Now();
        RegJuegoRuleta.Estado:= 1;
        RegJuegoRuleta.Bolilla:= Lib_Auxiliar.PosNula;
        ME_JUEGO.ModificoUltimo(ME_JUEGORULETA, RegJuegoRuleta);

        Lbl_estadoJuego.Caption:='"Hagan sus apuestas"';
```

```
        if (Spbtn_activarFicticios.Enabled = true) then
          Spbtn_activarFicticios.Enabled:=false;

        Spbtn_obsequiarCred.Enabled:=false;

        //Se llevaran a cabo las apuestas de los jugadores ficticios que esten activos
        Lib_AuxJuego.Cant_FicticiosActivos(ME_JUGADORES.Raiz(ME_JUGADOR),cantActivos);
        if (cantActivos > 0) then
          Lib_AuxJuego.Apuestas_JugadoresFicticios(ME_JUGADORES.Raiz(ME_JUGADOR),
cantActivos);
      end;

end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Spbtn_noVaMasClick(Sender: TObject);
begin
  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar
  Spbtn_noVaMas.Enabled:=false;
  Spbtn_bloqDesbloq.Enabled:= false;
  Spbtn_tirarBolilla.Enabled:=true;
  panel_tablero.Enabled:= false;

  ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);
      if (RegJuegoRuleta.Estado = 1) then
      begin
        RegJuegoRuleta.NroJugada:= RegJuegoRuleta.NroJugada;
        RegJuegoRuleta.FechaHora:= Now();
        RegJuegoRuleta.Estado:= 2;
        RegJuegoRuleta.Bolilla:= Lib_Auxiliar.PosNula;
        ME_JUEGO.ModificoUltimo(ME_JUEGORULETA, RegJuegoRuleta);

        Lib_AuxJuego.MovimientoHaberApostadores();
```

```
        Lbl_estadoJuego.Caption:='"No va más"';


        Spbtn_obsequiarCred.Enabled:=false;
      end;
end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.Spbtn_bloqDesbloqClick(Sender: TObject);
var
  Cant: tCantidad; // cantidad jugadores en general
begin
  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar
  Cant:= 0;
  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);


  if (not(ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR)) and (cant>1)) then //si cant es
>1 entonces al menos hay un jugador que no es el administrador
    begin
      F_PanelCaseButtons.MostrarEnGridJugadores();
    end
    else
      begin
        MessageDlg('No existen jugadores!!',mtWarning, [mbOK], 0);
        Form_Juego.F_Juego.Panel_JuegoGral.Enabled:= true;
      end;


  F_PanelCaseButtons.Panel_LockUnlock.Enabled:= true;
  F_PanelCaseButtons.Panel_LockUnlock.show;


  F_PanelCaseButtons.Panel_RegalarCredito.Enabled:= false;
  F_PanelCaseButtons.Panel_RegalarCredito.Hide;
  F_PanelCaseButtons.Lbl_regalarCred.Hide;
end;
```

```
{-------------------------------------------------------------------------------}


procedure TF_Juego.Spbtn_activarFicticiosClick(Sender: TObject);

begin

  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar

  F_PanelFicticios.Show;

end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.Spbtn_obsequiarCredClick(Sender: TObject);

begin

  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar

  F_PanelCaseButtons.Show;

  F_PanelCaseButtons.Panel_LockUnlock.hide;

  F_PanelCaseButtons.Panel_LockUnlock.Enabled:= false;


  F_PanelCaseButtons.Panel_RegalarCredito.show;

  F_PanelCaseButtons.Panel_RegalarCredito.Enabled:= true;

  F_PanelCaseButtons.Lbl_bloquearDesbloquear.hide;

  F_PanelCaseButtons.Lbl_regalarCred.show;

  F_PanelCaseButtons.Edit_cantCredito.Enabled:= false;

end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.Spbtn_refreshClick(Sender: TObject);

var

  cantEnLinea: tCantidad;

begin

  cantEnLinea:= 0;

  //Actualizo fecha y hora a mostrar

  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual);
```

```
  //Al refrezcar vuelvo a mostrar la cantidad de jugadores que se van uniendo a la
partida y estaran en linea (activos) una vez creada la partida
  Lib_AuxJuego.Cant_JugadoresEnLinea(ME_JUGADORES.Raiz(ME_JUGADOR), cantEnLinea);
  F_Juego.Lbl_cantEnLinea.Caption:= intTostr(cantEnLinea); //menos el croupier
end;


procedure TF_Juego.Spbtn_repartirPremiosClick(Sender: TObject);
var
  cantGanadores: tCantidad;
  cantFictActivos: tCantidad;
begin
  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar
  cantFictActivos:= 0;
  cantGanadores:= 0;
  Spbtn_trampa.Enabled:=false;

  ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);

    if (RegJuegoRuleta.Estado = 3) then
    begin
        RegJuegoRuleta.NroJugada:= RegJuegoRuleta.NroJugada;
        RegJuegoRuleta.FechaHora:= Now();
        RegJuegoRuleta.Estado:= 4;
        RegJuegoRuleta.Bolilla:= RegJuegoRuleta.Bolilla;
        ME_JUEGO.ModificoUltimo(ME_JUEGORULETA, RegJuegoRuleta);

        //Reparto premio de los apostadores de la ultima jugada
        ME_JUEGO.UltimaJugada(ME_JUEGORULETA, regJuegoRuleta);
        Lib_AuxJuego.repartirPremio_xJugada(ME_JUGADORES.Raiz(ME_JUGADOR),
regJuegoRuleta.NroJugada);

        //Desactivo jugadores ficticios
        Lib_AuxJuego.Desactivar_JugadoresFicticios(ME_JUGADORES.Raiz(ME_JUGADOR));
```

```
        //Muestro cantidad de ficticios activos una vez culminada la partida. Siempre
sera cero

        Lib_AuxJuego.Cant_FicticiosActivos(ME_JUGADORES.Raiz(ME_JUGADOR),
cantFictActivos);

        F_Juego.Lbl_CantFictActivados.Caption:= intTostr(cantFictActivos);


        //Muestro la cantidad de ganadores si es que los hay para que luego pueda
listarlo en el menu de arriba (Premiados)

        if not (ME_GANADORES.MeVacio_Ganadores(ME_GANADOR)) then

        begin

           Lib_AuxJuego.CantGanadores_EnUnaJugada(ME_GANADORES.Raiz(ME_GANADOR),
RegJuegoRuleta.NroJugada,cantGanadores);

           if (cantGanadores>0) then

            MessageDlg('HAY ' +intTostr(cantGanadores)+ ' GANADORES EN JUGADA NRO '
+RegJuegoRuleta.NroJugada,mtInformation , [mbOK], 0)

           else

            MessageDlg('NO HAY GANADORES EN JUGADA NRO '
+RegJuegoRuleta.NroJugada,mtInformation , [mbOK], 0);

        end

        else

          MessageDlg('NO HAY GANADORES EN JUGADA NRO '
+RegJuegoRuleta.NroJugada,mtInformation , [mbOK], 0);


    end;


    //Invoco al procedimiento que va a mostrar la pantalla de inicio para el croupier
una vez culminada una partida

    pantallaIniJuegoCrupier();

end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.Spbtn_tirarBolillaClick(Sender: TObject);

begin

  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar

  Spbtn_tirarBolilla.Enabled:=false;

  Spbtn_repartirPremios.Enabled:=true;
```

```
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);
    if (RegJuegoRuleta.Estado = 2) then
    begin
        RegJuegoRuleta.NroJugada:= RegJuegoRuleta.NroJugada;
        RegJuegoRuleta.FechaHora:= Now();
        RegJuegoRuleta.Estado:= 3;
        RegJuegoRuleta.Bolilla:= (random(36));
        ME_JUEGO.ModificoUltimo(ME_JUEGORULETA, RegJuegoRuleta);

        Lbl_estadoJuego.Caption:='"Bolilla tirada"';

        Lbl_nroBolillaG.Caption:= intTostr(RegJuegoRuleta.Bolilla);
        Spbtn_tirarBolilla.Enabled:= false;
        Spbtn_trampa.Enabled:=true;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Spbtn_trampaClick(Sender: TObject);
begin
  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar
  MessageDlg('SELECCIONE NUMERO DE LA TABLA PARA HACER TRAMPA.',mtInformation , [mbOK],
0);
  BtnTrampaIsActive:=true;
  ActiveBtnTrampa();
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image0Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);
```

```
  tabla_Image0.Width := tabla_Image0.Width + 5;
  tabla_Image0.Height := tabla_Image0.Height + 25;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(0);
  nameOp:= intTostr(0);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(0)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
        begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
        end
    end;
end;

{-------------------------------------------------------------------------------}
```

```
procedure TF_Juego.tabla_Image3Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image3.Width := tabla_Image3.Width + 5;
  tabla_Image3.Height := tabla_Image3.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(3);
  nameOp:= intTostr(3);

  //Si es el croupier y se activo el boton trampa, llamo a TableroTrampaSeleccionado
parametrizando numero de bolilla a hacer trampa
  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(3)
  else //es el croupier o jugador comun que estan apostando
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      //si el estado del juego es distinto a 2 (no va mas) esntonces parametrizo
nomenclado, valor, y nombre de opcion (nombre de valores que no son numero planos)
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else //el estado del juego es 2 (no va mas) entonces de deshabilita tablero, se
refrezca pantalla del usuario comun, y el tamaño del tablero vuelve a su estado
principal
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
```

```
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image10Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image10.Width := tabla_Image10.Width + 5;
  tabla_Image10.Height := tabla_Image10.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(10);
  nameOp:= intTostr(10);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(10)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
```

```
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image11Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image11.Width := tabla_Image11.Width + 5;
  tabla_Image11.Height := tabla_Image11.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(11);
  nameOp:= intTostr(11);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(11)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
```

```
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image12Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image12.Width := tabla_Image12.Width + 5;
  tabla_Image12.Height := tabla_Image12.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(12);
  nameOp:= intTostr(12);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(12)
```

```
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image13Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image13.Width := tabla_Image13.Width + 5;
  tabla_Image13.Height := tabla_Image13.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
```

```
  valor:= intTostr(13);
  nameOp:= intTostr(13);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(13)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image14Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);
```

```
  tabla_Image14.Width := tabla_Image14.Width + 5;
  tabla_Image14.Height := tabla_Image14.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(14);
  nameOp:= intTostr(14);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(14)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}
```

```
procedure TF_Juego.tabla_Image15Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image15.Width := tabla_Image15.Width + 5;
  tabla_Image15.Height := tabla_Image15.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(15);
  nameOp:= intTostr(15);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(15)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
```

```
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image16Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image16.Width := tabla_Image16.Width + 5;
  tabla_Image16.Height := tabla_Image16.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(16);
  nameOp:= intTostr(16);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(16)
  else
    begin
      ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
      if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a
cabo la apuesta ...
      begin
        if (RegJuegoRuleta.Estado<2) then
          TableroLegalSeleccionado(Nomenclador,valor,nameOp)
        else
        begin
          F_Juego.panel_tablero.Enabled:=false;
          refrezcarPantallaNormalUser();
          AutoSize_ImagenesTablero();
        end;
      end
```

```
      else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}
procedure TF_Juego.tabla_Image17Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image17.Width := tabla_Image17.Width + 5;
  tabla_Image17.Height := tabla_Image17.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(17);
  nameOp:= intTostr(17);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(17)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
```

```
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image18Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image18.Width := tabla_Image18.Width + 5;
  tabla_Image18.Height := tabla_Image18.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(18);
  nameOp:= intTostr(18);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(18)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
```

```
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image19Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image19.Width := tabla_Image19.Width + 5;
  tabla_Image19.Height := tabla_Image19.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(19);
  nameOp:= intTostr(19);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(19)
  else
```

```
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image1Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image1.Width := tabla_Image1.Width + 5;
  tabla_Image1.Height := tabla_Image1.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(1);
```

```
  nameOp:= intTostr(1);


  if (BtnTrampaIsActive) then
    TableroTrampaSeleccionado(1)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.tabla_Image20Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);


  tabla_Image20.Width := tabla_Image20.Width + 5;
```

```
  tabla_Image20.Height := tabla_Image20.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(20);
  nameOp:= intTostr(20);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(20)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}
procedure TF_Juego.tabla_Image21Click(Sender: TObject);
begin
```

```
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image21.Width := tabla_Image21.Width + 5;
  tabla_Image21.Height := tabla_Image21.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(21);
  nameOp:= intTostr(21);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(21)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;
```

```
{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image22Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image22.Width := tabla_Image22.Width + 5;
  tabla_Image22.Height := tabla_Image22.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(22);
  nameOp:= intTostr(22);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(22)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
```

```
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}
procedure TF_Juego.tabla_Image23Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image23.Width := tabla_Image23.Width + 5;
  tabla_Image23.Height := tabla_Image23.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(23);
  nameOp:= intTostr(23);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(23)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
```

```
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image24Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image24.Width := tabla_Image24.Width + 5;
  tabla_Image24.Height := tabla_Image24.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(24);
  nameOp:= intTostr(24);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(24)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
```

```
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image25Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image25.Width := tabla_Image25.Width + 5;
  tabla_Image25.Height := tabla_Image25.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(25);
  nameOp:= intTostr(25);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(25)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
```

```
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image26Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image26.Width := tabla_Image26.Width + 5;
  tabla_Image26.Height := tabla_Image26.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(26);
  nameOp:= intTostr(26);
```

```
  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(26)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image27Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image27.Width := tabla_Image27.Width + 5;
  tabla_Image27.Height := tabla_Image27.Height + 15;
  panel_tablero.Enabled:= false;
```

```
  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(27);
  nameOp:= intTostr(27);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(27)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image28Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
```

```
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);


  tabla_Image28.Width := tabla_Image28.Width + 5;
  tabla_Image28.Height := tabla_Image28.Height + 15;
  panel_tablero.Enabled:= false;


  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(28);
  nameOp:= intTostr(28);


  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(28)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;
```

```
{-------------------------------------------------------------------------------}


procedure TF_Juego.tabla_Image29Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image29.Width := tabla_Image29.Width + 5;
  tabla_Image29.Height := tabla_Image29.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(29);
  nameOp:= intTostr(29);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(29)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
```

```
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image2Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image2.Width := tabla_Image2.Width + 5;
  tabla_Image2.Height := tabla_Image2.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(2);
  nameOp:= intTostr(2);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(2)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
```

```
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image30Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image30.Width := tabla_Image30.Width + 5;
  tabla_Image30.Height := tabla_Image30.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(30);
  nameOp:= intTostr(30);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(30)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
```

```
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image31Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image31.Width := tabla_Image31.Width + 5;
  tabla_Image31.Height := tabla_Image31.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(31);
  nameOp:= intTostr(31);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(31)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
```

```
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image32Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image32.Width := tabla_Image32.Width + 5;
  tabla_Image32.Height := tabla_Image32.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(32);
  nameOp:= intTostr(32);
```

```
  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(32)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image33Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image33.Width := tabla_Image33.Width + 5;
  tabla_Image33.Height := tabla_Image33.Height + 15;
  panel_tablero.Enabled:= false;
```

```
  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(33);
  nameOp:= intTostr(33);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(33)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image34Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
```

```
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);


  tabla_Image34.Width := tabla_Image34.Width + 5;
  tabla_Image34.Height := tabla_Image34.Height + 15;
  panel_tablero.Enabled:= false;


  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(34);
  nameOp:= intTostr(34);


  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(34)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;
```

```
{-------------------------------------------------------------------------------}


procedure TF_Juego.tabla_Image35Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image35.Width := tabla_Image35.Width + 5;
  tabla_Image35.Height := tabla_Image35.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(35);
  nameOp:= intTostr(35);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(35)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
```

```
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image36Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image36.Width := tabla_Image36.Width + 5;
  tabla_Image36.Height := tabla_Image36.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(36);
  nameOp:= intTostr(36);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(36)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
```

```
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image4Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image4.Width := tabla_Image4.Width + 5;
  tabla_Image4.Height := tabla_Image4.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(4);
  nameOp:= intTostr(4);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(4)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
```

```
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image5Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image5.Width := tabla_Image5.Width + 5;
  tabla_Image5.Height := tabla_Image5.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(5);
  nameOp:= intTostr(5);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(5)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
```

```
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image6Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image6.Width := tabla_Image6.Width + 5;
  tabla_Image6.Height := tabla_Image6.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(6);
  nameOp:= intTostr(6);
```

```
  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(6)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}


procedure TF_Juego.tabla_Image7Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image7.Width := tabla_Image7.Width + 5;
  tabla_Image7.Height := tabla_Image7.Height + 15;
```

```
  panel_tablero.Enabled:= false;


  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(7);
  nameOp:= intTostr(7);


  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(7)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image8Click(Sender: TObject);
begin
```

```
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image8.Width := tabla_Image8.Width + 5;
  tabla_Image8.Height := tabla_Image8.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(8);
  nameOp:= intTostr(8);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(8)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;
```

```
{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_Image9Click(Sender: TObject);
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image9.Width := tabla_Image9.Width + 5;
  tabla_Image9.Height := tabla_Image9.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Pleno;
  valor:= intTostr(9);
  nameOp:= intTostr(9);

  if ((AdminLog) and (BtnTrampaIsActive)) then
    TableroTrampaSeleccionado(9)
  else
    begin
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
```

```
          MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
          SalirDelJuego();
      end
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_ImageImparesClick(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_ImageImpares.Width := tabla_ImageImpares.Width + 5;
  tabla_ImageImpares.Height := tabla_ImageImpares.Height + 10;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_ParImpar;
  valor:= 'I';
  nameOp:= '"IMPARES"';


    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
```

```
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_ImageNegroClick(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_ImageNegro.Width := tabla_ImageNegro.Width + 5;
  tabla_ImageNegro.Height := tabla_ImageNegro.Height + 10;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Rojonegro;
  valor:= 'N';
  nameOp:= '"Negro"';


    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
```

```
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.tabla_ImageParesClick(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);


  tabla_ImagePares.Width := tabla_ImagePares.Width + 5;
  tabla_ImagePares.Height := tabla_ImagePares.Height + 10;
  panel_tablero.Enabled:= false;


  Nomenclador:= Type_APUESTA.Nom_ParImpar;
  valor:= 'P';
  nameOp:= '"PARES"';



    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
```

```
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_ImagePrimeraColClick(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_ImagePrimeraCol.Width := tabla_ImagePrimeraCol.Width + 5;
  tabla_ImagePrimeraCol.Height := tabla_ImagePrimeraCol.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Columna;
  valor:= intTostr(1);
  nameOp:= '"Primer columna"';


    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
```

```
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_ImagePrimeros12Click(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_ImagePrimeros12.Width := tabla_ImagePrimeros12.Width + 20;
  tabla_ImagePrimeros12.Height := tabla_ImagePrimeros12.Height + 5;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Docena;
  valor:= intTostr(1);
  nameOp:= '"Primeros doce"';

    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
```

```
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_ImageRojoClick(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_ImageRojo.Width := tabla_ImageRojo.Width + 5;
  tabla_ImageRojo.Height := tabla_ImageRojo.Height + 10;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Rojonegro;
  valor:= 'R';
  nameOp:= '"Rojo"';

    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
```

```
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_ImageSegundaColClick(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_ImageSegundaCol.Width := tabla_ImageSegundaCol.Width + 5;
  tabla_ImageSegundaCol.Height := tabla_ImageSegundaCol.Height + 15;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Columna;
  valor:= intTostr(2);
  nameOp:= '"Segunda columna"';
```

```
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.tabla_ImageSegundos12Click(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_ImageSegundos12.Width := tabla_ImageSegundos12.Width + 20;
  tabla_ImageSegundos12.Height := tabla_ImageSegundos12.Height + 5;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_Docena;
  valor:= intTostr(2);
```

```
  nameOp:= '"Segundos doce"';



    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;

{-------------------------------------------------------------------------------}



procedure TF_Juego.tabla_ImageTerceraColClick(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);


  tabla_ImageTerceraCol.Width := tabla_ImageTerceraCol.Width + 5;
  tabla_ImageTerceraCol.Height := tabla_ImageTerceraCol.Height + 15;
  panel_tablero.Enabled:= false;
```

```
  Nomenclador:= Type_APUESTA.Nom_Columna;
  valor:= intTostr(3);
  nameOp:= '"Tercer columna"';



    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;


{-------------------------------------------------------------------------------}



procedure TF_Juego.tabla_ImageTerceros12Click(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);
```

```
  tabla_ImageTerceros12.Width := tabla_ImageTerceros12.Width + 20;
  tabla_ImageTerceros12.Height := tabla_ImageTerceros12.Height + 5;
  panel_tablero.Enabled:= false;



  Nomenclador:= Type_APUESTA.Nom_Docena;
  valor:= intTostr(3);
  nameOp:= '"Terceros doce"';



    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;

{-------------------------------------------------------------------------------}



procedure TF_Juego.tabla_Image_19a36Click(Sender: TObject);
//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa
```

```
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  tabla_Image_19a36.Width := tabla_Image_19a36.Width + 5;
  tabla_Image_19a36.Height := tabla_Image_19a36.Height + 10;
  panel_tablero.Enabled:= false;

  Nomenclador:= Type_APUESTA.Nom_PasaFalta;
  valor:= 'F';
  nameOp:= '"Falta(19-36)"';



    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);
    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...
    begin
      if (RegJuegoRuleta.Estado<2) then
        TableroLegalSeleccionado(Nomenclador,valor,nameOp)
      else
      begin
        F_Juego.panel_tablero.Enabled:=false;
        refrezcarPantallaNormalUser();
        AutoSize_ImagenesTablero();
      end;
    end
    else
      begin
        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);
        SalirDelJuego();
      end;
end;

{-------------------------------------------------------------------------------}
```

```
procedure TF_Juego.tabla_Image_1a18Click(Sender: TObject);

//Esta casilla sera estará disponible si se lleva a cabo una apuesta y no cuando hace
trampa

begin

  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Lbl_usuarioActual.Caption);

  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);


  tabla_Image_1a18.Width := tabla_Image_1a18.Width + 5;

  tabla_Image_1a18.Height := tabla_Image_1a18.Height + 10;

  panel_tablero.Enabled:= false;


  Nomenclador:= Type_APUESTA.Nom_PasaFalta;

  valor:= 'P';

  nameOp:= '"Pasa (1-18)"';



    ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);

    if not (regJugador.Bloqueado) then //si no esta bloqueado quien va a llevar a cabo
la apuesta ...

    begin

      if (RegJuegoRuleta.Estado<2) then

        TableroLegalSeleccionado(Nomenclador,valor,nameOp)

      else

      begin

        F_Juego.panel_tablero.Enabled:=false;

        refrezcarPantallaNormalUser();

        AutoSize_ImagenesTablero();

      end;

    end

    else

      begin

        MessageDlg('Lo siento, ha sido bloqueado!!' ,mtWarning , [mbOK], 0);

        SalirDelJuego();

      end;

end;
```

```
{------------------------------------------------------------------------------}


//El tablero volvera a su tamaño original luego de que se haya efecuado un 'click'
Procedure TF_Juego.AutoSize_ImagenesTablero;
begin
  tabla_Image0.AutoSize := true;tabla_Image0.AutoSize := false;
  tabla_Image1.AutoSize := true;tabla_Image1.AutoSize := false;
  tabla_Image2.AutoSize := true;tabla_Image2.AutoSize := false;
  tabla_Image3.AutoSize := true;tabla_Image3.AutoSize := false;
  tabla_Image3.AutoSize := true;tabla_Image3.AutoSize := false;
  tabla_Image3.AutoSize := true;tabla_Image3.AutoSize := false;
  tabla_Image4.AutoSize := true;tabla_Image4.AutoSize := false;
  tabla_Image5.AutoSize := true;tabla_Image5.AutoSize := false;
  tabla_Image6.AutoSize := true;tabla_Image6.AutoSize := false;
  tabla_Image7.AutoSize := true;tabla_Image7.AutoSize := false;
  tabla_Image8.AutoSize := true;tabla_Image8.AutoSize := false;
  tabla_Image9.AutoSize := true;tabla_Image9.AutoSize := false;
  tabla_Image10.AutoSize := true;tabla_Image10.AutoSize := false;
  tabla_Image11.AutoSize := true;tabla_Image11.AutoSize := false;
  tabla_Image12.AutoSize := true;tabla_Image12.AutoSize := false;
  tabla_Image13.AutoSize := true;tabla_Image13.AutoSize := false;
  tabla_Image14.AutoSize := true;tabla_Image14.AutoSize := false;
  tabla_Image15.AutoSize := true;tabla_Image15.AutoSize := false;
  tabla_Image16.AutoSize := true;tabla_Image16.AutoSize := false;
  tabla_Image17.AutoSize := true;tabla_Image17.AutoSize := false;
  tabla_Image18.AutoSize := true;tabla_Image18.AutoSize := false;
  tabla_Image19.AutoSize := true;tabla_Image19.AutoSize := false;
  tabla_Image20.AutoSize := true;tabla_Image20.AutoSize := false;
  tabla_Image21.AutoSize := true;tabla_Image21.AutoSize := false;
  tabla_Image22.AutoSize := true;tabla_Image22.AutoSize := false;
  tabla_Image23.AutoSize := true;tabla_Image23.AutoSize := false;
  tabla_Image24.AutoSize := true;tabla_Image24.AutoSize := false;
  tabla_Image25.AutoSize := true;tabla_Image25.AutoSize := false;
  tabla_Image26.AutoSize := true;tabla_Image26.AutoSize := false;
  tabla_Image27.AutoSize := true;tabla_Image27.AutoSize := false;
```

```
  tabla_Image28.AutoSize := true;tabla_Image28.AutoSize := false;
  tabla_Image29.AutoSize := true;tabla_Image29.AutoSize := false;
  tabla_Image30.AutoSize := true;tabla_Image30.AutoSize := false;
  tabla_Image31.AutoSize := true;tabla_Image31.AutoSize := false;
  tabla_Image32.AutoSize := true;tabla_Image32.AutoSize := false;
  tabla_Image33.AutoSize := true;tabla_Image33.AutoSize := false;
  tabla_Image34.AutoSize := true;tabla_Image34.AutoSize := false;
  tabla_Image35.AutoSize := true;tabla_Image35.AutoSize := false;
  tabla_Image36.AutoSize := true;tabla_Image36.AutoSize := false;

  tabla_ImagePrimeraCol.AutoSize:= true;tabla_ImagePrimeraCol.AutoSize:= false;
  tabla_ImageSegundaCol.AutoSize:= true;tabla_ImageSegundaCol.AutoSize:= false;
  tabla_ImageTerceraCol.AutoSize:= true;tabla_ImageTerceraCol.AutoSize:= false;

  tabla_ImagePrimeros12.AutoSize:= true;tabla_ImagePrimeros12.AutoSize:= false;
  tabla_ImageSegundos12.AutoSize:= true;tabla_ImageSegundos12.AutoSize:= false;
  tabla_ImageTerceros12.AutoSize:= true;tabla_ImageTerceros12.AutoSize:= false;

  tabla_Image_1a18.AutoSize:= true;tabla_Image_1a18.AutoSize:= false;
  tabla_Image_19a36.AutoSize:= true;tabla_Image_19a36.AutoSize:= false;

  tabla_ImagePares.AutoSize:= true;tabla_ImagePares.AutoSize:= false;
  tabla_ImageImpares.AutoSize:= true;tabla_ImageImpares.AutoSize:= false;

  tabla_ImageRojo.AutoSize:= true;tabla_ImageRojo.AutoSize:= false;
  tabla_ImageNegro.AutoSize:= true;tabla_ImageNegro.AutoSize:= false;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Btn_closePnelLoserClick(Sender: TObject);
var
 i: integer;
begin
  for i:=0 to menuFormJuego.Items.Count-1 do
```

```
    menuFormJuego.Items[I].Enabled:= true;

  //Un ves mostrado el panel de ganador o perdedor, dejo a usuario ver premios
   menuFormJuego.Items[2].Visible:= true;

  Panel_Jugadores.Enabled:= true;
  Panel_Winner.Visible:= false;
  Panel_Loser.Visible:= false;
  Panel_Winner.Color:= clRed;
  Lbl_waiting.Font.Color:= clBlack;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Btn_closePnelWinnerClick(Sender: TObject);
var
 i: integer;
begin
   for i:=0 to menuFormJuego.Items.Count-1 do
    menuFormJuego.Items[I].Enabled:= true;

   Panel_Jugadores.Enabled:= true;
   Panel_Winner.Visible:= false;
   Panel_ShowWinner.Visible:= false;

   (gifWinner.Picture.Graphic as TGIFImage).Animate:= false;
   gifWinner.Visible:= false;
   Panel_Winner.Color:= clRed;
   Lbl_waiting.Font.Color:= clBlack;
end;

{-------------------------------------------------------------------------------}

Procedure
TF_Juego.TableroLegalSeleccionado(nomenclador:Type_APUESTA.tNomenclador;valor:Type_APUE
STA.tNomenclador; nameOp:string);
```

```
var
  Cant: tCantidad; // cantidad jugadores en general
  Cant_Desbloq: tCantidad; // cantidad de jugadores desbloqueados
  Cant_Activos: tCantidad; // cantidad de jugadores activos
begin
  Cant_Desbloq:= 0;
  Cant_Activos:= 0;

  cant:= ME_JUGADORES.Cantidad_Jugadores(ME_JUGADOR);
  Lib_AuxJuego.Cant_JugadoresDesbloqueados(ME_JUGADORES.Raiz(ME_JUGADOR),
Cant_Desbloq);
  Lib_AuxJuego.Cant_JugadoresActivos(ME_JUGADORES.Raiz(ME_JUGADOR), Cant_Activos);



//Usuario comun
  if not (Form_login.AdminLog) then
  begin
    if Lib_Auxiliar.Mensaje_Confirmacion('¿Esta seguro que quiere apostar por:
'+nameOp+ ' ?') then
    begin
      SoloListar:= False; //
      F_Apostar.Show;

      F_Apostar.Lbl_SelectJugador.hide;
      F_Apostar.Lbl_ListaJugadores.hide;
      F_Apostar.Grid_ListaJugadores.hide;

      F_Apostar.Lbl_ListaDeApuestas.Show;
      F_Apostar.Grid_ListaApuestas.Show;

      F_Apostar.Panel_DatosApostar.show;
      F_Apostar.Edit_nomenclador.Text:= nomenclador;
      F_Apostar.Edit_valor.Text:=valor;
      F_Apostar.Edit_nameJugador.Text:= NickLogueado;
      Form_Apostar.F_Apostar.Edit_importe.SetFocus;
```

```
      F_Juego.ListandoApuestas();
    end
    else
      begin
        AutoSize_ImagenesTablero();
        panel_tablero.Enabled:= true;
      end;
  end
  else  //Croupier
    begin
        F_Apostar.Lbl_ListaJugadores.hide;
        F_Apostar.Lbl_ListaDeApuestas.hide;
        F_Apostar.Grid_ListaApuestas.hide
        ;
        F_Apostar.Lbl_SelectJugador.Show;
        F_Apostar.Grid_ListaJugadores.Show;

        F_Apostar.Edit_nameJugador.Clear;

        if (not ME_JUGADORES.MeVacio_Jugadores(ME_JUGADOR) and (cant>1)) then //si cant
es >1 entonces al menos hay un jugador que no es el administrador
        begin
        //Si al menos hay un jugador desbloqueado
          if (Cant_Desbloq >=1) then
          begin
          //Si hay jugadores activos
            if (Cant_Activos>=1) then
            begin
              if Lib_Auxiliar.Mensaje_Confirmacion('¿Esta seguro que quiere apostar
por: '+nameOp+ ' ?') then
              begin
                F_Apostar.Show;
                F_Apostar.Panel_DatosApostar.show;
                F_Apostar.Edit_nomenclador.Text:= nomenclador;
                F_Apostar.Edit_valor.Text:=valor;
                F_Apostar.ListandoJugadores();
```

```
              end;
            end
            else
              MessageDlg('No hay jugadores activos!',mtWarning, [mbOK], 0);
          end
          else
            MessageDlg('Existen jugadores, pero estan bloqueados',mtWarning, [mbOK],
0);
        end
        else
          MessageDlg('No existen jugadores!!',mtWarning, [mbOK], 0);

      AutoSize_ImagenesTablero();
      panel_tablero.Enabled:= true;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.TableroTrampaSeleccionado(num_bolilla:Type_JUEGO.tBolilla);
begin //Soy el Croupier y estoy haciendo trampa
  if Lib_Auxiliar.Mensaje_Confirmacion('¿Esta seguro que quiere cometer trampa con la
bolilla Nro: ' +intTostr(num_bolilla)+ ' ?') then
  begin
    Lbl_nroBolillaG.Caption:= intTostr(num_bolilla);
    Lbl_nroBolillaG.Font.Color:=clRed;
    Lbl_bolillaASalir.Font.Color:=clRed;
    Spbtn_trampa.Enabled:= false;

    //Se que estoy en estado 3 (bolilla tirada), actualizo regJuegoRuleta con la
"bolilla ganadora trampa"
    ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);
    RegJuegoRuleta.NroJugada:= RegJuegoRuleta.NroJugada;
    RegJuegoRuleta.FechaHora:= Now();
    RegJuegoRuleta.Estado:= RegJuegoRuleta.Estado;
    RegJuegoRuleta.Bolilla:= num_bolilla;
```

```
    ME_JUEGO.ModificoUltimo(ME_JUEGORULETA, RegJuegoRuleta);

    Spbtn_tirarBolilla.Enabled:=false;
    Spbtn_repartirPremios.Enabled:=true;

    BtnTrampaIsActive:= false;
    DesactiveBtnTrampa();
  end
  else
    begin
      panel_tablero.Enabled:= true;
      F_Apostar.Edit_nomenclador.Clear;
      F_Apostar.Edit_valor.Clear;
    end;

  AutoSize_ImagenesTablero();
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showWinner2Timer(Sender: TObject);
begin

  Timer_showWinner2.Enabled:= false;
  Panel_Winner.Visible:= true;
  Timer_showWinner3.Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showWinner3Timer(Sender: TObject);
begin
  Timer_showWinner3.Enabled:= false;
  Panel_Winner.Visible:= false;
  Timer_showWinner4.Enabled:= true;
```

```
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showWinner4Timer(Sender: TObject);
begin
  Timer_showWinner4.Enabled:= false;
  Panel_Winner.Visible:= true;
  Panel_Winner.Color:= clBlack;
  Lbl_waiting.Font.Color:= clRed;
  Lbl_waiting.Caption:= 'Aguarde ' +#13+ '  unos' +#13+ 'segundos ..';
  Timer_showWinner5.Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showWinner5Timer(Sender: TObject);
begin
  Timer_showWinner5.Enabled:= false;
  Panel_Winner.Visible:= false;
  Timer_showWinner6.Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showWinner6Timer(Sender: TObject);
begin
  Timer_showWinner6.Enabled:= false;
  Panel_Winner.Visible:= true;

  if (ShowWinner) then
  begin
    Panel_ShowWinner.Visible:= true;
    gifWinner.Visible:= true;
    (gifWinner.Picture.Graphic as TGIFImage).Animate:= true;
```

```
  end
  else
    Panel_Loser.Visible:= true;

  Lbl_saldoAcumulado.Show;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showLoser1Timer(Sender: TObject);
begin
  Timer_showLoser1.Enabled:= false;
  Panel_Loser.Visible:= false;

  Timer_showLoser2.Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showLoser2Timer(Sender: TObject);
begin
  Panel_Loser.Visible:= true;
  Timer_showLoser2.Enabled:= false;

  Timer_showLoser3.Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showLoser3Timer(Sender: TObject);
begin
  Timer_showLoser3.Enabled:= false;
  Panel_Loser.Visible:= false;

  Timer_showLoser4.Enabled:= true;
```

```
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showLoser4Timer(Sender: TObject);
begin
  Timer_showLoser4.Enabled:= false;
  Panel_Loser.Visible:= true;
  Timer_showLoser5.Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showLoser5Timer(Sender: TObject);
begin
  Timer_showLoser5.Enabled:= false;
  Panel_Loser.Visible:= false;
  Timer_showLoser6.Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showLoser6Timer(Sender: TObject);
begin
  Timer_showLoser6.Enabled:= false;
  Panel_Loser.Visible:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.Timer_showWinner1Timer(Sender: TObject);
begin
  Timer_showWinner1.Enabled:= false;
  Timer_showWinner2.Enabled:= true;
end;
```

```
{-------------------------------------------------------------------------------}

procedure TF_Juego.refrezcarPantallaNormalUser();
var
AuxRegJuego: Type_JUEGO.TipoRegDatos;
saldoActual: tImporte;
cant:tCantidad;
i: integer;

corte: boolean;
monto: tImporte;

begin
  //Actualizo fecha y hora a mostrar
  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual);

  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, AuxRegJuego);

  //Si el estado es mayor o igual a 2 (no va mas), dejo a usuario ver apuestas
efectuadas
  if ((AuxRegJuego.estado>=2) and (AuxRegJuego.estado<4)) then
    menuFormJuego.Items[1].Visible:= true;

      CASE (AuxRegJuego.estado) of
      0:begin
         Lbl_estadoJuego.Caption:='"Juego Creado"';
         //acumulacionDeApuestas:= 0;
         saldoAcumulado:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,
regJugador.Nick);
         Lbl_saldoAcumulado.Caption:= '$ ' + inttostr(saldoAcumulado);
        end;//end estado=0
      1:begin
         Lbl_estadoJuego.Caption:='"Hagan sus apuestas"';
         panel_tablero.Enabled:= true;
         saldoActual:= F_Apostar.obtenerSaldoActualJugador(NickLogueado);
```

```
          F_Juego.Lbl_saldoAcumulado.Caption:= '$ ' + intTostr(saldoActual);
        end;//end estado=1
      2:begin
          Lbl_estadoJuego.Caption:='"No va más"';
          MessageDlg('Se han cerrado las apuestas!',mtInformation , [mbOK], 0);
          //acumulacionDeApuestas:= 0;
          AutoSize_ImagenesTablero();
          saldoAcumulado:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,
regJugador.Nick);
          Lbl_saldoAcumulado.Caption:= '$ ' + inttostr(saldoAcumulado);
        end;//end estado=2
      3:begin
          Lbl_estadoJuego.Caption:='"Bolilla tirada"';
          saldoAcumulado:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,
regJugador.Nick);
          Lbl_saldoAcumulado.Caption:= '$ ' + inttostr(saldoAcumulado);
        end;//end estado=3
      4:begin
          monto:= 0;
          Lbl_estadoJuego.Caption:='"Premios repartidos y fin de juego"';
          saldoAcumulado:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,
regJugador.Nick);
          Lbl_saldoAcumulado.Caption:= '$ ' + inttostr(saldoAcumulado);
          Lbl_saldoAcumulado.Hide;

         //Muestro formulario ganador

Lib_AuxJuego.Premios_AcumuladosXJugadorUnaJugada(ME_GANADORES.Raiz(ME_GANADOR),
regJugador.Nick, RegJuegoRuleta.NroJugada, monto);
          if (monto > 0) then
          begin
            ShowWinner:= true;
            Timer_showWinner1.Enabled:= true;
            Lbl_montoWinner.Caption:= intTostr(monto);
            Lbl_bolillaGWinner.Caption:= intTostr(AuxRegJuego.Bolilla);
          end
          else
```

```
            begin
              ShowWinner:= false;
              Timer_showWinner1.Enabled:= true;
              Lbl_bolillaGLoser.Caption:= intTostr(AuxRegJuego.Bolilla);
            end;
            Lbl_waiting.Caption:= 'Aguarde ' +#13+ '  unos' +#13+ 'segundos';

          //Deshabilito panel jugadores
          Panel_Jugadores.Enabled:= false;

          //Deshabilito menu principal del form_juego al usuario comun
          for i:=0 to menuFormJuego.Items.Count-1 do
            menuFormJuego.Items[I].Enabled:= false;



        end;//end estado=4
      end;//end CASE

      //consulto si el estado del juego es >= 2 para asegurarme que nadie puede apostar
si el ya 'no va mas'
      if (AuxRegJuego.Estado>=2) then
      begin
        //Oculto tablero del jugador
        panel_tablero.Hide;
        Panel_NameTablero.Hide;
      end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.SalirDelJuego();
begin
 Form_Juego.F_Juego.close;
 Form_Jugadores.F_Jugadores.Show;
end;
```

```
{-------------------------------------------------------------------------------}


procedure TF_Juego.pantallaIniJuegoCrupier();

begin

  panel_tablero.Enabled:= false;

  AutoSize_ImagenesTablero();

  Lbl_usuarioActual.Caption:= NickLogueado;

  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar

  //--------------------------------

  Panel_Jugadores.Hide;

  Panel_Crupier.Show;

  F_Juego.Lbl_CantFictActivados.Caption:='--';;

  Lbl_cantFicticios.Caption:='--';

  Lbl_nroBolillaG.Caption:='--';

  //--------------------------------

  Spbtn_CrearPartida.Enabled:=true;

  Spbtn_bloqDesbloq.Enabled:= true;

  Spbtn_haganApuestas.Enabled:=false;

  Spbtn_noVaMas.Enabled:=false;

  Spbtn_tirarBolilla.Enabled:=false;

  Spbtn_repartirPremios.Enabled:=false;


  Spbtn_activarFicticios.Enabled:= false;

  Spbtn_obsequiarCred.Enabled:= false;

  Spbtn_trampa.Enabled:= false;

end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.pantallaIniJuegoJugadores();

var

  i:tPos;

begin
```

```
  panel_tablero.Enabled:= false;
  AutoSize_ImagenesTablero();
  Lbl_usuarioActual.Caption:= NickLogueado;
  ActualizarFechaHora(Lbl_fechaActual, Lbl_horaActual); //Actualizo fecha y hora a
mostrar

  Panel_Crupier.hide;
  Panel_Jugadores.Show;
  ME_JUEGO.RegistroNulo(RegJuegoRuleta);
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta); //
  Lbl_nroPartida.Caption:= RegJuegoRuleta.NroJugada; //
  panel_tablero.Show;
  Panel_NameTablero.Show;
  Lbl_ApuestaMax.Caption:= '$ ' + intTostr(apuestaMax);
  Lbl_ApuestaMin.Caption:= '$ ' + intTostr(apuestaMin);

  menuFormJuego.Items[0].Visible:= true;
  menuFormJuego.Items[1].Visible:= false;
  menuFormJuego.Items[2].Visible:= false;
  //for i:=1 to menuFormJuego.Items.Count-1 do
  //  menuFormJuego.Items[i].Visible:= false;
end;

{-------------------------------------------------------------------------------}

procedure TF_Juego.ListarPremiados(Nro_Jugada: Type_JUEGO.tClave);
begin
  ME_JUEGO.RegistroNulo(RegJuegoRuleta);
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, RegJuegoRuleta);

  Form_Premios.F_Premios.show;
  Lib_Auxiliar.limpiarGrid(F_Premios.Grid_Premiados);

  Lib_Auxiliar.EncabezadoGridPremios(F_Premios.Grid_Premiados);
  F_Premios.Grid_premiados.RowCount:=1;
```

```
  Lib_AuxJuego.Listado_Premios(ME_GANADORES.Raiz(ME_GANADOR), Nro_Jugada,
F_Premios.Grid_Premiados);

  F_Premios.Grid_premiados.FixedRows:=1;

end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.ListandoApuestas();

begin

  Lib_Auxiliar.limpiarGrid(F_Apostar.Grid_ListaApuestas);

  F_Apostar.Grid_ListaApuestas.RowCount:=1;

  Lib_Auxiliar.EncabezadosComunApuestas(F_Apostar.Grid_ListaApuestas);


  ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);


  if (Lib_AuxJuego.JugadorTieneApuestasEnPartida(RegJuegoRuleta.NroJugada,
NickLogueado)) then

  begin

    F_Apostar.Grid_ListaApuestas.RowCount:=1;

    Lib_AuxJuego.Listado_ApuestasUnJugador(F_Apostar.Grid_ListaApuestas,
RegJuegoRuleta.NroJugada, NickLogueado);

    //F_Apostar.Grid_ListaApuestas.FixedRows:=1;

  end;

end;


{-------------------------------------------------------------------------------}


procedure TF_Juego.ActualizarFechaHora(var Lbl_fecha:TLabel; var Lbl_hora: TLabel);

begin

  Lbl_fecha.Caption:= datetostr(now);

  Lbl_hora.Caption:= timetostr(now);

end;

END.
```

3.2.5 Formulario Apostar

```
unit Form_Apostar;


interface


uses
  ME_JUGADORES,
  Type_JUGADOR,
  ME_CTACTE,
  Type_ALMACEN,
  ME_JUEGO,
  Type_JUEGO,
  ME_APUESTAS,
  Type_APUESTA,
  ME_GANADORES,
  Type_GANADOR,
  Lib_Auxiliar,
  Lib_AuxJuego,
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.StdCtrls, Vcl.Buttons,
  Vcl.ExtCtrls;


type
  TF_Apostar = class(TForm)
    Panel_Listado: TPanel;
    Lbl_SelectJugador: TLabel;
    Panel_DatosApostar: TPanel;
    Lbl_nameJugador: TLabel;
    Edit_nameJugador: TEdit;
```

```
    btn_okeyApostar: TBitBtn;
    Panel_grid: TPanel;
    Lbl_nomenclador: TLabel;
    Edit_nomenclador: TEdit;
    Lbl_valor: TLabel;
    Edit_valor: TEdit;
    Lbl_importe: TLabel;
    Edit_importe: TEdit;
    Lbl_ListaDeApuestas: TLabel;
    Lbl_ListaJugadores: TLabel;
    Grid_ListaApuestas: TStringGrid;
    Grid_ListaJugadores: TStringGrid;
    BitBtn1: TBitBtn;
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    Procedure Listado_GeneralJugadores(RaizJugadores: Lib_Auxiliar.tPos);
    Function obtenerSaldoActualJugador(clave:tClave):tImporte;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Btn_ListarJugadoresClick(Sender: TObject);
    procedure Grid_ListaApuestasClick(Sender: TObject);
    procedure Grid_ListaJugadoresDrawCell(Sender: TObject; ACol, ARow: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure Grid_ListaJugadoresClick(Sender: TObject);
    procedure btn_okeyApostarClick(Sender: TObject);
    procedure Edit_importeKeyPress(Sender: TObject; var Key: Char);
    procedure Grid_ListaApuestasDrawCell(Sender: TObject; ACol, ARow: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure ListandoJugadores();
  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  F_Apostar: TF_Apostar;
  SaldoAcumulado, SaldoApuestas_EnPartida, saldoActual:tImporte;

implementation

uses
  Form_login, Form_Juego;

{$R *.dfm}

{-------------------------------------------------------------------------------}

procedure TF_Apostar.BitBtn1Click(Sender: TObject);
begin
  F_Apostar.Close;
end;

{-------------------------------------------------------------------------------}



procedure TF_Apostar.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  F_Juego.Enabled:= true;
  F_Juego.AutoSize_ImagenesTablero();
  F_Juego.panel_tablero.Enabled:= true;

  if ((not AdminLog) and (not SoloListar)) then
  begin
    saldoActual:= F_Apostar.obtenerSaldoActualJugador(NickLogueado);
    F_Juego.Lbl_saldoAcumulado.Caption:= '$ ' + intTostr(saldoActual);
  end;
end;
```

```
{-------------------------------------------------------------------------------}

procedure TF_Apostar.FormCreate(Sender: TObject);
begin
  self.Position := poScreenCenter;
end;

{-------------------------------------------------------------------------------}

procedure TF_Apostar.FormShow(Sender: TObject);
begin
  F_Juego.Enabled:= false;
  F_Apostar.Grid_ListaJugadores.RowCount:=1;
end;

{-------------------------------------------------------------------------------}

procedure TF_Apostar.Grid_ListaApuestasClick(Sender: TObject);
var
  i, posElim: tPos;
  Nick: type_JUGADOR.tClave;
  Importe: tImporte;
  enc: boolean;
begin
  enc:= false;
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA, regJuegoRuleta);

  if (not SoloListar) then
  begin
    //Si el estado es igual a 1 'hagan sus apuestas'
    if (regJuegoRuleta.Estado = 1) then
    begin
      if (Grid_ListaApuestas.Row <> 0) then
      begin
        Nick:= Grid_ListaApuestas.Cells[0, Grid_ListaApuestas.Row];
```

```
        Nomenclador:= Grid_ListaApuestas.Cells[1, Grid_ListaApuestas.Row];
        Valor:= Grid_ListaApuestas.Cells[2, Grid_ListaApuestas.Row];
        Importe:= strToint(Grid_ListaApuestas.Cells[3, Grid_ListaApuestas.Row]);

        i:= ME_APUESTAS.Primero(ME_APUESTA);
        while ((i <> ME_APUESTAS.PosNula_Apuestas(ME_APUESTA)) or (not enc)) do
        begin
          ME_APUESTAS.CapturarInfoME_Apuestas(ME_APUESTA, i, RegApuesta);
          if (RegApuesta.NroJugada = RegJuegoRuleta.NroJugada) and (RegApuesta.Nick =
Nick) then
            if ((RegApuesta.Nomenclador = Nomenclador) and (RegApuesta.Valor = Valor)
and (RegApuesta.Importe = Importe)) then
            begin
              enc:= true;
              posElim:= i;
            end;
          i:= ME_APUESTAS.Proximo(ME_APUESTA, i);
        end;//while

        if Lib_Auxiliar.Mensaje_Confirmacion('¿Esta seguro que quiere eliminar la
apuesta?') then
        begin
          ME_APUESTAS.EliminarInfoME_Apuestas(ME_APUESTA, posElim);
          F_Juego.ListandoApuestas();
        end;
      end;
    end;
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Apostar.Grid_ListaApuestasDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
  with (Sender as TStringGrid) do
```

```
    begin
      if (ARow = 0)
      then
        begin
          Canvas.Brush.Color := clBtnFace;
          Canvas.TextRect(Rect, Rect.Left + (Rect.Right - Rect.Left -
Canvas.TextWidth(Grid_ListaApuestas.Cells[ACol,ARow]) + 1) div 2, Rect.Top + 2,
Grid_ListaApuestas.Cells[ACol,ARow]);
        end
      else
      begin
        Canvas.Font.Color := clblack;
        if  (ARow mod 2 = 0)
        then
          Canvas.Brush.Color := $00E1FFF9
        else
          Canvas.Brush.Color := $00FFEBDF;
        Canvas.TextRect(Rect, Rect.Left + 2, Rect.Top + 2, cells[acol, arow]);
        Canvas.FrameRect(Rect);
      end;
    end;
end;


{-------------------------------------------------------------------------------}


procedure TF_Apostar.Grid_ListaJugadoresClick(Sender: TObject);
begin
  if (AdminLog) then
  begin
    if (Grid_ListaJugadores.Row <> 0) then
    begin
      Edit_nameJugador.Text := Grid_ListaJugadores.Cells[0,Grid_ListaJugadores.Row];
    end;


    if (Edit_nameJugador.Text<>'') then
    begin
```

```
      Edit_importe.Enabled:=true;
      Edit_importe.SetFocus;
    end
    else
      Edit_importe.Enabled:= false;
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Apostar.Grid_ListaJugadoresDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
with (Sender as TStringGrid) do
  begin
    if (ARow = 0)
    then
      begin
        Canvas.Brush.Color := clBtnFace;
        Canvas.TextRect(Rect, Rect.Left + (Rect.Right - Rect.Left -
Canvas.TextWidth(Grid_ListaJugadores.Cells[ACol,ARow]) + 1) div 2, Rect.Top + 2,
Grid_ListaJugadores.Cells[ACol,ARow]);
      end
    else
    begin
      Canvas.Font.Color := clblack;
      if  (ARow mod 2 = 0)
      then
        Canvas.Brush.Color := $00E1FFF9
      else
        Canvas.Brush.Color := $00FFEBDF;
      Canvas.TextRect(Rect, Rect.Left + 2, Rect.Top + 2, cells[acol, arow]);
      Canvas.FrameRect(Rect);
    end;
  end;
end;
```

```
{-------------------------------------------------------------------------------}
procedure TF_Apostar.Btn_ListarJugadoresClick(Sender: TObject);
begin
  ListandoJugadores();
end;
{-------------------------------------------------------------------------------}


procedure TF_Apostar.btn_okeyApostarClick(Sender: TObject);
begin
  if (Edit_nameJugador.Text<>'') then
  begin
      if (Edit_importe.Text<>'') then
      begin

ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,posJugador,Edit_nameJugador.Text);
        ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,regJugador,posJugador);
        if not (regJugador.Bloqueado) then
        begin
          if (strToint(Edit_Importe.Text) >= ME_JUEGO.ApuestaMinima(ME_JUEGORULETA))
and (strToint(Edit_Importe.Text) <= ME_JUEGO.ApuestaMaxima(ME_JUEGORULETA)) then
          begin
          ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);
          SaldoApuestas_EnPartida:=
ME_APUESTAS.SaldoApuestasPartida(ME_APUESTA,Edit_nameJugador.Text,RegJuegoRuleta.NroJug
ada)+strToint(Edit_importe.Text);
          SaldoAcumulado:=
ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,Edit_nameJugador.Text);


            if ((SaldoApuestas_EnPartida <= SaldoAcumulado)) then
            begin
              if (RegJuegoRuleta.Estado = 1) then
              begin //todo correcto para llevarse acabo la apuesta ...
                //acumulacionDeApuestas:= acumulacionDeApuestas +
strToint(Edit_importe.Text);
                //Armo registroApusta para insertar la nueva apuesta
                regApuesta.NroJugada:= RegJuegoRuleta.NroJugada;
```

```
                regApuesta.Nick:= Edit_nameJugador.Text;
                regApuesta.Nomenclador:= Edit_nomenclador.Text;
                regApuesta.Valor:=Edit_valor.Text;
                regApuesta.Importe:= strToint(Edit_importe.Text);

ME_APUESTAS.InsertarInfoME_Apuestas(ME_APUESTA,regApuesta,ME_APUESTAS.PosNula_Apuestas(
ME_APUESTA));


                //Obtengo el sueldo actual del jugador que se vera reflejado en su
cuenta cuando el estado sea "fin del juego"
                saldoActual:= obtenerSaldoActualJugador(Edit_nameJugador.Text);
                F_Juego.Lbl_saldoAcumulado.Caption:=intTostr(saldoActual);


                Edit_importe.Clear;
                Edit_importe.SetFocus;


                //Grilla apuestas
              Lib_Auxiliar.limpiarGrid(Grid_ListaApuestas);
              Lib_Auxiliar.EncabezadosComunApuestas(Grid_ListaApuestas);
              F_Apostar.Grid_ListaApuestas.RowCount:=1;
              Lib_AuxJuego.Listado_GeneralApuestas(Grid_ListaApuestas,
RegJuegoRuleta.NroJugada);
              F_Apostar.Grid_ListaApuestas.FixedRows:=1;
             //-------------------------------------------------------------------------
-----------
              Form_Juego.F_Juego.AutoSize_ImagenesTablero();
              Form_Juego.F_Juego.panel_tablero.Enabled:= true;
              F_Apostar.Close;
              MessageDlg('Apuesta realizada con éxito!',mtInformation, [mbOK], 0);
              end
              else
                //doble validacion, ya el el boton para abrir form de apuestas estara
deshabilitado si el tiempo de apostar culmino
                MessageDlg('Ya no se permite apuestas. Finalizo el tiempo de
apostar',mtInformation, [mbOK], 0);
            end
            else
              begin
```

```
                MessageDlg('Su saldo no es suficiente para llevar a cabo la
apuesta!',mtWarning, [mbOK], 0);
                Edit_importe.Clear;
                Edit_importe.SetFocus;
              end;
          end
          else
            begin
              MessageDlg('La apuesta minima permitida es de $ '
+intTostr(ME_JUEGO.ApuestaMinima(ME_JUEGORULETA))+
              ', y la apuesta maxima permitida es de $ '
+intTostr(ME_JUEGO.ApuestaMaxima(ME_JUEGORULETA)),mtWarning, [mbOK], 0);
              Edit_importe.Clear;
              Edit_importe.SetFocus;
            end;
        end
        else
          MessageDlg('El jugador se encuentra bloqueado!',mtWarning, [mbOK], 0);
      end
      else
        MessageDlg('Complete el importe a apostar!',mtWarning, [mbOK], 0);
      end
    else
      MessageDlg('Por favor, seleccione un jugador de la grilla!',mtWarning, [mbOK],
0);
end;

{-------------------------------------------------------------------------------}

procedure TF_Apostar.Edit_importeKeyPress(Sender: TObject; var Key: Char);
const
  CARAC_HABILITADOS = ['0'..'9', #0..#27];//solo numero y tecla borrar
begin
  if not (Key IN CARAC_HABILITADOS) then
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
end;
```

```
{-------------------------------------------------------------------------------}

Procedure TF_Apostar.Listado_GeneralJugadores(RaizJugadores: Lib_Auxiliar.tPos);
var
RD:Type_JUGADOR.tRegDatos;

begin
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin
    ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);

    if (RD.Nick <> 'ADMINISTRADOR') then
    BEGIN
      if not (RD.Bloqueado) then
      begin
        if (RD.Estado) then
        begin
          Grid_ListaJugadores.RowCount:= Grid_ListaJugadores.RowCount + 1;
          {Agrego renglon}
          Grid_ListaJugadores.Cells[0, Grid_ListaJugadores.RowCount-1] := RD.Nick;
          Grid_ListaJugadores.Cells[1, Grid_ListaJugadores.RowCount-1] := RD.Nombre;
          Grid_ListaJugadores.Cells[2, Grid_ListaJugadores.RowCount-1] := RD.Apellido;

          if RD.TipoJugador then
          begin
            Grid_ListaJugadores.Cells[3, Grid_ListaJugadores.RowCount-1] :='Real';
          end
          else
            begin
              Grid_ListaJugadores.Cells[3, Grid_ListaJugadores.RowCount-1]
:='Ficticio';
            end;

          if RD.Bloqueado then
```

```
          begin
            Grid_ListaJugadores.Cells[4, Grid_ListaJugadores.RowCount-1] :='SI';
          end
          else
            begin
              Grid_ListaJugadores.Cells[4, Grid_ListaJugadores.RowCount-1] :='NO';
            end;
          Grid_ListaJugadores.Cells[5, Grid_ListaJugadores.RowCount-1] :=
datetimetostr(RD.UltimaConexion);
        end;
      end;
    END;
    Listado_GeneralJugadores(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores));
    Listado_GeneralJugadores(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores));
  end;
end;




{-------------------------------------------------------------------------------}
//Obtengo el saldo que tiene el jugador mientras apuestas para mostrarlo. Se vera
reflejado en la cuenta del jugador cuando
//es estado sea igual a 2 "no va mas".
Function TF_Apostar.obtenerSaldoActualJugador(clave:tClave):tImporte;
begin
  ME_JUEGO.UltimaJugada(ME_JUEGORULETA,RegJuegoRuleta);


  SaldoAcumulado:= ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE,clave);
  SaldoApuestas_EnPartida:= ME_APUESTAS.SaldoApuestasPartida(ME_APUESTA,clave,
RegJuegoRuleta.NroJugada);


  obtenerSaldoActualJugador:= SaldoAcumulado - SaldoApuestas_EnPartida;
end;


{-------------------------------------------------------------------------------}
```

```
procedure TF_Apostar.ListandoJugadores();
begin
  Lib_Auxiliar.limpiarGrid(Grid_ListaJugadores);
  Lib_Auxiliar.EncabezadosComunJugadores(Grid_ListaJugadores);
  F_Apostar.Grid_ListaJugadores.RowCount:=1;
  F_Apostar.Listado_GeneralJugadores(ME_JUGADORES.Raiz(ME_JUGADOR));
  F_Apostar.Grid_ListaJugadores.FixedRows:=1;
end;
END.
```

3.2.6 Formulario Ficticios

```
unit Form_PanelFicticios;

interface

uses
  ME_JUGADORES,
  Type_JUGADOR,
  ME_CTACTE,
  Type_ALMACEN,
  ME_JUEGO,
  Type_JUEGO,
  ME_APUESTAS,
  Type_APUESTA,
  ME_GANADORES,
  Type_GANADOR,
  Lib_Auxiliar,
  Lib_AuxJuego,
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.StdCtrls, Vcl.Buttons,
  Vcl.ExtCtrls;

type
```

```
  TF_PanelFicticios = class(TForm)
    Panel_Listado: TPanel;
    Panel_grid: TPanel;
    BitBtn1: TBitBtn;
    Panel_CrearFicticios: TPanel;
    Lbl_ActivarFicticios: TLabel;
    Btn_ActivarFicticios: TButton;
    Cbox_CantFicticios: TComboBox;
    Panel_FicticiosActivos: TPanel;
    Label2: TLabel;
    Grid_FicticiosActivos: TStringGrid;
    Panel2: TPanel;
    Label1: TLabel;
    Grid_ListadoFicticios: TStringGrid;
    procedure FormShow(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure Grid_ListadoFicticiosDrawCell(Sender: TObject; ACol,
      ARow: Integer; Rect: TRect; State: TGridDrawState);
    Procedure Listado_GeneralFicticios(RaizJugadores: Lib_Auxiliar.tPos);
    Procedure Listado_FicticiosActivos(RaizJugadores: Lib_Auxiliar.tPos);
    procedure Cbox_CantFicticiosSelect(Sender: TObject);
    procedure mostrarFicticiosEnGrid();
    procedure mostrarFicticiosActivosEnGrid();
    procedure Btn_ActivarFicticiosClick(Sender: TObject);
    procedure Grid_FicticiosActivosDrawCell(Sender: TObject; ACol,
      ARow: Integer; Rect: TRect; State: TGridDrawState);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);

  private
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var
  F_PanelFicticios: TF_PanelFicticios;
  contador:tCantidad;

implementation

uses
  Form_Juego, Form_Login, Form_Apostar;

{$R *.dfm}

{-------------------------------------------------------------------------------}

procedure TF_PanelFicticios.BitBtn1Click(Sender: TObject);
begin
  Form_Juego.F_Juego.Panel_JuegoGral.Enabled:=true;
  F_PanelFicticios.Close;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelFicticios.FormClose(Sender: TObject; var Action: TCloseAction);
var
  cantFictActivos: tCantidad;
begin
  cantFictActivos:= 0;
  F_Juego.Enabled:=true;
  totalFicticios:= ME_JUGADORES.UltFicticio(ME_JUGADOR);
  F_Juego.Lbl_cantFicticios.Caption:= intTostr(totalFicticios);

  Lib_AuxJuego.Cant_FicticiosActivos(ME_JUGADORES.Raiz(ME_JUGADOR), cantFictActivos);
  F_Juego.Lbl_CantFictActivados.Caption:= intTostr(cantFictActivos);
end;

{-------------------------------------------------------------------------------}
```

```
procedure TF_PanelFicticios.FormCreate(Sender: TObject);

begin

  self.Position := poScreenCenter;

  Lbl_ActivarFicticios.Caption:= 'Seleccione ' + #10#13 + 'la cantidad ' + #10#13 + 'de
ficticios ' + #10#13 + 'a activar: ';

end;


{-------------------------------------------------------------------------------}


procedure TF_PanelFicticios.FormShow(Sender: TObject);

var

 i:integer;

begin

  F_Juego.Enabled:=false;

  mostrarFicticiosEnGrid();


  Panel_CrearFicticios.Show;

  Panel_FicticiosActivos.Hide;

end;


{-------------------------------------------------------------------------------}


procedure TF_PanelFicticios.Grid_FicticiosActivosDrawCell(Sender: TObject; ACol,

  ARow: Integer; Rect: TRect; State: TGridDrawState);

begin

  with (Sender as TStringGrid) do

  begin

    if (ARow = 0)

    then

      begin

        Canvas.Brush.Color := clBtnFace;

        Canvas.TextRect(Rect, Rect.Left + (Rect.Right - Rect.Left -
Canvas.TextWidth(Grid_FicticiosActivos.Cells[ACol,ARow]) + 1) div 2, Rect.Top + 2,
Grid_FicticiosActivos.Cells[ACol,ARow]);

      end
```

```
    else
    begin
      Canvas.Font.Color := clblack;
      // Pinto el fondo segun elestado de la Atencion
      if  (ARow mod 2 = 0)
      then
        Canvas.Brush.Color := $00E1FFF9
      else
        Canvas.Brush.Color := $00FFEBDF;
      Canvas.TextRect(Rect, Rect.Left + 2, Rect.Top + 2, cells[acol, arow]);
      Canvas.FrameRect(Rect);
    end;
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelFicticios.Grid_ListadoFicticiosDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
  with (Sender as TStringGrid) do
  begin
    if (ARow = 0)
    then
      begin
        Canvas.Brush.Color := clBtnFace;
        Canvas.TextRect(Rect, Rect.Left + (Rect.Right - Rect.Left -
Canvas.TextWidth(Grid_ListadoFicticios.Cells[ACol,ARow]) + 1) div 2, Rect.Top + 2,
Grid_ListadoFicticios.Cells[ACol,ARow]);
      end
    else
    begin
      Canvas.Font.Color := clblack;
      // Pinto el fondo segun elestado de la Atencion
      if  (ARow mod 2 = 0)
      then
```

```
        Canvas.Brush.Color := $00E1FFF9
      else
        Canvas.Brush.Color := $00FFEBDF;
      Canvas.TextRect(Rect, Rect.Left + 2, Rect.Top + 2, cells[acol, arow]);
      Canvas.FrameRect(Rect);
    end;
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelFicticios.Btn_ActivarFicticiosClick(Sender: TObject);
var
  cantFict_Activar:tCantidad;
begin
  if Cbox_CantFicticios.ItemIndex <> Lib_Auxiliar.PosNula then
  begin
    cantFict_Activar:= strToint(Cbox_CantFicticios.Text);
    Lib_AuxJuego.Activar_JugadoresFicticios(Form_Login.ME_JUGADOR, cantFict_Activar);

    Panel_FicticiosActivos.Show;
    mostrarFicticiosActivosEnGrid();

    Cbox_CantFicticios.ItemIndex:= Lib_Auxiliar.PosNula;
    Panel_CrearFicticios.Hide;
    Form_Juego.F_Juego.Spbtn_activarFicticios.Enabled:=false;
  end
  else
    MessageDlg('Seleccione cantidad de jugadores a activar!',mtWarning , [mbOK], 0)
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelFicticios.Cbox_CantFicticiosSelect(Sender: TObject);
var
```

```
  op, cantCrear:integer;
  numFicticio:tCantidad;
  msjMostrar:string;
  cantFict_Activar, cantFict_Activos:tCantidad;
begin
  op:= strToint(Cbox_CantFicticios.Text);
  numFicticio:= ME_JUGADORES.UltFicticio(ME_JUGADOR);

  if (op > numFicticio)then
  begin
    cantCrear:= op - numFicticio;
    if (numFicticio <> 0) then
      msjMostrar:= 'Solo hay ' + intTostr(numFicticio) + ' jugadores ficticios
creados.' + #13 + #13 +'¿Quiere crear ' +intTostr(cantCrear)+ ' ficticios mas?'
    else
      msjMostrar:= 'No hay jugadores ficticios creados ¿Quiere crearlos?';
    if Lib_Auxiliar.Mensaje_Confirmacion(msjMostrar) then
    begin
      Lib_AuxJuego.Insertar_JugadoresFicticios(ME_JUGADOR, cantCrear);
      mostrarFicticiosEnGrid();
      cantFict_Activar:= strToint(Cbox_CantFicticios.Text);
      if Lib_Auxiliar.Mensaje_Confirmacion('¿CONFIRMA QUE QUIERE ACTIVAR
'+intTostr(cantFict_Activar)+' JUGADORES FICTICIOS?') then
      begin
        Lib_AuxJuego.Activar_JugadoresFicticios(Form_Login.ME_JUGADOR,
cantFict_Activar);
        Panel_FicticiosActivos.Show;
        mostrarFicticiosActivosEnGrid();
        Cbox_CantFicticios.ItemIndex:= Lib_Auxiliar.PosNula;
        Panel_CrearFicticios.Hide;
        Form_Juego.F_Juego.Spbtn_activarFicticios.Enabled:=false;
      end;
    end;
  end;
end;
```

```
{<-------------------------------------------------------------------------------}

Procedure TF_PanelFicticios.Listado_GeneralFicticios(RaizJugadores: Lib_Auxiliar.tPos);
var
RD:Type_JUGADOR.tRegDatos;
begin
  if (RaizJugadores<>Lib_Auxiliar.PosNula) then
  begin

          ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);

          if not (RD.TipoJugador)then ///  false: entonces es un ficticio
          begin
            Grid_ListadoFicticios.RowCount:= Grid_ListadoFicticios.RowCount + 1;
            {Agrego renglon}
            Grid_ListadoFicticios.Cells[0, Grid_ListadoFicticios.RowCount-1] :=
RD.Nick;
            Grid_ListadoFicticios.Cells[1, Grid_ListadoFicticios.RowCount-1] :=
RD.Nombre;
            Grid_ListadoFicticios.Cells[2, Grid_ListadoFicticios.RowCount-1] :=
RD.Apellido;

            if RD.TipoJugador then
            begin
              Grid_ListadoFicticios.Cells[3, Grid_ListadoFicticios.RowCount-1]
:='Real';
            end
            else
              begin
                Grid_ListadoFicticios.Cells[3, Grid_ListadoFicticios.RowCount-1]
:='Ficticio';
              end;

            if RD.Estado then
            begin
```

```
              Grid_ListadoFicticios.Cells[4, Grid_ListadoFicticios.RowCount-1]
:='ACTIVO';

            end

            else

              begin

                Grid_ListadoFicticios.Cells[4, Grid_ListadoFicticios.RowCount-1] :='NO
ACTIVO';

              end;


            saldoActual:= F_Apostar.obtenerSaldoActualJugador(RD.Nick);


            Grid_ListadoFicticios.Cells[5, Grid_ListadoFicticios.RowCount-1] :=
intTostr(saldoActual);

            Grid_ListadoFicticios.Cells[6, Grid_ListadoFicticios.RowCount-1] :=
datetimetostr(RD.Alta);


          end;

          Listado_GeneralFicticios(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores));

          Listado_GeneralFicticios(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores));

  end;

end;


{-------------------------------------------------------------------------------}


Procedure TF_PanelFicticios.Listado_FicticiosActivos(RaizJugadores: Lib_Auxiliar.tPos);


var

RD:Type_JUGADOR.tRegDatos;

begin

  if (RaizJugadores<>Lib_Auxiliar.PosNula) then

  begin

          ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR,RD,RaizJugadores);


          if not (RD.TipoJugador)then ///  false: entonces es un ficticio

          begin

            if (RD.Estado) then ///true: entonces esta activo
```

```
            begin
                Grid_FicticiosActivos.RowCount:= Grid_FicticiosActivos.RowCount + 1;
                {Agrego renglon}
                Grid_FicticiosActivos.Cells[0, Grid_FicticiosActivos.RowCount-1] :=
RD.Nick;
                Grid_FicticiosActivos.Cells[1, Grid_FicticiosActivos.RowCount-1] :=
RD.Nombre;
                Grid_FicticiosActivos.Cells[2, Grid_FicticiosActivos.RowCount-1] :=
RD.Apellido;


                if RD.TipoJugador then
                begin
                  Grid_FicticiosActivos.Cells[3, Grid_FicticiosActivos.RowCount-1]
:='Real';
                end
                else
                  begin
                    Grid_FicticiosActivos.Cells[3, Grid_FicticiosActivos.RowCount-1]
:='Ficticio';
                  end;


                 if RD.Estado then
                 begin
                    Grid_FicticiosActivos.Cells[4, Grid_FicticiosActivos.RowCount-1]
:='ACTIVO';
                 end
                 else
                  begin
                    Grid_FicticiosActivos.Cells[4, Grid_FicticiosActivos.RowCount-1]
:='NO ACTIVO';
                  end;


                saldoActual:= F_Apostar.obtenerSaldoActualJugador(RD.Nick);


                Grid_FicticiosActivos.Cells[5, Grid_FicticiosActivos.RowCount-1] :=
intTostr(saldoActual);
                Grid_FicticiosActivos.Cells[6, Grid_FicticiosActivos.RowCount-1] :=
datetimetostr(RD.Alta);
```

```
            end;
          end;
          Listado_FicticiosActivos(ME_JUGADORES.ProximoIzq(ME_JUGADOR,RaizJugadores));
          Listado_FicticiosActivos(ME_JUGADORES.ProximoDer(ME_JUGADOR,RaizJugadores));
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelFicticios.mostrarFicticiosActivosEnGrid();

begin
  Lib_Auxiliar.limpiarGrid(Grid_FicticiosActivos);
  Lib_Auxiliar.EncabezadosComunFicticios(Grid_FicticiosActivos);

  Grid_FicticiosActivos.RowCount:=1;
  Listado_FicticiosActivos(ME_JUGADORES.Raiz(ME_JUGADOR));
  Grid_FicticiosActivos.FixedRows:=1;
end;

{-------------------------------------------------------------------------------}
procedure TF_PanelFicticios.mostrarFicticiosEnGrid();
var
  numFicticio: Lib_Auxiliar.tCantidad;
begin
  Lib_Auxiliar.limpiarGrid(Grid_ListadoFicticios);
  Lib_Auxiliar.EncabezadosComunFicticios(Grid_ListadoFicticios);
  numFicticio:= ME_JUGADORES.UltFicticio(ME_JUGADOR);
  if (numFicticio>0) then
  begin
    Grid_ListadoFicticios.RowCount:=1;
    Listado_GeneralFicticios(ME_JUGADORES.Raiz(ME_JUGADOR));
    Grid_ListadoFicticios.FixedRows:=1;
  end;
end;
```

```
END.
```

3.2.7 Formulario Premios

```
unit Form_Premios;


interface


uses
  ME_JUGADORES,
  Type_JUGADOR,
  ME_CTACTE,
  Type_ALMACEN,
  ME_JUEGO,
  Type_JUEGO,
  ME_APUESTAS,
  Type_APUESTA,
  ME_GANADORES,
  Type_GANADOR,
  Lib_Auxiliar,
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Buttons, Vcl.Grids,
  Vcl.ExtCtrls;


type
  TF_Premios = class(TForm)
    Panel_Premiados: TPanel;
    Label7: TLabel;
    BitBtn1: TBitBtn;
    Grid_premiados: TStringGrid;
    procedure FormShow(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
```

```
    procedure FormCreate(Sender: TObject);
    procedure Grid_premiadosDrawCell(Sender: TObject; ACol, ARow: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_Premios: TF_Premios;

implementation

Uses
  Form_Juego, Form_login;



{$R *.dfm}

{-------------------------------------------------------------------------------}

procedure TF_Premios.BitBtn1Click(Sender: TObject);
begin
  Form_Premios.F_Premios.Close;
  Form_Juego.F_Juego.Panel_JuegoGral.Enabled:=true;
end;

{-------------------------------------------------------------------------------}

procedure TF_Premios.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Form_Juego.F_Juego.Panel_JuegoGral.Enabled:= true;
end;
```

```
{-------------------------------------------------------------------------------}

procedure TF_Premios.FormCreate(Sender: TObject);
begin
  self.Position := poScreenCenter;
end;

{-------------------------------------------------------------------------------}

procedure TF_Premios.FormShow(Sender: TObject);
begin
  Form_Juego.F_Juego.Panel_JuegoGral.Enabled:=false;
end;

{-------------------------------------------------------------------------------}

procedure TF_Premios.Grid_premiadosDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
with (Sender as TStringGrid) do
  begin
    if (ARow = 0)
    then
      begin
        Canvas.Brush.Color := clBtnFace;
        Canvas.TextRect(Rect, Rect.Left + (Rect.Right - Rect.Left -
Canvas.TextWidth(Grid_premiados.Cells[ACol,ARow]) + 1) div 2, Rect.Top + 2,
Grid_premiados.Cells[ACol,ARow]);
      end
    else
    begin
      Canvas.Font.Color := clblack;
      if  (ARow mod 2 = 0)
      then
        Canvas.Brush.Color := $00E1FFF9
```

```
      else
        Canvas.Brush.Color := $00FFEBDF;
      Canvas.TextRect(Rect, Rect.Left + 2, Rect.Top + 2, cells[acol, arow]);
      Canvas.FrameRect(Rect);
    end;
  end;
end;
END.
```

3.2.8 Formulario Listados

```
unit Form_Listados;

interface

uses
  ME_JUGADORES,
  Type_JUGADOR,
  ME_CTACTE,
  Type_ALMACEN,
  ME_JUEGO,
  Type_JUEGO,
  ME_APUESTAS,
  Type_APUESTA,
  ME_GANADORES,
  Type_GANADOR,
  Lib_Auxiliar,
  Lib_AuxJuego,
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.ExtCtrls, Vcl.StdCtrls, Vcl.Buttons,
  Vcl.Grids;

type
  TF_Listados = class(TForm)
```

```
    Panel_gralListados: TPanel;
    BitBtn1: TBitBtn;
    Grid_ListadoxFiltro: TStringGrid;
    Panel_filtro: TPanel;
    Lbl_tittle: TLabel;
    Panel_Nomenclador: TPanel;
    Lbl_Nomenclador: TLabel;
    Cbox_Nomenclador: TComboBox;
    Panel_CantJugadores: TPanel;
    Lbl_cantJugadores: TLabel;
    Cbox_Cantidad: TComboBox;
    Panel_nickJugador: TPanel;
    Edit_NickJugador: TEdit;
    Btn_aceptar: TBitBtn;
    Lbl_NickJugador: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Lbl_NameJugador: TLabel;
    Lbl_FechaIngreso: TLabel;
    Lbl_PremiosAcumulados: TLabel;
    procedure FormShow(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Cbox_CantidadSelect(Sender: TObject);
    procedure Cbox_NomencladorSelect(Sender: TObject);
    procedure Btn_aceptarClick(Sender: TObject);
    procedure Grid_ListadoxFiltroDrawCell(Sender: TObject; ACol, ARow: Integer;
      Rect: TRect; State: TGridDrawState);


  private
    { Private declarations }
  public
```

```
    { Public declarations }
  end;

var
  F_Listados: TF_Listados;
  opListado: integer;

implementation

uses
  Form_croupier, Form_Jugadores, Form_login;

{$R *.dfm}

{-------------------------------------------------------------------------------}

procedure TF_Listados.BitBtn1Click(Sender: TObject);
begin
  F_Listados.Close;
end;

{-------------------------------------------------------------------------------}

procedure TF_Listados.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  if AdminLog then
    F_Croupier.Enabled:= true
  else
  begin
    F_Jugadores.Enabled:= true;
    Lib_Auxiliar.limpiarGrid(Grid_ListadoxFiltro);

    if (OpListado = 2) then
      Cbox_Nomenclador.ItemIndex:= -1;
```

```
    if (OpListado = 3) then
      Cbox_Cantidad.ItemIndex:= -1;

    if (OpListado = 5) then
      Edit_NickJugador.Clear;
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Listados.FormCreate(Sender: TObject);
begin
  self.Position := poScreenCenter;

  Lbl_Nomenclador.Caption:= 'Seleccione nomenclador' + #10#13 + 'para filtrar.';
  Lbl_cantJugadores.Caption:= 'Seleccione cantidad' + #10#13 + 'para filtrar.';
  Lbl_NickJugador.Caption:= 'INGRESE NICK DE UN JUGADOR';
end;


{-------------------------------------------------------------------------------}

procedure TF_Listados.FormShow(Sender: TObject);
begin
  if AdminLog then
    F_Croupier.Enabled:= false
  else
    F_Jugadores.Enabled:= false;

  Lib_Auxiliar.limpiarGrid(Grid_ListadoxFiltro);
  Grid_ListadoxFiltro.RowCount:= 1;
  Panel_Nomenclador.Hide;
  Panel_CantJugadores.Hide;
  Panel_nickJugador.Hide;
  Cbox_Nomenclador.ItemIndex:= Lib_Auxiliar.PosNula;
```

```
    case (OpListado) of
    1: begin //Listado General de Usuarios (jugadores)
      Lbl_tittle.Caption:='Listado general de jugadores';
      Lib_Auxiliar.EncabezadoDetalladoJugadores(Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.RowCount:=1;
      Lib_AuxJuego.ListadoGeneral_JugadoresEnGrid(ME_JUGADORES.Raiz(ME_JUGADOR),
Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.FixedRows:=1;
    end;
    2:begin ////jugadores que ganaron determinado premio seleccionando nomenclador
      Lbl_tittle.Caption:='Jugadores que ganaron determinado premio';
      Panel_Nomenclador.Show;
      Cbox_Nomenclador.ItemIndex:= Lib_Auxiliar.PosNula;
      Lib_Auxiliar.EncabezadoGanadoresXPremio(Grid_ListadoxFiltro);
    end;
    3:begin //<n> jugadores que más dinero ganaron a través del tiempo.
      Lbl_tittle.Caption:='Jugadores que más dinero ganaron a través del tiempo';
      Panel_CantJugadores.Show;
      Cbox_Cantidad.ItemIndex:= Lib_Auxiliar.PosNula;
      Lib_Auxiliar.EncabezadoJugadoresMasGanaron(Grid_ListadoxFiltro);
    end;
    4:begin //jugadores que nunca ganaron apuestas.
      Lbl_tittle.Caption:='Jugadores que nunca ganaron apuestas';
      Lib_Auxiliar.EncabezadoDetalladoJugadores(Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.RowCount:=1;
      Lib_AuxJuego.ListadoJugadoresNuncaGanaron(ME_JUGADORES.Raiz(ME_JUGADOR),
Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.FixedRows:=1;
    end;
    5:begin //Ficha de un jugador. Ingresando nick podra listar su cta cte con todos
sus movimientos ordenados por fecha y hora de manera cronologia ascendente
      Lbl_tittle.Caption:='Ficha de un Jugador';
      Lib_Auxiliar.EncabezadoFiltroCtaCte(Grid_ListadoxFiltro);
      Panel_nickJugador.show;
      Lbl_NameJugador.Caption:= '';
```

```
      Lbl_FechaIngreso.Caption:= '';
      Lbl_PremiosAcumulados.Caption:= '';
      Edit_NickJugador.Clear;
      Edit_NickJugador.SetFocus;
    end;  //Listado general de todas las partidas jugadas hasta el momento
    6:begin
      Lbl_tittle.Caption:='Listado general de partidas';
      Lib_Auxiliar.EncabezadoListadoPartidas(Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.RowCount:=1;
      Lib_AuxJuego.ListadoPartidas(Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.FixedRows:=1;
    end;
    7:begin //Listado de arbol balanceado de Jugadores
      Lbl_tittle.Caption:='Listado Árbol Binario AVL de Jugadores';
      EncabezadoArbolJugadores(Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.RowCount:=1;
      Lib_AuxJuego.ListadoArbolJugadores(ME_JUGADORES.Raiz(ME_JUGADOR),
Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.FixedRows:=1;
    end;
    8:begin //Listado de arbol balanceado de Ganadores
      Lbl_tittle.Caption:='Listado Árbol Trinario AVL de Ganadores';
      Lib_Auxiliar.EncabezadoArbolJugadores(Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.RowCount:=1;
      Lib_AuxJuego.ListadoArbolGanadores(ME_GANADORES.Raiz(ME_GANADOR),
Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.FixedRows:=1;
    end;
    9:begin //Listado de jugadores que se encuentran activos (activos = logueados)
      Lbl_tittle.Caption:='Listado Jugadores Activos';
      Lib_Auxiliar.EncabezadoDetalladoJugadores(Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.RowCount:=1;
      Lib_AuxJuego.Listado_JugadoresActivos(ME_JUGADORES.Raiz(ME_JUGADOR),
Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.FixedRows:=1;
    end;
```

```
    end;//case
end;

{-------------------------------------------------------------------------------}

procedure TF_Listados.Btn_aceptarClick(Sender: TObject);
var
  Nick: tClave;
  cantPremios: tCantidad;
  Monto: tImporte;
begin
  Nick:= UpperCase(Edit_NickJugador.Text);
  cantPremios:= 0;
  Monto:= 0;

   if not (ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR,PosJugador, Nick)) then
   begin //Nick incorrecto
    MessageDlg('Nick ingresado no existe!',mtError , [mbOK], 0);
    Edit_NickJugador.Clear;
    Edit_NickJugador.SetFocus;
   end
   else
    begin
      ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regjugador, PosJugador);
      Lbl_NameJugador.Caption:= regjugador.Nick;
      Lbl_FechaIngreso.Caption:= DatetoStr(regJugador.Alta);

      //Llamo a procedimiento recursivo que acumula en monto y cantidad los premios que
gano
      Lib_AuxJuego.Premios_AcumuladosXJugador(ME_GANADORES.Raiz(ME_GANADOR), Nick,
cantPremios, Monto);
      Lbl_PremiosAcumulados.Caption:= intTostr(cantPremios) + ' PREMIOS POR '
+intTostr(monto);

      //Llamo a procedimiento recursivo para que muestre los movimientos de ctacte de
un jugador
```

```
      Grid_ListadoxFiltro.RowCount:=1;

Lib_AuxJuego.ListadoCtaCte_deUnJugador(ME_CTACTE.PrimeroLdParcial(ME_CUENTACORRIENTE,Ni
ck), Nick, Grid_ListadoxFiltro);
      Grid_ListadoxFiltro.FixedRows:=1;


      Edit_NickJugador.Clear;
      Edit_NickJugador.SetFocus;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Listados.Grid_ListadoxFiltroDrawCell(Sender: TObject; ACol,
  ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
  with (Sender as TStringGrid) do
    begin
      if (ARow = 0)
      then
        begin
          Canvas.Brush.Color := clBtnFace;
          Canvas.TextRect(Rect, Rect.Left + (Rect.Right - Rect.Left -
Canvas.TextWidth(Grid_ListadoxFiltro.Cells[ACol,ARow]) + 1) div 2, Rect.Top + 2,
Grid_ListadoxFiltro.Cells[ACol,ARow]);
        end
      else
      begin
        Canvas.Font.Color := clblack;
        if  (ARow mod 2 = 0)
        then
          Canvas.Brush.Color := $00E1FFF9
        else
          Canvas.Brush.Color := $00FFEBDF;
        Canvas.TextRect(Rect, Rect.Left + 2, Rect.Top + 2, cells[acol, arow]);
        Canvas.FrameRect(Rect);
```

```
      end;
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_Listados.Cbox_CantidadSelect(Sender: TObject);
var
  num:tCantidad;
  importe: tImporte;
  cantFilas, cantGanadores: integer;
begin
  importe:= 0;
  num:= strToint(Cbox_Cantidad.Text);
  cantFilas:= 1;

  if not (ME_GANADORES.MeVacio_Ganadores(ME_GANADOR)) then
    begin
      cantGanadores:= ME_GANADORES.Cantidad_Ganadores(ME_GANADOR);
      if (num<=cantGanadores) then
      begin
      Grid_ListadoxFiltro.RowCount:=1;
      Lib_AuxJuego.ListadoMasPremiados(ME_GANADORES.Raiz(ME_GANADOR), importe,
Grid_ListadoxFiltro, cantFilas);
      Grid_ListadoxFiltro.FixedRows:=1;

      Grid_ListadoxFiltro.RowCount:=1;
      Lib_AuxJuego.ReordenarMasPremiados(Grid_ListadoxFiltro, num, cantFilas);
      Grid_ListadoxFiltro.FixedRows:=1;
      end
      else
        MessageDlg('TENGA PRESENTE QUE : ' +#13+#13 + 'La máxima cantidad a flitrar es
de ' + intTostr(cantGanadores) + ' ganadores!', mtWarning, [mbOK], 0);
    end;
end;
```

```
{-------------------------------------------------------------------------------}


procedure TF_Listados.Cbox_NomencladorSelect(Sender: TObject);

var

  nomenclador: tNomenclador;

  cantGanadores: tCantidad;

begin

  nomenclador:= Cbox_Nomenclador.Text;

  cantGanadores:= 0;


  Lib_AuxJuego.CantGanadores_xNomenclador(ME_GANADORES.Raiz(ME_GANADOR), nomenclador,
cantGanadores);


  if (cantGanadores>0) then

  begin

    Grid_ListadoxFiltro.RowCount:=1;

    Lib_AuxJuego.ListadoPremiados_Nomenclador(ME_GANADORES.Raiz(ME_GANADOR),
nomenclador, Grid_ListadoxFiltro);

    Grid_ListadoxFiltro.FixedRows:=1;

  end

  else

    MessageDlg('NO HAY GANADORES : ' +#13+#13 + 'No hay ganadores con el nomenclador '
+ nomenclador, mtInformation, [mbOK], 0);

end;

END.
```

3.2.9 Formulario Premio/Bloqueo

```
//Formulario que segun eliga el croupier puede ser para bloquear/ desbloquear u
obsequiar crédito

unit Form_PanelCaseButtons;
```

```
interface


uses
  ME_JUGADORES,
  Type_JUGADOR,
  ME_CTACTE,
  Type_ALMACEN,
  ME_JUEGO,
  Type_JUEGO,
  ME_APUESTAS,
  Type_APUESTA,
  ME_GANADORES,
  Type_GANADOR,
  Lib_Auxiliar,
  Lib_AuxJuego,
  Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
  Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Grids, Vcl.StdCtrls, Vcl.Buttons,
  Vcl.ExtCtrls;


type
  TF_PanelCaseButtons = class(TForm)
    Panel_Listado: TPanel;
    Panel_RegalarCredito: TPanel;
    btn_okRegalarCredito: TBitBtn;
    Panel_grid: TPanel;
    Edit_cantCredito: TEdit;
    Lbl_cantCredito: TLabel;
    Lbl_bloquearDesbloquear: TLabel;
    Lbl_regalarCred: TLabel;
    Grid_ListadoJugadores: TStringGrid;
    Label1: TLabel;
    Edit_Jugador: TEdit;
    BitBtn1: TBitBtn;
```

```
    Panel_LockUnlock: TPanel;
    Label3: TLabel;
    btn_okLockUnlock: TBitBtn;
    Edit_NickJugador: TEdit;
    procedure limpiarGrid();
    procedure MostrarEnGridJugadores();
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure Grid_ListadoJugadoresDrawCell(Sender: TObject; ACol,
      ARow: Integer; Rect: TRect; State: TGridDrawState);
    procedure Grid_ListadoJugadoresClick(Sender: TObject);
    procedure btn_okRegalarCreditoClick(Sender: TObject);
    procedure Edit_cantCreditoKeyPress(Sender: TObject; var Key: Char);
    procedure btn_okLockUnlockClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  F_PanelCaseButtons: TF_PanelCaseButtons;

implementation

uses
  Form_login, Form_Juego, Form_Croupier;

{$R *.dfm}

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.BitBtn1Click(Sender: TObject);
```

```
begin
  Form_Juego.F_Juego.Panel_JuegoGral.Enabled:=true;
  F_PanelCaseButtons.Close;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  if not (F_Juego.Enabled) then
    F_Juego.Enabled:= true;

  if not (F_Croupier.Enabled) then
    F_Croupier.Enabled:= true;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.FormCreate(Sender: TObject);
begin
  self.Position := poScreenCenter;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.FormShow(Sender: TObject);
begin
  if (F_Juego.Enabled) then
    F_Juego.Enabled:= false;

  if (F_Croupier.Enabled) then
    F_Croupier.Enabled:= false;

  MostrarEnGridJugadores();
```

```
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.Grid_ListadoJugadoresClick(Sender: TObject);
begin
    if Grid_ListadoJugadores.Row <> 0
    then
    begin
      if (Panel_RegalarCredito.Enabled) then //Croupier va a regalar credito
      begin
      Edit_Jugador.Text := Grid_ListadoJugadores.Cells[2,Grid_ListadoJugadores.Row];
        if (Edit_Jugador.Text<>'') then
        begin
          Edit_cantCredito.Enabled:=true;
          Edit_cantCredito.SetFocus;
        end
        else
          Edit_cantCredito.Enabled:= false;
      end
      else //Croupier va a bloquear/ desbloquear
        Edit_NickJugador.Text :=
Grid_ListadoJugadores.Cells[2,Grid_ListadoJugadores.Row];
    end;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.Grid_ListadoJugadoresDrawCell(Sender: TObject;
  ACol, ARow: Integer; Rect: TRect; State: TGridDrawState);
begin
with (Sender as TStringGrid) do
  begin
    if (ARow = 0)
    then
```

```
      begin
        Canvas.Brush.Color := clBtnFace;
        Canvas.TextRect(Rect, Rect.Left + (Rect.Right - Rect.Left -
Canvas.TextWidth(Grid_ListadoJugadores.Cells[ACol,ARow]) + 1) div 2, Rect.Top + 2,
Grid_ListadoJugadores.Cells[ACol,ARow]);
      end
    else
    begin
      Canvas.Font.Color := clblack;
      if  (ARow mod 2 = 0)
      then
        Canvas.Brush.Color := $00E1FFF9
      else
        Canvas.Brush.Color := $00FFEBDF;
      Canvas.TextRect(Rect, Rect.Left + 2, Rect.Top + 2, cells[acol, arow]);
      Canvas.FrameRect(Rect);
    end;
  end;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.limpiarGrid();
var
  i:integer;
begin
  with Grid_ListadoJugadores  do
    for i := 0 to RowCount - 1 do
    Rows[i].Clear;
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.btn_okLockUnlockClick(Sender: TObject);
var
  sMsj:string;
```

```
begin
  ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR, posJugador, Edit_NickJugador.Text);
  ME_JUGADORES.CapturarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

  if (regJugador.Bloqueado) then
  begin
    sMsj:= 'desbloquear';
    regJugador.Nick:= regJugador.Nick;
    regJugador.Contrasenia:= regJugador.Contrasenia;
    regJugador.Nombre:= regJugador.Nombre;
    regJugador.Apellido:= regJugador.Apellido;
    regJugador.Alta:= regJugador.Alta;
    regJugador.UltimaConexion:= Now();
    regJugador.Bloqueado:= false; // desbloqueo jugador
    regJugador.Estado:= regJugador.Estado; // si lo desbloqueo, le doy el estado que ya
tiene (desactivo)
  end
  else
    begin
      sMsj:= 'bloquear';
      regJugador.Nick:= regJugador.Nick;
      regJugador.Contrasenia:= regJugador.Contrasenia;
      regJugador.Nombre:= regJugador.Nombre;
      regJugador.Apellido:= regJugador.Apellido;
      regJugador.Alta:= regJugador.Alta;
      regJugador.UltimaConexion:= Now();
      regJugador.Bloqueado:= true; // bloqueo jugador
      regJugador.Estado:= false; // si lo bloqueo, lo desconecto
    end;

  if (Edit_NickJugador.Text<>'') then
  begin
    if Lib_Auxiliar.Mensaje_Confirmacion('¿Esta seguro que quiere ' +sMsj+ ' a '
+Edit_NickJugador.Text + ' ?') then
    begin
```

```
      ME_JUGADORES.BuscarInfoME_Jugadores(ME_JUGADOR, posJugador,
Edit_NickJugador.Text);
      ME_JUGADORES.ModificarInfoME_Jugadores(ME_JUGADOR, regJugador, posJugador);

      //Elimino las apuestas efectuadas del jugador recientemente bloquado
      EliminarApuestasUnJugador(Edit_NickJugador.Text);

      MostrarEnGridJugadores();
    end;
  end
  else
    MessageDlg('Debe seleccionar un jugador de la lista!!',mtWarning , [mbOK], 0);
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.btn_okRegalarCreditoClick(Sender: TObject);
begin
  if (Edit_Jugador.Text<>'') then
  begin
    if (Edit_cantCredito.Text<>'') then
    begin
      if (strtoint(Edit_cantCredito.Text) >= ME_JUEGO.ApuestaMinima(ME_JUEGORULETA))
and (strtoint(Edit_cantCredito.Text) <= ME_JUEGO.ApuestaMaxima(ME_JUEGORULETA)) then
      begin
        if Lib_Auxiliar.Mensaje_Confirmacion('¿Seguro que quiere obsequiar $'+
Edit_cantCredito.Text +' a ' + Edit_Jugador.Text + ' ?') then
        begin
          regCtaCte.Nick:= Edit_Jugador.Text;
          regCtaCte.FechaHora:= Now();
          regCtaCte.Concepto:= Type_ALMACEN.tipoConceptos.Credito_Regalado;
          regCtaCte.Debe:= strToint(Edit_cantCredito.Text);
          regCtaCte.haber:= 0;
          regCtaCte.saldo:= regCtaCte.Debe +
ME_CTACTE.SaldoAcumulado(ME_CUENTACORRIENTE, Edit_Jugador.Text);
          ME_CTACTE.InsertarAlmacen(ME_CUENTACORRIENTE, RegCtaCte);
```

```
          messagedlg('Crédito obsequiado con éxito!', mtInformation, [mbOk], 0);
          Edit_Jugador.Clear;
          Edit_cantCredito.Clear;
          MostrarEnGridJugadores();
        end;
      end
      else
        begin
          MessageDlg('El crédito minimo debe ser de $ '
+intTostr(ME_JUEGO.ApuestaMinima(ME_JUEGORULETA))+
                ', y el crédito máximo debe ser de $ '
+intTostr(ME_JUEGO.ApuestaMaxima(ME_JUEGORULETA)),mtWarning, [mbOK], 0);
          Edit_cantCredito.Clear;
          Edit_cantCredito.SetFocus;
        end;

    end
    else
      begin
        MessageDlg('Debe ingresar crédito a obsequiar!',mtError , [mbOK], 0);
        Edit_cantCredito.SetFocus;
      end;
  end
  else
    MessageDlg('Debe seleccionar un jugador de la lista!',mtError , [mbOK], 0);
end;

{-------------------------------------------------------------------------------}

procedure TF_PanelCaseButtons.Edit_cantCreditoKeyPress(Sender: TObject;
  var Key: Char);
const
  CARAC_HABILITADOS = ['0'..'9', #0..#27];//solo numero y tecla borrar
begin
  if not (Key IN CARAC_HABILITADOS) then
    Key:=#0; //la tecla vale null si presiona caracteres no habilitados
```

```
end;


{-------------------------------------------------------------------------------}


procedure TF_PanelCaseButtons.MostrarEnGridJugadores();

begin

  F_PanelCaseButtons.Show;

  Lib_Auxiliar.limpiarGrid(Grid_ListadoJugadores);

  Lib_Auxiliar.EncabezadoDetalladoJugadores(Grid_ListadoJugadores);

  Grid_ListadoJugadores.RowCount:=1;

  Lib_AuxJuego.ListadoGeneral_JugadoresEnGrid(ME_JUGADORES.Raiz(ME_JUGADOR),
Grid_ListadoJugadores);

  Grid_ListadoJugadores.FixedRows:=1;

end;


END.
```