Trabajo Final - Programación 2

Integración de Métodos - Cursado 2020

Prof. Claudio Álvarez

Tecnicatura Sup. en Análisis de Sistemas

Instituto Superior Juan XXIII

Acosta Martín - Nro. Libreta 26475

DOCUMENTACIÓN DE PROGRAMA

El presente informe se realizó a modo de documentación del programa final de la materia Programación 2, correspondiente al cursado del año 2020.

El programa ha sido desarrollado haciendo uso del entorno de desarrollo de software Delphi (Versión 7) .

La ruta de almacenamiento de archivos se toma automáticamente al ejecutar el software, en la carpeta donde esté guardado el programa, dentro de la carpeta “Archivos”.

ÍNDICE

Introducción

A grandes rasgos el programa fue desarrollado haciendo uso de la interfaz gráfica que ofrece el IDE Delphi7, haciendo uso de los componentes que trae por defecto e instalando un componente extra llamado “QuickReport” para realizar reportes en papel de un par de listados (Facturas y listados de artículos en stock).

En tanto al almacenamiento de la información, los datos son guardados en archivos creados automáticamente la primera vez que se ejecuta el programa, en el directorio donde se encuentre éste, dentro de la carpeta “Archivos”.

Con respecto a las librerías operacionales, se realizaron en total 5 librerías correspondientes a 5 TDAs (Tipos de Datos Abstractos). Éstos son estructuras de datos utilizadas a modo de índice para el ordenamiento de los datos de cada entidad.

Tenemos entonces, las estructuras de “Índice”, “Lista (doblemente enlazada)”, “Pila”, “Cola”, y por último, “ABB (Árbol Binario de Búsqueda)”. Cada una de estas estructuras están implementadas en una unidad distinta, cada una con sus tipos de datos y funcionalidades.

Las entidades que entran en juego en el funcionamiento del programa son las de “Cliente”, “Rubro”, “Artículo”, “Cuentas Corriente”, y las facturas (ventas), constituídas por un “Comprobante” y uno o varios “Detalle/s”.

Cada una de éstas entidades cuenta con su librería operacional de datos (desde donde se hace el alta y guardado de archivo de datos) y su ordenamiento desde una librería en particular. Estas son:

“Clientes” y “Rubros” = LO_DobleEnlace (Lista Doblemente Enlazada)

“Artículos” = LO_ABB (Arbol Binario de Búsqueda)

“Comprobantes” = LO_Cola

“Detalles” = LO_Pila

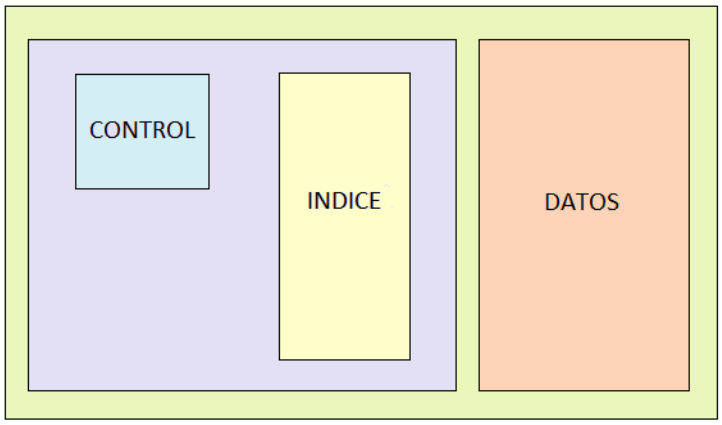
“Cuentas Corrientes” = LO_Pila

Al ejecutarse el programa pedirá ingresar los valores de porcentajes a tolerancia de desequilibrio para el árbol, una vez ingresados se crearán todos los datos en el directorio antes mencionado, y las entidades son controladas en base a 5 variables definidas en la Unit1. Estas son ME_Clientes, ME_Rubros, ME_Articulos, ME_Ventas, ME_CuentasCorrientes.

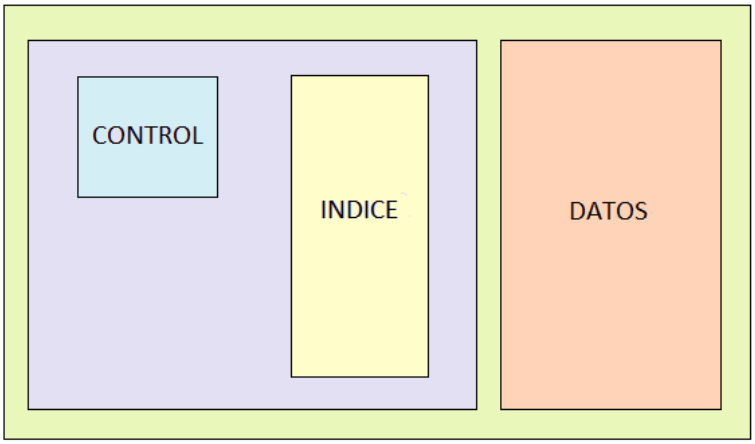
Gráficos

Para una mejor comprensión de las entidades del programa tenemos los siguientes gráficos.

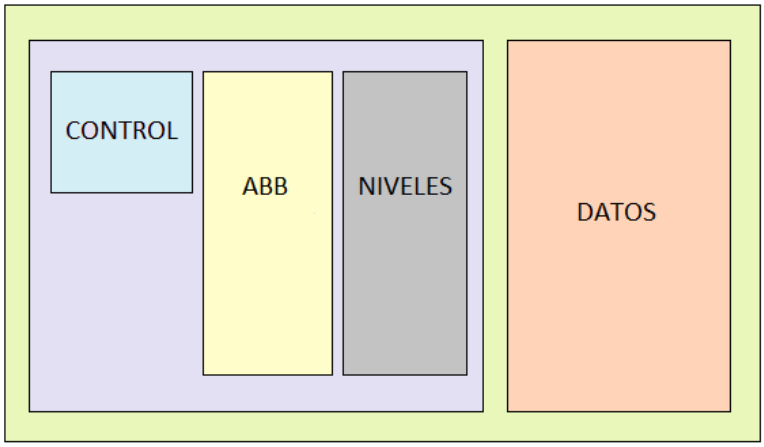
ME_Clientes



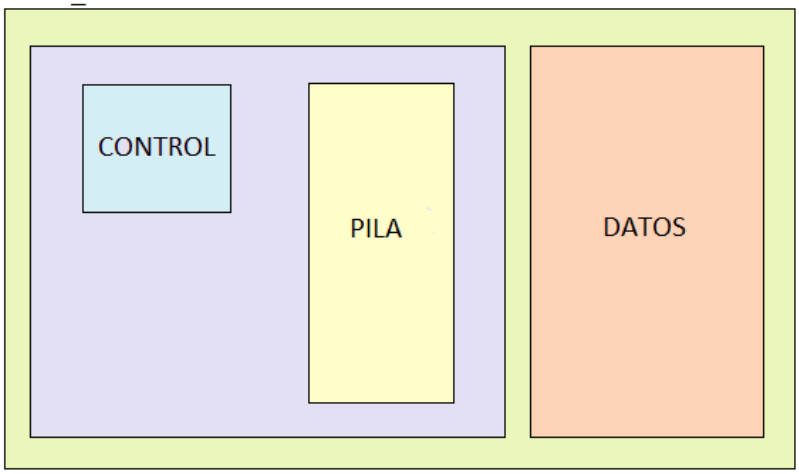
ME_RUBROS



ME_ARTICULOS



ME_CuentasCorrientes



ME_VENTAS

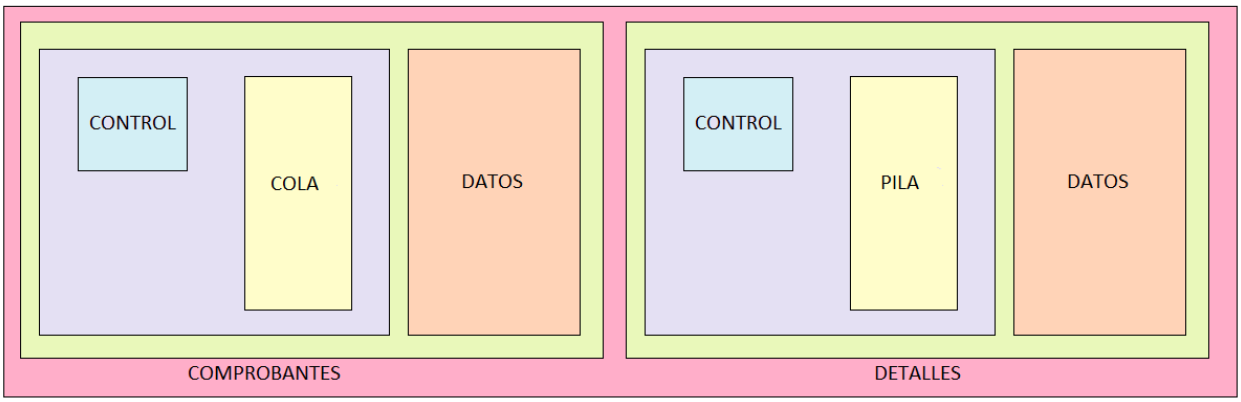
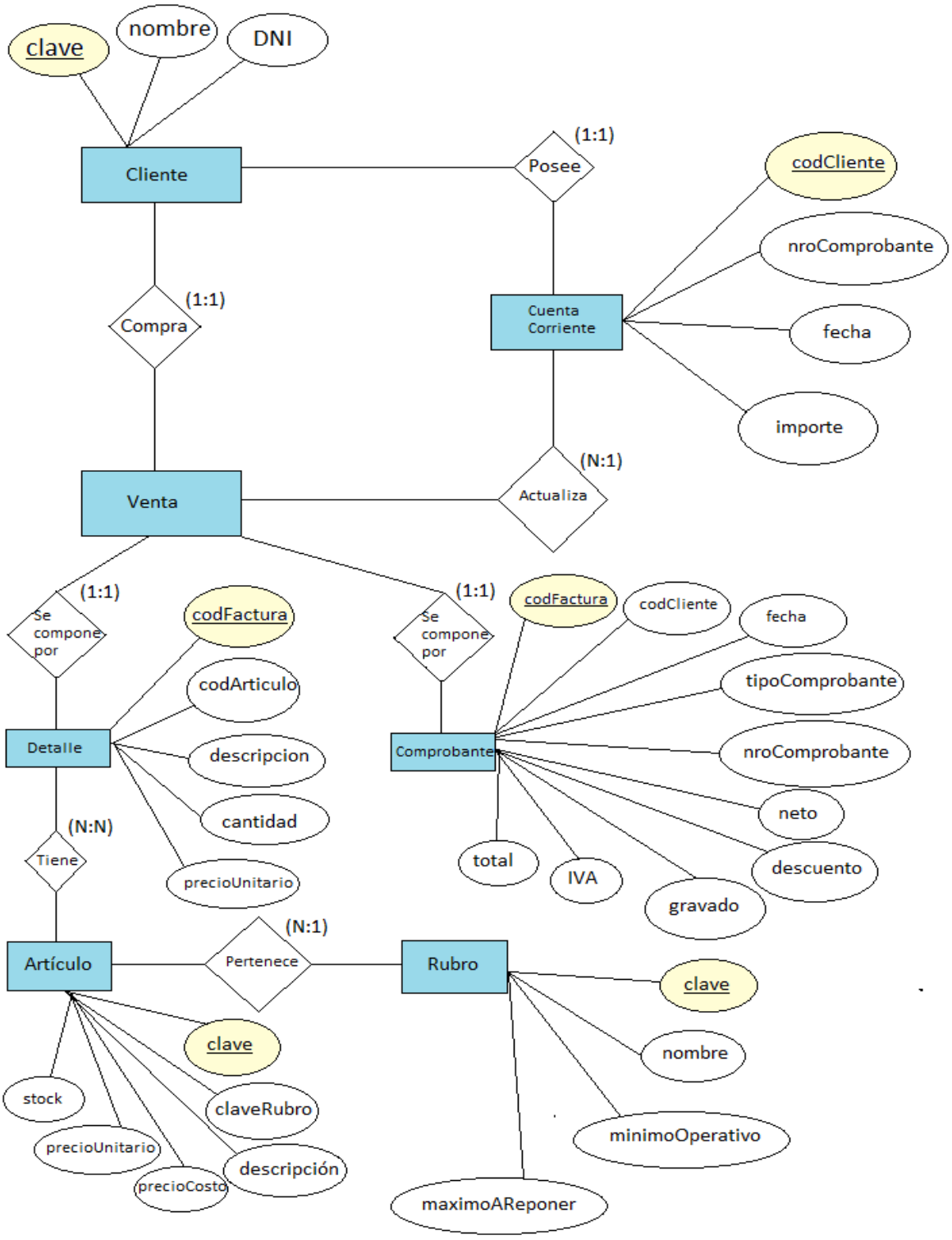


DIAGRAMA ENTIDAD RELACIÓN:



Librería Operacional Índice - “LO_Índice”

Esta librería está dedicada al funcionamiento de un índice simple.

No es usada para ordenar ningún conjunto de datos de entidad, sino que es usada para procesos específicos. Estos son, el rebalanceo de la estructura arbórea binaria (se crea un índice temporal que almacena todos los nodos punteros del árbol), y otro índice temporal para almacenar el ranking de artículos vendidos. Para ésto último no es usado como un índice en sí, pero se aprovechan los procedimientos ya hechos en la librería para ordenar de mayor a menor la cantidad vendida por artículo.

Dentro se definen las siguientes funcionalidades y tipos de datos.

Tipo_Indice: es un tipo de registro que contiene 2 archivos. Uno dedicado al almacenamiento de registros del índice y otro que contiene un registro de control.

Tipo_Clave:Longint;

Tipo_Posicion: LongInt;

Tipo_Registro_Indice: Un registro que se insertará en el archivo de indice. Contiene la clave a relacionar y la posición en el archivo de datos.

function Indice_Crear(var Indice: Tipo_Indice; sRuta, sNombre: String): boolean;

La función crear del índice, crea los archivos de control y de índice si no existen en la ruta que se le pasa por parámetro, con el nombre que se proporcione. Si ya existen simplemente los abre.

Luego se asignan estos archivos a la variable Índice que se proporcione.

procedure Indice_Abrir(var Indice: Tipo_Indice);

Abre los archivos de la variable pasada por referencia.

procedure Indice_Cerrar(var Indice: Tipo_Indice);

Cierra los archivos de la variable pasada por referencia.

function Indice_Buscar(var Indice: Tipo_Indice; Clave: Tipo_Clave; var Pos: Tipo_Posicion): Boolean;

Busca una clave pasa por parámetro en el archivo indice de la variable Indice. Si la encuentra devuelve un true y la variable Pos pasada por referencia toma el valor de la posición del archivo indice en la que se encuentra la variable.

Si no la encuentra devuelve un False y pos toma el valor de la posición en la que debería estar ubicada la clave.

procedure Indice_Insertar(var Indice: Tipo_Indice; pos: Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice);

Inserta el RegistroIndice en el archivo indice, en la posición pasada por parámetro.

procedure Indice_Modificar(var Indice: Tipo_Indice; pos: Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice);

Graba el registro indice RegistroIndice que se pase por parámetro en la posición del archivo indicada por pos

procedure Indice_Capturar(var Indice: Tipo_Indice; pos: Tipo_Posicion; var RegistroIndice: Tipo_Registro_Indice);

Captura el registro de la posición que indique pos del archivo indice en el RegistroIndice.

procedure Indice_Eliminar(var Indice: Tipo_Indice; pos: Tipo_Posicion);

Elimina del índice lo que se encuentre en la posicion que indica pos.

function Indice_Primero (var Indice:Tipo_Indice): Tipo_Posicion;

Devuelve la posicion del primer elemento del índice.

function Indice_Ultimo (var Indice:Tipo_Indice): Tipo_Posicion;

Devuelve la posición del último elemento del índice.

function Indice_Proximo (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;

Devuelve la posición del elemento siguiente al de la posicion pasada por parametro en pos.

function Indice_Anterior (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;

Devuelve la posición del elemento anterior al de la posición pasada por parametro en pos.

function Indice_Vacio (var Indice:Tipo_Indice): boolean;

Devuelve true si en el índice no hay elementos insertados, o false si hay al menos uno.

function Indice_ClaveNula (var Indice: Tipo_Indice): longint;

Devuelve la clave Nula de la librería operacional, en este caso -1.

function Indice_PosNula (var Indice: Tipo_Indice): longint;

Devuelve la posición nula de la librería operacional, en este caso -1.

procedure Indice_Destruir (var Indice: Tipo_Indice);

Borra los archivos de la variable Indice del disco duro.

Librería Operacional Índice - “LO_DobleEnlace”

Esta librería está dedicada al funcionamiento de una lista doblemente enlazada.

Es usada para ordenar los datos de las entidades “Clientes” y “Rubros”. Ambas entidades son ordenadas con listas distintas, con tipos de clave distintas aunque la clave en ambos casos es de tipo string.

Dentro se definen las siguientes funcionalidades y tipos de datos.

Tipo_Posicion: Longint;

Tipo_Clave: String [3];

Tipo_Registro_Indice: Registro que se almacenará en el archivo de lista. Contiene un campo clave de Tipo_Clave, un campo posición en datos de Tipo_Posicion, un campo Ant que guarda la posición del elemento anterior en la lista, y un Sig que guarda la posición del elemento siguiente en la lista:

Tipo_Indice: Registro que contiene el archivo de lista y el archivo de control de la lista.

function DobleEnlace_Crear(var Indice: Tipo_Indice; sRuta, sNombre: Tipo_Cadena): boolean;

Crea los archivo de Lista y Control si no estan creado en la ruta pasada por parametro con el nombre proporcionado. Si ya existen simplemente los abre..

procedure DobleEnlace_Abrir(var Indice: Tipo_Indice);

Abre los archivos de lista y control.

procedure DobleEnlace_Cerrar(var Indice: Tipo_Indice);

Cierra los archivos de lista y control.

function DobleEnlace_Buscar(var Indice: Tipo_Indice; clave: Tipo_Clave; var Posicion: Tipo_Posicion): boolean;

Busca en la lista la clave pasada por parámetro y devuelve true si lo encontró (y devuelve en la variable posicion el valor de la posición donde lo encontró) Si no lo encuentra devuelve false y además la posición donde debería estar.

Function DobleEnlace_Buscar_Mejorada (Var Indice: Tipo_Indice; clave: Tipo_Clave; Var Posicion : Tipo_Posicion): Boolean ;

Busca en la lista la clave pasada por parámetro y devuelve true si lo encontró (y devuelve en la variable posicion el valor de la posición donde lo encontró) Si no lo encuentra devuelve false y además la posición donde debería estar.

Procedure DobleEnlace_Insertar (var Indice: Tipo_Indice; pos:Tipo_Posicion; Reg:Tipo_Registro_Indice);

Inserta un elemento pasado por parámetro a la lista en la posición proporcionada.

Procedure DobleEnlace_Modificar(var Indice: Tipo_Indice; pos: Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice);

Inserta el elemento pasado por parámetro suplantando al que esté en la posición proporcionada

procedure DobleEnlace_Capturar(var Indice: Tipo_Indice; pos: Tipo_Posicion; var RegistroIndice: Tipo_Registro_Indice);

Toma un elemento de la lista (en la posición pos) y lo guarda en el RegistroIndice.

procedure DobleEnlace_Eliminar (var Indice: Tipo_Indice; pos: Tipo_Posicion);

Elimina un elemento de la lista (el de la posición pos), desenlazandolo de su predecesor y sucesor, y enlazandolo a una lista de elementos eliminados que se encuentra dentro del mismo archivo de lista.

function DobleEnlace_Primero (var Indice:Tipo_Indice): Tipo_Posicion;

Devuelve la posición del primer elemento de la lista.

function DobleEnlace_Primero_E(var indice:Tipo_Indice): TIpo_Posicion;

Devuelve la posición del primer elemento de la lista de elementos eliminados.

function DobleEnlace_Ultimo (var Indice:Tipo_Indice): Tipo_Posicion;

Devuelve el último elemento de la lista

function DobleEnlace_Proximo (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;

Devuelve la posicion del próximo elemento al pasado por parametro

function DobleEnlace_Anterior (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;

Devuelve la posición del anterior elemento al pasado por parametro

function DobleEnlace_Vacio (var Indice:Tipo_Indice): boolean;

Devuelve true si la estructura está vacía

function DobleEnlace_PosNula(var Indice:Tipo_Indice): integer;

Devuelve la posición nula de la librería operacional, en este caso -1.

function DobleEnlace_ClaveNula(var Indice:Tipo_Indice): string;

Devuelve la clave nula de la librería operacional, en este caso '' (un string vacío).

procedure DobleEnlace_Destruir (var Indice: Tipo_Indice);

Elimina los archivos de lista y control del disco duro.

function CompararClaves (Clave1, Clave2: Tipo_Clave): integer;

Es de uso exclusivo interno de la librería. Recibe por parámetro la primera clave y la segunda. Devuelve un 1 si la clave 1 es mayor, un 2 si la clave 2 es mayor, y un 0 si son iguales. Funciona así para poder comparar eficientemente las claves de “Clientes” y “Rubros” en la misma librería.

Librería Operacional Pila - “LO_Pila”

Esta librería está dedicada al funcionamiento de una estructura de pila o stack que sigue la lógica LIFO.

Funciona haciendo uso de la ya explicada librería doble enlace, ya que se considera un caso “especial” de una lista doblemente enlazada, en la que siempre se inserta en el primer posicion y siempre se elimina y captura en la primer posicion.

Es usada para ordenar los datos de las entidades “Detalles” y “Cuentas Corrientes”.

Dentro se definen las siguientes funcionalidades y tipos de datos.

function Pila_Crear (var Pila: Tipo_Indice; sRuta, sNombre: Tipo_Cadena): boolean;

Crea los archivos de pila y control si no estan creados en la ruta y nombre proporcionados, y si ya existen simplemente los abre.

procedure Pila_Abrir (var Pila: Tipo_Indice) ;

Abre los archivos de pila y control.

procedure Pila_Cerrar (Var Pila: Tipo_Indice);

Cierra los archivos de pila y control.

procedure Pila_Apilar (var Pila: Tipo_Indice ; RegistroIndice: Tipo_Registro_Indice) ;

Inserta el elemento RegistroIndice en la pila.

procedure Pila_Desapilar (Var Pila: Tipo_Indice);

Elimina el elemento de la primer posicion de la pila.

procedure Pila_Tope (var Pila: Tipo_Indice ; var RegistroIndice: Tipo_Registro_Indice);

Captura el elemento de la primer posicion de la pila y lo almacena en el registro indice.

function Pila_Vacia (var Pila: Tipo_Indice): boolean ;

Devuelve true si la pila esta vacía y false si no lo está.

function Pila_ClaveNula (var Pila:Tipo_Indice): string;

Devuelve la clave nula de la librería operacional, en este caso la misma de doble enlace o sea un string vacío.

function Pila_PosNula (var Pila:Tipo_Indice): longint;

Devuelve la clave nula de la librería operacional, en este caso la misma de doble enlace o sea un - 1.

procedure Pila_Destruir (var Pila:Tipo_Indice);

Elimina los archivos de pila y control del disco duro.

Librería Operacional Cola - “LO_Cola”

Esta librería está dedicada al funcionamiento de una estructura de cola o queue que sigue la lógica FIFO.

Funciona haciendo uso de la ya explicada librería doble enlace, ya que se considera un caso “especial” de una lista doblemente enlazada, en la que siempre se inserta en la última posición y siempre se elimina y captura en la primer posición.

Es usada para ordenar los datos de las entidades “Comprobantes”.

Dentro se definen las siguientes funcionalidades y tipos de datos.

```
function Cola_Crear ( var Cola: Tipo_Indice; sRuta, sNombre: Tipo_Cadena): boolean;
```

Crea los archivos de cola y control si no están creados en la ruta y nombre proporcionados, y si ya existen simplemente los abre.

```
procedure Cola_Abrir ( var Cola: Tipo_Indice ) ;
```

Abre los archivos de cola y control.

```
procedure cola_Cerrar ( Var Cola: Tipo_Indice );
```

Cierra los archivos de cola y control.

```
procedure Cola_Encolar (var Cola: Tipo_Indice ; RegistroIndice: Tipo_Registro_Indice ) ;
```

Inserta el elemento RegistroIndice en la cola.

procedure Cola_EncolarConFecha (var Cola: Tipo_Indice ; RegistroIndice: Tipo_Registro_Indice; Tipo_Comprobante, Fecha: String) ;

Inserta el elemento RegistroIndice en la cola, y además actualiza en el control la fecha de la última factura emitida del tipo que se pase por parámetro.

function UltimaFechaEmitida(var Cola: Tipo_Indice; tipoComprobante: string): string;

Devuelve la fecha de la ultima factura del tipoComprobante que se pase por parámetro

function Cola_UltimoNroComprobante(var Cola:Tipo_Indice): integer;

Devuelve el numero del ultimo comprobante emitido.

procedure Colla_Decolar (Var Colala: Tipo_Indice);

Elimina el elemento de la primer posicion de la cola.

procedure cola_Frente (var Cola: Tipo_Indice ; var RegistroIndice: Tipo_Registro_Indice);

Captura el elemento de la primer posicion de la cola y lo almacena en el registro indice.

function Cola_Vacia (var Cola: Tipo_Indice): boolean ;

Devuelve true si la cola esta vacía y false si no lo está.

function Cola_ClaveNula (var Cola: Tipo_Indice): string;

Devuelve la clave nula de la librería operacional, en este caso la misma de doble enlace o sea un string vacío.

function Cola_PosNula (var Cola: Tipo_Indice): longint;

Devuelve la clave nula de la librería operacional, en este caso la misma de doble enlace o sea un - 1.

procedure Cola_Tope (var Cola:Tipo_Indice; var RegistroIndice: Tipo_Registro_Indice);

Captura el elemento de la ultima posicion de la cola y lo almacena en el registroIndice.

procedure Cola_Destruir (var Cola:Tipo_Indice);

Elimina los archivos de cola y control del disco duro.

Librería Operacional ABB (Árbol Binario de Búsqueda) - “LO_ABB”

Esta librería está dedicada al funcionamiento de una estructura de árbol binario con control de balanceo por medio de un archivo que almacena la cantidad de elementos por nivel.

Es usada para ordenar los datos de las entidades “Artículos”.

Dentro se definen las siguientes funcionalidades y tipos de datos.

Tipo_Posicion: Longint ;

Tipo_Clave : Longint ;;

Tipo_Cantidad: Longint;

Tipo_Porcentaje: 1..100;

function Arbol_Crear (var Arbol: Tipo_Arbol; sRuta, sNombre: Tipo_Cadena; porcentajeTolerancia, porcentajeNiveles: Tipo_Porcentaje): boolean;

Crea los archivos de Control Índice y Niveles referidos al ABB si no están creados en la ruta y con nombre proporcionados. Si ya existen simplemente los abre.

Se debe proporcionar el porcentaje de tolerancia a desequilibrio (porcentajeTolerancia) y el porcentaje de niveles a evaluar (porcentajeNiveles) para luego, al definir el equilibrio del árbol, realizarlo en base a estos porcentajes.

function Arbol_Creado (var Arbol: Tipo_Arbol; sRuta, sNombre: Tipo_Cadena): boolean;

Devuelve true si los archivos del ABB existen y false si no existen.

procedure Arbol_Abrir (var Arbol: Tipo_Arbol);

Abre los archivos control, índice y niveles de la variable Arbol.

procedure Arbol_Cerrar(var Arbol: Tipo_Arbol);

Cierra los archivos control, índice, y niveles de la variable Arbol.

function Arbol_Buscar (var Arbol: Tipo_Arbol; Clave:Tipo_Clave; var pos:Tipo_Posicion; Var Nivel: Tipo_Cantidad): Boolean;

Busca en el archivo índice una clave, si no la encuentra devuelve false y pos toma la posición del que debería ser el padre, y nivel el nivel a donde debería pertenecer la clave.

Si la encuentra devuelve true, y las variables pos y nivel toman la posición y nivel donde se encontró el elemento.

procedure Arbol_Insertar(var Arbol: Tipo_Arbol; pos:Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice; Nivel: Tipo_Cantidad);

Inserta elemento RegistroIndice como hoja al árbol

procedure Arbol_Eliminar (var Arbol: Tipo_Arbol; pos: Tipo_Posicion; nivel: Tipo_Cantidad);

Elimina el nodo del árbol que se encuentra en la posicion y nivel indicados por parametro.

function Arbol_Raiz (var Arbol: Tipo_Arbol): Tipo_Posicion;

Devuelve la posición de la raíz del árbol

function Arbol_HijoIzquierdo (Var Arbol: Tipo_Arbol; pos:Tipo_Posicion): Tipo_Posicion;

Devuelve la posición del hijo izquierdo de un nodo (el de la posicion pos).

function Arbol_HijoDerecho (Var Arbol: Tipo_Arbol; pos:Tipo_Posicion): tipo_posicion;

Devuelve la posición del hijo derecho de un nodo (el de la posicion pos)

function Arbol_Padre (Var Arbol: Tipo_Arbol; pos:Tipo_Posicion): tipo_posicion;

Devuelve la posicion del padre de un nodo (el de la posicion pos).

function Arbol_Vacio (var Arbol: Tipo_Arbol): Boolean ;

Devuelve true si el árbol se encuentra vacío y false si no.

procedure Arbol_Capturar (Var Arbol: Tipo_Arbol; pos:Tipo_Posicion; var RegistroIndice:Tipo_Registro_Indice);

Captura un nodo de la posicion pasada por parámetro en el RegistroIndice

procedure Arbol_Modificar (Var Arbol: Tipo_Arbol; pos:Tipo_Posicion; RegistroIndice:Tipo_Registro_Indice);

Modifica el nodo que se encuentra en la posicion pos, grabando en el archivo indice, en la posición pos, el RegistroIndice

function Arbol_PosNula (var Arbol: Tipo_Arbol): longint;

Devuelve la posicion nula de la librería operacional, en este caso - 1.

function Arbol_ClaveNula (var Arbol: Tipo_Arbol): Tipo_Clave;

Devuelve la clave nula de la librería operacional, en este caso - 1.

procedure Arbol_Destruir (var Arbol: Tipo_Arbol);

Elimina los archivos referidos al ABB del disco duro.

function Arbol_Equilibrado (var Arbol: Tipo_Arbol) : boolean;

Devuelve true si el árbol se encuentra equilibrado y false si no

procedure Arbol_Rebalanceo (var Arbol: Tipo_Arbol; sRuta, sNombre: Tipo_Cadena);

Crea un nuevo árbol a partir del ya existente pasado por referencia. El nuevo árbol se encontrará en la misma ruta con el mismo nombre original pero estará balanceado.

function Arbol_UltimoNivel(var Arbol: Tipo_Arbol): Tipo_Cantidad;

Devuelve el último nivel con elementos del árbol.

procedure Arbol_CapturarNivel(var Arbol: Tipo_Arbol; pos: Tipo_Posicion; var RegistroNivel: Tipo_Registro_Nivel);

Captura el elemento que hay en el archivo de niveles en la posición indicada y los guarda en el RegistroNIvel

function Arbol_PorcentajeTolerancia (var Arbol: Tipo_Arbol): Tipo_Porcentaje;

Devuelve el porcentaje de tolerancia a desequilibrio establecido en el control del ABB.

function Arbol_PorcentajeNiveles(var Arbol: Tipo_Arbol): Tipo_Porcentaje;

Devuelve el porcentaje de niveles a evaluar establecido en el control del ABB.

function Arbol_CantidadElementos (var Arbol: Tipo_Arbol): integer;

Devuelve la cantidad de elementos que hay en el árbol.

procedure Arbol_CambiarPorcentajeTolerancia(var Arbol: Tipo_Arbol; porcentajeTolerancia: Tipo_Porcentaje);

Cambia el porcentaje de tolerancia a desequilibrio, estableciendo el pasado por parámetro.

procedure Arbol_cambiarPorcentajeNiveles(var Arbol: Tipo_Arbol; porcentajeNivel: Tipo_Porcentaje);

Cambia el porcentaje de niveles a evaluar,, estableciendo el pasado por parámetro.

Librerías Operacionales de Datos

En el programa se trabaja con 6 librerías de datos correspondientes a las 6 entidades identificadas anteriormente. Lo único que cambia en ellas son los tipos de registros, es decir, los atributos de cada entidad, detallados anteriormente en el diagrama entidad-relación. Pero las funcionalidades se comportan de la misma manera en todas ellas.

Estas funcionalidades son:

function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;

Crea el archivo de datos, con el nombre indicado en la ruta indicada si es que no existe aún. Si existe simplemente lo abre.

procedure Datos_Abrir(var ME: Tipo_Datos);

Abre el archivo de datos.

procedure Datos_Cerrar(var ME: Tipo_Datos);

Cierra el archivo de datos

procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);

Inserta en una posición nueva el registro de datos pasado por parámetro

procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);

Modifica el registro que se encuentra en la posición pos del archivo de datos, almacenando en esta posición el nuevo registro reg.

procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);

Captura el registro que se encuentra en la posición pos del archivo de datos y lo guarda en la variable reg.

procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);

Elimina lógicamente el registro que se encuentra en la posición pos del archivo de datos.

procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);

Elimina del disco duro el archivo de datos.

Código Fuente

A continuación se presenta el código fuente del programa, en total son 22 archivos “.pas” (más los generados automáticamente por el IDE)

PROJECT1

```
program Project1;
uses
  Forms,
  Unit1 in 'Unit1.pas' {Form_MenuPrincipal},
  LO_Datos_Clientes in 'LO_Datos_Clientes.pas',
  LO_DobleEnlace in 'LO_DobleEnlace.pas',
  Unit2 in 'Unit2.pas' {Form_Clientes},
  Unit3 in 'Unit3.pas' {Form_Rubros},
  LO_Datos_Rubros in 'LO_Datos_Rubros.pas',
  LO_Datos_Articulos in 'LO_Datos_Articulos.pas',
  Unit4 in 'Unit4.pas' {Form_Articulos},
  LO_ABB in 'LO_ABB.pas',
  LO_Indice in 'LO_Indice.pas',
  Unit5 in 'Unit5.pas' {Form_Mantenimiento},
  Unit6 in 'Unit6.pas' {Form_Facturacion},
  LO_Datos_Comprobantes in 'LO_Datos_Comprobantes.pas',
  LO_Datos_Detalles in 'LO_Datos_Detalles.pas',
  LO_Cola in 'LO_Cola.pas',
  LO_Pila in 'LO_Pila.pas',
  Unit7 in 'Unit7.pas' {Form_Reporte},
  LO_Datos_Asientos in 'LO_Datos_Asientos.pas',
  Unit8 in 'Unit8.pas' {Form_Reporte_Articulos},
  Unit9 in 'Unit9.pas' {Form_Listados},
  Unit10 in 'Unit10.pas' {Form_ListadosCC};

{$R *.res}
begin
  Application.Initialize;
  Application.CreateForm(TForm_MenuPrincipal, Form_MenuPrincipal);
  Application.CreateForm(TForm_Clientes, Form_Clientes);
  Application.CreateForm(TForm_Rubros, Form_Rubros);
  Application.CreateForm(TForm_Articulos, Form_Articulos);
  Application.CreateForm(TForm_Mantenimiento, Form_Mantenimiento);
  Application.CreateForm(TForm_Facturacion, Form_Facturacion);
```

```
  Application.CreateForm(TForm_Reporte, Form_Reporte);
  Application.CreateForm(TForm_Reporte_Articulos, Form_Reporte_Articulos);
  Application.CreateForm(TForm_Listados, Form_Listados);
  Application.CreateForm(TForm_ListadosCC, Form_ListadosCC);
  Application.Run;
end.
```

UNIT 1

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Buttons, StdCtrls, jpeg, ExtCtrls, LO_Datos_Clientes, LO_DobleEnlace,
  LO_Datos_Rubros, LO_ABB, LO_Datos_Articulos, FileCtrl, LO_Cola, LO_Pila, LO_Datos_Comprobantes,
  LO_Datos_Detalles, LO_Indice, LO_Datos_Asientos;
type
  TForm_MenuPrincipal = class(TForm)
    Image1: TImage;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    BitBtn1: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Button6: TButton;
    DirectoryListBox1: TDirectoryListBox;
    Button5: TButton;
    procedure FormCreate(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
```

```
public
  { Public declarations }
end;

Tipo_ME_Clientes = Record
                    Indice: LO_DobleEnlace.Tipo_Indice;
                    Datos: LO_Datos_Clientes.Tipo_Datos;
                   end;

Tipo_ME_Rubros = record
                   Indice: LO_DobleEnlace.Tipo_Indice;
                   Datos: LO_Datos_Rubros.Tipo_Datos;
                  end;

Tipo_Me_Articulos = record
                     Arbol: LO_ABB.Tipo_Arbol;
                     Datos: LO_Datos_Articulos.Tipo_Datos;
                    end;

Tipo_Comprobante = record
                    Cola: LO_DobleEnlace.Tipo_Indice;
                    Datos: LO_Datos_Comprobantes.Tipo_Datos;
                   end;

Tipo_Detalle = record
                 Pila: LO_DobleEnlace.Tipo_Indice;
                 Datos: LO_Datos_Detalles.Tipo_Datos;
                end;

Tipo_ME_Ventas = record
                   Comprobante: Tipo_Comprobante;
                   Detalle: Tipo_Detalle;
                  end;

Tipo_ME_CuentasCorrientes = record
                              Pila: LO_DobleEnlace.Tipo_Indice;
                              Datos: LO_Datos_Asientos.Tipo_Datos;
                             end;

var
  Form_MenuPrincipal: TForm_MenuPrincipal;
  ME_Clientes: Tipo_ME_Clientes;
  ME_Rubros: Tipo_ME_Rubros;
  ME_Articulos: Tipo_ME_Articulos;
```

```
  ME_Ventas: Tipo_ME_Ventas;
  ME_CuentasCorrientes: Tipo_ME_CuentasCorrientes;

  sRuta, sNombreDatos, sNombreIndice, sNombreRubros, sNombreArticulos,
  sNombreComprobantes, sNombreDetalles, sNombreCuentasCorrientes: string;

implementation

uses Unit2, Unit3, Unit4, Unit5, Unit6, Unit9;

{$R *.dfm}

function VerificarPorcentaje (palabra: string): boolean;
var
  bResultado: boolean;
  letra: string;
  n: integer;
begin
  bResultado:= true;
  for n:= 1 to (Length(palabra)) do
  begin
    letra := palabra[n];
    if (letra<'0') or (letra>'9') then bResultado:=false;
  end;

  VerificarPorcentaje:= bResultado;
end;

procedure TForm_MenuPrincipal.FormCreate(Sender: TObject);
var
  sNiveles, sTolerancia: string;
begin

  sRuta:= Form_MenuPrincipal.DirectoryListBox1.Directory + '\Archivos';
  //Nombres Clientes
  sNombreDatos:='Datos';
  sNombreIndice:='Clientes';
  //Nombre Rubros
  sNombreRubros:='Rubros';
  //Nombre Articulos
  sNombreArticulos:= 'Articulos';
  //Nombre Ventas
  sNombreComprobantes:= 'Comprobantes';
  sNombreDetalles:= 'Detalles';
```

```
//Nombre cuentas corrientes
sNombreCuentasCorrientes:= 'Asientos';


//Archivos de Clientes
LO_Datos_Clientes.Datos_Crear(Me_Clientes.Datos, sRuta, sNombreDatos);
LO_DobleEnlace.DobleEnlace_Crear(ME_Clientes.Indice, sRuta, sNombreIndice);
LO_Datos_Clientes.Datos_Abrir(ME_Clientes.Datos);
LO_DobleEnlace.DobleEnlace_Abrir(ME_Clientes.Indice);

//Archivos de Rubros
LO_Datos_Rubros.Datos_Crear(Me_Rubros.Datos, sRuta, sNombreRubros);
LO_DobleEnlace.DobleEnlace_Crear(ME_Rubros.Indice, sRuta, sNombreRubros);
LO_Datos_Rubros.Datos_Abrir(ME_Rubros.Datos);
LO_DobleEnlace.DobleEnlace_Abrir(ME_Rubros.Indice);

//Archivo Articulos
//Antes de crear hay que ver si ya estan creados
//Si ya estan creados simplemente se abren
//Si no estan creados se deben crear a partir de los porcentajes ingresados por el usuario
LO_Datos_Articulos.Datos_Crear(ME_Articulos.Datos, sRuta, sNombreArticulos);
LO_Datos_Articulos.Datos_Abrir(ME_Articulos.Datos);

if (LO_ABB.Arbol_Creado(ME_Articulos.Arbol, sRuta, sNombreArticulos) = true) then
begin
  LO_ABB.Arbol_Abrir(ME_Articulos.Arbol);
end
else
begin
  Showmessage('A continuación deberá definir los porcentajes de tolerancia a desequilibrio y niveles a controlar en balanceo.');


  repeat
  if InputQuery('Porcentajes de Árbol.','Ingrese el porcentaje de niveles a tener en cuenta al momento de determinar equilibrio.',
sNiveles) = false
  then showmessage('El usuario canceló la operacion y este programa deberia cerrarse')
  until VerificarPorcentaje(sNiveles) = true;


  repeat
  if InputQuery('Porcentajes de Árbol.','Ingrese el porcentaje de tolerancia a desbalanceo de niveles.', sTolerancia) = false
  then showmessage('El usuario canceló la operacion y este programa deberia cerrarse')
  until VerificarPorcentaje(sTolerancia) = true;
  LO_ABB.Arbol_Crear(ME_Articulos.Arbol, sRuta, sNombreArticulos, StrToInt(sTolerancia), StrToInt(sNiveles));
  LO_ABB.Arbol_Abrir(ME_Articulos.Arbol);
  showmessage('Se han creado los archivos del ABB. Si desea cambiar los valores en un futuro dirigase al menu mantenimiento.');
```

```
 end;
 //Archivo Ventas
 //Cola
 LO_Datos_Comprobantes.Datos_Crear(ME_Ventas.Comprobante.Datos, sRuta, sNombreComprobantes);
 LO_Cola.Cola_Crear(ME_Ventas.Comprobante.Cola, sRuta, sNombreComprobantes);
 LO_Datos_Comprobantes.Datos_Abrir(Me_Ventas.Comprobante.Datos);
 LO_Cola.Cola_Abrir(ME_Ventas.Comprobante.Cola);
 //Pila
 LO_Datos_Detalles.Datos_Crear(ME_Ventas.Detalle.Datos, sRuta, sNombreDetalles);
 LO_Pila.Pila_Crear(ME_Ventas.Detalle.Pila, sRuta, sNombreDetalles);
 LO_Datos_Detalles.Datos_Abrir(Me_Ventas.Detalle.Datos);
 LO_Pila.Pila_Abrir(ME_Ventas.Detalle.Pila);

 //Archivo Cuentas Corrientes
 LO_Datos_Asientos.Datos_Crear(ME_CuentasCorrientes.Datos, sRuta, sNombreCuentasCorrientes);
 LO_Pila.Pila_Crear(ME_CuentasCorrientes.Pila, sRuta, sNombreCuentasCorrientes);
 LO_Datos_Asientos.Datos_Abrir(ME_CuentasCorrientes.Datos);
 LO_Pila.Pila_Abrir(ME_CuentasCorrientes.Pila);
end;

procedure TForm_MenuPrincipal.BitBtn1Click(Sender: TObject);
begin
 //Cerrar clientes
 LO_Datos_Clientes.Datos_Cerrar(ME_Clientes.Datos);
 LO_DobleEnlace.DobleEnlace_Cerrar(Me_Clientes.Indice);

 //Cerrar rubros
 Lo_Datos_Rubros.Datos_Cerrar(ME_Rubros.Datos);
 LO_DobleEnlace.DobleEnlace_Cerrar(ME_Rubros.Indice);

 //Cerrar articulos
 LO_Datos_Articulos.Datos_Cerrar(ME_Articulos.Datos);
 LO_ABB.Arbol_Cerrar(ME_Articulos.Arbol);

 //Cerrar ventas
 LO_Datos_Comprobantes.Datos_Cerrar(ME_Ventas.Comprobante.Datos);
 LO_Cola.Cola_Cerrar(ME_Ventas.Comprobante.Cola);

 LO_Datos_Detalles.Datos_Cerrar(ME_Ventas.Detalle.Datos);
 LO_Pila.Pila_Cerrar(ME_Ventas.Detalle.Pila);

 //Cerrar CuentasCorrientes
 LO_Datos_Asientos.Datos_Cerrar(ME_CuentasCorrientes.Datos);
 LO_Pila.Pila_Cerrar(ME_CuentasCorrientes.Pila);
```

```
end;

procedure TForm_MenuPrincipal.Button1Click(Sender: TObject);
begin
  Form_MenuPrincipal.Hide;
  Form_Clientes.Show;
end;

procedure TForm_MenuPrincipal.Button2Click(Sender: TObject);
begin
  Form_MenuPrincipal.Hide;
  Form_Rubros.show;
end;

procedure TForm_MenuPrincipal.Button3Click(Sender: TObject);
begin
  Form_MenuPrincipal.Hide;
  Form_Articulos.Show;

end;

procedure TForm_MenuPrincipal.Button6Click(Sender: TObject);
begin
  Form_MenuPrincipal.Hide;
  Form_Mantenimiento.Show;
end;

procedure TForm_MenuPrincipal.Button4Click(Sender: TObject);
begin
  Form_Facturacion.Show;
  Form_MenuPrincipal.Hide;
end;

procedure TForm_MenuPrincipal.Button5Click(Sender: TObject);
begin
  Form_Listados.show;
end;

end.
```

UNIT 2

```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls, Buttons, Lo_DobleEnlace, Lo_Datos_Clientes,
  Grids, ComCtrls, LO_Datos_Asientos, LO_Pila, LO_Datos_Detalles;
type
  TForm_Clientes = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Button1: TButton;
    GroupBox1: TGroupBox;
    BitBtn1: TBitBtn;
    Edit1: TEdit;
    Label2: TLabel;
    Edit2: TEdit;
    Label3: TLabel;
    Edit3: TEdit;
    Label4: TLabel;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    StringGrid1: TStringGrid;
    Label5: TLabel;
    ComboBox1: TComboBox;
    DateTimePicker1: TDateTimePicker;
    DateTimePicker2: TDateTimePicker;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Edit4: TEdit;
    procedure Button1Click(Sender: TObject);
```

```
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Edit1Change(Sender: TObject);

    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
    procedure DateTimePicker1Change(Sender: TObject);
    procedure DateTimePicker2Change(Sender: TObject);
    procedure Edit4Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Clientes: TForm_Clientes;
implementation
uses Unit1, Unit6;

{$R *.dfm}

procedure LimpiarTablaClientes;
var
  i: longint;
begin
  for i:= 1 to Form_Clientes.StringGrid1.RowCount  do
  begin
    Form_clientes.StringGrid1.Rows[i].Clear;
  end;
end;
```

```
procedure LimpiarAbmClientes;
begin
  Form_Clientes.Edit1.Clear;
  Form_Clientes.Edit2.Clear;
  Form_Clientes.Edit3.Clear;
end;

procedure ListarClientes;
var
  RegistroDatos: LO_Datos_Clientes.Tipo_Registro_Datos;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  posicion, posDatos: LO_DobleEnlace.Tipo_Posicion;
  i: longint;
begin
  LimpiarTablaClientes;
  Form_Clientes.StringGrid1.Cells[0,0]:= 'Cód.';
  Form_Clientes.StringGrid1.Cells[1,0]:= 'Nombre';
  Form_Clientes.StringGrid1.Cells[2,0]:= 'DNI';
  Form_Clientes.StringGrid1.RowCount:= 1;
  posicion:= LO_DobleEnlace.DobleEnlace_Primero(Unit1.ME_Clientes.Indice);
  i:= 1;
  while (posicion <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.ME_Clientes.Indice)) do
  begin
    Form_Clientes.StringGrid1.RowCount:= Form_Clientes.StringGrid1.RowCount + 1;
    Lo_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, posicion, RegistroIndice);
    posDatos:= registroIndice.Posicion;
    LO_Datos_Clientes.Datos_Capturar(Unit1.ME_clientes.Datos, posDatos, RegistroDatos);
    Form_Clientes.StringGrid1.Cells[0, i]:= RegistroDatos.Clave;
    Form_clientes.StringGrid1.Cells[1, i]:= RegistroDatos.Nombre;
    Form_clientes.StringGrid1.Cells[2, i]:= RegistroDatos.Dni;
    i:=i+1;
    posicion:= RegistroIndice.Sig;
```

```
  end;//while
end;  //End ListarClientes

function VerificarClave(Clave: String): boolean;
var
  bResultado: boolean;
begin
  bResultado:= false;
  if ( Clave[1] >= 'A' ) and (Clave[1] <= 'Z') then
  begin
    if ( Ord(Clave[2]) >= 48) and ( Ord(Clave[2]) <= 57) then
      if ( Ord(Clave[3]) >= 48) and ( Ord(Clave[3]) <= 57) then bResultado:=true
  end;
  VerificarClave:= bResultado;
end;    //End VerificarClave

function VerificarFormatoDni(Dni: String): boolean;
var
  bResultado, bCorte: boolean;
  pos: integer;
begin
  bResultado:= true;
  bCorte:= false;
  pos:=1;
  if (Length(dni) <> 13) then bResultado:=false
  else
  begin
    while (bCorte = false) and (pos < Length(dni)+1) do
    begin
    if ((pos = 1) or (pos = 2) or ((pos >=4) and (pos <=11)) or (pos = 13)) then
    begin
      if ((DNI[pos] < '0') or (DNI[pos] > '9')) then bResultado:=false;
    end;
```

```
    if (pos = 3) or (pos = 12) then
    begin
      if DNI[pos] <> '-' then bResultado:= false;
    end;
    pos:=pos+1
    end;
  end;
  VerificarFormatoDNI:= bResultado;
end;

function CorregirClave(Clave: string): string;
var
  nuevaClave: string;
begin
  nuevaClave:=clave;
  if (Clave[1]>= 'a') and (Clave[1]<='z') then
  begin
    Clave[1]:= chr ( Ord(Clave[1]) - 32 ); //no se usa Uppercase() por incompatibilidad entre string y char
    nuevaClave:= clave;
  end;
  CorregirClave:= nuevaClave;
end;

function CapturarDatos(Clave: string): boolean;
var
  Pos: LO_DobleEnlace.Tipo_Posicion;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  posDatos: LO_Datos_Clientes.Tipo_Posicion;
  RegistroDatos: LO_Datos_clientes.Tipo_Registro_Datos;
  bResultado: boolean;
begin
  bResultado:= false;
  if (Lo_DobleEnlace.DobleEnlace_Buscar(Unit1.Me_Clientes.Indice, clave, pos)=true) then
```

```
  begin
    LO_DobleEnlace.DobleEnlace_Capturar(Unit1.Me_Clientes.Indice,pos, RegistroIndice);
    posDatos:= RegistroIndice.Posicion;
    LO_Datos_Clientes.Datos_Capturar(Unit1.ME_Clientes.Datos, posDatos, RegistroDatos);
    Form_Clientes.Edit2.Text:= RegistroDatos.Nombre;
    Form_Clientes.Edit3.Text:= RegistroDatos.Dni;
    Form_Clientes.BitBtn1.Enabled:=false;
    Form_Clientes.BitBtn2.Enabled:=true;
    Form_Clientes.BitBtn3.Enabled:=true;
    bResultado:=true;
  end;
  CapturarDatos:=bResultado;
end;

procedure ActualizarClientesEnFacturacion;
var
  pos: integer;
  clave: String;
  RegistroIndiceCliente: LO_DobleEnlace.Tipo_Registro_indice;
begin
  //mostrar clientes en el combobox
  Form_Facturacion.ComboBox1.Items.Clear;
  pos:= LO_DobleEnlace.DobleEnlace_Primero(Unit1.ME_Clientes.Indice);
  while (pos <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.ME_Clientes.Indice)) do
  begin
    LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, pos, RegistroIndiceCliente);
    clave:= RegistroIndiceCliente.Clave;
    Form_Facturacion.ComboBox1.AddItem(clave, Form_Facturacion.ComboBox1);
    pos:= RegistroIndiceCliente.Sig;
  end;
end;

procedure TForm_Clientes.Button1Click(Sender: TObject);
```

```
begin
  Form_Clientes.Close;
  Form_MenuPrincipal.Show;
end;

procedure TForm_Clientes.BitBtn1Click(Sender: TObject);
var
  RegistroDatos: LO_Datos_Clientes.Tipo_Registro_Datos;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  clave: LO_Datos_Clientes.Tipo_Clave;
  nombre: LO_Datos_Clientes.Tipo_Nombre;
  dni: LO_Datos_Clientes.Tipo_Dni;
  pos: LO_DobleEnlace.Tipo_Posicion;
  posDatos: integer;
begin
  clave:= Form_Clientes.Edit1.Text;
  clave:= CorregirClave(clave);
  nombre:= Form_Clientes.Edit2.Text;
  dni:= Form_Clientes.Edit3.Text;

  if (nombre='') or (dni='') then showmessage('Por favor, ingrese todos los campos')
  else
  begin
   RegistroDatos.Clave:=clave;
   RegistroDatos.Nombre:=nombre;
   RegistroDatos.Dni:=dni;

   posDatos:= FileSize(Unit1.ME_Clientes.Datos.D);
   RegistroIndice.Clave:=clave;
   RegistroIndice.Posicion:= posDatos;

   if VerificarFormatoDni(Dni) = false then Showmessage('Error. El formato del DNI debe ser 99-99999999-9')
   else
```

```
  begin
  if (LO_DobleEnlace.DobleEnlace_Buscar(Unit1.ME_Clientes.Indice, clave, pos) = false ) then
  begin

    LO_Datos_Clientes.Datos_Insertar(Unit1.Me_Clientes.Datos, RegistroDatos);
    LO_DobleEnlace.DobleEnlace_Insertar(Unit1.ME_Clientes.Indice, pos, RegistroIndice );

    Showmessage('Se ha insertado el cliente.');

    LimpiarAbmClientes;
    ListarClientes;
    ActualizarClientesEnFacturacion;
  end else showmessage('opaaa no se ha insertado por algun error xd');
  //SINO NO PODRA INSERTARLO PORQUE EL BOTON INSERTAR SE DESHABILITARÄ
  end
  end
end;

procedure TForm_Clientes.FormCreate(Sender: TObject);
begin
  ListarClientes;
  Form_Clientes.ComboBox1.ItemIndex:=0;
  Form_Clientes.DateTimePicker1.Date:= now;
  Form_Clientes.DateTimePicker2.Date:= now;
  edit1.MaxLength:=3;
  edit2.MaxLength:=30;
  edit3.MaxLength:=13;
  Edit4.MaxLength:=4;
end;

procedure TForm_Clientes.Edit1Change(Sender: TObject);
var
  Clave: string;
```

```
begin
  clave:= edit1.Text;

  if Length(Clave) = 1 then
  begin
    if (Clave <'A') or (Clave >'Z') then
    begin
      if (Clave<'a') or (Clave>'z') then Form_Clientes.Edit1.Clear;
    end
  end;
  if (Length(Clave) <> 3) then
  begin
    Form_Clientes.BitBtn1.Enabled:=false;
    Form_Clientes.BitBtn2.Enabled:=false;
    Form_Clientes.BitBtn3.Enabled:=False;
    Form_Clientes.Edit2.Clear;
    Form_Clientes.Edit3.Clear;
  end;
  if (Length(Clave) = 3) then
  begin
    clave:= CorregirClave(Clave);
    if (VerificarClave(Clave) = true) then
    begin
      if CapturarDatos(Clave) = false then Form_Clientes.BitBtn1.Enabled:=true;
    end else Showmessage('Se ha ingresado una clave incorrecta. (El formato debe ser "A23") ');
  end;
end;

procedure TForm_Clientes.BitBtn3Click(Sender: TObject);
var
  RegistroDatos: LO_Datos_Clientes.Tipo_Registro_Datos;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  clave: LO_Datos_Clientes.Tipo_Clave;
```

```
  nombre: LO_Datos_Clientes.Tipo_Nombre;
  dni: LO_Datos_Clientes.Tipo_Dni;
  pos: LO_DobleEnlace.Tipo_Posicion;
  posDatos: integer;
begin
  clave:= Form_Clientes.Edit1.Text;
  clave:= CorregirClave(clave);
  nombre:= Form_Clientes.Edit2.Text;
  dni:= Form_Clientes.Edit3.Text;
  if (nombre='') or (dni='') then showmessage('Por favor, ingrese todos los campos')
  else
  begin
   RegistroDatos.Clave:=clave;
   RegistroDatos.Nombre:=nombre;
   RegistroDatos.Dni:=dni;

   if VerificarFormatoDni(Dni) = false then Showmessage('Error. El formato del DNI debe ser 99-99999999-9')
   else
   begin
    if (LO_DobleEnlace.DobleEnlace_Buscar(Unit1.ME_Clientes.Indice, clave, pos) = true ) then
   begin
     LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, pos, RegistroIndice);
     posDatos:= RegistroIndice.Posicion;
     LO_Datos_Clientes.Datos_Modificar(Unit1.Me_Clientes.Datos, posDatos, RegistroDatos);
     Showmessage('Se ha modificado el cliente.');
     LimpiarAbmClientes;
     ListarClientes;
   end
   end
  end
end;

function AutorizarEliminacionCliente(clave: string): boolean;
```

```
var
  bResultado: boolean;
  suma: real;
  RegistroPila: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroDatosAsiento: LO_Datos_Asientos.Tipo_Registro_Datos;
  PilaAux: LO_DobleEnlace.Tipo_Indice;
  posDatos: longint;
begin
  Pila_Crear(PilaAux, unit1.sRuta, 'pilaTemp');
  Pila_Abrir(PilaAux);
  bResultado:= true;
  suma:= 0;
  while (Pila_Vacia(Unit1.ME_CuentasCorrientes.Pila) = false) do
  begin
    Pila_Tope(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);
    if (RegistroPila.Clave = clave) then
    begin
        posDatos:= REgistroPila.Posicion;
        LO_Datos_Asientos.Datos_Capturar(Unit1.ME_CuentasCorrientes.Datos, posDatos, RegistroDatosAsiento);
        suma:= suma+RegistroDatosAsiento.importe;
    end;
    Pila_Desapilar(Unit1.ME_CuentasCorrientes.Pila);
    Pila_Apilar(pilaAux, RegistroPila);
  end;//while
  //Volvemos a dejar pila CC como estaba
  while (Pila_Vacia(pilaAux) = false) do
  begin
    Pila_Tope(pilaAux, RegistroPila);
    Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);
    Pila_Desapilar(pilaAux);
  end;
  if Suma <> 0 then bResultado:= false;
  Pila_Destruir(pilaAux);
```

```
  AutorizarEliminacionCliente:= bResultado;
end;

procedure TForm_Clientes.BitBtn2Click(Sender: TObject);
var
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  clave: LO_Datos_Clientes.Tipo_Clave;
  pos: LO_DobleEnlace.Tipo_Posicion;
  posDatos: integer;
begin
  clave:= Form_Clientes.Edit1.Text;
  clave:=CorregirClave(clave);
  if (LO_DobleEnlace.DobleEnlace_Buscar(Unit1.ME_Clientes.Indice, clave, pos) = true ) then
  begin
    LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, pos, RegistroIndice);
    posDatos:= RegistroIndice.Posicion;
    if Dialogs.MessageDlg('¿Está seguro que desea eliminar este cliente?', mtConfirmation,[mbYes,mbNo], 0) = mrYes then
    begin
      if (AutorizarEliminacionCliente(clave) = true) then
      begin
        LO_Datos_Clientes.Datos_Eliminar(Unit1.Me_Clientes.Datos, posDatos);
        LO_DobleEnlace.DobleEnlace_Eliminar(Unit1.Me_Clientes.Indice, pos);
        Showmessage('Se ha eliminado un cliente.');
        ActualizarClientesEnFacturacion;
        LimpiarAbmClientes;
        ListarClientes;
      end
      else
        showmessage('No se puede eliminar el cliente. Tiene saldo deudor.')
    end;
  end;
end;
```

```
procedure TForm_Clientes.Button3Click(Sender: TObject);
var
  reg: LO_DobleEnlace.Tipo_Registro_indice;
begin
  if LO_DobleEnlace.DobleEnlace_Ultimo(ME_Clientes.Indice) <> -1 then
  begin
    LO_DobleEnlace.DobleEnlace_Capturar(ME_Clientes.Indice, LO_DobleEnlace.DobleEnlace_Ultimo(ME_Clientes.Indice), Reg);
    Showmessage('Pos ultimo: '+IntToStr(LO_DobleEnlace.DobleEnlace_Ultimo(ME_Clientes.Indice))+', clave: '+Reg.Clave);
  end
end;

procedure TForm_Clientes.ComboBox1Change(Sender: TObject);
begin
  if (Form_Clientes.ComboBox1.ItemIndex = 0) then
    begin
      ListarClientes;
      Form_Clientes.Label6.Visible:= false;
      Form_Clientes.Label7.Visible:= false;
      Form_Clientes.DateTimePicker1.Visible:= false;
      Form_Clientes.DateTimePicker2.Visible:= false;
      Form_Clientes.Label8.Visible:= false;
      Form_Clientes.Edit4.Visible:= false;
    end
    else
    if (Form_Clientes.ComboBox1.ItemIndex = 1) then
    begin
      Form_Clientes.Label6.Visible:= true;
      Form_Clientes.Label7.Visible:= true;
      Form_Clientes.DateTimePicker1.Visible:= true;
      Form_Clientes.DateTimePicker2.Visible:= true;
      Form_Clientes.Label8.Visible:= false;
      Form_Clientes.Edit4.Visible:= false;
    end
```

```
    else
    if (Form_Clientes.ComboBox1.ItemIndex = 2) then
    begin
      Form_Clientes.Label6.Visible:= false;
      Form_Clientes.Label7.Visible:= false;
      Form_Clientes.DateTimePicker1.Visible:= false;
      Form_Clientes.DateTimePicker2.Visible:= false;
      Form_Clientes.Label8.Visible:= true;
      Form_Clientes.Edit4.Visible:= true;
    end;
end;

procedure ListarPorFechas( Fecha1, Fecha2: TDateTime);
var
  posicion, posDatos: longint;
  RegistroPila: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroDatosAsiento: LO_Datos_Asientos.Tipo_Registro_Datos;
  RegistroDatosCliente: LO_Datos_Clientes.Tipo_Registro_Datos;
  pilaAux: LO_DobleEnlace.Tipo_Indice;
  bCorte: boolean;
  i: longint;
begin
  //Por cada uno de los clientes recorremos la pila
  //Si encontramos una fecha asociada a él entre las dos fechas pasadas por parametro
  //Lo escribimos en la grilla
  LimpiarTablaClientes;
  Form_Clientes.StringGrid1.RowCount:= 1;
  posicion := LO_DobleEnlace.DobleEnlace_Primero(Unit1.ME_Clientes.Indice);
  i:=1;
  LO_Pila.Pila_Crear(pilaAux, Unit1.sRuta, 'pilaTemp');
  LO_Pila.Pila_Abrir(pilaAux);
  while (posicion <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.ME_Clientes.Indice)) do
```

```
begin
 LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Clientes.Indice,posicion, RegistroIndice);
 bCorte:=false;
 while (LO_Pila.Pila_Vacia(Unit1.ME_CuentasCorrientes.Pila) = false) and (bCorte = false) do
 begin
  Pila_Tope(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);
  if ( (RegistroPila.Clave) = (RegistroIndice.Clave) ) then
  begin
   posDatos:= RegistroPila.Posicion;
   LO_Datos_Asientos.Datos_Capturar(Unit1.ME_CuentasCorrientes.Datos, posDatos, RegistroDatosAsiento);
   if (StrToDate(RegistroDatosAsiento.fecha) >= fecha1) and (StrToDate(RegistroDatosAsiento.fecha) <= fecha2) then
   begin
    bCorte:= true;
    posDatos:= RegistroIndice.Posicion;
    LO_Datos_Clientes.Datos_Capturar(Unit1.ME_Clientes.Datos, posdatos, RegistroDatosCliente);
    Form_Clientes.StringGrid1.RowCount:=  Form_Clientes.StringGrid1.RowCount + 1;



    Form_Clientes.StringGrid1.Cells[0,i]:= RegistroDatosCliente.Clave;
    Form_Clientes.StringGrid1.Cells[1,i]:= RegistroDatosCliente.Nombre;
    Form_Clientes.StringGrid1.Cells[2,i]:= RegistroDatosCliente.Dni;
    i:=i+1;
   end
   else
   begin
    Pila_Apilar(pilaAux, RegistroPila);
    Pila_Desapilar(Unit1.ME_CuentasCorrientes.Pila);
   end
  end
  else
  begin
   Pila_Apilar(pilaAux, RegistroPila);
   Pila_Desapilar(Unit1.ME_CuentasCorrientes.Pila);
```

```
    end
  end;//While
  //Volver a dejar pila como estaba:
  While ( Pila_Vacia(pilaAux) = false) do
  begin
    Pila_Tope(pilaAux, RegistroPila);
    Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, registroPila);
    Pila_Desapilar(pilaAux);
  end ;
  posicion:= RegistroIndice.Sig;
 end;//while
 LO_Pila.Pila_Destruir(pilaAux);
end;


procedure TForm_Clientes.DateTimePicker1Change(Sender: TObject);
begin
  if Form_Clientes.DateTimePicker1.Date < Form_Clientes.DateTimePicker2.Date then
 begin
   ListarPorFechas( Form_Clientes.DateTimePicker1.Date, Form_Clientes.DateTimePicker2.Date );
 end
end;

procedure TForm_Clientes.DateTimePicker2Change(Sender: TObject);
begin
 if Form_Clientes.DateTimePicker1.Date < Form_Clientes.DateTimePicker2.Date then
  begin
   ListarPorFechas( Form_Clientes.DateTimePicker1.Date, Form_Clientes.DateTimePicker2.Date );
  end
end;

procedure ListarClientesPorArticulo(codArticulo: longint);
var
```

```
  pos, posDatos, nroComprobante, i: longint;
  PilaAuxCC, PilaAuxDetalles: LO_DobleEnlace.Tipo_Indice;
  RegistroIndice, RegistroPilaCC, RegistroPilaDetalle: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroDatosAsiento: LO_Datos_Asientos.Tipo_Registro_Datos;
  RegistroDatosDetalle: LO_Datos_Detalles.Tipo_Registro_Datos;
  RegistroDatosCliente: LO_Datos_Clientes.Tipo_Registro_Datos;
  bCorte, bEncontrado: boolean;
begin
pos:= LO_DobleEnlace.DobleEnlace_Primero(Unit1.ME_Clientes.Indice);
  Pila_Crear(PilaAuxCC, Unit1.sRuta, 'pilaTempCC');
  Pila_Abrir(PilaAuxCC);
  Pila_Crear(PilaAuxDetalles, unit1.sRuta, 'pilaTempDetalles');
  Pila_Abrir(PilaAuxDetalles);

  LimpiarTablaClientes;
  Form_Clientes.StringGrid1.RowCount:= 1;
  i:=1;

  while pos <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.ME_Clientes.Indice) do
  begin
    DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, pos, RegistroIndice);
    bCorte:= false;

    while (Pila_Vacia(Unit1.ME_CuentasCorrientes.Pila) = false) and (bCorte = false) do
    begin
      Pila_Tope(Unit1.ME_CuentasCorrientes.Pila, RegistroPilaCC);

      if (RegistroPilaCC.Clave = RegistroIndice.Clave) then
      begin
        posDatos:= RegistroPilaCC.posicion;
        LO_Datos_Asientos.Datos_Capturar(Unit1.ME_CuentasCorrientes.Datos, posDatos, RegistroDatosAsiento);
        nroComprobante:= RegistroDatosAsiento.nroComprobante;
```

```
while (Pila_Vacia(Unit1.ME_Ventas.Detalle.Pila) = false) and (bCorte = false) do
begin
 Pila_Tope(Unit1.ME_Ventas.Detalle.Pila, RegistroPilaDetalle);

 if ( StrToINt(RegistroPilaDetalle.Clave) = nroComprobante) then
 begin
  posDatos:= RegistroPilaDetalle.Posicion;
  LO_Datos_Detalles.Datos_Capturar(Unit1.ME_Ventas.Detalle.Datos, posDatos, RegistroDatosDetalle);

  if RegistroDatosDetalle.codArticulo = codArticulo then
  begin
   bCorte:= true;
   posDatos:= RegistroIndice.Posicion;

   //Agregamos el cliente a la tabla
   LO_Datos_Clientes.Datos_Capturar(Unit1.ME_Clientes.Datos, posdatos, RegistroDatosCliente);
   Form_Clientes.StringGrid1.RowCount:=  Form_Clientes.StringGrid1.RowCount + 1;
   Form_Clientes.StringGrid1.Cells[0,i]:= RegistroDatosCliente.Clave;
   Form_Clientes.StringGrid1.Cells[1,i]:= RegistroDatosCliente.Nombre;
   Form_Clientes.StringGrid1.Cells[2,i]:= RegistroDatosCliente.Dni;
   i:= i +1;
  end
  else
  begin
   Pila_Desapilar(Unit1.ME_Ventas.Detalle.Pila);
   Pila_Apilar(PilaAuxDetalles, RegistroPilaDetalle);
  end
 end
 else
 begin
  Pila_Desapilar(Unit1.ME_Ventas.Detalle.Pila);
  Pila_Apilar(PilaAuxDetalles, RegistroPilaDetalle);
 end;
```

```
      end;//While pila detalles
    end;
    //Volvemos a dejar como estaba la pila de detalles
      while Pila_Vacia(pilaAuxDetalles) = false do
      begin
        Pila_Tope(pilaAuxDetalles, RegistroPilaDetalle);
        Pila_Apilar(Unit1.ME_Ventas.Detalle.Pila, RegistroPilaDetalle);
        Pila_Desapilar(pilaAuxDetalles);
      end;

      Pila_Desapilar(Unit1.ME_CuentasCorrientes.Pila);
      Pila_Apilar(pilaAuxCC, RegistroPilaCC);
  end;//while pila cc

  //Volvemos a dejar la pila de CC como estaba
  while Pila_Vacia(PilaAuxCC) = false do
  begin
    Pila_Tope(PilaAuxCC, RegistroPilaCC);
    Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, RegistroPilaCC);
    Pila_Desapilar(PilaAuxCC);
  end;

  pos:= RegistroIndice.Sig;
end; //while recorrer listadoble

if bCorte=false then showmessage('No se encontraron compras de este articulo');

Pila_Destruir(pilaAuxCC);
Pila_Destruir(pilaAuxDetalles);

end;
```

```
procedure TForm_Clientes.Edit4Change(Sender: TObject);
var
  s: String;
  n: longint;
begin
  s:= Form_Clientes.Edit4.Text;

  if ( Length(s) = 4 ) and ( TryStrToInt(s, n) ) then
  begin
    ListarClientesPorArticulo( n );
  end
end;
end.
```

UNIT 3

```
unit Unit3;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls, Buttons, LO_DobleEnlace, LO_Datos_Rubros,
  Grids, LO_ABB, LO_Datos_Articulos;
type
  TForm_Rubros = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Button1: TButton;
    GroupBox1: TGroupBox;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
```

```
    Label5: TLabel;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    BitBtn1: TBitBtn;
    BitBtn2: TBitBtn;
    BitBtn3: TBitBtn;
    StringGrid1: TStringGrid;
    BitBtn4: TBitBtn;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Rubros: TForm_Rubros;
implementation
uses Unit1, Unit4;
{$R *.dfm}

procedure LimpiarAbmRubros;
begin
  Form_Rubros.Edit1.Clear;
  Form_Rubros.Edit2.Clear;
  Form_Rubros.Edit3.Clear;
```

```
  Form_Rubros.Edit4.Clear;
end;

procedure LimpiarTablaRubros;
var
  i: longint;
begin
  for i:= 1 to Form_Rubros.StringGrid1.RowCount  do
  begin
    Form_Rubros.StringGrid1.Rows[i].Clear;
  end;

end;

procedure ContarArticulos( var ME_Articulos: unit1.Tipo_ME_Articulos; posNodo: LO_ABB.Tipo_posicion; claveRubro:string; Var
Suma: Longint);
var
  RegistroNodo: LO_ABB.Tipo_Registro_Indice;
  posDatos: longint;
  RegistroDatosArticulo: LO_Datos_Articulos.Tipo_Registro_Datos;
begin

  if (posNodo <> LO_ABB.Arbol_PosNula(ME_Articulos.Arbol)) then
  begin
    LO_ABB.Arbol_Capturar(ME_Articulos.Arbol, posNodo, RegistroNodo);
    posDatos:= RegistroNodo.Posicion;
    LO_Datos_Articulos.Datos_Capturar(ME_Articulos.Datos, posDatos, RegistroDatosArticulo);

    if (claveRubro = RegistroDatosArticulo.ClaveRubro) then suma:=suma+1;

    ContarArticulos(ME_Articulos, RegistroNodo.HijoIzq, claveRubro, suma);
    ContarArticulos(ME_Articulos, RegistroNodo.HijoDer, claveRubro, suma);

  end
```

```
end;

function CantArticulosEnRubro(clave: String): longint;
var
  suma: Longint;
  posRaiz: LO_ABB.Tipo_Posicion;
begin

  posRaiz:=LO_ABB.Arbol_Raiz(Unit1.ME_Articulos.Arbol) ;

  suma:=0;

  ContarArticulos(Unit1.ME_Articulos, posRaiz, clave, suma);

  CantArticulosEnRubro:= suma;




end;

procedure ListarRubros;
var
  RegistroDatosRubro: LO_Datos_Rubros.Tipo_Registro_Datos;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;

  posicion, posDatos: LO_DobleEnlace.Tipo_Posicion;

  i: longint;
begin
  LimpiarTablaRubros;
  Form_Rubros.StringGrid1.Cells[0,0]:= 'Cód.';
  Form_Rubros.StringGrid1.Cells[1,0]:= 'Nombre';
  Form_Rubros.StringGrid1.Cells[2,0]:= 'Min. OP.';
```

```
 Form_Rubros.StringGrid1.Cells[3,0]:= 'Max. a Reponer';
 Form_Rubros.StringGrid1.Cells[4,0]:= 'Cant. de articulos';


 Form_Rubros.StringGrid1.RowCount:=1;
 i:= 1;


 posicion:= LO_DobleEnlace.DobleEnlace_Primero(Unit1.ME_Rubros.Indice);
 while posicion <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.ME_Rubros.Indice) do
 begin
  LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Rubros.Indice, posicion, RegistroIndice);

  posDatos:= RegistroIndice.Posicion;

  LO_Datos_Rubros.Datos_Capturar(Unit1.ME_Rubros.Datos, posDatos, RegistroDatosRubro);

  Form_Rubros.StringGrid1.RowCount:= Form_Rubros.StringGrid1.RowCount + 1;
  Form_Rubros.StringGrid1.Cells[0, i]:= RegistroDatosRubro.Clave;
  Form_Rubros.StringGrid1.Cells[1, i]:= RegistroDatosRubro.Nombre;
  Form_Rubros.StringGrid1.Cells[2, i]:= IntToStr(RegistroDatosRubro.minimoOperativo);
  Form_Rubros.StringGrid1.Cells[3, i]:= IntToStr(RegistroDatosRubro.maximoAReponer);
  Form_Rubros.StringGrid1.Cells[4, i]:= IntToStr(CantArticulosEnRubro(RegistroDatosRubro.Clave));

  i:=i+1;
  posicion:= RegistroIndice.Sig;
 end;


end;  //End ListarRubros

function VerificarMinMax(sMinimo, sMaximo: string): boolean;
var
```

```
  bResultado: boolean;
  letra: string;
  n,m: integer;
begin
  bResultado:= true;

  if TryStrToInt(sMinimo, n) = false then bResultado:=false ;
  if TryStrToInt(sMaximo, m) = false then bResultado:= false;

  if (bResultado = true) then
  begin
    if n>m then
    begin
      bResultado:=false;
      showmessage('El minimo operativo no puede ser mayor al máximo a reponer.')
    end;
  end else showmessage('Verifique el minimo y el máximo, deben ser numeros enteros.');

  VerificarMinMax:= bResultado;
end;

function VerificarClave(Clave: string):boolean;
var
  bResultado: boolean;
begin
  bResultado:= false;

  if ( Clave[1] >= '0' ) and (Clave[1] <= '9') then
  begin
    if ( Clave[2] >= '0') and ( Clave[2] <= '9') then
      if ( Clave[3] >= '0' ) and ( Clave[3] <= '9' ) then bResultado:=true
  end;
  VerificarClave:= bResultado;
```

```
end;

function CapturarDatos(Clave: string): boolean;
var
  Pos: LO_DobleEnlace.Tipo_Posicion;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  posDatos: LO_Datos_Rubros.Tipo_Posicion;
  RegistroDatos: LO_Datos_Rubros.Tipo_Registro_Datos;
  bResultado: boolean;
begin
  bResultado:= false;

  if (Lo_DobleEnlace.DobleEnlace_Buscar_Mejorada(Unit1.Me_Rubros.Indice, clave, pos)=true) then
  begin
    LO_DobleEnlace.DobleEnlace_Capturar(Unit1.Me_Rubros.Indice,pos, RegistroIndice);
    posDatos:= RegistroIndice.Posicion;

    LO_Datos_Rubros.Datos_Capturar(Unit1.ME_Rubros.Datos, posDatos, RegistroDatos);

    Form_Rubros.Edit2.Text:= RegistroDatos.Nombre;
    Form_Rubros.Edit3.Text:= IntToStr(RegistroDatos.minimoOperativo);
    Form_Rubros.Edit4.Text:= IntToStr(RegistroDatos.maximoAReponer);


    Form_Rubros.BitBtn1.Enabled:=false;
    Form_Rubros.BitBtn2.Enabled:=true;
    Form_Rubros.BitBtn3.Enabled:=true;

    bResultado:=true;
  end;

  CapturarDatos:=bResultado;
end;
```

```
procedure TForm_Rubros.Button1Click(Sender: TObject);
begin
  Form_Rubros.Close;
  Form_MenuPrincipal.show;
end;

procedure TForm_Rubros.FormCreate(Sender: TObject);
begin
  ListarRubros;
  edit1.MaxLength:=3;
  edit2.MaxLength:=40;
end;

procedure TForm_Rubros.Edit1Change(Sender: TObject);
var
  Clave: string;
begin
  clave:= edit1.Text;

  if Length(Clave) = 1 then
  begin
    if (Clave <'0') or (Clave >'9') then Form_Rubros.Edit1.Clear;
  end;

  if (Length(Clave) <> 3) then
  begin
    Form_Rubros.BitBtn1.Enabled:=false;
    Form_Rubros.BitBtn2.Enabled:=false;
    Form_Rubros.BitBtn3.Enabled:=False;

    Form_Rubros.Edit2.Clear;
    Form_Rubros.Edit3.Clear;
```

```
    Form_Rubros.Edit4.Clear;
  end;

  if (Length(Clave) = 3) then
  begin
    if (VerificarClave(Clave) = true) then
    begin
      if CapturarDatos(Clave) = false then Form_Rubros.BitBtn1.Enabled:=true;
    end else Showmessage('Se ha ingresado una clave incorrecta. (El formato debe ser "123") ');
  end;

end;

procedure ActualizarRubrosEnArticulos(clave: String);

begin
    Form_Articulos.ComboBox1.AddItem(clave, Form_Articulos.ComboBox1);
    Form_Articulos.ComboBox2.AddItem(clave, Form_Articulos.ComboBox2);
end;

procedure TForm_Rubros.BitBtn1Click(Sender: TObject);
var
  RegistroDatos: LO_Datos_Rubros.Tipo_Registro_Datos;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  clave: LO_Datos_Rubros.Tipo_Clave;
  nombre: LO_Datos_Rubros.Tipo_Nombre;
  sMinimo, sMaximo: string;
  pos: LO_DobleEnlace.Tipo_Posicion;
  posDatos: integer;

begin
  clave:= Form_Rubros.Edit1.Text;
  nombre:= Form_Rubros.Edit2.Text;
```

```
 sMinimo:= Form_Rubros.Edit3.Text;
 sMaximo:= Form_Rubros.Edit4.Text;
 if (nombre='') or (sMinimo='') or (sMaximo='') then showmessage('Por favor, ingrese todos los campos')
 else
 begin
  if ((VerificarMinMax(sMinimo, sMaximo)) = true) then

  begin
    RegistroDatos.Clave:=clave;
    RegistroDatos.Nombre:=nombre;
    RegistroDatos.minimoOperativo:=StrToInt(sMinimo);
    RegistroDatos.maximoAReponer:= StrToInt(sMaximo);
    posDatos:= FileSize(Unit1.ME_Rubros.Datos.D);
    RegistroIndice.Clave:=clave;
    RegistroIndice.Posicion:= posDatos;

    if (LO_DobleEnlace.DobleEnlace_Buscar_Mejorada(Unit1.ME_Rubros.Indice, clave, pos) = false ) then
    begin

      LO_Datos_Rubros.Datos_Insertar(Unit1.Me_Rubros.Datos, RegistroDatos);
      LO_DobleEnlace.DobleEnlace_Insertar(Unit1.Me_Rubros.Indice, pos, RegistroIndice );

      Showmessage('Se ha insertado el rubro.');

      ActualizarRubrosEnArticulos(RegistroIndice.Clave);
      LimpiarAbmRubros;
      ListarRubros;
    end
    //SINO NO PODRA INSERTARLO PORQUE EL BOTON INSERTAR SE DESHABILITARÄ
  end
 end
end;
```

```
function AutorizarEliminacionRubro(clave: string): boolean;
var
  bResultado: boolean;
begin

  if CantArticulosEnRubro(clave) = 0 then bResultado:= true else bResultado:= false;

  AutorizarEliminacionRubro:= bResultado;
end;

procedure TForm_Rubros.BitBtn2Click(Sender: TObject);
var
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  clave: LO_Datos_Rubros.Tipo_Clave;
  pos: LO_DobleEnlace.Tipo_Posicion;
  posDatos: integer;
begin
  clave:= Form_Rubros.Edit1.Text;

  if (LO_DobleEnlace.DobleEnlace_Buscar_Mejorada(Unit1.ME_Rubros.Indice, clave, pos) = true ) then
  begin
    LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Rubros.Indice, pos, RegistroIndice);
    posDatos:= RegistroIndice.Posicion;

    if Dialogs.MessageDlg('¿Está seguro que desea eliminar este rubro?', mtConfirmation,[mbYes,mbNo], 0) = mrYes then
    begin
      if (AutorizarEliminacionRubro(clave) = true) then
      begin
        LO_Datos_Rubros.Datos_Eliminar(Unit1.ME_Rubros.Datos, posDatos);
        LO_DobleEnlace.DobleEnlace_Eliminar(Unit1.ME_Rubros.Indice, pos);

        Showmessage('Se ha eliminado un rubro.');
```

```
      LimpiarAbmRubros;
      ListarRubros;
     end
     else
      showmessage('No se puede eliminar este rubro. Tiene artículos asociados');
   end;
 end;



end;

procedure TForm_Rubros.BitBtn3Click(Sender: TObject);
var
 RegistroDatos: LO_Datos_Rubros.Tipo_Registro_Datos;
 RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
 clave: LO_Datos_Rubros.Tipo_Clave;
 nombre: LO_Datos_Rubros.Tipo_Nombre;
 pos: LO_DobleEnlace.Tipo_Posicion;
 posDatos: integer;
 sMinimo, sMaximo: String;
begin
 clave:= Form_Rubros.Edit1.Text;
 nombre:= Form_Rubros.Edit2.Text;
 sMinimo:= Form_Rubros.Edit3.Text;
 sMaximo:= Form_Rubros.Edit4.Text;

 if (nombre='') or (sMinimo='') or (sMaximo='') then showmessage('Por favor, ingrese todos los campos')
 else
 begin
  if (VerificarMinMax(sMinimo,sMaximo) = false) then showmessage('Verificar minimo operativo y maximo a reponer. (Deben ser
numeros enteros)')
   else
   begin
     RegistroDatos.Clave:=clave;
```

```
      RegistroDatos.Nombre:=nombre;
      RegistroDatos.minimoOperativo:= StrToInt(sMinimo);
      RegistroDatos.maximoAReponer:= StrToInt(sMaximo);

      if (LO_DobleEnlace.DobleEnlace_Buscar_Mejorada(Unit1.ME_Rubros.Indice, clave, pos) = true ) then
      begin
        LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Rubros.Indice, pos, RegistroIndice);
        posDatos:= RegistroIndice.Posicion;
        LO_Datos_Rubros.Datos_Modificar(Unit1.ME_Rubros.Datos, posDatos, RegistroDatos);
        Showmessage('Se ha modificado el rubro.');
        LimpiarABMRubros;
        ListarRubros;
      end
    end
  end
end;
procedure TForm_Rubros.BitBtn4Click(Sender: TObject);
begin
  ListarRubros;
end;

end.
```

UNIT 4

```
unit Unit4;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls, Buttons, LO_DobleEnlace, LO_ABB, LO_Datos_Articulos,
  Grids, LO_Pila, LO_Datos_Detalles;
type
  TForm_Articulos = class(TForm)
```

```
Button1: TButton;
Image1: TImage;
GroupBox1: TGroupBox;
Label1: TLabel;
Edit1: TEdit;
Label2: TLabel;
ComboBox1: TComboBox;
Label3: TLabel;
Label4: TLabel;
Memo1: TMemo;
Label5: TLabel;
Edit2: TEdit;
Label6: TLabel;
Edit3: TEdit;
Label7: TLabel;
Edit4: TEdit;
BitBtn1: TBitBtn;
BitBtn2: TBitBtn;
BitBtn3: TBitBtn;
StringGrid1: TStringGrid;
Label8: TLabel;
ComboBox2: TComboBox;
Button2: TButton;
procedure Button1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure Edit1Change(Sender: TObject);
procedure ComboBox1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure ComboBox1KeyPress(Sender: TObject; var Key: Char);
procedure ComboBox1KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
procedure BitBtn2Click(Sender: TObject);
```

```
    procedure BitBtn3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure ComboBox2Change(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form_Articulos: TForm_Articulos;

implementation
uses Unit1, Unit5, Unit8, Unit6, Unit3
{$R *.dfm}

procedure LimpiarTablaArticulos;
var
  i: longint;
begin
  for i:= 1 to Form_Articulos.StringGrid1.RowCount  do
  begin
    Form_Articulos.StringGrid1.Rows[i].Clear;
  end;
end;

procedure mostrarDatosArticulo(posNodo: LO_ABB.Tipo_Posicion);
var
  posDatos: LO_Datos_Articulos.Tipo_posicion;
  RegistroDatos: LO_Datos_Articulos.Tipo_Registro_Datos;
  clave, codRubro, desc, precioCosto, precioUnitario, stock: string;
  i: longint;
begin
```

```
  posDatos:= posNodo;

  LO_Datos_Articulos.Datos_Capturar(Unit1.Me_articulos.Datos, posDatos, RegistroDatos);
  clave:= IntToStr(RegistroDatos.Clave);
  codRubro:= RegistroDatos.ClaveRubro;
  desc:= RegistroDatos.Descripcion;
  precioCosto:= FloatToStr(RegistroDatos.PrecioCosto);
  precioUnitario:= FloatToStr(RegistroDatos.PrecioUnitario);
  stock:= IntToStr(RegistroDatos.stock);

  Form_Articulos.StringGrid1.RowCount:= Form_Articulos.StringGrid1.RowCount+1;
  i:= Form_Articulos.StringGrid1.RowCount-1;

  Form_articulos.StringGrid1.Cells[0,i]:= clave;
  Form_articulos.StringGrid1.Cells[1,i]:= codRubro;
  Form_articulos.StringGrid1.Cells[2,i]:= desc;
  Form_articulos.StringGrid1.Cells[3,i]:= '$ '+precioCosto;
  Form_articulos.StringGrid1.Cells[4,i]:= '$ '+precioUnitario;
  Form_articulos.StringGrid1.Cells[5,i]:= stock;
end;

procedure MostrarArbolInOrden(Arbol: LO_ABB.Tipo_Arbol; posNodo: integer);
var
  RegistroArbol: LO_ABB.Tipo_REgistro_Indice;
begin
  if (posNodo <> -1) then
  begin
    LO_ABB.Arbol_Capturar(Unit1.ME_Articulos.Arbol, posNodo, RegistroArbol);

    MostrarArbolInOrden(Arbol, RegistroArbol.HijoIzq);

    mostrarDatosArticulo(RegistroArbol.Posicion);
```

```
    MostrarArbolInOrden(Arbol, RegistroARbol.HijoDer)
  end
end;

procedure ListarArticulos;
begin
  LimpiarTablaArticulos;


  Form_Articulos.StringGrid1.Cells[0,0]:= 'CÓD.';
  Form_Articulos.StringGrid1.Cells[1,0]:= 'RUBRO';
  Form_Articulos.StringGrid1.Cells[2,0]:= 'DESCRIPCION';
  Form_Articulos.StringGrid1.Cells[3,0]:= 'PRECIO COSTO';
  Form_Articulos.StringGrid1.Cells[4,0]:= 'PRECIO U.';
  Form_Articulos.StringGrid1.Cells[5,0]:= 'STOCK';


  Form_articulos.StringGrid1.RowCount:=1;

  if (LO_ABB.Arbol_Vacio(Unit1.ME_Articulos.Arbol) = false) then
  MostrarArbolInOrden(Unit1.Me_Articulos.Arbol, LO_ABB.Arbol_Raiz(Unit1.ME_Articulos.Arbol));

end;

procedure TForm_Articulos.Button1Click(Sender: TObject);
begin
  Form_Articulos.Close;
  Form_MenuPrincipal.show;
end;

procedure CargarRubros; //Toma los elementos que hay actualmente en el archivo de rubros y llena el comboBox con las claves
actuales
var
  pos: LO_DobleEnlace.Tipo_Posicion;
```

```
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
begin
  if (LO_DobleEnlace.DobleEnlace_Vacio(Unit1.Me_Rubros.Indice) = false) then
  begin
    Form_Articulos.ComboBox1.Items.Clear;
    Form_Articulos.ComboBox2.Items.Clear;
    Form_Articulos.ComboBox2.AddItem('-- VER TODOS --', Form_Articulos.ComboBox2);
    pos:= LO_DobleEnlace.DobleEnlace_Primero(Unit1.Me_Rubros.Indice);

    while pos <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.Me_Rubros.Indice) do
    begin
      LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Rubros.Indice, pos, RegistroIndice);
      Form_Articulos.ComboBox1.AddItem(RegistroIndice.Clave, Form_Articulos.ComboBox1);
      Form_Articulos.ComboBox2.AddItem(RegistroIndice.Clave, Form_Articulos.ComboBox2);
      pos:= RegistroIndice.Sig;
    end
  end
end;

procedure TForm_Articulos.FormCreate(Sender: TObject);
begin
  Form_Articulos.Memo1.Text:= '';
  Form_Articulos.Edit1.MaxLength:= 4;
  Form_Articulos.Memo1.MaxLength:= 80;


  ListarArticulos;
  CargarRubros;
end;

procedure TForm_Articulos.BitBtn1Click(Sender: TObject);
var
  numero: double;
```

```
  RegistroArbol: LO_ABB.Tipo_Registro_Indice;
  RegistroDatos: LO_Datos_Articulos.Tipo_Registro_Datos;
  pos, posDatos: LO_ABB.Tipo_posicion;
  Nivel: LO_ABB.Tipo_Cantidad;
begin
  if (Form_Articulos.Edit1.Text='') or (Form_Articulos.ComboBox1.Text='') or (Form_Articulos.Memo1.Text='') or
(Form_Articulos.Edit2.Text='') or (Form_Articulos.Edit3.Text='') or (Form_Articulos.Edit4.Text='') then
  begin
    showmessage('Por favor, ingrese todos los campos');
  end
  else
  begin
    if TryStrToFloat(Edit1.Text, numero) = false then  showmessage('No ha insertado una clave numerica (1000 .. 9999')
    else
      if (TryStrToFloat(Edit2.Text, numero) = false) or  (TryStrToFloat(Edit3.Text, numero) = false) or (TryStrToFloat(Edit4.Text,
numero) = false) then showmessage('Verifique los precios y el stock.')
      else
      begin
        RegistroDatos.Clave:= StrToInt(Edit1.Text);
        RegistroDatos.ClaveRubro:= ComboBox1.Text;
        RegistroDatos.Descripcion:= Memo1.Text;
        RegistroDatos.PrecioCosto:= StrToFloat(Edit2.Text);
        RegistroDatos.PrecioUnitario:= StrToFloat(Edit3.Text);
        RegistroDatos.stock:= StrToInt(Edit4.Text);
        RegistroDatos.Borrado:= false;

        posDatos:= FileSize(Unit1.ME_Articulos.Datos.D);
        RegistroArbol.Clave:= StrToInt(Edit1.Text);
        RegistroArbol.Posicion:= posDatos;

        if (LO_ABB.Arbol_Buscar(Unit1.Me_Articulos.Arbol, RegistroArbol.Clave, pos, Nivel) = true) then showmessage('La clave ya
existe')
        else
        begin
```

```
        LO_Datos_Articulos.Datos_Insertar(Unit1.ME_Articulos.Datos, RegistroDatos);
        LO_ABB.Arbol_Insertar(ME_Articulos.Arbol, pos, RegistroArbol, Nivel);
        showmessage('Se ha insertado un articulo.');
        Form_Mantenimiento.TreeView1.Items.Clear;
        Form_Mantenimiento.ListBox1.Clear;
        Form_Articulos.Edit1.Clear;
        Form_Articulos.ComboBox1.Clear;
        Form_Articulos.Memo1.Clear;
        Form_Articulos.Edit2.Clear;
        Form_Articulos.Edit3.Clear;
        Form_Articulos.Edit4.Clear;
        ListarArticulos;
      end

    end
  end
end;

procedure CapturarDatosArticulos(pos: integer);
var
  RegistroArbol: LO_ABB.Tipo_Registro_Indice;
  posDatos: LO_Datos_Articulos.Tipo_Posicion;
  RegistroDatos: LO_Datos_Articulos.Tipo_Registro_Datos;
begin


  LO_ABB.Arbol_Capturar(Unit1.ME_Articulos.Arbol ,pos, RegistroArbol);
  posDatos:= RegistroArbol.Posicion;

  LO_Datos_Articulos.Datos_Capturar(Unit1.ME_Articulos.Datos, posDatos, RegistroDatos);

  Form_Articulos.ComboBox1.Text:= RegistroDatos.ClaveRubro;
  Form_Articulos.Memo1.Text:= REgistroDatos.Descripcion;
```

```
  Form_Articulos.Edit2.Text:= FloatToStr(RegistroDatos.PrecioCosto);
  Form_Articulos.Edit3.Text:= FloatToStr(RegistroDatos.PrecioUnitario);
  Form_Articulos.Edit4.Text:= IntToStr(RegistroDatos.stock);
end;

procedure LimpiarAbmArticulos;
begin
    Form_Articulos.ComboBox1.Text:='';
    Form_Articulos.Memo1.Text:='';
    Form_Articulos.Edit2.Text:='';
    Form_Articulos.Edit3.Text:='';
    Form_Articulos.Edit4.Text:='';
end;

procedure TForm_Articulos.Edit1Change(Sender: TObject);
var
  sNumero: string;
  nNumero: integer;
  numero: double;
  pos: LO_ABB.Tipo_Posicion;
  nivel: LO_ABB.Tipo_Cantidad;
begin
  sNumero:= Form_Articulos.Edit1.Text;
  if  (Length(sNumero) = 1) and (sNumero = '0') then Form_Articulos.Edit1.clear;
  if (Length(sNumero) = 1) and ((sNumero < '1') or (sNumero > '9') ) then Form_Articulos.Edit1.Clear;



  if( Length(sNumero) = 4 ) and ( TryStrToFloat(sNumero,numero) ) then
  begin
    nNumero:= StrToInt(sNumero);
    if (LO_ABB.Arbol_Buscar(Unit1.Me_Articulos.Arbol, nNumero, pos, nivel) = false ) then
    begin
```

```
      Form_Articulos.BitBtn1.Enabled:= true;
      Form_Articulos.BitBtn2.Enabled:=false;
      Form_Articulos.BitBtn3.Enabled:=false;
    end
    else
    begin
      CapturarDatosArticulos(pos);
      Form_Articulos.BitBtn1.Enabled:= false;
      Form_Articulos.BitBtn2.Enabled:= true;
      Form_Articulos.BitBtn3.Enabled:= true;
    end
  end
  else
  begin
    Form_Articulos.BitBtn1.Enabled:=false;
    Form_Articulos.BitBtn2.Enabled:=false;
    Form_Articulos.BitBtn3.Enabled:=false;
    LimpiarAbmArticulos;
    CargarRubros;
  end
end;

procedure TForm_Articulos.ComboBox1KeyDown(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  Form_Articulos.ComboBox1.Text:='';
end;

procedure TForm_Articulos.ComboBox1KeyPress(Sender: TObject;
  var Key: Char);
begin
  Form_Articulos.ComboBox1.Text:='';
end;
```

```
procedure TForm_Articulos.ComboBox1KeyUp(Sender: TObject; var Key: Word;
  Shift: TShiftState);
begin
  Form_Articulos.ComboBox1.Text:='';
end;

function VerificarEliminacionArticulo(clave: integer): boolean;
var
  bResultado, bCorte: boolean;
  RegistroPila: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroDatosDetalle: LO_Datos_Detalles.Tipo_Registro_Datos;
  pilaAuxDet: LO_DobleEnlace.Tipo_Indice;
  posDatos: longint;

begin
  Pila_Crear(PilaAuxDet, unit1.sRuta, 'pilaTempDet');
  Pila_Abrir(pilaAuxDet);
  bCorte:= false;

  while (Pila_Vacia(Unit1.ME_Ventas.Detalle.Pila) = false) and (bCorte=false) do
  begin
    Pila_Tope(Unit1.ME_Ventas.Detalle.Pila, RegistroPila);

    posDatos:= RegistroPila.Posicion;
    LO_Datos_Detalles.Datos_Capturar(Unit1.ME_Ventas.Detalle.Datos, posDatos, RegistroDatosDetalle);

    if clave = (RegistroDatosDetalle.codArticulo) then
    bCorte:= true;
    Pila_Apilar(pilaAuxDet, RegistroPila);
    Pila_Desapilar(Unit1.ME_Ventas.Detalle.Pila);
  end;
  if (bCorte = true) then
```

```
  bResultado:= false //No se deja eliminar
  else
  bResultado:= true; //Se deja eliminar

  //Volvemos a dejar la pila de detalles como estaba
  while (Pila_Vacia(pilaAuxDet) = false) do
  begin
    Pila_Tope(pilaAuxDet, RegistroPila);
    Pila_Apilar(Unit1.ME_Ventas.Detalle.Pila, RegistroPila);
    Pila_Desapilar(pilaAuxDet);
  end;
  Pila_Destruir(pilaAuxDet);
  VerificarEliminacionArticulo:= bResultado;
end;

procedure TForm_Articulos.BitBtn2Click(Sender: TObject);
var
  RegistroArbol: LO_ABB.Tipo_Registro_Indice;
  clave: LO_Datos_Articulos.Tipo_Clave;
  pos: LO_ABB.Tipo_Posicion;
  nivel: LO_ABB.Tipo_Cantidad;
  posDatos: integer;
begin
  clave:= StrToInt ( Form_Articulos.Edit1.Text );

  if (LO_ABB.Arbol_Buscar(Unit1.ME_Articulos.Arbol, clave, pos, nivel) = true ) then
  begin
    LO_ABB.Arbol_Capturar(Unit1.ME_Articulos.Arbol, pos, RegistroArbol);
    posDatos:= RegistroArbol.Posicion;

    if Dialogs.MessageDlg('¿Está seguro que desea eliminar este articulo?', mtConfirmation,[mbYes,mbNo], 0) = mrYes then
    begin
      if (VerificarEliminacionArticulo(clave) = true) then
```

```
      begin
        LO_Datos_Articulos.Datos_Eliminar(Unit1.ME_Articulos.Datos, posDatos);
        LO_ABB.Arbol_Eliminar(Unit1.ME_Articulos.Arbol, pos, nivel);
        Showmessage('Se ha eliminado un articulo.');
        LimpiarAbmArticulos;
        ListarArticulos;
      end else showmessage('No se puede eliminar el artículo, tiene facturas asociadas');
    end;
  end;
end;

procedure TForm_Articulos.BitBtn3Click(Sender: TObject);
var
  clave, claveRubro, descripcion, precioCosto, precioUnitario, stock: string;
  nPrecioCosto, nPrecioUnitario : real;
  numero: double;
  nStock, numeroInt,nClave: integer;
  RegistroArbol: LO_ABB.Tipo_Registro_Indice;
  RegistroDatos: LO_Datos_Articulos.Tipo_Registro_Datos;
  posDatos: LO_Datos_Articulos.Tipo_Posicion;
  pos: LO_ABB.Tipo_posicion;
  Nivel: LO_ABB.Tipo_Cantidad;
begin
  clave:= Form_Articulos.Edit1.Text;
  claveRubro:= Form_Articulos.ComboBox1.Text;
  descripcion:= Form_Articulos.Memo1.Text;
  precioCosto:= (Form_articulos.Edit2.Text);
  precioUnitario:= (Form_articulos.Edit3.Text);
  stock:= (Form_articulos.Edit4.Text);

  if (claveRubro = '') or (descripcion = '') or (precioCosto = '') or (precioUnitario = '') or (stock = '') then Showmessage('Por favor,
complete todos los campos.')
  else
  begin
```

```
    if (TryStrToFloat(precioCosto, numero) = false) or  (TryStrToFloat(precioUnitario, numero) = false) or (TryStrToInt(stock,
numeroInt) = false) then showmessage('Verifique los precios y el stock.')
    else
    begin
      nPrecioCosto:= StrToFloat(precioCosto);
      nPrecioUnitario:= StrToFloat(precioUnitario);
      nStock:= StrToInt(stock);
      nClave:= StrToInt(clave);
      LO_ABB.Arbol_Buscar(Unit1.Me_Articulos.Arbol, nClave, pos, nivel);
      LO_ABB.Arbol_Capturar(Unit1.Me_Articulos.Arbol, pos, RegistroArbol);
      posDatos:= RegistroArbol.Posicion;
      RegistroDatos.Clave:= nClave;
      RegistroDatos.ClaveRubro:= ClaveRubro;
      RegistroDatos.Descripcion:= Descripcion;
      RegistroDatos.PrecioCosto:= nPrecioCosto;
      RegistroDatos.PrecioUnitario:= nPrecioUnitario;
      RegistroDatos.stock:= nStock ;
      LO_Datos_Articulos.Datos_Modificar(Unit1.Me_Articulos.Datos, posDatos, RegistroDatos);
      showmessage('Se ha modificado el articulo!');
      ListarArticulos;
    end
  end
end;


procedure TForm_Articulos.Button2Click(Sender: TObject);
var
  fecha: string;
begin
  fecha:= DateToStr(now);
  Form_Reporte_Articulos.QRLabel_Fecha.caption:= fecha;
  Form_Reporte_Articulos.QuickRep1.Preview;
end;
```

```
procedure RecorrerArbol(var ME_Articulos: Unit1.Tipo_ME_Articulos; posNodo: LO_ABB.Tipo_Posicion; clave: string);
var
  RegistroNodo: LO_ABB.Tipo_registro_indice;
  RegistroDatosArticulo: LO_Datos_Articulos.Tipo_Registro_datos;
  posDatos, i: longint;
begin
  if (posNodo <> LO_ABB.Arbol_PosNula(ME_articulos.Arbol)) then
  begin
    LO_ABB.Arbol_Capturar(ME_Articulos.Arbol, posNodo, RegistroNodo);
    posDatos:= registroNodo.Posicion;
    LO_Datos_articulos.Datos_Capturar(ME_Articulos.Datos, posDatos, registroDatosArticulo);



    if (Clave = registroDatosArticulo.ClaveRubro) then
    begin
      Form_Articulos.StringGrid1.RowCount:= Form_Articulos.StringGrid1.RowCount + 1;
      i:= Form_Articulos.StringGrid1.RowCount-1;
      Form_Articulos.StringGrid1.Cells[0, i]:= IntToStr(RegistroDatosArticulo.Clave);
      Form_Articulos.StringGrid1.Cells[1, i]:= (RegistroDatosArticulo.ClaveRubro);
      Form_Articulos.StringGrid1.Cells[2, i]:= (RegistroDatosArticulo.Descripcion);
      Form_Articulos.StringGrid1.Cells[3, i]:= '$ '+FloatToStr(RegistroDatosArticulo.PrecioCosto);
      Form_Articulos.StringGrid1.Cells[4, i]:= '$ '+FloatToStr(RegistroDatosArticulo.PrecioUnitario);
      Form_Articulos.StringGrid1.Cells[5, i]:= intToStr(RegistroDatosArticulo.stock);
    end;
    RecorrerArbol(ME_Articulos, RegistroNodo.HijoIzq, clave);
    RecorrerArbol(ME_Articulos, RegistroNodo.HijoDer, clave);
  end;
end;

procedure ListarArticulosPorRubro(clave: String);
var
  posRaiz: LO_ABB.Tipo_Posicion;
begin
```

```
  posRaiz:= LO_ABB.Arbol_Raiz(Unit1.Me_Articulos.Arbol);
  LimpiarTablaArticulos;
  Form_articulos.StringGrid1.RowCount:=1;
  RecorrerArbol(Unit1.ME_Articulos, posRaiz, clave);
end;

procedure TForm_Articulos.ComboBox2Change(Sender: TObject);
var
  s: String;
begin
  s:= Form_Articulos.ComboBox2.Text;

  if s = '-- VER TODOS --' then ListarArticulos
  else
  ListarArticulosPorRubro(s);
end;
end.
```

UNIT 5

```
unit Unit5;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ComCtrls, LO_ABB, LO_DobleEnlace, LO_Datos_Rubros, LO_Datos_Articulos,
  jpeg, ExtCtrls, LO_Datos_Clientes, LO_Datos_Asientos, LO_Datos_Detalles, LO_Datos_Comprobantes,
  LO_Pila, LO_Cola;
type
  TForm_Mantenimiento = class(TForm)
    TreeView1: TTreeView;
    RadioButton1: TRadioButton;
    RadioButton2: TRadioButton;
```

```
RadioButton3: TRadioButton;
RadioButton4: TRadioButton;
Button2: TButton;
Button3: TButton;
ListBox1: TListBox;
Button4: TButton;
Button5: TButton;
Button6: TButton;
Button7: TButton;
Button8: TButton;
Button9: TButton;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
procedure Button9Click(Sender: TObject);
procedure RadioButton4Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure RadioButton1Click(Sender: TObject);
procedure RadioButton2Click(Sender: TObject);
procedure RadioButton3Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Button8Click(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
```

```
    procedure Button5Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form_Mantenimiento: TForm_Mantenimiento;

implementation

uses Unit1, Unit3, Unit4;
{$R *.dfm}
procedure TForm_Mantenimiento.Button9Click(Sender: TObject);
begin
  Form_Mantenimiento.RadioButton1.SetFocus;
  Form_MenuPrincipal.Show;
  Form_Mantenimiento.Close;
end;
procedure recorrerArbol(var Arbol: LO_ABB.Tipo_Arbol; posNodo: LO_ABB.Tipo_Posicion);
var
  RegistroNodo: LO_ABB.Tipo_Registro_Indice;
  node, nodeAux: TTreeNode;
begin
  if posNodo <> LO_ABB.Arbol_PosNula(Unit1.ME_Articulos.Arbol) then
  begin
    LO_ABB.Arbol_Capturar(Unit1.ME_Articulos.Arbol, posNodo, RegistroNodo);

    node:= Form_Mantenimiento.TreeView1.Items.AddChild(Form_Mantenimiento.TreeView1.Selected,
IntToStr(RegistroNodo.clave));
    nodeAux:=node;
    Form_Mantenimiento.TreeView1.Select(node);
    recorrerArbol(Arbol, RegistroNodo.HijoIzq);
    Form_Mantenimiento.TreeView1.Select(nodeAux);
```

```
    recorrerArbol(Arbol, RegistroNodo.HijoDer);
  end
end;

procedure DibujarArbol;
var
  posRaiz: LO_ABB.Tipo_Posicion;
begin
    posRaiz:= LO_Abb.Arbol_Raiz(Unit1.Me_Articulos.Arbol);
    Form_Mantenimiento.TreeView1.Items.Clear;
    recorrerArbol(Unit1.ME_Articulos.Arbol, posRaiz);
end;

procedure ListarNiveles;
var
  i: integer;
  RegistroNivel: LO_ABB.Tipo_Registro_Nivel;
begin
  Form_Mantenimiento.ListBox1.Clear;
  Form_Mantenimiento.ListBox1.AddItem('Niveles', Form_Mantenimiento.ListBox1);
  Form_Mantenimiento.ListBox1.AddItem('-------------------------------------------------', Form_Mantenimiento.ListBox1);

  if LO_ABB.Arbol_Vacio(Unit1.ME_Articulos.Arbol) then
  begin
    showmessage('No hay niveles que mostrar porque el arbol esta vacio.');
    Form_Mantenimiento.Label1.visible:= false;
  end
  else
  begin
    for i:= 0 to LO_Abb.Arbol_UltimoNivel(Unit1.ME_Articulos.Arbol) do
    begin
      LO_Abb.Arbol_CapturarNivel(Unit1.ME_Articulos.Arbol, i, RegistroNivel);
      Form_Mantenimiento.ListBox1.AddItem('Nivel '+IntToStr(i)+': '+ IntToStr(RegistroNivel.CantidadElementos)+' elementos.',
Form_Mantenimiento.ListBox1);
```

```
    end;
  end
end;
procedure VerificarEquilibrio ;
begin
  if (LO_ABB.Arbol_Equilibrado(Unit1.Me_Articulos.Arbol) = false) then
    begin
      Form_Mantenimiento.Button6.Enabled:= true;
      Form_Mantenimiento.Label2.Show;
      Form_Mantenimiento.Label3.Show;
      Form_Mantenimiento.Label1.Hide;
    end
    else
    begin
      Form_Mantenimiento.Button6.Enabled:= false;
      Form_Mantenimiento.Label2.hide;
      Form_Mantenimiento.Label3.hide;
      Form_Mantenimiento.Label1.show;
    end;

  if (LO_ABB.Arbol_Vacio(Unit1.ME_Articulos.Arbol)) then Form_Mantenimiento.Label1.Hide;
end ;
procedure TForm_Mantenimiento.RadioButton4Click(Sender: TObject);
begin
  Form_Mantenimiento.Button2.Visible:= false;
  Form_Mantenimiento.Button3.Visible:= false;

  Form_Mantenimiento.Button4.Visible:= false;
  Form_Mantenimiento.Button5.Visible:= false;
  Form_Mantenimiento.Button6.Visible:= true;
  Form_Mantenimiento.Button7.Visible:= true;
  Form_Mantenimiento.Button8.Visible:= true;
```

```
  Form_Mantenimiento.Label4.Show;
  DibujarArbol;
  ListarNiveles;
  VerificarEquilibrio;
end;
procedure TForm_Mantenimiento.Button3Click(Sender: TObject);
begin
   if Dialogs.MessageDlg('¿Está seguro que desea eliminar los archivos de Rubros?', mtConfirmation,[mbYes,mbNo], 0) = mrYes
then
    begin
      LO_Datos_Rubros.Datos_Destruir(Unit1.ME_Rubros.Datos, Unit1.sRuta, Unit1.sNombreRubros);
      LO_DobleEnlace.DobleEnlace_Destruir(Unit1.ME_Rubros.Indice);
      Showmessage('Se han eliminado todos los datos de Rubros.');
      Form_Articulos.ComboBox1.Items.Clear;
      LO_Datos_Rubros.Datos_Crear(Unit1.Me_Rubros.Datos, Unit1.sRuta, unit1.sNombreRubros);
      LO_DobleEnlace.DobleEnlace_Crear(Unit1.ME_Rubros.Indice, Unit1.sRuta, Unit1.sNombreRubros);
      LO_Datos_Rubros.Datos_Abrir(Unit1.Me_Rubros.Datos);
      LO_DobleEnlace.DobleEnlace_Abrir(Unit1.Me_Rubros.Indice);
    end;
end;
procedure TForm_Mantenimiento.RadioButton1Click(Sender: TObject);
begin
  Form_Mantenimiento.Button2.Visible:= true;
  Form_Mantenimiento.Button3.Visible:= true;
  Form_Mantenimiento.Button4.Visible:= false;
  Form_Mantenimiento.Button5.Visible:= false;
  Form_Mantenimiento.Button6.Visible:= false;
  Form_Mantenimiento.Button7.Visible:= false;
  Form_Mantenimiento.Button8.Visible:= false;
  Form_Mantenimiento.Label1.Hide;
  Form_Mantenimiento.Label2.Hide;
  Form_Mantenimiento.label3.Hide;
end;
```

```
procedure TForm_Mantenimiento.RadioButton2Click(Sender: TObject);
begin
  Form_Mantenimiento.Button2.Visible:= false;
  Form_Mantenimiento.Button3.Visible:= false;

  Form_Mantenimiento.Button4.Visible:= true;

  Form_Mantenimiento.Button5.Visible:= false;
  Form_Mantenimiento.Button6.Visible:= false;
  Form_Mantenimiento.Button7.Visible:= false;
  Form_Mantenimiento.Button8.Visible:= false;

  Form_Mantenimiento.Label1.Hide;
  Form_Mantenimiento.Label2.Hide;
  Form_Mantenimiento.label3.Hide;
end;

procedure TForm_Mantenimiento.RadioButton3Click(Sender: TObject);
begin
  Form_Mantenimiento.Button2.Visible:= false;
  Form_Mantenimiento.Button3.Visible:= false;

  Form_Mantenimiento.Button4.Visible:= false;
  Form_Mantenimiento.Button5.Visible:= true;
  Form_Mantenimiento.Button6.Visible:= false;
  Form_Mantenimiento.Button7.Visible:= false;
  Form_Mantenimiento.Button8.Visible:= false;

  Form_Mantenimiento.Label1.Hide;
  Form_Mantenimiento.Label2.Hide;
  Form_Mantenimiento.label3.Hide;
end;
```

```
procedure TForm_Mantenimiento.Button6Click(Sender: TObject);
begin
  if Dialogs.MessageDlg('¿Está seguro que desea rebalancear la estructura? Esto puede tardar mucho dependiendo de la cantidad de elementos.', mtConfirmation,[mbYes,mbNo], 0) = mrYes then
    begin
      LO_ABB.Arbol_Rebalanceo(Unit1.Me_Articulos.Arbol, Unit1.sRuta, Unit1.sNombreArticulos);
      showmessage('Se ha balanceado con exito!');

      Form_Mantenimiento.RadioButton1.SetFocus;
      Form_Mantenimiento.RadioButton4.SetFocus;
    end;
end;

procedure MostrarPorcentaje;
var
  tolerancia, niveles: LO_ABB.Tipo_Porcentaje;
begin
  tolerancia:= LO_ABB.Arbol_PorcentajeTolerancia(Unit1.ME_Articulos.Arbol);
  niveles:= LO_ABB.Arbol_PorcentajeNiveles(Unit1.ME_Articulos.Arbol);
  Form_Mantenimiento.Label7.Caption:= IntToStr(tolerancia);
  Form_Mantenimiento.Label8.Caption:= IntToStr(niveles);
end;

procedure TForm_Mantenimiento.FormCreate(Sender: TObject);
begin
  MostrarPorcentaje;
end;
procedure TForm_Mantenimiento.Button8Click(Sender: TObject);
var
  tolerancia, niveles: LO_ABB.Tipo_Porcentaje;
begin
  if Dialogs.MessageDlg('¿Está seguro que desea eliminar los archivos de Articulos?', mtConfirmation,[mbYes,mbNo], 0) = mrYes then
    begin
```

```
      tolerancia:= LO_ABB.Arbol_PorcentajeTolerancia(Unit1.Me_Articulos.Arbol);
      niveles:= LO_ABB.Arbol_porcentajeNiveles(Unit1.Me_Articulos.Arbol);
      LO_Datos_Articulos.Datos_Destruir(Unit1.ME_Articulos.Datos);
      LO_ABB.Arbol_Destruir(Unit1.ME_Articulos.Arbol);
      Showmessage('Se han eliminado todos los datos de Articulos.');

      Form_Mantenimiento.TreeView1.Items.Clear;
      Form_Mantenimiento.ListBox1.Clear;
      Form_Mantenimiento.Label1.Hide;
      Form_Mantenimiento.Label2.Hide;
      Form_Mantenimiento.label3.Hide;
      LO_Datos_articulos.Datos_Crear(Unit1.ME_Articulos.Datos, Unit1.sRuta, Unit1.sNombreArticulos);
      LO_ABB.Arbol_Crear(Unit1.ME_Articulos.Arbol, Unit1.sRuta, Unit1.sNombreArticulos, tolerancia, niveles);
      LO_Datos_Articulos.Datos_Abrir(Unit1.ME_Articulos.Datos);
      LO_ABB.Arbol_Abrir(UNit1.ME_ARticulos.Arbol);
    end;
end;

function VerificarPorcentaje (palabra: string): boolean;
var
  bResultado: boolean;
  letra: string;
  n: integer;
begin
  bResultado:= true;
  for n:= 1 to (Length(palabra)) do
  begin
    letra := palabra[n];
    if (letra<'0') or (letra>'9') then bResultado:=false;
  end;
  VerificarPorcentaje:= bResultado;
end;
procedure TForm_Mantenimiento.Button7Click(Sender: TObject);
```

```
var sNiveles, sTolerancia : string;
begin
   Showmessage('A continuación deberá definir los porcentajes de tolerancia a desequilibrio y niveles a controlar en balanceo.');

   repeat
   if InputQuery('Porcentajes de Árbol.','Ingrese el porcentaje de niveles a tener en cuenta al momento de determinar equilibrio.', sNiveles) = false
   then showmessage('El usuario canceló la operacion')
   until VerificarPorcentaje(sNiveles) = true;
   repeat
   if InputQuery('Porcentajes de Árbol.','Ingrese el porcentaje de tolerancia a desbalanceo de niveles.', sTolerancia) = false
   then showmessage('El usuario canceló la operacion')
   until VerificarPorcentaje(sTolerancia) = true;
   if (sNiveles <> '') then
   begin
     LO_ABB.Arbol_cambiarPorcentajeNiveles(Unit1.ME_Articulos.Arbol, StrToInt(sNiveles));
     Form_Mantenimiento.Label8.Caption:= sNiveles;
   end ;
   if (sTolerancia <> '') then
   begin
     LO_ABB.Arbol_CambiarPorcentajeTolerancia(Unit1.ME_Articulos.Arbol, StrToInt(sTolerancia));
     Form_Mantenimiento.Label7.Caption:= sTolerancia;
   end ;
   VerificarEquilibrio;
   Showmessage('Se han actualizado los porcentajes');
end;
procedure TForm_Mantenimiento.Button2Click(Sender: TObject);
begin
  if Dialogs.MessageDlg('¿Está seguro que desea eliminar los archivos de Clientes?', mtConfirmation,[mbYes,mbNo], 0) = mrYes
then
     begin
     LO_Datos_Clientes.Datos_Destruir(Unit1.ME_Clientes.Datos, Unit1.sRuta, Unit1.sNombreDatos);
      LO_DobleEnlace.DobleEnlace_Destruir(Unit1.ME_Clientes.Indice);
      Showmessage('Se han eliminado todos los datos de Clientes.');
```

```
      Form_Articulos.ComboBox1.Items.Clear;
      LO_Datos_Clientes.Datos_Crear(Unit1.ME_Clientes.Datos, Unit1.sRuta, unit1.sNombreDatos);
      LO_DobleEnlace.DobleEnlace_Crear(Unit1.ME_Clientes.Indice, Unit1.sRuta, Unit1.sNombreIndice);

      LO_Datos_Clientes.Datos_Abrir(Unit1.ME_Clientes.Datos);
      LO_DobleEnlace.DobleEnlace_Abrir(Unit1.ME_Clientes.Indice);
    end;
end;
procedure TForm_Mantenimiento.Button4Click(Sender: TObject);
begin
  if Dialogs.MessageDlg('¿Está seguro que desea eliminar los archivos de Detalles y Cuentas Corrientes?',
mtConfirmation,[mbYes,mbNo], 0) = mrYes then
    begin
      LO_Datos_Detalles.Datos_Destruir(Unit1.ME_Ventas.Detalle.Datos, Unit1.sRuta, Unit1.sNombreDetalles);
      LO_Pila.Pila_Destruir(Unit1.ME_Ventas.Detalle.Pila);
      LO_Datos_Asientos.Datos_Destruir(Unit1.ME_CuentasCOrrientes.Datos, Unit1.sRuta, unit1.sNombreCuentasCorrientes);
      LO_Pila.Pila_Destruir(Unit1.ME_CuentasCorrientes.Pila);
      Showmessage('Se han eliminado todos los datos de Detalles y Cuentas Corrientes.');
      LO_Datos_Detalles.Datos_Crear(Unit1.ME_Ventas.Detalle.Datos, Unit1.sRuta, unit1.sNombreDetalles);
      LO_Pila.Pila_Crear(Unit1.ME_Ventas.Detalle.Pila, Unit1.sRuta, unit1.sNombreDetalles);
      LO_Datos_Asientos.Datos_Crear(Unit1.ME_CuentasCorrientes.Datos, Unit1.sRuta, Unit1.sNombreCuentasCorrientes);
      LO_Pila.Pila_Crear(Unit1.ME_CuentasCorrientes.Pila, unit1.sRuta, Unit1.sNombreCuentasCorrientes);
      LO_Datos_Detalles.Datos_Abrir(Unit1.ME_Ventas.Detalle.Datos);
      LO_Pila.Pila_Abrir(Unit1.ME_Ventas.Detalle.Pila);

      LO_Datos_Asientos.Datos_Abrir(Unit1.ME_CuentasCorrientes.Datos);
      LO_Pila.Pila_Abrir(Unit1.ME_CuentasCorrientes.Pila);
    end;
end;
procedure TForm_Mantenimiento.Button5Click(Sender: TObject);
begin
 if Dialogs.MessageDlg('¿Está seguro que desea eliminar los archivos de Comprobantes?', mtConfirmation,[mbYes,mbNo], 0) =
mrYes then
     begin
```

```
    LO_Datos_Comprobantes.Datos_Destruir(Unit1.ME_Ventas.Comprobante.Datos, Unit1.sRuta,
Unit1.sNombreComprobantes);
    LO_Cola.Cola_Destruir(Unit1.ME_Ventas.Comprobante.Cola);
    Showmessage('Se han eliminado todos los datos de Comprobantes.');
    LO_Datos_Comprobantes.Datos_Crear(Unit1.ME_Ventas.Comprobante.Datos, Unit1.sRuta, unit1.sNombreComprobantes);
    LO_Cola.Cola_Crear(Unit1.ME_Ventas.Comprobante.Cola, Unit1.sRuta, Unit1.sNombreComprobantes);

    LO_Datos_Comprobantes.Datos_Abrir(Unit1.ME_Ventas.Comprobante.Datos);
    LO_Cola.Cola_Abrir(Unit1.ME_Ventas.Comprobante.Cola);
  end;
end;
end.
```

UNIT 6

```
unit Unit6;
interface
uses
 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
 Dialogs, StdCtrls, jpeg, ExtCtrls, Menus, Grids, Buttons, LO_ABB, LO_DobleEnlace,
 LO_Datos_Clientes, LO_Datos_Articulos, ComCtrls, LO_Datos_Comprobantes, LO_Datos_Detalles,
 LO_Cola, LO_Pila, LO_Datos_Asientos;

type
 TForm_Facturacion = class(TForm)
   Image1: TImage;
   Button1: TButton;
   StringGrid1: TStringGrid;
   GroupBox1: TGroupBox;
   ComboBox1: TComboBox;
   Edit1: TEdit;
   Edit2: TEdit;
   Label1: TLabel;
```

```
Label2: TLabel;
Label3: TLabel;
BitBtn1: TBitBtn;
BitBtn2: TBitBtn;
BitBtn3: TBitBtn;
Label5: TLabel;
Edit3: TEdit;
BitBtn4: TBitBtn;
Memo1: TMemo;
Label6: TLabel;
Label7: TLabel;
Label9: TLabel;
Label10: TLabel;
Label11: TLabel;
ComboBox2: TComboBox;
Label4: TLabel;
Label8: TLabel;
Edit4: TEdit;
Label12: TLabel;
ComboBox3: TComboBox;
DateTimePicker1: TDateTimePicker;
Label13: TLabel;
Label_Neto: TLabel;
Label14: TLabel;
procedure Button1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);

procedure BitBtn2Click(Sender: TObject);
procedure StringGrid1SelectCell(Sender: TObject; ACol, ARow: Integer;
  var CanSelect: Boolean);
procedure BitBtn3Click(Sender: TObject);
procedure BitBtn1Click(Sender: TObject);
procedure BitBtn4Click(Sender: TObject);
```

```
    procedure Edit1Change(Sender: TObject);
    procedure Edit3Change(Sender: TObject);
    procedure ComboBox2Change(Sender: TObject);
    procedure Edit4Change(Sender: TObject);
    procedure ComboBox3Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Facturacion: TForm_Facturacion;

implementation

uses Unit1, Unit7;

{$R *.dfm}

procedure TForm_Facturacion.Button1Click(Sender: TObject);
begin
  Form_MenuPrincipal.Show;
  Form_Facturacion.Close;
end;
function NormalizarCadena(cadena: string): string;
begin
  NormalizarCadena:= StringReplace(cadena,'$', '',[rfReplaceAll]);;
end;
procedure ActualizarNeto;
var
  i: Longint;
  total: String;
```

```
  neto: real;
begin
  neto:=0;
  for i:= 1 to  ( (Form_Facturacion.StringGrid1.RowCount) -1 ) do
  begin
     total:= Form_Facturacion.StringGrid1.Cells[4, i];
     total:= StringReplace(total, '$', '', [rfReplaceAll]);

     neto:= neto+StrToFloat(total);
  end;

  Form_Facturacion.Label_Neto.Caption:= FloatToStr(neto);
end;

procedure NombrarCeldas;
begin
  Form_Facturacion.StringGrid1.Cells[0,0]:= 'COD. ARTICULO';
  Form_Facturacion.StringGrid1.Cells[1,0]:= 'DETALLE';
  Form_Facturacion.StringGrid1.Cells[2,0]:= 'CANT.';
  Form_Facturacion.StringGrid1.Cells[3,0]:= 'PRECIO UNITARIO';
  Form_Facturacion.StringGrid1.Cells[4,0]:= 'TOTAL';

end;

procedure ActualizarClientesEnFacturacion;
var
  pos: integer;
  clave: String;
  RegistroIndiceCliente: LO_DobleEnlace.Tipo_Registro_indice;
begin
  //mostrar clientes en el combobox
  Form_Facturacion.ComboBox1.Items.Clear;
```

```
  pos:= LO_DobleEnlace.DobleEnlace_Primero(Unit1.ME_Clientes.Indice);

  while (pos <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.ME_Clientes.Indice)) do
  begin
    LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, pos, RegistroIndiceCliente);
    clave:= RegistroIndiceCliente.Clave;
    Form_Facturacion.ComboBox1.AddItem(clave, Form_Facturacion.ComboBox1);

    pos:= RegistroIndiceCliente.Sig;
  end;
end;

procedure TForm_Facturacion.FormCreate(Sender: TObject);
begin
  NombrarCeldas;
  Memo1.Text:='';
  Form_Facturacion.StringGrid1.RowCount:=1;
  Form_Facturacion.Edit1.MaxLength:= 4;
  Form_Facturacion.Edit4.MaxLength:= 3;
  Form_Facturacion.DateTimePicker1.Date:= now;


  ActualizarClientesEnFacturacion;


end;

procedure EliminarFila(Fila: Integer);
var
  i: Integer;
begin
  for i := Fila to Form_Facturacion.StringGrid1.RowCount - 1 do
    Form_Facturacion.StringGrid1.Rows[i].Assign(Form_Facturacion.StringGrid1.Rows[i + 1]);
```

```
  Form_Facturacion.StringGrid1.RowCount := Form_Facturacion.StringGrid1.RowCount - 1;
end;

procedure TForm_Facturacion.BitBtn2Click(Sender: TObject);
begin
  if Form_Facturacion.Label5.Caption = '' then Showmessage('Seleccione en la tabla una fila a eliminar')
  else
  begin
    if (Form_Facturacion.Label5.Caption = '0') then Showmessage('No se puede eliminar esa fila')
    else
    begin
      EliminarFila( StrToInt(Form_Facturacion.Label5.Caption));
      Form_Facturacion.Label5.Caption:='';


      //Vemos si el recuento de filas es = 1, si lo es entonces deshabilitamos la emision de fact.
      if Form_Facturacion.StringGrid1.RowCount = 1
      then
      begin
        Form_Facturacion.Label_Neto.Caption:='';
        Form_Facturacion.BitBtn4.Enabled:= false;
      end
      else
      ActualizarNeto;

    end
  end

end;

procedure TForm_Facturacion.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
```

```
begin
  Form_Facturacion.Label5.Caption:= IntToStr(ARow);
end;

procedure LimpiarTabla;
var
  i: integer;
begin
  for i:= 1 to (Form_Facturacion.StringGrid1.RowCount) do
    Form_Facturacion.StringGrid1.Rows[i].Clear;
end;

procedure TForm_Facturacion.BitBtn3Click(Sender: TObject);
begin
  LimpiarTabla;
  Form_Facturacion.StringGrid1.Cells[0,0]:= 'COD. ARTICULO';
  Form_Facturacion.StringGrid1.Cells[1,0]:= 'DETALLE';
  Form_Facturacion.StringGrid1.Cells[2,0]:= 'CANT.';
  Form_Facturacion.StringGrid1.Cells[3,0]:= 'PRECIO UNITARIO';
  Form_Facturacion.StringGrid1.Cells[4,0]:= 'TOTAL';
  Form_Facturacion.StringGrid1.RowCount:=1;

  Form_Facturacion.ComboBox1.Clear;
  Form_Facturacion.ComboBox2.Clear;
  Form_Facturacion.ComboBox2.AddItem('01', Form_Facturacion.ComboBox2);
  Form_Facturacion.ComboBox2.AddItem('02', Form_Facturacion.ComboBox2);
  Form_Facturacion.ComboBox2.AddItem('03', Form_Facturacion.ComboBox2);
  Form_Facturacion.ComboBox2.AddItem('50', Form_Facturacion.ComboBox2);

  Form_Facturacion.ComboBox3.Clear;
  Form_Facturacion.ComboBox3.AddItem('1: Contado', Form_Facturacion.ComboBox3 );
  Form_Facturacion.ComboBox3.AddItem('2: Crédito', Form_Facturacion.ComboBox3 );
  Form_Facturacion.Edit1.Clear;
```

```
  Form_Facturacion.Edit2.Clear;
  Form_Facturacion.Edit3.Clear;
  Form_Facturacion.Edit4.Clear;

  ActualizarClientesEnFacturacion;

end;

function VerStock(codArt ,cant: longint): boolean;
var
  bResultado: boolean;
  pos: LO_ABB.Tipo_posicion;
  nivel: LO_ABB.Tipo_Cantidad;
  RegistroDatosArticulo: LO_DAtos_Articulos.Tipo_Registro_Datos;
begin
  LO_ABB.Arbol_Buscar(Unit1.ME_Articulos.Arbol, codArt, pos, nivel);
  LO_Datos_Articulos.Datos_Capturar(Unit1.ME_Articulos.Datos,pos, RegistroDatosArticulo);

  if cant > RegistroDatosArticulo.stock then bResultado:= false
  else bResultado:= true;

  VerStock:= bResultado;
end;

procedure AgregarRegistroATabla( codArticulo, detalle, cant, unitario, total: string);
var
  i: longint;
begin
      Form_Facturacion.StringGrid1.RowCount:= Form_Facturacion.StringGrid1.RowCount + 1;

       i:= Form_Facturacion.StringGrid1.RowCount-1;

       Form_Facturacion.StringGrid1.Cells[0,i]:= codArticulo;
```

```
      Form_Facturacion.StringGrid1.Cells[1,i]:= detalle;
      Form_Facturacion.StringGrid1.Cells[2,i]:= cant;
      Form_Facturacion.StringGrid1.Cells[3,i]:= unitario;
      Form_Facturacion.StringGrid1.Cells[4,i]:= total;
      Form_Facturacion.Edit1.Clear;
      Form_Facturacion.Edit1.SetFocus;
      Form_Facturacion.Memo1.Text:='';
      Form_Facturacion.Edit2.Clear;
      Form_Facturacion.Edit3.Clear;
      Form_Facturacion.Label11.Caption:='';
      ActualizarNeto;
      //Vemos si ya esta ingresado el descuento, el tipo comprobante y la forma de pago para habilitar la emision de fact.
      if (Form_Facturacion.Edit4.Text <> '') and  (Form_Facturacion.ComboBox2.Text <> '') and (Form_Facturacion.ComboBox3.Text
<> '')
      then Form_Facturacion.BitBtn4.Enabled:= true;
end;


procedure TForm_Facturacion.BitBtn1Click(Sender: TObject);
var
  detalle, codArticulo, cant, total, unitario: string;
  n: integer;
  bEncontrado: boolean;
begin

  if (Form_Facturacion.ComboBox1.Text='') or (Form_Facturacion.Edit1.Text='') or
  (Form_Facturacion.Memo1.Text='') or (Form_Facturacion.Edit2.Text='') or
  (Form_Facturacion.Edit3.Text='') then showmessage('Por favor, complete todos los campos.')
  else
  begin
    codArticulo:= Form_Facturacion.Edit1.Text;
    cant:= Form_Facturacion.Edit3.Text;
    detalle:= Form_Facturacion.Memo1.Text;
    unitario:= '$'+Form_Facturacion.Edit2.Text;
```

```
total:= '$'+Form_Facturacion.Label11.Caption;

if ( Form_Facturacion.StringGrid1.RowCount <= 1 ) then
begin
  if (VerStock( StrToInt(codArticulo), strToInt( cant) ) = false) then showmessage('No hay suficiente stock para esa cantidad')
  else
    AgregarRegistroATabla(codArticulo, detalle, cant, unitario, total);
end
else  //SINO... Si ya hay registros en la tabla...
begin
  n:= 1;
  bEncontrado:= false;
  while (bEncontrado = false) and (n <> Form_facturacion.StringGrid1.RowCount) do
  begin
    if (codArticulo = Form_Facturacion.StringGrid1.Cells[0, n]) then
    begin
      bEncontrado:= true;
      cant:= IntToStr(strToInt(cant) + strToint(Form_Facturacion.StringGrid1.Cells[2, n] ));
      if (VerStock(StrToInt(codArticulo), StrToInt(cant)) = false) then showmessage ('No hay suficiente stock para esa cantidad')
      else
      begin //hay que editar la cantidad en el registro
        Form_Facturacion.StringGrid1.Cells[2,n]:= cant;
        Form_Facturacion.StringGrid1.Cells[4,n]:= '$'+FloatToStr(strtoint(cant) *
strtoFloat(normalizarCadena(Form_Facturacion.StringGrid1.Cells[3,n])));
        ActualizarNeto;
        Showmessage('Se ha modificado la cantidad de un articulo.')
      end
    end
      else n:=n+1;
  end;//while

  if (n = Form_Facturacion.StringGrid1.RowCount) then //No encontro el registro, se debe insertar normalmente
    if (VerStock( StrToInt(codArticulo), strToInt( cant) ) = false) then showmessage('No hay suficiente stock para esa cantidad')
    else
```

```
        AgregarRegistroATabla(codArticulo, detalle, cant, unitario, total);


    end
  end//else completo todos los campos
end; //clickagregar

procedure ActualizarDatosReporte;
var
  claveCliente, nombreCliente, dniCliente, fecha, nroComp, tipoComp, subtotal,
  descuento, gravado, iva, totalFacturado, formaDePago: string ;
  pos: LO_DobleEnlace.Tipo_Posicion;
  posDatos: LO_Datos_Clientes.Tipo_Posicion;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroDatos: LO_Datos_Clientes.Tipo_Registro_Datos;

begin
  claveCliente:= Form_Facturacion.ComboBox1.Text;

  //BuscamosDatosDeCliente
  LO_DobleEnlace.DobleEnlace_Buscar(Unit1.ME_Clientes.Indice, claveCliente, pos);
  LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, pos, RegistroIndice);
  posDAtos:= RegistroIndice.Posicion;
  LO_Datos_Clientes.Datos_Capturar(Unit1.ME_clientes.Datos, posDatos, RegistroDatos);
  nombreCliente:= RegistroDatos.Nombre;
  dniCliente:= RegistroDatos.Dni;

  //Fecha
  fecha:= DateToStr(Form_Facturacion.DateTimePicker1.Date);

  //Comprobante
  nroComp:= IntToStr(LO_Cola.Cola_UltimoNroComprobante(Unit1.ME_VEntas.Comprobante.Cola)); //Ver ultimo en pila/cola

  tipoComp:= Form_Facturacion.ComboBox2.Text;
```

```
 if tipoComp='01' then tipoComp:='A'
   else if tipoComp='02' then tipoComp:='deb. A'
     else if tipoComp='03' then tipoComp:='cred. A'
       else if tipoComp ='50' then tipoComp := 'Recibo X' ;

 //Footer
 subtotal:= Form_Facturacion.Label_Neto.Caption;
 descuento:= Form_Facturacion.Edit4.Text;
 gravado:= FormatFloat('#.##', StrToFloat(subtotal) - ( StrToFloat(subtotal) * ( StrToInt(descuento) / 100 )  ) );
 iva:= FormatFloat('#.##', 0.21 * StrToFloat(gravado)   );
 totalFacturado:= FloatToStr(StrToFloat(gravado) + StrToFloat(iva));
 FormaDePago:= Form_Facturacion.ComboBox3.Text;

 Form_Reporte.QRLabel_ClaveCliente.Caption:= claveCliente;
 Form_Reporte.QRLabel_NombreCliente.Caption:=nombreCliente;
 Form_Reporte.QRLabel_DNICliente.Caption:= dniCliente;
 Form_Reporte.QRLabel_Fecha.Caption:= fecha;
 Form_Reporte.QRLabel_NroComprobante.Caption:= nroComp  ;
 Form_Reporte.QRLabel_TipoComprobante.Caption:= tipoComp ;
 Form_Reporte.QRLabel_Subtotal.Caption:= '$'+subtotal;
 Form_Reporte.QRLabel_Descuento.Caption:= descuento;
 form_Reporte.QRLabel_Gravado.Caption:= '$'+gravado;
 form_reporte.QRLabel_IVA.Caption:= '$'+iva;
 form_reporte.QRLabel_TotalFacturado.Caption:= '$'+totalFacturado;
 form_reporte.QRLabel_FormaDePago.Caption:= formaDePago[1];

end;

procedure ActualizarStockArticulos;
var
 i: longint;
 cCodArticulo, cCant: string;
 pos: LO_ABB.Tipo_Posicion;
```

```
  nivel: LO_ABB.Tipo_Cantidad;
  RegistroArbol: LO_ABB.Tipo_Registro_Indice;
  RegistroDatosArticulo: LO_Datos_Articulos.Tipo_Registro_Datos;
  posDatos: longint;
begin
  i:= 1;

  while (i < Form_Facturacion.StringGrid1.RowCount) do
  begin
    cCodArticulo:= Form_Facturacion.StringGrid1.Cells[0, i];
    cCant:= Form_Facturacion.StringGrid1.Cells[2, i];

    LO_ABB.Arbol_Buscar(Unit1.ME_Articulos.Arbol, StrToInt(cCodArticulo), pos, nivel);
    LO_ABB.Arbol_Capturar(Unit1.ME_Articulos.Arbol, pos, RegistroArbol);

    posDatos:= RegistroARbol.Posicion;

    LO_Datos_articulos.Datos_Capturar(Unit1.ME_Articulos.Datos, posDatos, RegistroDatosArticulo);

    RegistroDatosArticulo.stock:= RegistroDatosArticulo.stock - strToInt(cCant);

    LO_Datos_Articulos.Datos_Modificar(Unit1.ME_articulos.Datos, posDatos, RegistroDatosArticulo);

    i:=i+1;
  end;
end;
function ValidarFecha(cFecha, cTipoComp: string): boolean; //Se fija si la fecha ingresada no es mayor que el dia actual +1 y que no
es menor a la fecha del ultimo comprobante del mismo tipo emitido
var
  bResultado: boolean;
  ultimaFechaEmitida: string;
  begin
  bResultado:= true;
  ultimafechaemitida:='-';
```

```
 if (Cola_Vacia(Unit1.ME_Ventas.Comprobante.cola) = true ) then bResultado:= true
 else
 if (StrToDate(cFecha) > now+1) then
 begin
  bResultado:= false;
  Showmessage('La fecha no puede ser mayor a la del dia de mañana.')

 end
 else
 begin

  if LO_Cola.UltimaFechaEmitida(Unit1.ME_Ventas.Comprobante.cola, cTipoComp) = '' then bResultado:=true
  else
    if StrToDate(cFecha) < StrToDate(LO_Cola.UltimaFechaEmitida(Unit1.ME_ventas.Comprobante.cola, cTipoComp)) then
    begin
      bResultado:= false;
      showmessage('La fecha ingresada es menor a de la ultima factura del mismo tipo emitida. Debe ser mayor o igual a:
'+(LO_Cola.UltimaFechaEmitida(Unit1.ME_ventas.Comprobante.cola, cTipoComp)));
    end
    else
    bResultado:=true;
 end;
 ValidarFecha:= bResultado;
end;

function AutorizarFactura( claveCliente: string): boolean; //A COMPLETAR se fija si el cliente esta autorizado a hacer mas
movimientos
var
 bResultado: boolean;
 suma: real;
 pilaAux: LO_DobleEnlace.Tipo_Indice;
 RegistroPila: LO_DobleEnlace.Tipo_Registro_Indice;
 RegistroDatosAsiento: LO_Datos_Asientos.Tipo_Registro_Datos;
```

```
 posDatos: longint;
begin
 suma:=0;
 bResultado:=true;
 LO_Pila.Pila_Crear(pilaAux, Unit1.sRuta, 'pilaTemp');
 LO_Pila.Pila_Abrir(pilaAux);
 if LO_Pila.Pila_Vacia(Unit1.ME_CuentasCorrientes.Pila) = true then bResultado:=true
 else
 begin
 //Recorrer pila y sumar importes del cliente , si da 0 entonces puede copmrar, sino no
  while Pila_Vacia(Unit1.ME_CuentasCorrientes.Pila) = false do
  begin
   Pila_Tope(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);

   if (RegistroPila.Clave = claveCliente) then
   begin
    posDatos:= RegistroPila.Posicion;
    LO_Datos_Asientos.Datos_Capturar(Unit1.ME_CuentasCorrientes.Datos, posDatos, RegistroDatosAsiento);
    suma:= suma + RegistroDatosAsiento.importe;
   end;
   Pila_Apilar(pilaAux, RegistroPila);
   Pila_Desapilar(Unit1.ME_CuentasCorrientes.Pila);
  end;//while
  //Volvemos a pasar los elementos a la original
  while (Pila_Vacia(pilaAux) = false) do
  begin
   Pila_Tope(pilaAux, RegistroPila);
   Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);
   Pila_Desapilar(pilaAux);
  end;//while
  if suma = 0 then bResultado:= true else bResultado:=false;
 end;
 Pila_Destruir(pilaAux);
```

```
  AutorizarFactura:= bResultado;
end;
procedure TForm_Facturacion.BitBtn4Click(Sender: TObject);    //Click en EMITIR FACTURA
var
  RegistroPila, RegistroCola: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroDatosComprobante: LO_Datos_Comprobantes.Tipo_Registro_Datos;
  RegistroDatosDetalle: LO_Datos_Detalles.Tipo_Registro_Datos;
  codCliente, cPrecioUnitario, cFecha, cTipoComprobante: string;
  i, posDatos, codFactura, nroComprobante: longint;
  RegistroAsiento: LO_Datos_Asientos.Tipo_Registro_Datos;
  nNeto, nGravado, nIVA, nTotal: real;
  nDescuento: integer;
begin
  codCliente:= Form_Facturacion.ComboBox1.Text;
  cFecha:= dateToStr(Form_Facturacion.DateTimePicker1.date);
  cTipoComprobante:= Form_Facturacion.ComboBox2.Text;

    if ( ValidarFecha(cFecha, cTipoComprobante) = true ) then
    begin
      //Cargar factura...

      //Num. de comprobante y cod factura siempre seran iguales. El codigo de la factura señala la vinculacion del comprobante con
el o los detalles.
      nroComprobante:= LO_Cola.Cola_UltimoNroComprobante(Unit1.Me_Ventas.Comprobante.Cola) + 1;
      codFactura:= LO_Cola.Cola_UltimoNroComprobante(Unit1.Me_Ventas.Comprobante.Cola) + 1;

      //Comprobante
      RegistroDatosComprobante.codFactura:= IntToStr(codFactura);
      RegistroDatosComprobante.codCliente:= codCliente;
      RegistroDatosComprobante.fecha:= DateToStr(Form_Facturacion.DateTimePicker1.date);
      RegistroDatosComprobante.tipoComprobante:= Form_Facturacion.ComboBox2.Text;
      RegistroDatosComprobante.nroComprobante:= nroComprobante;
      RegistroDatosComprobante.neto:= StrToFloat(Form_Facturacion.Label_Neto.Caption);
      RegistroDatosComprobante.descuento:= StrToInt(Form_Facturacion.Edit4.Text);
```

```
    RegistroDatosComprobante.gravado:= RegistroDatosComprobante.neto - (RegistroDatosComprobante.descuento/100 *
RegistroDatosComprobante.neto);
    RegistroDatosComprobante.IVA:= 0.21 * (RegistroDatosComprobante.gravado);
    RegistroDatosComprobante.total:= RegistroDatosComprobante.gravado + RegistroDatosComprobante.IVA;
    RegistroDatosComprobante.borrado:= false;

    posDatos:= FileSize(ME_Ventas.Comprobante.Datos.D);

    RegistroCola.Clave:= IntToStr(codFactura);
    RegistroCola.Posicion:= posDatos;

    LO_Datos_Comprobantes.Datos_Insertar(Unit1.ME_Ventas.Comprobante.Datos, RegistroDatosComprobante);
    LO_Cola.Cola_EncolarConFecha(Unit1.ME_Ventas.Comprobante.Cola,
RegistroCola,RegistroDatosComprobante.tipoComprobante, RegistroDatosComprobante.fecha );
    //DETALLE (UN REGISTRO POR CADA UNO DE LOS REGISTROS DEL STRINGGRID)
    for i:= 1 to (Form_Facturacion.StringGrid1.RowCount - 1 ) do
    begin
      cPrecioUnitario:= NormalizarCadena(Form_Facturacion.StringGrid1.Cells[3, i]);
      RegistroDatosDetalle.codFactura:= IntToStr(codFactura);
      RegistroDatosDetalle.codArticulo:= strToInt(Form_Facturacion.StringGrid1.Cells[0, i]);
      RegistroDatosDetalle.descripcion:= Form_Facturacion.StringGrid1.Cells[1, i];
      RegistroDatosDetalle.cant:= StrToInt(Form_Facturacion.StringGrid1.Cells[2, i]);
      RegistroDatosDetalle.precioUnitario:= StrToFloat(cPrecioUnitario);
      RegistroDatosDetalle.Borrado:=false;

      posDatos:= FileSize(Unit1.ME_Ventas.Detalle.Datos.D);

      RegistroPila.Clave:= IntToStr(codFactura);
      RegistroPila.Posicion:= posDatos;

      LO_Datos_Detalles.Datos_Insertar(Unit1.ME_Ventas.Detalle.Datos, RegistroDatosDetalle);
      Lo_Pila.Pila_Apilar(Unit1.ME_Ventas.Detalle.Pila, RegistroPila);
    end; //for detalle
```

```
//Asiento en Cta Corriente

nNeto:= StrToFloat(Form_Facturacion.Label_Neto.Caption);
nDescuento:= StrToInt(Form_Facturacion.Edit4.Text);
nGravado:= nNeto - ( ( nDescuento / 100 ) * nNeto );
nIVA:= 0.21 * nGravado;
nTotal:= nGravado + nIVA;

if (Form_Facturacion.ComboBox3.Text = '1: Contado') then
begin
  RegistroAsiento.CodCliente:= codCliente;
  RegistroAsiento.nroComprobante:=nroComprobante;
  RegistroAsiento.fecha:= DateToStr(Form_Facturacion.DateTimePicker1.date);
  RegistroAsiento.importe:= 0 - nTotal;
  posDatos:= FileSize(Unit1.ME_CuentasCorrientes.Datos.D) ;
  RegistroPila.Clave:= codCliente;
  RegistroPila.Posicion:= posDatos;
  LO_Datos_Asientos.Datos_Insertar(Unit1.ME_CuentasCorrientes.Datos, RegistroAsiento);
  LO_Pila.Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);  //CARGAMOS EL DEBE
  RegistroAsiento.CodCliente:= codCliente;
  RegistroAsiento.nroComprobante:=nroComprobante;
  RegistroAsiento.fecha:= DateToStr(Form_Facturacion.DateTimePicker1.date);
  RegistroAsiento.importe:= nTotal;

  posDatos:= FileSize(Unit1.ME_CuentasCorrientes.Datos.D);

  RegistroPila.Clave:= codCliente;
  RegistroPila.Posicion:= posDatos;

  LO_Datos_Asientos.Datos_Insertar(Unit1.ME_CuentasCorrientes.Datos, RegistroAsiento);
  LO_Pila.Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, RegistroPila); //CARGAMOS EL HABER
end
else
```

```
    begin
      //Cargamos solo el DEBE
      RegistroAsiento.CodCliente:= codCliente;
      RegistroAsiento.nroComprobante:=nroComprobante;
      RegistroAsiento.fecha:= DateToStr(Form_Facturacion.DateTimePicker1.date);
      RegistroAsiento.importe:= 0 - nTotal;

      posDatos:= FileSize(Unit1.ME_CuentasCorrientes.Datos.D) ;

      RegistroPila.Clave:= codCliente;
      RegistroPila.Posicion:= posDatos;

      LO_Datos_Asientos.Datos_Insertar(Unit1.ME_CuentasCorrientes.Datos, RegistroAsiento);
      LO_Pila.Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);  //CARGAMOS EL DEBE

    end;
    //Actualizar stock de articulos
    ActualizarStockArticulos;
    //Preparar reporte
    ActualizarDatosReporte;
    //Presentar listado.
    Form_Reporte.QuickRep1.Preview;
    Showmessage('Se ha guardado registro de factura.');
   end; //end if fecha
end;

procedure LimpiarCampos;
begin
  Form_Facturacion.Memo1.Clear;
  Form_Facturacion.Edit2.Clear;

  Form_Facturacion.Edit3.Clear;
  Form_Facturacion.Edit3.Enabled:=false;
```

```
  Form_Facturacion.Label11.Caption:= '';

end;

procedure TForm_Facturacion.Edit1Change(Sender: TObject);
var
  s: string;
  n: longint;
  pos: LO_ABB.Tipo_Posicion;
  nivel: LO_ABB.Tipo_Cantidad;
  RegistroArbol: LO_ABB.Tipo_Registro_Indice;

  RegistroDatos: LO_Datos_Articulos.Tipo_Registro_Datos;
  posDatos: LO_Datos_Articulos.Tipo_Posicion;

begin
  s:= Form_Facturacion.Edit1.Text;

  if Length(s) <> 4 then LimpiarCampos
  else
  begin
    if TryStrToInt(s, n) = false then showmessage('Ingrese un codigo de articulo valido. Formato "1111" ')
    else
    begin
      if (LO_ABB.Arbol_Buscar(Unit1.ME_Articulos.Arbol,StrToInt(s), pos, nivel) = true) then
        begin
          LO_ABB.Arbol_Capturar(Unit1.ME_Articulos.Arbol, pos, RegistroArbol);
          posDatos:= RegistroArbol.Posicion;

          LO_Datos_Articulos.Datos_Capturar(Unit1.ME_Articulos.Datos, posDatos, RegistroDatos);
          Form_Facturacion.Memo1.Text:= registroDatos.Descripcion;
          Form_Facturacion.Edit2.Text:= FloatToStr(RegistroDatos.PrecioUnitario);
          Form_Facturacion.Edit3.Enabled:=true;
```

```
        Form_Facturacion.Edit3.SetFocus;


      end
    end
  end
end;
procedure TForm_Facturacion.Edit3Change(Sender: TObject);
var
  s: String;
  n: integer;
begin
  s:= Form_Facturacion.Edit3.Text;
  if s='0' then Form_Facturacion.Edit3.Text:='';
  if (s = '') then Form_Facturacion.Label11.Caption:=''
  else
  begin
    if (TryStrToInt(s , n) = false) then
    begin
          showmessage('Ingrese cantidad valida');
          Form_Facturacion.Edit3.Text:='';
    end
   else
    begin
      Form_Facturacion.Label11.Caption:= FloatToStr( n * StrToFloat(Form_Facturacion.Edit2.Text) ) ;
    end
  end
end;


procedure TForm_Facturacion.ComboBox2Change(Sender: TObject);
begin

  if  (Form_Facturacion.ComboBox2.Text = '') then Form_Facturacion.BitBtn4.Enabled:= false;
```

```
 if (Form_Facturacion.StringGrid1.RowCount > 1) and (Form_Facturacion.Edit4.Text <> '') and
   (Form_Facturacion.ComboBox2.Text <> '') and (Form_Facturacion.ComboBox3.Text <> '')
  then
 begin
   Form_Facturacion.BitBtn4.Enabled:= true;
 end
 else
   Form_Facturacion.BitBtn4.Enabled:= false;
end;

procedure TForm_Facturacion.Edit4Change(Sender: TObject);
var
 s:String;
 n: longint;
begin
 s:= Form_Facturacion.Edit4.Text;

 if (s = '') then  Form_Facturacion.BitBtn4.Enabled:= false;
 if (TryStrToInt(s,n)) then
 begin
   if (Form_Facturacion.StringGrid1.RowCount > 1) and (Form_Facturacion.ComboBox2.Text <> '') and
   (Form_Facturacion.ComboBox3.Text <> '') and (Form_Facturacion.Edit4.Text <> '')
   then
     Form_Facturacion.BitBtn4.Enabled:= true
   else
     Form_Facturacion.BitBtn4.Enabled:= false;
   end
end;
procedure TForm_Facturacion.ComboBox3Change(Sender: TObject);
begin
if  (Form_Facturacion.ComboBox3.Text = '') then Form_Facturacion.BitBtn4.Enabled:= false;
 if (Form_Facturacion.StringGrid1.RowCount > 1) and (Form_Facturacion.Edit4.Text <> '') and
   (Form_Facturacion.ComboBox2.Text <> '') and (Form_Facturacion.ComboBox3.Text <> '')
```

```
  then
 begin
   Form_Facturacion.BitBtn4.Enabled:= true;
 end
 else
   Form_Facturacion.BitBtn4.Enabled:= false;
end;
end.
```

UNIT 7

```
unit Unit7;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, QuickRpt, QRCtrls;
type
  TForm_Reporte = class(TForm)
    QuickRep1: TQuickRep;
    QRBand1: TQRBand;
    QRBand2: TQRBand;
    QRLabel1: TQRLabel;
    QRLabel2: TQRLabel;
    QRLabel_NroComprobante: TQRLabel;
    QRLabel_TipoComprobante: TQRLabel;
    QRShape1: TQRShape;
    QRLabel5: TQRLabel;
    QRLabel_ClaveCliente: TQRLabel;
    QRLabel_6: TQRLabel;
    QRLabel_NombreCliente: TQRLabel;
    QRLabel3: TQRLabel;
    QRLabel_Fecha: TQRLabel;
    QRLabel4: TQRLabel;
```

```
QRLabel_DNICliente: TQRLabel;
QRShape2: TQRShape;
QRBand3: TQRBand;
GroupHeaderBand1: TQRBand;
QRLabel11: TQRLabel;
QRLabel6: TQRLabel;
QRLabel7: TQRLabel;
QRLabel8: TQRLabel;
QRLabel9: TQRLabel;
QRLabel12: TQRLabel;
QRShape3: TQRShape;
QRLabel13: TQRLabel;
QRLabel14: TQRLabel;
QRLabel_Subtotal: TQRLabel;
QRLabel_Descuento: TQRLabel;
QRLabel_Gravado: TQRLabel;
QRLabel10: TQRLabel;
QRLabel_IVA: TQRLabel;
QRLabel19: TQRLabel;
QRLabel_TotalFacturado: TQRLabel;
QRLabel21: TQRLabel;
QRLabel_FormaDePago: TQRLabel;
QRLabel23: TQRLabel;
QRShape4: TQRShape;
QRBand4: TQRBand;
QRLabel_codArt: TQRLabel;
QRLabel_Detalle: TQRLabel;
QRLabel_cant: TQRLabel;
QRLabel_uni: TQRLabel;
QRLabel_total: TQRLabel;
procedure QuickRep1NeedData(Sender: TObject; var MoreData: Boolean);
procedure QuickRep1BeforePrint(Sender: TCustomQuickRep;
  var PrintReport: Boolean);
```

```
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form_Reporte: TForm_Reporte;
  posArticulos: integer;

implementation
uses Unit6;
{$R *.dfm}

procedure TForm_Reporte.QuickRep1BeforePrint(Sender: TCustomQuickRep;
  var PrintReport: Boolean);
begin
  posArticulos  := 1;
end;

procedure TForm_Reporte.QuickRep1NeedData(Sender: TObject;
  var MoreData: Boolean);
begin
  MoreData := (  posArticulos < Form_Facturacion.StringGrid1.RowCount);
  if MoreData then
  begin
    QRLabel_codArt.Caption := Form_Facturacion.StringGrid1.Cells [0, posArticulos];
    QRLabel_Detalle.Caption    := Form_Facturacion.StringGrid1.Cells  [1, posArticulos];
    QRLabel_cant.Caption := Form_Facturacion.StringGrid1.Cells  [2, posArticulos];
    QRLabel_uni.Caption      := Form_Facturacion.StringGrid1.Cells  [3, posArticulos];
    QRLabel_total.Caption       := Form_Facturacion.StringGrid1.Cells  [4, posArticulos];
    // Proximo elemento de la grilla
    posArticulos := Succ (posArticulos)  ;
```

```
  end;
end;
end.
```

UNIT 8

```
unit Unit8;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ExtCtrls, QuickRpt, QRCtrls;
type
  TForm_Reporte_Articulos = class(TForm)
    QuickRep1: TQuickRep;
    QRBand1: TQRBand;
    QRLabel1: TQRLabel;
    QRLabel2: TQRLabel;
    QRBand2: TQRBand;
    QRBand3: TQRBand;
    QRBand4: TQRBand;
    QRLabel_CodArtheader: TQRLabel;
    QRLabel_Descheader: TQRLabel;
    QRLabel_Stockheader: TQRLabel;
    QRLabel_Costoheader: TQRLabel;
    QRLabel_Unitarioheader: TQRLabel;
    QRShape1: TQRShape;
    QRLabel_Fecha: TQRLabel;
    QRShape2: TQRShape;
    QRLabel_codArt: TQRLabel;
    QRLabel_Desc: TQRLabel;
    QRLabel_stock: TQRLabel;
    QRLabel_Costo: TQRLabel;
    QRLabel_unitario: TQRLabel;
```

```
    procedure QuickRep1BeforePrint(Sender: TCustomQuickRep;
      var PrintReport: Boolean);
    procedure QuickRep1NeedData(Sender: TObject; var MoreData: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form_Reporte_Articulos: TForm_Reporte_Articulos;
  posArticulos: longint;
implementation
uses Unit4;
{$R *.dfm}
procedure TForm_Reporte_Articulos.QuickRep1BeforePrint(
  Sender: TCustomQuickRep; var PrintReport: Boolean);
begin
  posArticulos  := 1;
end;

procedure TForm_Reporte_Articulos.QuickRep1NeedData(Sender: TObject;
  var MoreData: Boolean);
begin
  MoreData := (  posArticulos < Form_Articulos.StringGrid1.RowCount);
  if MoreData then
  begin
    QRLabel_codArt.Caption  := Form_Articulos.StringGrid1.Cells [0, posArticulos];
    QRLabel_Desc.Caption    := Form_Articulos.StringGrid1.Cells [2, posArticulos];
    QRLabel_stock.Caption   := Form_Articulos.StringGrid1.Cells [5, posArticulos];
    QRLabel_Costo.Caption   := Form_Articulos.StringGrid1.Cells [3, posArticulos];
    QRLabel_Unitario.Caption:= Form_Articulos.StringGrid1.Cells [4, posArticulos];

    // Proximo elemento de la grilla
```

```
    posArticulos := Succ (posArticulos)  ;
  end;
end;
end.
```

UNIT 9

```
unit Unit9;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls, LO_DobleEnlace, LO_ABB, LO_Datos_Articulos,
  LO_Datos_Rubros, LO_Indice, LO_Pila, LO_Cola, LO_Datos_Asientos, LO_Datos_Detalles;
type
  TForm_Listados = class(TForm)
    Image1: TImage;
    Button1: TButton;
    Label1: TLabel;
    ComboBox1: TComboBox;
    ListBox1: TListBox;
    procedure Button1Click(Sender: TObject);
    procedure ComboBox1Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_Listados: TForm_Listados;
implementation
uses Unit1, Unit4, Unit10;
{$R *.dfm}
```

```
procedure TForm_Listados.Button1Click(Sender: TObject);
begin
  Form_Listados.ListBox1.Clear;
  Form_Listados.Close;
  Form_MenuPrincipal.show;
end;
procedure RecorrerArbolInORden(ME_Articulos: Unit1.Tipo_ME_Articulos; posNodo: integer; claveRubro: String; minOp, max:
longint);
var
  RegistroArbol: LO_ABB.Tipo_REgistro_Indice;
  RegistroDatosArticulo: LO_Datos_Articulos.Tipo_Registro_Datos;
  numAReponer, posDatos: longint;
begin
  if (posNodo <> -1) then
  begin
    LO_ABB.Arbol_Capturar(ME_Articulos.Arbol, posNodo, RegistroArbol);
    RecorrerArbolInORden(ME_Articulos, RegistroArbol.HijoIzq, claveRubro, minOP, max);
    posDatos:= RegistroArbol.Posicion;
    LO_Datos_articulos.Datos_Capturar(ME_Articulos.Datos, posDAtos, RegistroDatosArticulo);

    if RegistroDatosArticulo.ClaveRubro = claveRubro then
    begin
      if RegistroDatosArticulo.stock < minOP then
      begin
        //LISTAR
        numAReponer:= max - RegistroDatosArticulo.stock;
        Form_Listados.ListBox1.AddItem('Articulo: '+IntToStr(RegistroDatosArticulo.Clave)+'. Num. a reponer:
'+intToStr(numAReponer), Form_Listados.ListBox1 );
      end;
    end;

    RecorrerArbolInORden(ME_Articulos, RegistroARbol.HijoDer, claveRubro, minOP, max)
  end
end;
```

```
procedure ListarArticulosConFaltaDeStock;
var
  pos: LO_DobleEnlace.Tipo_Posicion;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroDatosRubro: LO_Datos_Rubros.Tipo_Registro_Datos;
  posRaiz: LO_ABB.Tipo_Posicion;
begin
  pos:= LO_DobleEnlace.DobleEnlace_Primero(Unit1.ME_Rubros.Indice);
  posRaiz:= LO_ABB.Arbol_Raiz(Unit1.ME_Articulos.Arbol);
  Form_Listados.ListBox1.Clear;
  while pos <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.ME_Rubros.Indice) do
  begin
    LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Rubros.Indice, pos, RegistroIndice);
    LO_Datos_Rubros.Datos_Capturar(Unit1.ME_Rubros.Datos, RegistroIndice.Posicion, RegistroDatosRubro);
    Form_Listados.ListBox1.AddItem('Rubro '+RegistroIndice.Clave+':', Form_Listados.ListBox1);
    Form_Listados.ListBox1.AddItem('', Form_Listados.ListBox1);
    RecorrerArbolInOrden(Unit1.ME_Articulos, posRaiz, RegistroIndice.Clave, RegistroDatosRubro.minimoOperativo,
RegistroDatosRubro.maximoAReponer);
    Form_Listados.ListBox1.AddItem('-------------------------------------------------', Form_Listados.ListBox1);
    pos:= RegistroIndice.Sig;
  end; //While
end;
procedure ListarRanking;
var
  suma, claveArt, i, posDatos, posIndice: integer;
  IndiceAux: LO_Indice.Tipo_Indice;
  pilaAuxDet: LO_DobleEnlace.Tipo_Indice;
  RegistroPilaDetalle: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroIndice: LO_Indice.Tipo_Registro_Indice;
  RegistroDatosDetalle: LO_Datos_Detalles.Tipo_Registro_Datos;
  RegistroArbol: LO_ABB.Tipo_Registro_Indice;
  RegistroDatosArticulo: LO_Datos_Articulos.Tipo_Registro_Datos;
  nivel: LO_ABB.Tipo_Cantidad;
  posArbol: longint;
```

```
begin
  LO_Indice.Indice_Crear(IndiceAux, Unit1.sRuta, 'IndiceTemp');
  Lo_Indice.Indice_Abrir(IndiceAux);
  LO_Pila.Pila_Crear(pilaAuxDet, unit1.sRuta, 'pilaAuxDetTemp');
  LO_Pila.Pila_Abrir(pilaAuxDet);
  if LO_ABB.Arbol_Vacio(Unit1.ME_Articulos.Arbol) = false then
  begin
   for i:= 1 to (Form_Articulos.StringGrid1.RowCount-1) do
   begin
     suma:=0;
     claveArt:= StrToInt(Form_Articulos.StringGrid1.cells[0, i]);

       while (LO_Pila.Pila_Vacia(Unit1.ME_Ventas.Detalle.pila)=false) do
       begin
         Pila_Tope(Unit1.ME_Ventas.Detalle.Pila, RegistroPilaDetalle);
           posDatos:= registroPilaDetalle.Posicion;
           LO_Datos_Detalles.Datos_Capturar(ME_Ventas.Detalle.Datos, posDatos, registroDatosDetalle);
           if (RegistroDatosDetalle.codArticulo = claveArt) then  suma:= suma + RegistroDatosDetalle.cant;
         Pila_Desapilar(Unit1.ME_Ventas.Detalle.Pila);
         Pila_Apilar(pilaAuxDet, RegistroPilaDetalle);
       end;//end pilaDet
       //Volvemos a dejar como estaba la pilaDet
       while (Pila_Vacia(pilaAuxDet)=false) do
       begin
         Pila_Tope(pilaAuxDet, RegistroPilaDetalle);
         Pila_Apilar(Unit1.ME_Ventas.Detalle.Pila, RegistroPilaDetalle);
         Pila_Desapilar(pilaAuxDet);
       end;

      //Insertamos en un indice auxiliar para luego listarlo descendentemente
      LO_Indice.Indice_Buscar(IndiceAux, suma, posIndice);
      RegistroIndice.Clave:= suma;
      RegistroIndice.Puntero:= claveArt;
```

```
      LO_Indice.Indice_Insertar(IndiceAux, posIndice, RegistroIndice);
    end;
    for i:= LO_Indice.Indice_Ultimo(IndiceAux) downto LO_Indice.Indice_Primero(IndiceAux) do   //Recorremos descendentemente
para listar el ranking desde los mas vendidos a los menos vendidos
    begin
      LO_indice.Indice_Capturar(indiceAux, i, RegistroIndice);
      LO_ABB.Arbol_Buscar(Unit1.ME_Articulos.Arbol, RegistroIndice.Puntero, posArbol, nivel);
      LO_ABB.Arbol_Capturar(Unit1.ME_Articulos.Arbol, posArbol, RegistroArbol);
      posDatos:= RegistroArbol.Posicion;
      LO_Datos_articulos.Datos_Capturar(Unit1.ME_Articulos.Datos, posDatos, RegistroDatosArticulo);
      Form_Listados.ListBox1.AddItem('Cod. Articulo: '+IntToStr(RegistroIndice.Puntero), Form_Listados.ListBox1);
      Form_Listados.ListBox1.AddItem('Clave Rubro: '+RegistroDatosArticulo.ClaveRubro , Form_Listados.ListBox1);
      Form_Listados.ListBox1.AddItem('Descripcion: '+RegistroDatosARticulo.Descripcion, Form_Listados.ListBox1);
      Form_Listados.ListBox1.AddItem('Cantidad vendida: '+intToStr(RegistroIndice.Clave), Form_Listados.ListBox1);
      Form_Listados.ListBox1.AddItem('---------------------------------', Form_Listados.ListBox1);

     end;
   end;
   LO_Indice.Indice_Destruir(indiceAux);
   LO_Pila.Pila_Destruir(pilaAuxDet);
end;

procedure TForm_Listados.ComboBox1Change(Sender: TObject);
var
  s:String;
begin
  s:= Form_Listados.ComboBox1.Text;

  if s='Reposición de articulos' then
  begin
    Form_Listados.ListBox1.Clear;
    ListarArticulosConFaltaDeStock;
  end else
  if s='Ranking de Articulos' then
```

```
 begin
   Form_Listados.ListBox1.Clear;
   Form_Listados.ListBox1.AddItem('Ranking de articulos más vendidos: ', Form_Listados.ListBox1);
   Form_Listados.ListBox1.AddItem('-------------------------------------------------------------------------', Form_Listados.ListBox1);
   ListarRanking;
 end else
 if s='Cuentas Corrientes...' then
 begin
   Form_ListadosCC.show;
 end
end;
end.
```

UNIT 10

```
unit Unit10;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls, Grids, LO_Datos_Clientes, LO_DobleEnlace,
  LO_Datos_Asientos, LO_Pila, LO_Cola, LO_Datos_Comprobantes;

type
  TForm_ListadosCC = class(TForm)
    Image1: TImage;
    Button1: TButton;
    StringGrid1: TStringGrid;
    StringGrid_CCCliente: TStringGrid;
    ComboBox1: TComboBox;
    StringGrid_ResumenCC: TStringGrid;
    StringGrid_Comprobantes: TStringGrid;
    Label1: TLabel;
    Label2: TLabel;
```

```
    Label_SumaTotal: TLabel;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure StringGrid1SelectCell(Sender: TObject; ACol, ARow: Integer;
      var CanSelect: Boolean);
    procedure ComboBox1Change(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form_ListadosCC: TForm_ListadosCC;

implementation

{$R *.dfm}

uses
  Unit1;

procedure TForm_ListadosCC.Button1Click(Sender: TObject);
begin
  Form_ListadosCC.Close;
end;

procedure LimpiarTablaClientes;
var
  i: longint;
begin
  for i:= 1 to Form_ListadosCC.StringGrid1.RowCount  do
  begin
```

```
    Form_ListadosCC.StringGrid1.Rows[i].Clear;
  end;
end;

procedure ListarClientes;
var
  RegistroDatos: LO_Datos_Clientes.Tipo_Registro_Datos;
  RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  posicion, posDatos: LO_DobleEnlace.Tipo_Posicion;
  i: longint;
begin
  LimpiarTablaClientes;
  Form_ListadosCC.StringGrid1.Cells[0,0]:= 'Cód.';
  Form_ListadosCC.StringGrid1.Cells[1,0]:= 'Nombre';
  Form_ListadosCC.StringGrid1.Cells[2,0]:= 'DNI';
  Form_ListadosCC.StringGrid1.RowCount:= 1;
  posicion:= LO_DobleEnlace.DobleEnlace_Primero(Unit1.ME_Clientes.Indice);
  i:= 1;
  while (posicion <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.ME_Clientes.Indice)) do
  begin
    Form_ListadosCC.StringGrid1.RowCount:= Form_ListadosCC.StringGrid1.RowCount + 1;

    Lo_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, posicion, RegistroIndice);
    posDatos:= registroIndice.Posicion;
    LO_Datos_Clientes.Datos_Capturar(Unit1.ME_clientes.Datos, posDatos, RegistroDatos);

    Form_ListadosCC.StringGrid1.Cells[0, i]:= RegistroDatos.Clave;
    Form_ListadosCC.StringGrid1.Cells[1, i]:= RegistroDatos.Nombre;
    Form_ListadosCC.StringGrid1.Cells[2, i]:= RegistroDatos.Dni;
    i:=i+1;
    posicion:= RegistroIndice.Sig;
  end;//while
```

```
end;  //End ListarClientes

procedure TForm_ListadosCC.FormCreate(Sender: TObject);
begin
  ListarClientes;
end;

procedure ListarComprobantesPorCliente(claveCliente : string);
var
  pilaAuxCC, colaAuxComp: LO_DobleEnlace.Tipo_Indice;
  RegistroPila, RegistroCola: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroDatosAsiento: LO_Datos_Asientos.Tipo_Registro_Datos;
  RegistroDatosComprobante: LO_Datos_Comprobantes.Tipo_Registro_Datos;
  posDatos, i, nroComprobante, nroComprobanteAUX: longint;
  sumaTotal:real;
  bCorte: boolean;

begin
  Pila_Crear(pilaAuxCC, Unit1.sRuta, 'pilaTemp');
  Pila_Abrir(pilaAuxCC);
  Cola_Crear(colaAuxComp, unit1.sRuta, 'colaTemp');
  Cola_Abrir(colaAuxComp);

  Form_ListadosCC.StringGrid_Comprobantes.Cells[0,0]:='CLAVE';
  Form_ListadosCC.StringGrid_Comprobantes.Cells[1,0]:='FECHA';
  Form_ListadosCC.StringGrid_Comprobantes.Cells[2,0]:='TIPO';
  Form_ListadosCC.StringGrid_Comprobantes.Cells[3,0]:='NRO.';
  Form_ListadosCC.StringGrid_Comprobantes.Cells[4,0]:='NETO';
  Form_ListadosCC.StringGrid_Comprobantes.Cells[5,0]:='DESCUENTO';
  Form_ListadosCC.StringGrid_Comprobantes.Cells[6,0]:='GRAVADO';
  Form_ListadosCC.StringGrid_Comprobantes.Cells[7,0]:='IVA';
  Form_ListadosCC.StringGrid_Comprobantes.Cells[8,0]:='TOTAL';
```

```
Form_ListadosCC.StringGrid_Comprobantes.RowCount:=1;
i:=1;
sumaTotal:=0;

while (Pila_Vacia(Unit1.ME_CuentasCorrientes.Pila) = false) do
begin
 Pila_Tope(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);
 bCorte:=false;

 if (RegistroPila.Clave = claveCliente) then
 begin
  posDatos:= registroPila.Posicion;
  LO_Datos_Asientos.Datos_Capturar(Unit1.ME_CuentasCorrientes.Datos, posDatos, RegistroDatosASiento);
  nroComprobante:= RegistroDatosAsiento.nroComprobante;

  while (Cola_Vacia(Unit1.ME_Ventas.Comprobante.Cola) = false) and (bCorte = false) do
  begin
   Cola_Frente(Unit1.ME_Ventas.Comprobante.Cola, RegistroCola);

   if (RegistroCola.Clave = intToStr(nroComprobante)) then
   begin
    bCorte:=True;
    posDatos:= RegistroCOla.Posicion;
    LO_Datos_Comprobantes.Datos_Capturar(Unit1.ME_Ventas.Comprobante.Datos, posDatos, RegistroDatosComprobante);

    if RegistroDatosAsiento.importe < 0 then
    begin
       //Listar Reg
       Form_ListadosCC.StringGrid_Comprobantes.RowCount:= Form_ListadosCC.StringGrid_Comprobantes.RowCount + 1;

       Form_ListadosCC.StringGrid_Comprobantes.Cells[0,i]:= claveCliente;
       Form_ListadosCC.StringGrid_Comprobantes.Cells[1,i]:= RegistroDatosComprobante.fecha;
       Form_ListadosCC.StringGrid_Comprobantes.Cells[2,i]:= registroDatosComprobante.tipoComprobante;
```

```
          Form_ListadosCC.StringGrid_Comprobantes.Cells[3, i]:= intToStr(nroComprobante);
          Form_ListadosCC.StringGrid_Comprobantes.Cells[4, i]:= '$ '+FormatFloat('#.##', RegistroDatosComprobante.neto);
          Form_ListadosCC.StringGrid_Comprobantes.Cells[5, i]:= '% '+IntToStr(RegistroDatosComprobante.descuento);
          Form_ListadosCC.StringGrid_Comprobantes.Cells[6, i]:= '$ '+FormatFloat('#.##', RegistroDatosComprobante.gravado);
          Form_ListadosCC.StringGrid_Comprobantes.Cells[7, i]:= '$ '+FormatFloat('#.##', RegistroDatosComprobante.IVA);
          Form_ListadosCC.StringGrid_Comprobantes.Cells[8, i]:= '$ '+FormatFloat('#.##', RegistroDatosComprobante.total);
          sumaTotal:= sumaTotal + RegistroDatosComprobante.total;
          i:=i+1;
        end

    end;//endif

    Cola_Encolar(colaAuxComp, RegistroCola);
    Cola_Decolar(unit1.ME_Ventas.Comprobante.Cola);
  end;//while colaComp

  //Volvemos a dejar la colaCompr como estaba
  while (Cola_Vacia(colaAuxComp) = false) do
  begin
    Cola_Frente(colaAuxComp, REgistroCola);
    Cola_Encolar(Unit1.ME_Ventas.Comprobante.Cola, RegistroCola);
    Cola_Decolar(colaAuxComp);
  end;

 end;//End if CCclave = codCliente


 Pila_Desapilar(Unit1.ME_CuentasCorrientes.Pila);
 Pila_Apilar(pilaAuxCC, RegistroPila);
end;//pilaCC

//Volvemos a dejar la pila como estaba
while (Pila_Vacia(pilaAuxCC) = false) do
```

```
  begin
    Pila_Tope(pilaAuxCC, RegistroPila);
    Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);
    Pila_Desapilar(pilaAuxCC);
  end;

  Form_ListadosCC.Label_SumaTotal.Caption:= FormatFloat('#.##', sumaTotal);

  Pila_Destruir(pilaAuxCC);
  Cola_Destruir(colaAuxComp);
end;

procedure ListarResumenCC;
var
  pilaAuxCC: LO_DobleEnlace.Tipo_Indice;
  RegistroPila, RegistroIndice: LO_DobleEnlace.Tipo_Registro_Indice;
  RegistroDatosAsiento: LO_Datos_Asientos.Tipo_Registro_Datos;
  pos, posDatos, i: Longint;
  suma: real;
  RegistroDatosCliente: LO_Datos_Clientes.Tipo_Registro_Datos;
  nombreCliente: string;
begin
  LO_Pila.Pila_Crear(pilaAuxCC, unit1.sRuta, 'pilaTemp');
  LO_Pila.Pila_Abrir(pilaAuxCC);
  pos:= LO_DobleEnlace.DobleEnlace_Primero(Unit1.ME_Clientes.Indice);
  Form_ListadosCC.StringGrid_ResumenCC.Cells[0,0]:='CLAVE';
  Form_ListadosCC.StringGrid_ResumenCC.Cells[1,0]:='CLIENTE';
  Form_ListadosCC.StringGrid_ResumenCC.Cells[2,0]:='SUMA DE IMPORTES';
  Form_ListadosCC.StringGrid_ResumenCC.RowCount:=1;
  i:=1;


  if (LO_DobleEnlace.DobleEnlace_Vacio(Unit1.ME_Clientes.Indice) = false) then
```

```
begin
  while pos <> LO_DobleEnlace.DobleEnlace_PosNula(Unit1.ME_Clientes.Indice)) do
begin
  suma:=0;
  DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, pos, RegistroIndice);
  posDatos:= RegistroIndice.Posicion;
  LO_Datos_Clientes.Datos_Capturar(Unit1.ME_Clientes.Datos, posDatos, RegistroDatosCliente);
  nombreCliente:= RegistroDatosCliente.nombre;



  while (Pila_Vacia(Unit1.ME_CuentasCorrientes.Pila) = false) do
  begin
    Pila_Tope(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);


    if (RegistroPila.Clave = RegistroIndice.Clave) then
    begin
      posDatos:= RegistroPila.Posicion;
      LO_Datos_Asientos.Datos_Capturar(Unit1.ME_CuentasCorrientes.Datos, posDatos, RegistroDatosAsiento);
      if (RegistroDatosAsiento.importe > 0) then suma:= suma + RegistroDatosAsiento.importe;
    end;


    Pila_Apilar(pilaAuxCC, REgistroPila);
    Pila_Desapilar(Unit1.ME_CuentasCorrientes.Pila);
  end;//pila CC

  //Volvemos a dejar la pilaCC como estaba
  while Pila_Vacia(pilaAuxCC) = false do
  begin
    Pila_Tope(pilaAuxCC, RegistroPila);
    Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);
    Pila_Desapilar(pilaAuxCC);
  end;
```

```
    Form_listadosCC.StringGrid_ResumenCC.RowCount:= Form_ListadosCC.StringGrid_ResumenCC.RowCount + 1;
    Form_ListadosCC.StringGrid_ResumenCC.Cells[0, i]:= RegistroIndice.Clave;
    Form_ListadosCC.StringGrid_ResumenCC.Cells[1, i]:= nombreCliente;
    Form_ListadosCC.StringGrid_ResumenCC.Cells[2,i]:= '$ '+( FormatFloat('#.##', suma) );
    i:=i+1;
   pos:= RegistroIndice.Sig;
  end;//while
  end;

  LO_Pila.Pila_Destruir(pilaAuxCC);
end;

procedure ListarCCDeCliente(clave: string);
var
  PilaAuxCC, ColaAuxComp: LO_DobleEnlace.Tipo_Indice;
  RegistroPila, RegistroCola, RegistroCliente: LO_DobleEnlace.Tipo_Registro_indice;
  RegistroDatosCliente: LO_Datos_Clientes.Tipo_Registro_Datos;
  RegistroDatosAsiento: LO_Datos_Asientos.Tipo_Registro_Datos;
  RegistroDatosComprobante: LO_Datos_Comprobantes.Tipo_REgistro_Datos;
  posDatos, pos, nroComprobante, i: longint;

  nombreCliente, fecha, tipoComprobante, importe: string;
begin
  LO_Pila.Pila_Crear(pilaAuxCC, unit1.sRuta, 'pilaTemp');
  LO_Pila.Pila_Abrir(pilaAuxCC);
  LO_Cola.Cola_Crear(ColaAuxComp, unit1.sRuta, 'colaTemp');
  LO_Cola.Cola_Abrir(ColaAuxComp);

  LO_DobleEnlace.DobleEnlace_Buscar(Unit1.ME_Clientes.Indice, clave, pos);
  LO_DobleEnlace.DobleEnlace_Capturar(Unit1.ME_Clientes.Indice, pos, RegistroCliente);
  posDatos:= RegistroCliente.Posicion;
  LO_Datos_Clientes.Datos_Capturar(Unit1.ME_Clientes.Datos, posDatos, RegistroDatosCliente);
  Form_ListadosCC.StringGrid_CCCliente.RowCount:= 1;
```

```
Form_ListadosCC.StringGrid_CCCliente.Cells[0,0]:='CLAVE';
Form_ListadosCC.StringGrid_CCCliente.Cells[1,0]:='NOMBRE CLIENTE';
Form_ListadosCC.StringGrid_CCCliente.Cells[2,0]:='FECHA';
Form_ListadosCC.StringGrid_CCCliente.Cells[3,0]:='TIPO COMP.';
Form_ListadosCC.StringGrid_CCCliente.Cells[4,0]:='NRO. COMP.';
Form_ListadosCC.StringGrid_CCCliente.Cells[5,0]:='IMPORTE';

nombreCliente:= RegistroDatosCliente.Nombre;

while Pila_Vacia(Unit1.ME_CuentasCorrientes.Pila) = false do
begin
 Pila_Tope(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);

 if (RegistroPila.Clave = clave) then
 begin
  posDatos:= RegistroPila.Posicion;
  LO_Datos_Asientos.Datos_Capturar(Unit1.ME_CuentasCorrientes.Datos, posDatos, RegistroDatosAsiento);

  nroComprobante:= RegistroDatosAsiento.nroComprobante;

  while (Cola_Vacia(Unit1.ME_Ventas.Comprobante.Cola) = false) {and (RegistroDatosAsiento.importe > 0) }do
  begin
   Cola_Frente(Unit1.ME_Ventas.Comprobante.Cola, RegistroCola);

   if ( IntToSTr(nroComprobante) = RegistroCola.Clave ) then
   begin
    posDatos:= REgistroCOla.Posicion;
    LO_Datos_Comprobantes.Datos_Capturar(Unit1.ME_Ventas.Comprobante.Datos, posDatos, RegistroDatosComprobante);

    fecha:= registroDatosComprobante.fecha;
    tipoComprobante:= RegistroDatosComprobante.tipoComprobante;
    Importe:= FormatFloat('#.##',RegistroDatosAsiento.importe);
```

```
      Form_ListadosCC.StringGrid_CCCliente.RowCount:= Form_ListadosCC.StringGrid_CCCliente.RowCount + 1;

      for i:= Form_ListadosCC.StringGrid_CCCliente.RowCount - 1 downto 1 do
      begin
        Form_ListadosCC.StringGrid_CCCliente.Rows[i+1].Assign(Form_ListadosCC.StringGrid_CCCliente.Rows[i]);
      end;
      Form_ListadosCC.StringGrid_CCCliente.Cells[0,1]:= clave;
      Form_ListadosCC.StringGrid_CCCliente.Cells[1,1]:= nombreCliente;
      Form_ListadosCC.StringGrid_CCCliente.Cells[2,1]:= fecha;
      Form_ListadosCC.StringGrid_CCCliente.Cells[3,1]:= tipoComprobante;
      Form_ListadosCC.StringGrid_CCCliente.Cells[4,1]:= IntToStr(NroComprobante);
      Form_ListadosCC.StringGrid_CCCliente.Cells[5,1]:= '$ '+importe;
    end;

    Cola_Encolar(colaAuxComp, RegistroCola);
    Cola_Decolar(Unit1.ME_Ventas.Comprobante.Cola);
  end;//ColaComp
 end;

 //Dejamos cola como estaba
 while Cola_Vacia(ColaAuxComp) = false do
 begin
   Cola_Frente(colaAuxComp, RegistroCola);
   Cola_Encolar(Unit1.ME_Ventas.Comprobante.Cola, RegistroCola);
   Cola_Decolar(colaAuxComp);
 end;


 Pila_Apilar(PilaAuxCC, RegistroPila);
 Pila_Desapilar(Unit1.ME_CuentasCorrientes.Pila);
end; //PilaCC

//Dejamos pila CC como estaba
```

```
  while (Pila_Vacia(pilaAuxCC)=false) do
  begin
    Pila_Tope(pilaAuxCC, RegistroPila);
    Pila_Apilar(Unit1.ME_CuentasCorrientes.Pila, RegistroPila);
    Pila_Desapilar(pilaAuxCC);
  end;

  LO_Pila.Pila_Destruir(pilaAuxCC);
  LO_Cola.Cola_Destruir(ColaAuxComp);
end;

procedure TForm_ListadosCC.StringGrid1SelectCell(Sender: TObject; ACol,
  ARow: Integer; var CanSelect: Boolean);
begin
  if (Form_ListadosCC.ComboBox1.Text ='Cta. Corriente de un Cliente') and (aRow <> 0)  then
  begin
    ListarCCDeCliente(Form_ListadosCC.StringGrid1.Cells[0, aRow]);
  end else
  if (Form_ListadosCC.ComboBox1.Text = 'Comprobantes de un Cliente') and (aRow <> 0) then
  begin
    ListarComprobantesPorCliente(Form_ListadosCC.StringGrid1.Cells[0, aRow]);
  end
end;

procedure TForm_ListadosCC.ComboBox1Change(Sender: TObject);
var
  s: String;
begin
  s:= Form_ListadosCC.ComboBox1.Text;

  if s='Cta. Corriente de un Cliente' then
  begin
    Form_listadosCC.Label1.Visible:=true;
```

```
  Form_ListadosCC.StringGrid_CCCliente.Visible:=true;
  Form_ListadosCC.StringGrid_ResumenCC.Visible:= false;
  Form_ListadosCC.StringGrid_Comprobantes.Visible:=false;
  Form_listadoscc.Label2.Visible:=false;
  Form_listadoscc.Label_SumaTotal.Visible:=false;
  ListarClientes;
 end else
 if s= 'Resumen Cuentas Corrientes' then
 begin
  Form_listadosCC.Label1.Visible:=false;
  Form_ListadosCC.StringGrid_CCCliente.Visible:=false;
  Form_ListadosCC.StringGrid_ResumenCC.Visible:= true;
  Form_ListadosCC.StringGrid_Comprobantes.Visible:=false;
  Form_listadoscc.Label2.Visible:=false;
  Form_listadoscc.Label_SumaTotal.Visible:=false;

  ListarResumenCC;
 end else
 if s= 'Comprobantes de un Cliente'  then
 begin
  Form_listadosCC.Label1.Visible:=true;
  Form_ListadosCC.StringGrid_CCCliente.Visible:=false;
  Form_ListadosCC.StringGrid_ResumenCC.Visible:= false;
  Form_ListadosCC.StringGrid_Comprobantes.Visible:=true;
  Form_listadoscc.Label2.Visible:=true;
  Form_listadoscc.Label_SumaTotal.Visible:=true;
  ListarClientes;
 end
end;
end.
```

LO_ABB

```
Unit LO_ABB;

Interface

uses

SysUtils, Math, LO_Indice; //Se utiliza la LO_Indice para guardar elementos temporalmente en un indice (para reconstruir el arbol cuando se balancea)

Type

Tipo_Posicion = Longint ;

Tipo_Clave    = Longint ;

Tipo_Cadena   = String [255] ;

Tipo_Cantidad = Longint;

Tipo_Porcentaje = 1..100;


Tipo_Registro_Control = Record
                          Raiz, Borrados : Tipo_Posicion;
                          Ultimo: Tipo_Cantidad;
                          Nombre, ruta: Tipo_Cadena ;
                          porcentajeTolerancia: Tipo_Porcentaje;
                        porcentajeNiveles: Tipo_Porcentaje;
                         End;


Tipo_Archivo_Control = File of Tipo_Registro_Control ;


Tipo_Registro_Indice   = Record
                            Clave: Tipo_Clave ;
                         Posicion: Tipo_Posicion ;
                         Padre, HijoIzq, HijoDer : Tipo_Posicion ;
                        End;


Tipo_Archivo_Indice = File of Tipo_Registro_Indice ;


Tipo_Registro_Nivel = record
                          CantidadElementos: Tipo_Cantidad;
                       end;
Tipo_Archivo_Nivel = file of Tipo_Registro_Nivel;
```

```
Tipo_Arbol  = Record
                C: Tipo_Archivo_Control;
                I: Tipo_Archivo_Indice;
                N: Tipo_Archivo_Nivel;
              End;
```

```
 function Arbol_Crear (var Arbol: Tipo_Arbol; sRuta, sNombre: Tipo_Cadena; porcentajeTolerancia, porcentajeNiveles:
Tipo_Porcentaje): boolean; //Crea los archivos de Control Indice y Niveles referidos al ABB

 function Arbol_Creado (var Arbol: Tipo_Arbol; sRuta, sNombre: Tipo_Cadena): boolean;
//Devuelve true si los archivos del ABB existen y false si no existen

 procedure Arbol_Abrir (var Arbol: Tipo_Arbol);                                          //Abre los archivos

 procedure Arbol_Cerrar(var Arbol: Tipo_Arbol);                                          //Cierra los archivos

 function Arbol_Buscar (var Arbol: Tipo_Arbol; Clave:Tipo_Clave; var pos:Tipo_Posicion; Var Nivel: Tipo_Cantidad): Boolean;
//Busca de manera binaria un nodo, si no lo encuentra devuelve la posicion del que deberia ser el padre, y el nivel a donde deberia
pertenecer el nodo

 procedure Arbol_Insertar(var Arbol: Tipo_Arbol; pos:Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice; Nivel: Tipo_Cantidad);
//Inserta una hoja al arbol

 procedure Arbol_Eliminar (var Arbol: Tipo_Arbol; pos: Tipo_Posicion; nivel: Tipo_Cantidad);
//Elimina un nodo de un arbol

 function Arbol_Raiz (var Arbol: Tipo_Arbol): Tipo_Posicion;                             //Devuelve la
posicion de la raiz del arbol

 function Arbol_HijoIzquierdo ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion): Tipo_Posicion;
//Devuelve la posicion del hijo izquierdo de un nodo

 function Arbol_HijoDerecho ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion): tipo_posicion;
//Devuelve la posicion del hijo derecho de un nodo

 function Arbol_Padre ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion): tipo_posicion;        //Devuelve
la posicion del padre de un nodo

 function Arbol_Vacio ( var Arbol: Tipo_Arbol): Boolean ;                                //Devuelve true si el
arbol se encuentra vacio y flase si no

 procedure Arbol_Capturar  ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion; var RegistroIndice:Tipo_Registro_Indice );
//Captura un nodo de la posicion pasada por parametro en el RegistroIndice

 procedure Arbol_Modificar  ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion; RegistroIndice:Tipo_Registro_Indice );
//Modifica el nodo que se encuentra en la posicion pos

 function Arbol_PosNula (var Arbol: Tipo_Arbol): longint;                                //Devuelve la
posicion nula de la libreria operacional

 function Arbol_ClaveNula (var Arbol: Tipo_Arbol): Tipo_Clave;                           //Devuelve la
clave nula de la libreria operacional
```

```
  procedure Arbol_Destruir  (var Arbol: Tipo_Arbol);                                    //Elimina los
archivos referidos al ABB

  function Arbol_Equilibrado (var Arbol: Tipo_Arbol) : boolean;                         //Devuelve true si el
arbol se encuentra equilibrado y false si no

  procedure MigrarElementos (var Arbol: Tipo_Arbol; var Indice: LO_Indice.Tipo_Indice);  //Mueve
todos los elementos de un arbol a un archivo ordenado.

  procedure LlenarArbol(var Arbol: Tipo_Arbol; Indice: LO_Indice.Tipo_Indice; ini, fin: Tipo_Posicion);
//Algoritmo recursivo que crea un nuevo arbol a partir de una estructura de indice ordenada. (Se invoca en RebalancearArbol)

  procedure Arbol_Rebalanceo (var Arbol: Tipo_Arbol; sRuta, sNombre: Tipo_Cadena);      //Serie
de algoritmos para rebalancear arbol

  procedure ActualizarNiveles(var Arbol: Tipo_Arbol; posNodo: Tipo_Posicion; Nivel: Tipo_Cantidad);
//Utiliza recursividad para actualizar todos los niveles infieriores al nodo pasado por parametro (se utiliza esta funcion al eliminar
realizando un puente).

  function Arbol_UltimoNivel(var Arbol: Tipo_Arbol): Tipo_Cantidad;

  procedure Arbol_CapturarNivel(var Arbol: Tipo_Arbol; pos: Tipo_Posicion; var RegistroNivel: Tipo_Registro_Nivel);

  function Arbol_PorcentajeTolerancia (var Arbol: Tipo_Arbol): Tipo_Porcentaje;

  function Arbol_PorcentajeNiveles(var Arbol: Tipo_Arbol): Tipo_Porcentaje;

  function Arbol_CantidadElementos (var Arbol: Tipo_Arbol): integer;

  procedure Arbol_CambiarPorcentajeTolerancia(var Arbol: Tipo_Arbol; porcentajeTolerancia: Tipo_Porcentaje);

  procedure Arbol_cambiarPorcentajeNiveles(var Arbol: Tipo_Arbol; porcentajeNivel: Tipo_Porcentaje);


implementation


//-------------------------------------------------------------------------------------------


procedure ActualizarNiveles(var Arbol: Tipo_Arbol; posNodo: Tipo_Posicion; Nivel: Tipo_Cantidad);
var
  RegistroIndice: Tipo_Registro_Indice;
  RegistroNivel: Tipo_Registro_Nivel;
begin
  if (posNodo <> _posicion_Nula) then
  begin
    Seek(Arbol.I, posNodo);
    Read(Arbol.I, RegistroIndice);
```

```
    Seek(Arbol.N, Nivel);
    Read(Arbol.N, RegistroNivel);
    RegistroNivel.CantidadElementos:= RegistroNivel.CantidadElementos + 1 ;
    Seek(Arbol.N, Nivel);
    Write(Arbol.N, RegistroNivel);

    Seek(Arbol.N, Nivel+1);
    Read(Arbol.N, RegistroNivel);
    RegistroNivel.CantidadElementos:= RegistroNivel.CantidadElementos - 1 ;
    Seek(Arbol.N, Nivel+1);
    Write(Arbol.N, RegistroNivel);

    ActualizarNiveles(Arbol, RegistroIndice.HijoIzq, Nivel+1);
    ActualizarNiveles(Arbol, RegistroIndice.HijoDer, Nivel+1);

  end
end;//end ActualizarNiveles;

//---------------------------------------------------------------------------------------

function Arbol_Creado (var Arbol: Tipo_Arbol; sRuta, sNombre: Tipo_Cadena): boolean;
var
  sArchivoIndice, sArchivoControl, sArchivoNiveles: string;
  bHayError: boolean;
begin
    sArchivoIndice:= sRuta+'\'+sNombre+'.ntx';
    sArchivoControl:= sRuta+'\'+sNombre+'.con';
    sArchivoNiveles:= sRuta+'\'+sNombre+'.niv';

    Assign(Arbol.I, sArchivoIndice);     //Asignamos archivo de Indice
    Assign(Arbol.C, sArchivoControl);     //Asignamos archivo de Control
    Assign(Arbol.N, sArchivoNiveles); //Asignamos archivo de niveles
```

```
    {$I-}

    Reset(Arbol.I);
    Reset(Arbol.N);
    Reset(Arbol.C);

    bHayError:= IoResult <> 0;


    Arbol_Creado:= not bHayError;

    {$I+}

end; //End Arbol_Creado

//-------------------------------------------------------------------------------------

function  Arbol_Crear  ( var Arbol: Tipo_Arbol; sRuta, sNombre: Tipo_Cadena; porcentajeTolerancia, porcentajeNiveles:
Tipo_Porcentaje): boolean ;
var
  sArchivoIndice, sArchivoControl , sArchivoNiveles: Tipo_Cadena;
  bHayError: boolean;

  RegistroControl: Tipo_Registro_Control;
begin

    sArchivoIndice:= sRuta+'\'+sNombre+'.ntx';
    sArchivoControl:= sRuta+'\'+sNombre+'.con';
    sArchivoNiveles:= sRuta+'\'+sNombre+'.niv';

    Assign(Arbol.I, sArchivoIndice);     //Asignamos archivo de Indice
    Assign(Arbol.C, sArchivoControl);     //Asignamos archivo de Control
    Assign(Arbol.N, sArchivoNiveles); //Asignamos archivo de niveles
```

```
{$I-}

Reset(Arbol.I);

bHayError:= (IoResult <> 0);

if bHayError then Rewrite(Arbol.I);

close (Arbol.I);


//NIVELES
Reset(Arbol.N);

bHayError:= (IoResult <> 0);

if bHayError then Rewrite(Arbol.N);


//Ahora lo mismo para control e iniciarlo

Reset(Arbol.C);

bHayError:= IoResult <> 0;

if bHayError then
  begin

    Rewrite(Arbol.C);
    RegistroControl.Ruta    := sRuta;
    RegistroControl.Nombre  := sNombre;
    RegistroControl.Ultimo:= _posicion_nula;
    RegistroControl.Raiz:= _Posicion_Nula;
```

```
        RegistroControl.Borrados:= _Posicion_Nula;
      RegistroControl.porcentajeTolerancia:= porcentajeTolerancia;
        RegistroControl.porcentajeNiveles:= porcentajeNiveles;

        Seek(Arbol.C, 0);
        write(Arbol.C,RegistroControl);
      end;
    Arbol_Crear:= bHayError;
    {$I+}
end; //End ArbolCrear

//---------------------------------------------------------------------------------------

procedure Arbol_Abrir (var Arbol: Tipo_Arbol);
begin
 Reset(Arbol.I);
   Reset(Arbol.C);
 Reset(Arbol.N);
end; //End ArbolAbrir

//---------------------------------------------------------------------------------------

procedure Arbol_Cerrar (var Arbol: Tipo_Arbol);
begin
 Close(Arbol.C);
 Close(Arbol.I);
 Close(Arbol.N);
end; //End ArbolCerrar

//---------------------------------------------------------------------------------------

function Arbol_Buscar  ( var Arbol: Tipo_Arbol; Clave:Tipo_Clave; Var pos:Tipo_Posicion; Var Nivel: Tipo_Cantidad): Boolean;
{Si encuentra elemento devuelve TRUE y pos= posicion de elemento, nivel es el nivel del elemento.
```

```
IMPORTANTE SI NO lo encuentra devuelve FALSE y pos es la posicion del PADRE o sea donde deberia engancharse el
elemento, y nivel es el nivel donde DEBERIA pertenecer el nuevo elemento}
Var
  RegistroControl: Tipo_Registro_Control;
  RegistroIndice: Tipo_Registro_Indice;
  PosPadre: Tipo_Posicion;
  bEncontrado : Boolean ;
Begin

seek (Arbol.C, 0);
Read (Arbol.C, RegistroControl);

posPadre := _posicion_nula;
pos := RegistroControl.Raiz;
bEncontrado:= False;
Nivel:=0;

while (bEncontrado = false) and (pos <> _posicion_nula) do
begin
            Seek ( Arbol.I, pos);
            Read ( Arbol.I, RegistroIndice );

            if (RegistroIndice.Clave = clave) then bEncontrado := True
            else
              begin
                Nivel:= Succ(Nivel);
                posPadre:= pos;
                if ( clave <= RegistroIndice.Clave) Then  pos:= RegistroIndice.HijoIzq
                Else  pos:= RegistroIndice.HijoDer;
              end;
end;

if (bEncontrado = false) then
begin
```

```
  pos := PosPadre ;
end;

Arbol_Buscar:= bEncontrado;

End; //End ArbolBuscar

//-------------------------------------------------------------------------------------

procedure Arbol_Insertar ( var Arbol: Tipo_Arbol; pos:Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice; Nivel: Tipo_Cantidad);
var
 Rpadre: Tipo_Registro_Indice;
 RegistroControl: Tipo_Registro_Control;
 posGrabar: Tipo_Posicion;

 RegistroNivel: Tipo_Registro_Nivel;

Begin
   Seek (Arbol.C,  0);
   Read (Arbol.C, RegistroControl);

   posGrabar:= FileSize ( Arbol.I );

   if (RegistroControl.Raiz =  _posicion_nula)   or  (pos = _posicion_nula) then
  //insertar en Arbol vacio
  begin
                RegistroIndice.padre:= _posicion_nula;
                RegistroIndice.HijoIzq:= _posicion_nula;
                RegistroIndice.HijoDer:= _posicion_nula;

                RegistroControl.Raiz := posGrabar;

                RegistroControl.Ultimo:= 0;
```

```
            RegistroNivel.CantidadElementos:=1;
end
 else
//Insertar una hoja
begin
            RegistroIndice.padre:= pos ;
            RegistroIndice.HijoIzq:= _posicion_nula;
           RegistroIndice.HijoDer:= _posicion_nula;

            Seek (Arbol.I, pos );
            Read ( Arbol.I, Rpadre );

            if (RegistroIndice.Clave < Rpadre.Clave) then Rpadre.HijoIzq:= posGrabar
            else   Rpadre.HijoDer:= posGrabar;

            Seek (Arbol.I, pos);
            Write (Arbol.I, Rpadre);

            if (  Nivel < FileSize(Arbol.N)  ) then
            begin
                        Seek(Arbol.N, Nivel);
                        Read(Arbol.N, RegistroNivel);
                        RegistroNivel.CantidadElementos:= Succ(RegistroNIvel.CantidadElementos);
            end else
                        begin
                                    RegistroNivel.CantidadElementos:=1;
                                    RegistroControl.Ultimo:= nivel;
                        end

end;

 //SE graba todo.
```

```
  Seek  (Arbol.C, 0);
  Write (Arbol.C, RegistroControl);

  Seek  (Arbol.I, posGrabar);
  Write (Arbol.I, RegistroIndice);

  Seek( Arbol.N, nivel);
  Write(Arbol.N, RegistroNivel);

End; //End ArbolInsertar

//---------------------------------------------------------------------------------------

procedure Arbol_Eliminar (var Arbol: Tipo_Arbol; pos: Tipo_Posicion; nivel: Tipo_Cantidad);

var
 RegistroControl: Tipo_Registro_Control;
 Rpadre, RegistroPadre, Raux, RegistroIndice, RegPadre, RegIzq, RegDer, RegistroSustituto, RegistroPadreSustituto,
RegistroHijoIzq, RegistroHijoDer: Tipo_Registro_Indice;
 posPadre, posHijoIzq, posHijoDer, posCandidato, posSustitutoIzq, posSustitutoDer, posSustituto, posPadreSus : Tipo_Posicion;
  nivelARestar: integer;
  contadorPasosIzq, contadorPasosDer: integer;

  RegistroNivel: Tipo_Registro_Nivel;

begin
 Seek ( Arbol.C, 0);
 Read ( Arbol.C, RegistroControl);

 Seek ( Arbol.I, pos );
 Read ( Arbol.I, RegistroIndice );

  //CASO 1 - elimino y queda arbol vacio
  //El nivel a restar es el que indica la variable nivel
```

```
if (RegistroControl.raiz = pos) and (RegistroIndice.hijoIzq = _posicion_nula) and (RegistroIndice.HijoDer = _posicion_nula) then
 begin
  RegistroControl.Raiz := _posicion_nula;
  nivelARestar:= nivel;
 end
 else
 //CASO 2: elimino hoja
 //El nivel a restar es el que indica la variable nivel
 if (RegistroIndice.HijoIzq = _posicion_nula) and (RegistroIndice.HijoDer = _posicion_nula) then
 begin
  posPadre := RegistroIndice.padre;
  Seek ( Arbol.I, posPadre );
  Read  (Arbol.I, Rpadre );

  if (Rpadre.HijoIzq = pos) then Rpadre.HijoIzq := _posicion_nula
  else Rpadre.HijoDer := _posicion_nula;

  Seek ( Arbol.I, posPadre );
  Write (Arbol.I, Rpadre );

  nivelARestar:= nivel;

 end
 else
 //CASO 3:  es un padre con solo un nodo hijo. SE RESUELVE POR PUENTE.
 //Se deben actualizar los niveles inferiores al nivel pasado por parametro. Al resolver por puente el nivel pasado por parametro
quedará con la misma cantidad de elementos, pero hay que recorrer los inferiores.
 if ((RegistroIndice.Hijoizq = _posicion_nula) and (RegistroIndice.HijoDer <> _posicion_nula)) or ((RegistroIndice.HijoDer =
_posicion_nula) and(RegistroIndice.Hijoizq <> _posicion_nula)) then
 begin
  posPadre  := RegistroIndice.Padre;
  posHijoIzq:= RegistroIndice.HijoIzq;
  posHijoDer:= RegistroIndice.HijoDer;
```

```
    //Lo marcamos con un -1 para mas tarde identificar que se realizó un puente (Al final de este procedimiento realizamos la actualizacion de niveles)
    nivelARestar:= -1;

    //Eliminamos un elemento del nivel (luego lo repondremos con el algoritmo recursivo)
    Seek(Arbol.N, Nivel);
    Read(Arbol.N, RegistroNivel);
    RegistroNivel.CantidadElementos:= RegistroNivel.CantidadElementos-1;
    Seek(Arbol.N, Nivel);
    Write(Arbol.N, RegistroNivel);

    if (posHijoIzq = _posicion_nula) then posCandidato := posHijoDer else posCandidato := posHijoIzq;

    //Resuelvo el padre

    if (RegistroIndice.Padre = _posicion_nula) then RegistroControl.Raiz:= posCandidato
    else
    begin
      Seek ( Arbol.I, posPadre) ;
      Read ( Arbol.I, RegPadre );

      if (RegPadre.HijoIzq = pos) then RegPadre.HijoIzq := posCandidato else RegPadre.HijoDer := posCandidato;

      Seek ( Arbol.I, posPadre );
      Write ( Arbol.I, regPadre );
    end;

    // Resuelvo el hijo (Sustituto)
    if (posHijoIzq <> _posicion_nula) then
    // Es el HIJO IZQUIERDO
    begin
      Seek ( Arbol.I, posHijoIzq);
      Read ( Arbol.I, RegIzq );
```

```
    RegIzq.padre:= posPadre;
    Seek ( Arbol.I, posHijoIzq);
    Write ( Arbol.I, RegIzq );
  end
  else
  // Es el HIJO DERECHO
  begin
    Seek ( Arbol.I, posHijoDer);
    Read ( Arbol.I, RegDer );
    RegDer.padre:= posPadre;
    Seek ( Arbol.I, posHijoDer);
    Write ( Arbol.I, RegDer  );
  end

end

else
//CASO 4: general, elimino "al medio" usando sustitucion
//El nivel a restar será alguno de los niveles siguientes al nivel pasado por parametro (BARRER RESTANDO LOS NIVELES SIGUIENTES)
begin
  // Paso 1: buscar SUSTITUTO por izquierda (mayor de los menores)-----------------------------
  //buscar por izq y luego el mas a la derecha.....

  contadorPasosIzq:= 0;
  contadorPasosDer:= 0;


  posSustitutoIzq:= RegistroIndice.HijoIzq;
  Seek(Arbol.I, posSustitutoIzq);
  Read(Arbol.I, RegistroSustituto);
  ContadorPasosIzq:= Succ(ContadorPasosIzq);

  while (RegistroSustituto.HijoDer <> _posicion_nula) do
```

```
begin
  posSustitutoIzq:= RegistroSustituto.HijoDer;
  Seek(Arbol.I, posSustitutoIzq);
  Read(Arbol.I, RegistroSustituto);
  contadorPasosIzq:= Succ(contadorPasosIzq);
end;

// Paso 2: buscar SUSTITUTO por derecha (menor de los mayores)-------------------------------
//buscar por derecha y luego el mas a la izquierda.....

posSustitutoDer := RegistroIndice.HijoDer;
Seek(Arbol.I, posSustitutoDer);
Read(Arbol.I, RegistroSustituto);

ContadorPasosDer:= Succ(ContadorPasosDer);

while (RegistroSustituto.HijoIzq <> _posicion_nula) do
begin
  posSustitutoDer:= RegistroSustituto.HijoIzq;
  Seek(Arbol.I, posSustitutoDer);
  Read(Arbol.I, RegistroSustituto);
  contadorPasosDer:= Succ(contadorPasosDer);
end;

//Comparo que susituto es "mejor" (cual esta mas lejos del nodo a eliminar)
if (contadorPasosIzq > contadorPasosDer) then
begin
  posSustituto:= posSustitutoIzq;
  NivelARestar:= nivel + contadorPasosIzq ;
end
else
begin
  posSustituto:= posSustitutoDer;
```

```
  NivelARestar:= nivel + contadorPasosDer;
end;

// Paso 3: hago la sustitucion entre pos y posSustituto
// ..... algoritmo de cambio de enlaces ....

Seek(Arbol.I, posSustituto);
Read(Arbol.I, RegistroSustituto);

posPadreSus:= RegistroSustituto.Padre;
Seek(Arbol.I, posPadreSus);
Read(Arbol.I, RegistroPadreSustituto);

if (RegistroPadreSustituto.HijoIzq = posSustituto) then RegistroPadreSustituto.HijoIzq:= _posicion_nula else
RegistroPadreSustituto.HijoDer:= _posicion_nula;

Seek(Arbol.I, posPadreSus);
Write(Arbol.I, RegistroPadreSustituto);

posPadre:= RegistroIndice.Padre;
RegistroSustituto.Padre:= posPadre;

RegistroSustituto.HijoIzq:= RegistroIndice.HijoIzq;
RegistroSustituto.HijoDer:= RegistroIndice.HijoDer;

//Puede ser que pos sea la raiz
if (RegistroSustituto.Padre = _posicion_nula) then
begin
    RegistroControl.Raiz := posSustituto;
    Seek(Arbol.C, 0);
    Write(Arbol.C, RegistroControl);


    Seek(Arbol.I, posSustituto);
```

```
      Write(Arbol.I, RegistroSustituto);
  end
  else
  begin
    Seek(Arbol.I, posPadre);
    Read(Arbol.I, RegistroPadre);

    if (RegistroPadre.HijoIzq = pos) then RegistroPadre.HijoIzq:= posSustituto else RegistroPadre.HijoDer:= posSustituto;

    Seek(Arbol.I, posPadre);
    Write(Arbol.I, RegistroPadre);

    Seek(Arbol.I, posSustituto);
    Write(Arbol.I, RegistroSustituto);

    posHijoIzq:= RegistroSustituto.HijoIzq;
    posHijoDer:= RegistroSustituto.HijoDer;

    Seek(Arbol.I, posHijoIzq);
    Read(Arbol.I, RegistroHijoIzq);
    RegistroHijoIzq.Padre:= posSustituto;
    Seek(Arbol.I, posHijoIzq);
    Write(Arbol.I, RegistroHijoIzq);

    Seek(Arbol.I, posHijoDer);
    Read(Arbol.I, RegistroHijoDer);
    RegistroHijoDer.Padre:= posSustituto;
    Seek(Arbol.I, posHijoDer);
    Write(Arbol.I, RegistroHijoDer);
  end;

end;//end caso4
```

```
//-----------------------------------------------------
//Engancho al nodo en la secuencia de eliminados

if RegistroControl.Borrados <> _posicion_nula then
begin
        Seek (Arbol.I, RegistroControl.Borrados);
        Read (Arbol.I, Raux );
        Raux.HijoIzq := pos ;
        Seek (Arbol.I, RegistroControl.Borrados);
        Write (Arbol.I, Raux );
end
 else
 begin
   RegistroControl.Borrados:= pos;
   Seek(Arbol.C, 0);
   Write(Arbol.C, RegistroControl);
 end;

// Grabo RegistroIndice
 RegistroIndice.Padre := _posicion_nula;
RegistroIndice.HijoIzq:= _posicion_nula;
RegistroIndice.HijoDer:= RegistroControl.Borrados;

Seek ( Arbol.I, pos );
Write ( Arbol.I, RegistroIndice );

//Grabo Nivel

 if (nivelARestar = -1) then  //Se realizó un puente, hay que actualizar niveles inferiores
 begin
   ActualizarNiveles(Arbol, posCandidato, Nivel);
   Nivel:= RegistroControl.Ultimo;
   Seek(Arbol.N, Nivel);
```

```
  Read(Arbol.N, RegistroNivel);

  if (RegistroNivel.CantidadElementos=0) then
  begin
    RegistroControl.Ultimo:= RegistroControl.Ultimo - 1 ;
    Seek(Arbol.C, 0);
    Write(Arbol.C, RegistroControl);
  end

end
else
begin
  Seek(Arbol.N, nivelARestar);
 Read(Arbol.N, RegistroNivel);

 RegistroNivel.CantidadElementos:= RegistroNivel.CantidadElementos-1;

 Seek( Arbol.N, nivelARestar);
 Write(Arbol.N, RegistroNivel);

 if (RegistroNivel.CantidadElementos = 0) then
 RegistroControl.Ultimo:= RegistroControl.Ultimo - 1;

 Seek(Arbol.C, 0);
 Write(Arbol.C, RegistroControl);

 end



End; //End ArbolEliminar

//-------------------------------------------------------------------------------------
```

```
function Arbol_Raiz (var Arbol: Tipo_Arbol): Tipo_Posicion;
Var
 RegistroControl: Tipo_Registro_Control;

begin
 Seek ( Arbol.C, 0);
 Read ( Arbol.C, RegistroControl);
 Arbol_Raiz:= RegistroControl.Raiz ;

end; //End ArbolRaiz

//-------------------------------------------------------------------------------------

function Arbol_HijoIzquierdo ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion): Tipo_Posicion;
Var
 RegistroIndice: Tipo_Registro_Indice;
begin
 Seek ( Arbol.I, pos );
 Read ( Arbol.I, RegistroIndice);
 Arbol_HijoIzquierdo := RegistroIndice.HijoIzq ;
End;        //End ArbolHijoIzq

//-------------------------------------------------------------------------------------

function Arbol_HijoDerecho ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion): tipo_posicion;
Var
 RegistroIndice: Tipo_Registro_Indice;
begin
 seek ( Arbol.I, pos );
 Read ( Arbol.I, RegistroIndice);
 Arbol_HijoDerecho:= RegistroIndice.HijoDer ;
End; //end ArbolHijoDer
```

```
//-------------------------------------------------------------------------------------

function Arbol_Padre ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion): tipo_posicion;
Var
 RegistroIndice: Tipo_Registro_Indice;
begin
 seek ( Arbol.I, pos );
 Read ( Arbol.I, RegistroIndice);
 Arbol_Padre:= RegistroIndice.Padre;
End;

//-------------------------------------------------------------------------------------

function Arbol_Vacio ( var Arbol: Tipo_Arbol): Boolean ;
Var
 RegistroControl: Tipo_Registro_Control;
 bResultado: boolean;
begin
 seek ( Arbol.C, 0);
 Read ( arbol.C, RegistroControl);

 if RegistroControl.Raiz <> _posicion_nula then bResultado:= false
 else
 bResultado:= true;

 Arbol_Vacio:= bResultado;

end;

//-------------------------------------------------------------------------------------

procedure Arbol_Capturar  ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion; var RegistroIndice:Tipo_Registro_Indice );
```

```
begin
 seek ( Arbol.I, pos );
 Read ( Arbol.I, RegistroIndice);
end;

//-------------------------------------------------------------------------------------

procedure Arbol_Modificar  ( Var Arbol: Tipo_Arbol; pos:Tipo_Posicion; RegistroIndice:Tipo_Registro_Indice );
Var
 Raux: Tipo_Registro_Indice;
begin
 seek ( Arbol.I, pos );
 Read ( Arbol.I, Raux);

 seek ( Arbol.I, pos );
 Write ( Arbol.I, RegistroIndice);
end;

//-------------------------------------------------------------------------------------

function Arbol_PosNula (var Arbol: Tipo_Arbol): longint;
begin
 Arbol_PosNula:= _posicion_nula;
end;//End ArbolPosNUla

//-------------------------------------------------------------------------------------

Function Arbol_ClaveNula (var Arbol: Tipo_Arbol): Tipo_Clave;
begin
 Arbol_ClaveNula:= _Clave_nula;
end;//End ArbolClaveNUla

//-------------------------------------------------------------------------------------
```

```
Procedure Arbol_Destruir  (var Arbol: Tipo_Arbol);
 begin

   Close(Arbol.C);
   Close(Arbol.I);
   Close(Arbol.N);

   Erase(Arbol.C);
   Erase(Arbol.I);
   Erase(Arbol.N);

 end; //Arbol_Destruir
//-------------------------------------------------------------------------------------------


function Arbol_Equilibrado (var Arbol: Tipo_Arbol) : boolean;
var
RegistroControl: Tipo_Registro_Control;
nivelesDeTolerancia, nivelesAControlar, nivelesIncompletos : Tipo_Cantidad;
 pos: Tipo_posicion;
 RegistroNivel: Tipo_Registro_Nivel;
begin
Seek(Arbol.C, 0 );
Read(Arbol.C, RegistroControl);

 if (RegistroControl.Raiz= _posicion_nula) then Arbol_Equilibrado:= true
 else
 begin

   nivelesAControlar:= Trunc( (RegistroControl.porcentajeNiveles / 100) * RegistroControl.Ultimo );

 nivelesDeTolerancia:= Trunc ( (RegistroControl.porcentajeTolerancia / 100 ) * nivelesAControlar ) ;
```

```
 nivelesIncompletos := 0 ;

 pos:= nivelesAControlar;

 while ( pos >= 0 ) and (nivelesIncompletos <= nivelesDeTolerancia) do
 begin
        Seek(Arbol.N, pos);
        Read(Arbol.N, RegistroNivel);

        if ( RegistroNivel.CantidadElementos < Power(2, pos) ) then nivelesIncompletos := nivelesIncompletos + 1;

        pos:=pos-1
 end;

 Arbol_Equilibrado := nivelesIncompletos <= nivelesDeTolerancia;
 end

end;//End ArbolEquilibrado

procedure MigrarElementos (var Arbol: Tipo_Arbol; var Indice: LO_Indice.Tipo_Indice);// Mueve todos los elementos de un arbol a
un archivo ordenado.
var

 RegistroIndice: LO_Indice.Tipo_Registro_Indice;

 RegistroControl: Tipo_Registro_Control;
 RegistroIndiceA, RegistroHijoIzq, RegistroHijoDer: Tipo_Registro_Indice;
 RegistroNivel: Tipo_Registro_nivel;

 pos, posPadre, posHijoIzq, posHijoDer, posNodo: Tipo_Posicion;

 cantidadElem, i, visitados: integer;
```

```
begin

Seek(Arbol.C,0);
Read(Arbol.C, RegistroControl);



//Primero debemos saber que cantidad de elementos hay en el arbol

cantidadElem:=0;

for i:= 0 to RegistroControl.Ultimo do
begin
        Seek(Arbol.N, i);
        Read(Arbol.N, RegistroNivel);
        cantidadElem:=cantidadElem+RegistroNivel.CantidadElementos;
end;

//Recorrer Arbol e ir insertando en indice.  SE UTILIZA UN RECORRIDO PRE-ORDEN

visitados:=0;

Seek(Arbol.I, RegistroControl.Raiz);
Read(Arbol.I, RegistroIndiceA);

    posNodo:= RegistroControl.Raiz;

while (visitados <> cantidadElem) do
begin

        Seek(Arbol.I, posNodo);
        Read(Arbol.I, RegistroIndiceA);

        posPadre:= RegistroIndiceA.Padre;
```

```
posHijoIzq:= RegistroIndiceA.HijoIzq;
posHijoDer:= RegistroIndiceA.HijoDer;

if (RegistroIndiceA.Clave <> _clave_nula) then //Si no esta visitado ya...
begin
  RegistroIndice.Clave:= RegistroIndiceA.Clave;
  RegistroIndice.Puntero:= RegistroIndiceA.Posicion;

  LO_Indice.Indice_Buscar(Indice, RegistroIndice.Clave, pos);
  LO_Indice.Indice_Insertar(Indice, pos, RegistroIndice);

  RegistroIndiceA.Clave:= _Clave_nula;
  visitados:= visitados+1;

  Seek(Arbol.I, posNodo);
  Write(Arbol.I, RegistroIndiceA);

  if (posHijoIzq <> _posicion_nula) then
  begin
    Seek(Arbol.I, posHijoIzq);
    Read(Arbol.I, RegistroHijoIzq);
    if (RegistroHijoIzq.Clave = _clave_nula) then posNodo:=posPadre
    else posNodo:= posHijoIzq;
  end
  else
    if (posHijoDer <> _posicion_nula) then posNodo:=posHijoDer
    else
      if (posPadre <> _posicion_nula) then posNodo:=posPadre;


end
else //YA ESTA VISITADO, ENTONCES ir para la derecha. (porque hacemos recorrido PRE ORDEN)
begin
```

```
                if (posHijoDer <> _Posicion_nula) then
                begin
                  Seek(Arbol.I, posHijoDer);
                  Read(Arbol.I, RegistroHijoDer);

                  if (RegistroHijoDer.clave = _clave_nula) then posNodo:= posPadre
                  else posNodo:= posHijoDer;
                end
                else posNodo:= posPadre;
              end

end; //while visitados <> cantidadElem
end;

procedure LlenarArbol(var Arbol: Tipo_Arbol; Indice: LO_Indice.Tipo_Indice; ini, fin: Tipo_Posicion); //Crea un nuevo arbol a partir de una estructura de indice ordenada. Usa recursion
var
RegistroArbol: Tipo_Registro_Indice;
RegistroIndice: LO_Indice.Tipo_Registro_Indice;
pos: Tipo_Posicion;
Nivel: Tipo_Cantidad;
 medio:integer;
begin
if (ini <= fin) then
begin
              medio:= (ini+fin) div 2;
              Seek( Indice.I, medio);
              Read( Indice.I, RegistroIndice);
              RegistroArbol.Clave:= RegistroIndice.Clave;
              RegistroArbol.Posicion:= RegistroIndice.Puntero;
              Arbol_Buscar(Arbol, RegistroArbol.Clave , pos, Nivel);
              Arbol_Insertar(Arbol, pos, RegistroArbol, nivel);
              LlenarArbol(Arbol, Indice, ini, medio-1);
```

```
            LlenarArbol(Arbol, Indice, medio+1, fin);
end;
end;

procedure Arbol_Rebalanceo (var Arbol: Tipo_Arbol; sRuta, sNombre: Tipo_Cadena); //Pasa todos los elementos del arbol a un indice y crea un nuevo arbol a partir de estos elementos pero insertandolos ordenadamente (a partir de busqueda binaria)
var Indice: LO_Indice.Tipo_Indice;
    RegistroControl: Tipo_Registro_Control;
    tolerancia, nivelesAControlar: Tipo_Porcentaje;
begin
  Seek(Arbol.C, 0);
  Read(Arbol.C, RegistroControl);
  tolerancia:= RegistroControl.porcentajeTolerancia;
  nivelesAControlar:= RegistroControl.porcentajeNiveles;

// Primer paso. Mover elementos de arbol a indice:
LO_Indice.Indice_Crear( Indice, sRuta, 'IndiceTemporal');
LO_Indice.Indice_Abrir(Indice);
MigrarElementos(Arbol, Indice);

// Segundo paso. Eliminar arbol original (Archivos de indice (arbol), control y niveles).
Arbol_Destruir(Arbol);

//Tercer paso. Crear un arbol nuevo.
Arbol_Crear(Arbol, sRuta, sNombre, tolerancia, nivelesAControlar);
Arbol_Abrir(Arbol);

//Cuarto paso. Llenar el arbol a partir de logica de busqueda binaria en el indice.
LlenarArbol(Arbol, Indice, 0, (Filesize(Indice.I)-1));

//Quinto paso. Eliminar Indice temporal.
LO_Indice.Indice_Destruir(Indice);

end; // end ArbolRebalanceo
```

```
//---------------------------------------------------------------------------

function Arbol_UltimoNivel(var Arbol: Tipo_Arbol): Tipo_Cantidad;
var
  RegistroControl: Tipo_Registro_Control;
begin
  Seek(Arbol.C, 0);
  Read(Arbol.C, RegistroControl);

  Arbol_UltimoNivel:= RegistroControl.Ultimo;
end;

//---------------------------------------------------------------------------

procedure Arbol_CapturarNivel(var Arbol: Tipo_Arbol; pos: Tipo_Posicion; var RegistroNivel: Tipo_Registro_Nivel);
begin
  Seek(Arbol.N, pos);
  Read(Arbol.N, RegistroNivel);
end;

//---------------------------------------------------------------------------

function Arbol_PorcentajeTolerancia (var Arbol: Tipo_Arbol): Tipo_Porcentaje;
var
  RegistroControl: Tipo_Registro_Control;
begin
  Seek(Arbol.C, 0);
  Read(Arbol.C, RegistroControl);

  Arbol_PorcentajeTolerancia:= RegistroControl.porcentajeTolerancia;
end;
```

```
//---------------------------------------------------------------------------

function Arbol_PorcentajeNiveles(var Arbol: Tipo_Arbol): Tipo_Porcentaje;
var
  RegistroControl: Tipo_Registro_Control;
begin
  Seek(Arbol.C, 0);
  Read(Arbol.C, RegistroControl);

  Arbol_PorcentajeNiveles:= RegistroControl.porcentajeNiveles;
end;

//---------------------------------------------------------------------------

function Arbol_CantidadElementos (var Arbol: Tipo_Arbol): integer;
var
  nResultado, i: integer;
  RegistroControl: Tipo_Registro_Control;
  RegistroNivel: Tipo_Registro_Nivel;
begin
  Seek(Arbol.C, 0);
  Read(Arbol.C, RegistroControl);
  nResultado:=0;

  if RegistroControl.Ultimo <> _posicion_nula then
  begin

    for i:= 0 to RegistroControl.Ultimo do
    begin
      Seek(Arbol.N, i);
      Read(Arbol.N, RegistroNivel);

      nResultado:= nResultado + RegistroNivel.CantidadElementos;
```

```
    end;
  end;

  Arbol_CantidadElementos:= nResultado;
end;

//---------------------------------------------------------------------------

procedure Arbol_cambiarPorcentajeNiveles(var Arbol: Tipo_Arbol; porcentajeNivel: Tipo_Porcentaje);
var
  RegistroControl: Tipo_Registro_Control;
begin
  Seek(Arbol.C, 0);
  Read(Arbol.C, RegistroControl);

  RegistroControl.porcentajeNiveles:= porcentajeNivel;

  Seek(Arbol.C, 0);
  Write(Arbol.C, RegistroControl);
end;

//---------------------------------------------------------------------------

procedure Arbol_cambiarPorcentajeTolerancia(var Arbol: Tipo_Arbol; porcentajeTolerancia: Tipo_Porcentaje);
var
  RegistroControl: Tipo_Registro_Control;
begin
  Seek(Arbol.C, 0);
  Read(Arbol.C, RegistroControl);

  RegistroControl.porcentajeTolerancia:= porcentajeTolerancia;

  Seek(Arbol.C, 0);
```

```
 Write(Arbol.C, RegistroControl);
end;
end.
```

LO_COLA

```
unit LO_Cola;
Interface
Uses SysUtils, LO_DobleEnlace;
function Cola_Crear  ( var Cola:Tipo_Indice; ruta, nombre: Tipo_Cadena ) :boolean;
procedure Cola_Abrir ( var Cola              :Tipo_Indice );
procedure Cola_Cerrar ( var Cola: Tipo_Indice );
procedure Cola_Decolar ( var Cola: Tipo_Indice );
procedure Cola_Encolar ( var Cola:Tipo_Indice; RegistroIndice: Tipo_Registro_Indice );
Procedure Cola_EncolarConFecha ( var Cola:Tipo_Indice; RegistroIndice:Tipo_Registro_Indice; TipoComprobante, fecha: String );
procedure Cola_Frente ( var Cola:Tipo_Indice;  var  RegistroIndice: Tipo_Registro_Indice);
Function Cola_Vacia  ( var Cola:Tipo_Indice ): boolean;
Function Cola_PosNula  ( var Cola:Tipo_Indice): Tipo_Posicion ;
Function Cola_ClaveNula  ( var Cola:Tipo_Indice): Tipo_Clave ;
procedure Cola_Tope (var Cola: Tipo_Indice; var RegistroIndice: Tipo_Registro_Indice);
procedure Cola_Destruir(var Cola: Tipo_Indice);
function UltimaFechaEmitida(var Cola: Tipo_Indice; tipoComprobante: string): string;
function Cola_UltimoNroComprobante(var Cola:Tipo_Indice): integer;
Implementation
function Cola_Crear  ( var Cola:Tipo_Indice; ruta, nombre: Tipo_Cadena ) :boolean;
begin
 Cola_Crear:= LO_DobleEnlace.DobleEnlace_Crear ( Cola, ruta, nombre );
end;
//-------------------------------------------------------------
procedure Cola_Abrir ( var Cola              :Tipo_Indice );
begin
 LO_DobleEnlace.DobleEnlace_Abrir ( Cola );
end;
```

```
//---------------------------------------------------------------
Procedure Cola_Cerrar ( var Cola: Tipo_Indice );
begin
 LO_DobleEnlace.DobleEnlace_Cerrar ( Cola );
End;
//---------------------------------------------------------------
Procedure Cola_Decolar ( var Cola: Tipo_Indice );
begin
 LO_DobleEnlace.DobleEnlace_Eliminar ( Cola, LO_DobleEnlace.DobleEnlace_Primero(Cola) );
End;
//---------------------------------------------------------------------------
Procedure Cola_Encolar ( var Cola:Tipo_Indice; RegistroIndice:Tipo_Registro_Indice );
begin
 LO_DobleEnlace.DobleEnlace_Insertar ( Cola, -1  , RegistroIndice ) ; //Insertar de doble enlace inserta al final si pasas la posicion
nula
End;
//---------------------------------------------------------------------------
Procedure Cola_EncolarConFecha ( var Cola:Tipo_Indice; RegistroIndice:Tipo_Registro_Indice; TipoComprobante, fecha: String );
var
  RegistroControl: Tipo_Registro_Control;
begin
 LO_DobleEnlace.DobleEnlace_Insertar ( Cola, -1  , RegistroIndice ) ; //Insertar de doble enlace inserta al final si pasas la posicion
nula
  Seek(Cola.C, 0);
  Read(Cola.C, RegistroControl);
  RegistroControl.UltimoNroComprobante:= RegistroControl.UltimoNroComprobante + 1;
  if (TipoComprobante = '01') then
  begin
    RegistroControl.UltimaFechaEmitida01:= fecha;
  end else
    if (TipoComprobante = '02') then
    begin
      RegistroControl.UltimaFechaEmitida02:= fecha;
    end else
```

```
    if (TipoComprobante = '03') then
    begin
      RegistroControl.UltimaFechaEmitida03:= fecha;
    end else
      if (TipoComprobante = '50') then
      begin
        RegistroControl.UltimaFechaEmitida50:= fecha;
      end ;
  Seek(Cola.C, 0);
  Write(Cola.C, RegistroControl);
End;

//---------------------------------------------------------------------------
Procedure Cola_Frente ( var Cola:Tipo_Indice;  var  RegistroIndice: Tipo_Registro_Indice);
begin
 LO_DobleEnlace.DobleEnlace_Capturar ( Cola, LO_DobleEnlace.DobleEnlace_Primero(Cola) , RegistroIndice );
End;
//---------------------------------------------------------------
Function Cola_Vacia  ( var Cola:Tipo_Indice ): boolean;
begin
 Cola_Vacia :=  LO_DobleEnlace.DobleEnlace_Vacio ( Cola ) ;
end;
//---------------------------------------------------------------
Function Cola_PosNula  ( var Cola:Tipo_Indice): Tipo_Posicion ;
begin
 Cola_PosNula := LO_DobleEnlace.DobleEnlace_PosNula ( Cola );
end;
//---------------------------------------------------------------
Function Cola_ClaveNula  ( var Cola:Tipo_Indice): Tipo_Clave ;
begin
 Cola_ClaveNula := LO_DobleEnlace.DobleEnlace_ClaveNula( Cola);
End;
//---------------------------------------------------------------
```

```
procedure Cola_Tope (var Cola: Tipo_Indice; var RegistroIndice: Tipo_Registro_Indice);
begin
  LO_DobleEnlace.DobleEnlace_Capturar ( Cola,  LO_DobleEnlace.DobleEnlace_Ultimo (Cola), RegistroIndice);
end;
//---------------------------------------------------------------
procedure Cola_Destruir(var Cola: Tipo_Indice);
begin
  close(cola.C);
  close(Cola.I);
  Erase(Cola.C);
  Erase(Cola.I);
end ;
//-------------------------------------------------------------------------------
function UltimaFechaEmitida(var Cola: Tipo_Indice; tipoComprobante: string): string;
var
  RegistroControl: Tipo_Registro_Control;
begin
  Seek(Cola.C, 0);
  Read(Cola.C, REgistroControl);
  if tipoComprobante = '01' then UltimaFechaEmitida := RegistroControl.UltimaFechaEmitida01
  else
  if tipoComprobante = '02' then UltimaFechaEmitida := RegistroControl.UltimaFechaEmitida02
  else
  if tipoComprobante = '03' then UltimaFechaEmitida := RegistroControl.UltimaFechaEmitida03
  else
  if tipoComprobante = '50' then UltimaFechaEmitida := RegistroControl.UltimaFechaEmitida50
end;
//-------------------------------------------------------------------------------
function Cola_UltimoNroComprobante(var Cola:Tipo_Indice): integer;var
  RegistroControl: Tipo_Registro_control;
begin
    Seek(Cola.C, 0);
    Read(Cola.C, RegistroControl);
```

```
    Cola_UltimoNroComprobante:= RegistroControl.UltimoNroComprobante;
end;
end.
```

LO_DOBLEENLACE

```
unit LO_DobleEnlace;
interface
uses SysUtils;
Const
  _posicion_nula  = -1;
  _clave_nula           = '';
  _Clave_Inicial = 'A00';
  _Clave_media = 'M50';
  _Clave_Final = 'Z99' ;

Type
Tipo_Posicion  =  Longint;
Tipo_Clave     =  String [3]; //Esta clave valdrá tanto para claves entre A00 y Z99 como para claves entre 100 y 999
Tipo_Cadena    =  String [255];

Tipo_Registro_Indice =  Record
                                  Clave: Tipo_Clave;
                                  Posicion: Tipo_Posicion;
                                  Ant, Sig: Tipo_Posicion;
                               End;

Tipo_Archivo_Indice = File of Tipo_Registro_Indice;

Tipo_Registro_Control = Record
                          Ruta, Nombre: Tipo_Cadena ;
                          Primero, Ultimo, Primero_E, Ultimo_E: Tipo_Posicion;
                          UltimaFechaEmitida01, UltimaFechaEmitida02, UltimaFechaEmitida03, UltimaFechaEmitida50: Tipo_Cadena;
```

```
                UltimoNroComprobante: longint;
                        End;

Tipo_Archivo_Control = File of Tipo_Registro_Control;

Tipo_Indice = Record
                    C: Tipo_Archivo_Control ;
                    I: Tipo_Archivo_Indice ;
                End;
```

function DobleEnlace_Crear(var Indice: Tipo_Indice; sRuta, sNombre: Tipo_Cadena): boolean; //Crea los archivo de Indice, Control y Diccionario si no estan creados.

procedure DobleEnlace_Abrir(var Indice: Tipo_Indice); //Abre los archivos I,C,D

procedure DobleEnlace_Cerrar(var Indice: Tipo_Indice);
//Cierra los archivos I,C,D

function DobleEnlace_Buscar(var Indice: Tipo_Indice; clave: Tipo_Clave; var Posicion: Tipo_Posicion): boolean;
//Busca de manera serial algun elemento y devuelve true si lo encontró (y devuelve en la variable posicion el valor de la posicion donde lo encontró) Si no lo encuentra devuelve false y ademas la posicion donde deberia estar

function DobleEnlace_Buscar_Mejorada (Var Indice: Tipo_Indice; clave: Tipo_Clave; Var Posicion : Tipo_Posicion): Boolean ;
//Busca de manera mas eficiente haciendo busquedas ascendes o descendentes dependiendo si es mas grande o mas chica que la clave media M50

Procedure DobleEnlace_Insertar (var Indice: Tipo_Indice; pos:Tipo_Posicion; Reg:Tipo_Registro_Indice); //Inserta un elemento pasado por parametro a la lista doble en la posicion proporcionada

Procedure DobleEnlace_Modificar(var Indice: Tipo_Indice; pos: Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice);
//Inserta el elemento pasado por parametro suplantando al que esté en la posicion

procedure DobleEnlace_Capturar(var Indice: Tipo_Indice; pos: Tipo_Posicion; var RegistroIndice: Tipo_Registro_Indice);
//Toma un elemento de la lista (en la posicion pos) y lo guarda en el RegistroIndice

procedure DobleEnlace_Eliminar (var Indice: Tipo_Indice; pos: Tipo_Posicion);
//Elimina un elemento de la lista (el de la posicion pos)

function DobleEnlace_Primero (var Indice:Tipo_Indice): Tipo_Posicion;
    //Devuelve el primer elemento de la lista

function DobleEnlace_Primero_E(var indice:Tipo_Indice): TIpo_Posicion;
//Devuelve el primer elemento de la lista de elementos eliminados

function DobleEnlace_Ultimo (var Indice:Tipo_Indice): Tipo_Posicion;
//Devuelve el ultimo elemento de la lista

```
  function DobleEnlace_Proximo (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;
      //Devuelve la posicion del proximo elemento al pasado por parametro

  function DobleEnlace_Anterior (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;
//Devuelve la posicion del anterior elemento al pasado por parametro

  function DobleEnlace_Vacio (var Indice:Tipo_Indice): boolean;
      //Devuelve true si la estructura está vacia

  function DobleEnlace_PosNula(var Indice:Tipo_Indice): integer;
//Devuelve la posicion nula

  function DobleEnlace_ClaveNula(var Indice:Tipo_Indice): string;
//Devuelve la clave nula

  procedure DobleEnlace_Destruir (var Indice: Tipo_Indice);
//Elimina los archivos de Indice, Control y Diccionario


  function CompararClaves (Clave1, Clave2: Tipo_Clave): integer;
//Recibe por parametro la primer clave y la segunda. Devuelve un 1 si la clave 1 es mayor, un 2 si la 2 es mayor, y un 0 si son
iguales


implementation

  function DobleEnlace_Crear(var Indice: Tipo_Indice; sRuta, sNombre: Tipo_Cadena): boolean;

  var

    sArchivoIndice, sArchivoControl: Tipo_cadena;

    RegistroControl : Tipo_Registro_Control;

    bHayError: boolean;

  begin

    sArchivoIndice:= sRuta+'\'+sNombre+'.ntx';

    sArchivoControl:= sRuta+'\'+sNombre+'.con';


    Assign(Indice.I, sArchivoIndice);     //Asignamos archivo de Indice

    Assign(Indice.C, sArchivoControl);     //Asignamos archivo de Control

    {$I-}

    Reset(Indice.I);

    bHayError:= (IoResult <> 0);

    if bHayError then Rewrite(Indice.I);

    Close(Indice.I);

    //Ahora lo mismo para control e iniciarlo

    Reset(Indice.C);
```

```
  bHayError:= IoResult <> 0;
  if bHayError then
    begin
      Rewrite(Indice.C);
      RegistroControl.Ruta    := sRuta;
      RegistroControl.Nombre  := sNombre;
      RegistroControl.Primero:= _Posicion_Nula;
      RegistroControl.Ultimo:= _Posicion_Nula;
      RegistroControl.Primero_E:= _Posicion_Nula;
      RegistroControl.Ultimo_E:= _Posicion_Nula;
      RegistroControl.UltimaFechaEmitida01:='';
      RegistroControl.UltimaFechaEmitida02:='';
      RegistroControl.UltimaFechaEmitida03:='';
      RegistroControl.UltimaFechaEmitida50:='';
      RegistroControl.UltimoNroComprobante:=0;
      Seek(Indice.C, 0);
      write(Indice.C,RegistroControl);
    end;
  Close(Indice.C);
  DobleEnlace_Crear:= bHayError;
  {$I+}
end; //end DobleEnlace_Crear

//-----------------------------------------------------------------------------------------------------------------------------

  procedure DobleEnlace_Abrir(var Indice: Tipo_Indice);
  begin
    Reset(Indice.C);
    Reset(Indice.I);
  end; //DobleEnlace_Abrir

//-----------------------------------------------------------------------------------------------------------------------------

  procedure DobleEnlace_Cerrar(var Indice: Tipo_Indice);
```

```
 begin
   Close(Indice.C);
   Close(Indice.I);
 end; //DobleEnlace_Cerrar

//-------------------------------------------------------------------------------------------------------------------------------

 function DobleEnlace_Buscar(var Indice: Tipo_Indice; clave: Tipo_Clave; var Posicion: Tipo_Posicion): boolean;
 var
   bEncontrado, bCorte: boolean;
   RegistroControl: Tipo_Registro_Control;
   RegistroIndice: Tipo_Registro_Indice;
 begin
   Seek(Indice.C, 0);
   Read(Indice.C, RegistroControl);
   if (RegistroControl.Ultimo = _Posicion_Nula) then
   begin
     Posicion:= _posicion_nula;
     DobleEnlace_Buscar:= false;
   end
   else
     begin
       //Buscamos en forma serial
       bEncontrado:= false;
       bCorte:= false;
       posicion:= RegistroControl.Primero;

       while (bEncontrado = false) and (bCorte = false) and (posicion <> _Posicion_Nula) do
       begin
         Seek(Indice.I, posicion);
         Read(Indice.I, RegistroIndice);
         If (RegistroIndice.Clave = Clave) then bEncontrado:= True
         else
```

```
            if (CompararClaves(RegistroIndice.Clave, Clave) = 1) then bCorte:= true
            else
              posicion:= RegistroIndice.Sig;
          end;//While
          DobleEnlace_Buscar:= bEncontrado;
        end
      end; //DobleEnlace_Buscar

//-----------------------------------------------------------------------------------------------------------------------------------

 procedure DobleEnlace_Eliminar ( var Indice: Tipo_Indice; pos: Tipo_Posicion);
 var
   Reg, RegAnt, RegSig , RegX: Tipo_Registro_Indice ;
   RegistroControl: Tipo_Registro_Control;
   posAnt, posSig: Tipo_Posicion ;

 begin
    //LEER CABECERA (ya que dice donde está primero y ultimo) para reconocer en que posición estamos (pos)
    Seek ( Indice.C, 0 );
    Read ( Indice.C, RegistroControl) ;

    Seek ( Indice.I, pos);
    Read (Indice.I, Reg);

    if (RegistroControl.Primero = pos) and (RegistroControl.Ultimo = pos) then
    //ELIMINO Y QUEDA LISTA VACIA
    begin
      RegistroControl.Primero := _posicion_nula;
      RegistroControl.Ultimo  := _posicion_nula;
    end
    else
    if (RegistroControl.Primero = pos) then
      //ELIMINAR AL PRINCIPIO
```

```
Begin
  posSig := Reg.Sig ;
  Seek   ( Indice.I,  posSig);
  Read   ( Indice.I, RegSig );

  RegSig.Ant := _posicion_nula;
  RegistroControl.Primero :=  PosSig ;

  Seek (Indice.I, PosSig );
  Write ( Indice.I, RegSig );
end
else
if  (RegistroControl.Ultimo = pos) then
//ELIMINAR AL FINAL
begin
  posAnt := Reg.Ant ;
  Seek  ( Indice.I, posAnt );
  Read  ( Indice.I, RegAnt);

  RegAnt.Sig:= _posicion_nula;
  RegistroControl.ULtimo := PosAnt;
  Seek (Indice.I, PosAnt );
  Write ( Indice.I, RegAnt );
end
else
//ELIMINAR AL MEDIO
begin
  posSig:= Reg.Sig ;
  posAnt:= Reg.Ant ;
  Seek ( Indice.I, posSig);
  Read  ( Indice.I, RegSig );

  Seek  (Indice.I, posAnt );
```

```
    Read (Indice.I, RegAnt);

    RegAnt.Sig := posSig ;
    RegSig.Ant := posAnt ;

    Seek ( Indice.I, posAnt );
    Write (Indice.I, RegAnt);

    Seek  ( Indice.I, posSig );
    Write (Indice.I, RegSig );
  end;

 //COLOCAR ESE NODO EN LA SECUENCIA DE BORRADOS  ----------------------
 Reg.Sig:=  RegistroControl.Primero_E;
 Reg.Ant:= _posicion_nula ;

 if (RegistroControl.Primero_E <> _posicion_nula) Then
begin
            Seek ( Indice.I, RegistroControl.Primero_E );
            Read ( Indice.I, RegX );
            RegX.Ant:= pos ;
            Seek ( Indice.I, RegistroControl.Primero_E );
            Write( Indice.I, RegX );
end else
 begin
    RegistroControl.Ultimo_E:=Pos;
 end;
 RegistroControl.Primero_E:=Pos ;
//GRABO TODO LO QUE TENGO QUE GRABAR
 Seek (Indice.C, 0);
 Write (Indice.C, RegistroControl);
 Seek ( Indice.I, pos);
 Write (Indice.I, Reg );
```

```
 end;//end DobleEnlace_Eliminar
//-------------------------------------------------------------------------------------------------------------------------------------------------------

 Procedure DobleEnlace_Insertar ( var Indice: Tipo_Indice; pos:Tipo_Posicion; Reg:Tipo_Registro_Indice);
 var
  RegistroControl:Tipo_Registro_Control;
RegSig, RegAnt: Tipo_Registro_Indice;
posnueva, posUltimo, posAnt:Tipo_Posicion;
 begin

  Seek ( Indice.C, 0) ;
  Read ( Indice.C, RegistroControl);

  posNueva := FileSize ( Indice.I) ;

  if RegistroControl.Primero = _posicion_nula then // 1 - Inserto en estructura vacia

begin
            Reg.Sig := _posicion_nula;
            Reg.Ant := _posicion_nula;
            RegistroControl.Primero:= posNueva ;
            RegistroControl.Ultimo := posNueva ;
end

   else
if (RegistroControl.Primero = pos) then // 2 - Inserto al principio de la estructura
                        begin
                                    seek (Indice.I, pos);
                                    Read (Indice.I, RegSig );
                                    Reg.Sig   := pos ;
                                    Reg.Ant     := _posicion_nula;
                                    RegSig.Ant:= posNueva;
```

```
                RegistroControl.Primero:= posNueva;

                Seek (Indice.I, pos);
                Write (Indice.I, RegSig);
        end
else
   if (pos = _posicion_nula) then // 3 - Inserto al final de la estructura
                begin
                        posUltimo:= RegistroControl.Ultimo;
                        Seek (Indice.I , posUltimo ) ;
                        Read (Indice.I , RegAnt );

                        RegAnt.Sig := posNueva ;
                        Reg.Ant    := posUltimo;
                        Reg.Sig    := _posicion_nula;
                        RegistroControl.Ultimo  := posNueva ;

                        Seek (Indice.I, posUltimo);
                        Write (Indice.I, RegAnt);

                end

        else
                // 4 - Inserta al medio de la estructura
                begin
                        Seek ( Indice.I, pos);
                        Read (Indice.I, RegSig);
                        posAnt := RegSig.Ant ;
                        Seek  (Indice.I, posAnt);
                        Read  (Indice.I, RegAnt );

                        RegSig.Ant:= posNueva;
                        RegAnt.Sig:= posNueva;
```

```
                                    Reg.Ant     := posAnt;
                                    Reg.Sig   := pos ;

                                    Seek ( Indice.I, pos);
                                    Write (Indice.I, RegSig);
                                    Seek  (Indice.I, posAnt);
                                    Write (Indice.I, RegAnt) ;

                                end;

  //GRABAR TODOS
 Seek ( Indice.I, posNueva );
 Write (Indice.I, Reg );
 Seek  (Indice.C, 0);
 Write (Indice.C, RegistroControl);

end; //End DobleEnlace_Insertar

//-------------------------------------------------------------------------------------------------------------------------

Procedure DobleEnlace_Modificar(var Indice: Tipo_Indice; pos: Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice);
var
 Aux: Tipo_Registro_Indice;
begin
 Seek(Indice.I, pos);
 Read(Indice.I, Aux);
 RegistroIndice.Sig:= Aux.Sig;
 RegistroIndice.Ant:= Aux.Ant;
 Seek(Indice.I, pos);
 Write(Indice.I, RegistroIndice);
end;//End DobleEnlace_Modificar

//-------------------------------------------------------------------------------------------------------------------------
```

```
function DobleEnlace_Primero (var Indice:Tipo_Indice): Tipo_Posicion;
var
  RegistroControl: Tipo_registro_Control;
begin
  Seek(Indice.C,0);
  Read(Indice.C, RegistroControl);

  DobleEnlace_Primero:= RegistroControl.Primero;
end; //end DobleEnlace_Primero

//-----------------------------------------------------------------------------------------------------------------------------

function DobleEnlace_Primero_E (var Indice:Tipo_Indice): Tipo_Posicion;
var
  RegistroControl: Tipo_registro_Control;
begin
  Seek(Indice.C,0);
  Read(Indice.C, RegistroControl);

  DobleEnlace_Primero_E:= RegistroControl.Primero_E;
end; //end DobleEnlace_Primero_E

//-----------------------------------------------------------------------------------------------------------------------------

function DobleEnlace_Ultimo (var Indice:Tipo_Indice): Tipo_Posicion;
var
  RegistroControl: Tipo_registro_Control;
begin
  Seek(Indice.C,0);
  Read(Indice.C, RegistroControl);

  DobleEnlace_Ultimo:= RegistroControl.Ultimo;
```

```
end; //end DobleEnlace_Ultimo

//-----------------------------------------------------------------------------------------------------------------------------

function DobleEnlace_Ultimo_E (var Indice:Tipo_Indice): Tipo_Posicion;
var
  RegistroControl: Tipo_registro_Control;
begin
  Seek(Indice.C,0);
  Read(Indice.C, RegistroControl);

  DobleEnlace_Ultimo_E:= RegistroControl.Ultimo_E;
end; //end DobleEnlace_UltimoE

//-----------------------------------------------------------------------------------------------------------------------------

function DobleEnlace_Proximo (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;
var
  RegistroIndice: Tipo_registro_Indice;
begin
  Seek(Indice.I,pos);
  Read(Indice.I, RegistroIndice);

  DobleEnlace_Proximo:= RegistroIndice.Sig;
end; //end DobleEnlace_Proximo

//-----------------------------------------------------------------------------------------------------------------------------

function DobleEnlace_Anterior (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;
var
  RegistroIndice: Tipo_registro_Indice;
begin
  Seek(Indice.I,pos);
```

```
  Read(Indice.I, RegistroIndice);
  DobleEnlace_Anterior:= RegistroIndice.Ant;
end; //end DobleEnlace_Anterior
//-----------------------------------------------------------------------------------------------------------------------------------
procedure DobleEnlace_Capturar (var Indice: Tipo_Indice; pos: Tipo_posicion; var RegistroIndice: Tipo_Registro_Indice);
begin
  Seek(Indice.I, pos);
  Read(Indice.I, RegistroIndice);
end;//End DobleEnlace_Capturar
//-----------------------------------------------------------------------------------------------------------------------------------
function DobleEnlace_Vacio (var Indice:Tipo_Indice): boolean;
var
RegistroControl: Tipo_Registro_Control;
vacio: boolean;
begin
  Seek(Indice.C, 0);
  Read(Indice.C, RegistroControl);
  if RegistroControl.Primero = _posicion_nula then
    vacio:= true
    else
    vacio:= false;
DobleEnlace_Vacio:= vacio;
end;//End DobleEnlace_Vacio

//-----------------------------------------------------------------------------------------------------------------------------------

function DobleEnlace_PosNula(var Indice:Tipo_Indice): integer;
begin
  DobleEnlace_PosNula := _Posicion_Nula;
end;

function DobleEnlace_ClaveNula(var Indice:Tipo_Indice): string;
begin
```

```
  DobleEnlace_ClaveNula := _Clave_Nula;
end;

//-----------------------------------------------------------------------------------------------------------------------------

procedure DobleEnlace_Destruir (var Indice: Tipo_Indice);
begin
 Close(Indice.C);
 Close(Indice.I);

 Erase(Indice.C);
 Erase(Indice.I);
end; //End DobleEnlace_Destruir

//-----------------------------------------------------------------------------------------------------------------------------

Function DobleEnlace_Buscar_Mejorada ( Var Indice: Tipo_Indice; clave: Tipo_Clave; Var Posicion : Tipo_Posicion): Boolean ;
  //Busca <clave> en el INDICE. Usa búsqueda binaria. Si no la encuentra, devuelve la posición donde debería estar.
  var
    bEncontrado, bCorte: boolean ;
  cTipoBusqueda: char;
    RegistroControl: Tipo_Registro_Control;
    RegistroIndice: Tipo_Registro_Indice;
  begin
    Seek(Indice.C, 0);
    Read(Indice.C, RegistroControl);
    if (Clave <= _clave_media) then cTipoBusqueda := 'A' else cTipoBusqueda := 'D';
    if (RegistroControl.Ultimo =  _posicion_nula) then
    begin
     Posicion := _posicion_nula ;
     DobleEnlace_Buscar_Mejorada:= false ;
    end
    else
```

```
begin
  //Busco en forma serial
  bEncontrado:= false;
  bCorte:= false ;
 if cTipoBusqueda = 'A' then
  begin
    //Busco desde el principio
    posicion:= RegistroControl.Primero;
    while (bEncontrado = false) and (bCorte = false) and (posicion <> _posicion_nula) do
    begin
      Seek (Indice.I, posicion );
      Read (Indice.I, RegistroIndice);
      if RegistroIndice.Clave = Clave then bEncontrado := true
      else
        if (RegistroIndice.clave > clave) then bCorte:= True else posicion := RegistroIndice.Sig ;
    end;//while
  end//if tipo a
  else
  begin
  //Busco desde el final
    posicion:= RegistroControl.Ultimo ;

    while (bEncontrado = false) and ( bCorte = false ) and (posicion <> _posicion_nula ) do
    begin
              seek (Indice.I, posicion );
      Read ( Indice.I, RegistroIndice ) ;
              if RegistroIndice.clave = Clave then bEncontrado := true
              else
                if (RegistroIndice.clave < clave) then bCorte:= True else posicion := RegistroIndice.Ant ;
    end;//while

    if (bEncontrado = false) then posicion:= RegistroIndice.Sig;          //dado que siempre devuelvo la posicion del inmediato
siguiente
  end//else (tipo d)
```

```
    end;//ultimo es distinto a posicionnula
   DobleEnlace_Buscar_Mejorada:=bEncontrado;
 end;//DobleEnlace_Buscar_Mejorada


//-----------------------------------------------------------------------------------------------------------------------------------------

function CompararClaves (Clave1, Clave2: Tipo_Clave): integer; //Recibe por parametro la primer clave y la segunda. Devuelve un
1 si la clave 1 es mayor, un 2 si la 2 es mayor, y un 0 si son iguales
var
 nResultado: integer;
letra1, letra2: char;
  nClave1, nClave2: longint;


numeroMedio1, numeroMedio2: integer;


numero1,numero2: integer;
begin
letra1:=Clave1[1];
letra2:=Clave2[1];


if (Clave1 = Clave2) then nResultado:=0
else
begin
  if ((letra1 >= '0') and (letra2<='9')) then    //ESTOY COMPARANDO NUMEROS
      begin
        nClave1:=StrToInt(Clave1);
        nClave2:=StrToInt(Clave2);
        if (nClave1> nClave2) then nResultado:= 1 else nResultado:= 2;
      end
      else
      if (letra1>letra2) then nResultado:= 1
    else
    if (letra2>letra1) then nResultado:= 2
      else
```

```
    if (letra1=letra2) then
    begin
        //hay que ver que numero es mas grande pero antes hay que ver que el numero central no sea 0
        numeroMedio1:= StrToInt( Clave1[2] );
        numeroMedio2:= strToInt( Clave2[2] );

        if (numeroMedio1 = 0) then numero1:= StrToInt(Clave1[3]) else numero1:=StrToInt(Clave1[2]+Clave1[3]);
        if (numeroMedio2 = 0) then numero2:= StrToInt(Clave2[3]) else numero2:=StrToInt(Clave2[2]+Clave2[3]);
            if (numero1 > numero2) then nResultado:= 1 else nResultado:= 2

   end;
end;

ComparарClaves:= nResultado;
end; //End_ComparарClaves
//-------------------------------------------------------------------------------------------------------------------------------
end.
```

LO_INDICE

```
unit LO_Indice;
interface
 uses
 SysUtils;
 const
 _Posicion_Nula = -1;
 _Clave_Nula = -1;
 type
 Tipo_Clave = Longint;
 Tipo_Cadena = String [255];
 Tipo_Posicion = LongInt;
```

```
Tipo_Registro_Indice = record
                         Clave: Tipo_Clave;
                         Puntero: Tipo_Posicion;
                       end;

Tipo_Archivo_Indice = file of Tipo_Registro_Indice;

Tipo_Registro_Control = record
                         Ruta: Tipo_Cadena;
                         Nombre: Tipo_Cadena;
                         Ultimo: Tipo_Posicion;
                        end;

Tipo_Archivo_Control = file of Tipo_Registro_Control;

Tipo_Indice = record
                        C: Tipo_Archivo_Control;
                        I: Tipo_Archivo_Indice;
                       end;

function Indice_Crear(var Indice: Tipo_Indice; sRuta, sNombre: String): boolean;     //Crea los archivos de Datos y Control y los asigna a la variable Clientes
procedure Indice_Abrir(var Indice: Tipo_Indice);
procedure Indice_Cerrar(var Indice: Tipo_Indice);
function Indice_Buscar(var Indice: Tipo_Indice; Clave: Tipo_Clave; var Pos: Tipo_Posicion): Boolean;
procedure Indice_Insertar(var Indice: Tipo_Indice; pos: Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice);
procedure Indice_Modificar(var Indice: Tipo_Indice; pos: Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice);
procedure Indice_Capturar(var Indice: Tipo_Indice; pos: Tipo_Posicion; var RegistroIndice: Tipo_Registro_Indice);
procedure Indice_Eliminar(var Indice: Tipo_Indice; pos: Tipo_Posicion);
function Indice_Primero (var Indice:Tipo_Indice): Tipo_Posicion;
function Indice_Ultimo (var Indice:Tipo_Indice): Tipo_Posicion;
function Indice_Proximo (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;
function Indice_Anterior (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;
```

```
  function Indice_Vacio (var Indice:Tipo_Indice): boolean;
  procedure Indice_Destruir (var Indice: Tipo_Indice);
  function Indice_ClaveNula (var Indice: Tipo_Indice): longint;
  function Indice_PosNula (var Indice: Tipo_Indice): longint;

implementation

  function Indice_Crear(var Indice: Tipo_Indice; sRuta, sNombre: String): boolean;
  var
    sArchivoIndice, sArchivoControl: Tipo_cadena;
    RegistroControl : Tipo_Registro_Control;
    bHayError: boolean;
  begin
    sArchivoIndice:= sRuta+'\'+sNombre+'.ntx';
    sArchivoControl:= sRuta+'\'+sNombre+'.con';

    Assign(Indice.I, sArchivoIndice);     //Asignamos archivo de Indice
    Assign(Indice.C, sArchivoControl);     //Asignamos archivo de Control

    {$I-}

    Reset(Indice.I);
    bHayError:= IoResult <> 0;
    if bHayError then Rewrite(Indice.I);
    Close(Indice.I);

    //Ahora lo mismo para control e iniciarlo

    Reset(Indice.C);
    bHayError:= IoResult <> 0;
    if bHayError then
      begin
```

```
      Rewrite(Indice.C);
      RegistroControl.Ruta    := sRuta;
      RegistroControl.Nombre  := sNombre;
      RegistroControl.Ultimo:= _Posicion_Nula;
      Seek(Indice.C, 0);
      write(Indice.C,RegistroControl);

    end;
  Close(Indice.C);
  Indice_Crear:= bHayError;
  {$I+}

end; //end Indice_Crear

procedure Indice_Abrir(var Indice: Tipo_Indice);
begin
  reset(Indice.I);
  reset(Indice.C);
end;//end Indice_Abrir

procedure Indice_Cerrar(var Indice: Tipo_Indice);
begin
  close(Indice.I);
  close(Indice.C);
end; //end Indice_Cerrar

function Indice_Buscar(var Indice: Tipo_Indice; Clave: Tipo_Clave; var Pos: Tipo_Posicion): Boolean;
var
  nInicio, nFin, nMedio: Tipo_Posicion;
  bEncontrado:boolean;
  RegistroIndice: Tipo_Registro_Indice;
  RegistroControl: Tipo_Registro_Control;
begin
```

```
bEncontrado:=false;
Seek(Indice.C, 0);
Read(Indice.C, RegistroControl);

if RegistroControl.Ultimo = _Posicion_Nula then   //El indice está vacío
begin
  Pos:= 0;
  Indice_Buscar:=false;
end
else
begin
  nInicio:=0;
  nFin:=RegistroControl.Ultimo;

  while (bEncontrado=false) and (nInicio <= nFin) do
  begin
    nMedio := (nInicio + nFin) div 2;
    Seek(Indice.I, nMedio);
    Read(Indice.I, RegistroIndice);

    if Clave = RegistroIndice.Clave then
    begin
      bEncontrado:= true;
      pos:= nMedio;
    end
    else
    if Clave < RegistroIndice.Clave then
      nFin:= nMedio-1
    else
      nInicio:= nMedio+1;
  end;//While

  if bEncontrado = true then Indice_Buscar:= true
```

```
    else
    begin
      Indice_Buscar:= false;
      Seek(Indice.I, nMedio);
      Read(Indice.I, RegistroIndice);
      if RegistroIndice.Clave < Clave then nMedio:= nMedio + 1 ;
      pos:= nMedio;
    end;
  end;
end;//End ME_Buscar


procedure Indice_Insertar(var Indice: Tipo_Indice; pos: Tipo_Posicion; RegistroIndice: Tipo_Registro_Indice);
var
  RegistroControl: Tipo_Registro_Control;
  RegistroIndiceNuevo, RegistroIndiceAux: Tipo_Registro_Indice;
  i: Tipo_Posicion;
begin
  Seek(Indice.C, 0);
  Read(Indice.C, RegistroControl);
  if (RegistroControl.Ultimo < FileSize(Indice.I) - 1 ) then
  begin
    {Esto quiere decir que hay elementos dados de baja en el indice, entonces movemos el ultimo+1 al final fisico para poder
insertar el nuevo elemento antes}
    Seek(Indice.I, RegistroControl.Ultimo+1);
    Read(Indice.I, RegistroIndiceAux);
    Seek(Indice.I, FileSize(Indice.I));
    Write(Indice.I, RegistroIndiceAux);
  end;//if
  {Ahora abrimos la estructura}
  if (RegistroControl.Ultimo = _Posicion_Nula) then
  begin
    Seek(Indice.I, 0);
    Write(Indice.I, RegistroIndice);
  end
```

```
  else
  begin
    For i:= RegistroControl.Ultimo downto pos do
    begin
     Seek(Indice.I, i);
     Read(Indice.I, RegistroIndiceNuevo);
     Seek(Indice.I, Succ(i));
     Write(Indice.I, RegistroIndiceNuevo);
    end;//for
    Seek(Indice.I, pos);
    Write(Indice.I, RegistroIndice);
  end ;
 RegistroControl.Ultimo:= Succ(RegistroControl.Ultimo);
 Seek(Indice.C, 0);
 Write(Indice.C, RegistroControl);
end; //End Me_Insertar

procedure Indice_Modificar(var Indice: Tipo_Indice; pos: Tipo_Posicion{posicion que devuelve el Indice_Buscar}; RegistroIndice:
Tipo_Registro_Indice);
 begin
   Seek(Indice.I, pos);
   Write(Indice.I, RegistroIndice);
end; //End ME_modificar

procedure Indice_Capturar(var Indice: Tipo_Indice; pos: Tipo_Posicion; var RegistroIndice: Tipo_Registro_Indice);
begin
    Seek(Indice.I, pos);
    Read(Indice.I, RegistroIndice);
end; //End Indice_Capturar

procedure Indice_Eliminar(var Indice: Tipo_Indice; pos: Tipo_Posicion);
var
  RegistroIndice, RegistroIndiceAux: Tipo_Registro_Indice;
  RegistroControl: Tipo_Registro_Control;
```

```
  i: Tipo_Posicion;
begin
  Seek(Indice.C, 0);
  Read(Indice.C, RegistroControl);
  Seek(Indice.I, pos);
  Read(Indice.I, RegistroIndiceAux); //Guardamos en un auxiliar el eleIndicento a borrar para luego guardarlo por delante del ultimo elemento logico (por si en un futuro se quiere recuperar)
  for i:= pos+1 to  RegistroControl.Ultimo do
  begin
    Seek(Indice.I, i);
    Read(Indice.I, RegistroIndice);
    Seek(Indice.I, Pred(i));
    Write(Indice.I, RegistroIndice);
  end;//for
  Seek(Indice.I, RegistroControl.Ultimo);
  Write(Indice.I, RegistroIndiceAux);
  RegistroControl.Ultimo:= Pred(RegistroControl.Ultimo);
  Seek(Indice.C,0);
  Write(Indice.C, RegistroControl);
end; //End Indice_Eliminar

function Indice_Primero (var Indice:Tipo_Indice): Tipo_Posicion;
var
RegistroControl: Tipo_Registro_Control;
begin
  Seek(Indice.C, 0);
  Read(Indice.C, RegistroControl);
  if RegistroControl.Ultimo = _Posicion_Nula then
    Indice_Primero:=_Posicion_Nula
    else
      Indice_Primero:=0;
end; //End Indice_Primero

function Indice_Ultimo(var Indice:Tipo_Indice): Tipo_Posicion;
```

```
var
RegistroControl: Tipo_Registro_Control;
begin
  Seek(Indice.C, 0);
  Read(Indice.C, RegistroControl);
  Indice_Ultimo:= RegistroControl.Ultimo;
end; //Indice_Ultimo

function Indice_Proximo (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;
var
RegistroControl: Tipo_Registro_Control;
begin
  Seek(Indice.C, 0);
  Read(Indice.C, RegistroControl);
  if pos=RegistroControl.Ultimo then
  Indice_Proximo:=_Posicion_Nula
  else
    Indice_Proximo:= Succ(pos);
end;//End Indice_Proximo

function Indice_Anterior (var Indice:Tipo_Indice; pos: Tipo_Posicion): Tipo_Posicion;
var
RegistroControl: Tipo_Registro_Control;
begin
  Seek(Indice.C, 0);
  Read(Indice.C, RegistroControl);
  if pos = 0 then
    Indice_Anterior := _Posicion_Nula
    else
      Indice_Anterior := Pred(pos);
end;//End Indice_Anterior

function Indice_Vacio (var Indice:Tipo_Indice): boolean;
```

```
 var
   RegistroControl: Tipo_Registro_Control;
 begin
   Seek(Indice.C,0);
   Read(Indice.C, RegistroControl);

   if (RegistroControl.Ultimo = _Posicion_Nula) then
      Indice_Vacio := true
      else
      Indice_Vacio := false
 end;//END Indice_VACIO

 procedure Indice_Destruir (var Indice: Tipo_Indice);
 var
   cArchivoIndice, cArchivoControl: Tipo_Cadena;
   RegistroControl: Tipo_Registro_Control;
 begin
   Close(Indice.C);
   Close(Indice.I);
   Erase(Indice.C);
   Erase(Indice.I);
 end; //Indice_Destruir

//---------------------------------------------------------------------------

 function Indice_ClaveNula (var Indice: Tipo_Indice): longint;
 begin
   Indice_ClaveNula:= _Clave_Nula;
 end;

//---------------------------------------------------------------------------

 function Indice_PosNula (var Indice: Tipo_Indice): longint;
```

```
 begin
   Indice_PosNula:= _posicion_Nula;
 end;
end.
```

LO_PILA

```
Unit LO_Pila;
interface
uses SysUtils, LO_DobleEnlace;
function Pila_Crear ( var Pila: Tipo_Indice; sRuta, sNombre: Tipo_Cadena): boolean;
procedure Pila_Abrir ( var Pila: Tipo_Indice ) ;
procedure Pila_Cerrar ( Var Pila: Tipo_Indice );
procedure Pila_Apilar  (var Pila: Tipo_Indice ; RegistroIndice: Tipo_Registro_Indice ) ;
procedure Pila_Desapilar  ( Var Pila: Tipo_Indice );
procedure Pila_Tope ( var Pila: Tipo_Indice ; var RegistroIndice: Tipo_Registro_Indice );
function Pila_Vacia ( var Pila: Tipo_Indice ): boolean ;
function Pila_ClaveNula  ( var Pila:Tipo_Indice  ): string;
function Pila_PosNula  ( var Pila:Tipo_Indice  ): longint;
procedure Pila_Destruir  ( var Pila:Tipo_Indice  );

implementation

function Pila_Crear ( var Pila: Tipo_Indice ; sRuta, sNombre: Tipo_Cadena ): boolean;
begin
  LO_DobleEnlace.DobleEnlace_Crear(Pila, sRuta, sNombre);
End; //End Pila_Crear
//-------------------------------------------------------
Procedure Pila_Abrir ( var Pila: Tipo_Indice) ;
Begin
  LO_DobleEnlace.DobleEnlace_Abrir(Pila);
End;
//-------------------------------------------------------
```

```
Procedure Pila_Cerrar ( Var Pila: Tipo_Indice);
begin
  LO_DobleEnlace.DobleEnlace_Cerrar(Pila);
End;
//-------------------------------------------------------
Procedure Pila_Apilar  (var Pila: Tipo_Indice; RegistroIndice: Tipo_Registro_Indice ) ;
begin

  LO_DobleEnlace.DobleEnlace_Insertar(Pila, LO_DobleEnlace.DobleEnlace_Primero(Pila), RegistroIndice);

end;
//-------------------------------------------------------
Procedure Pila_Desapilar  ( Var Pila: Tipo_Indice);
begin
  LO_DobleEnlace.DobleEnlace_Eliminar(Pila, LO_DobleEnlace.DobleEnlace_Primero(Pila) );
End;
//-------------------------------------------------------
Procedure Pila_Tope ( var Pila: Tipo_Indice; var RegistroIndice: Tipo_Registro_Indice);
begin
 LO_DobleEnlace.DobleEnlace_Capturar(Pila, LO_DobleEnlace.DobleEnlace_Primero(Pila), RegistroIndice);
end;
//-------------------------------------------------------
Function Pila_Vacia ( var Pila: Tipo_Indice): boolean ;
begin
 Pila_Vacia:= LO_DobleEnlace.DobleEnlace_Vacio(Pila)
end;
//-------------------------------------------------------
function Pila_ClaveNula  ( var Pila:Tipo_Indice ): String;
begin
 Pila_ClaveNula:= LO_DobleEnlace.DobleEnlace_ClaveNula(Pila);
End;
//-------------------------------------------------------
function Pila_PosNula  ( var Pila: Tipo_Indice ): longint;
```

```
begin
 Pila_PosNula:= LO_DobleEnlace.DobleEnlace_PosNula(Pila);
End;
//---------------------------------------------------------


procedure Pila_Destruir (Var Pila: Tipo_Indice);
begin
 LO_DobleEnlace.DobleEnlace_Destruir(Pila);
end;
end.
```

LO_DATOS_ARTICULOS

```
unit LO_Datos_Articulos;
interface
  uses
    SysUtils;
  type
    Tipo_Clave = Longint; // 1000 .. 9999
    Tipo_Clave_Rubro = string [3];
    Tipo_Cadena = String [40];
    Tipo_Precio = real;
    Tipo_Posicion = LongInt;

    Tipo_Registro_Datos = record
                        Clave: Tipo_Clave;
                                    ClaveRubro: Tipo_Clave_Rubro;
                        Descripcion: Tipo_Cadena;
                                    PrecioCosto: Tipo_Precio;
                                    PrecioUnitario: Tipo_Precio;
                              stock: longint;
                        Borrado: boolean;
                       end;
```

```
  Tipo_Archivo_Datos = file of Tipo_Registro_Datos;

  Tipo_Datos = record
                 D: Tipo_Archivo_Datos;
               end;
 var
  Datos: Tipo_Datos;
 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
 procedure Datos_Abrir(var ME: Tipo_Datos);
 procedure Datos_Cerrar(var ME: Tipo_Datos);
 procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
 procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
 procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
 procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
 procedure Datos_Destruir(var ME: Tipo_Datos);
implementation

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
  var
   sArchivoDatos: String;
   bHayError: boolean;
 begin
   sArchivoDatos:= sRuta+'\'+sNombre+'.dat';
   Assign(ME.D, sArchivoDatos);     //Asignamos archivo de Datos

   {$I-}
   Reset(Me.D);
   bHayError:= IoResult <> 0;
   if bHayError then Rewrite(Me.D);
   Close(Me.D);
   Datos_Crear:= bHayError;
   {$I+}
```

```
end;//DatosCrear

procedure Datos_Abrir(var ME: Tipo_Datos);
begin
  reset(Me.D);
end;//DatosAbrir

procedure Datos_Cerrar(var ME: Tipo_Datos);
begin
  close(Me.D);
end;//DatosCerrar

procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, FileSize(Me.D));
  write(Me.D, reg);
end;//DatosInsertar

procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, pos);
  write(Me.D, reg);
end;//DatosModificar

procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
begin
  Seek(Me.D, pos);
  Read(Me.D, Reg);
end;//DatosCapturar

procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
```

```
  var
  RegistroDatos: Tipo_Registro_Datos;
  begin
    Seek(Me.D, pos);
    Read(Me.D, RegistroDatos);
    RegistroDatos.borrado:=true;

    Seek(ME.D, pos);
    Write(Me.D, RegistroDatos);
  end;//DatosCapturar

  procedure Datos_Destruir(var ME: Tipo_Datos);
  begin
    Close(Me.D);
    Erase(Me.D);
  end; //DatosDestruir
end.
```

LO_DATOS_ASIENTOS

```
unit LO_Datos_Asientos;
interface
  uses
    SysUtils;

  type
    Tipo_Clave = string [255];
    Tipo_Cadena = String [255];
    Tipo_Posicion = LongInt;
    Tipo_Precio = real;
```

```
  Tipo_Registro_Datos = record
                         CodCliente: Tipo_Clave;
                         nroComprobante: longint;
                         fecha: Tipo_Cadena;
                                        importe: Tipo_Precio;
                         Borrado: boolean;
                        end;

  Tipo_Archivo_Datos = file of Tipo_Registro_Datos;

  Tipo_Datos = record
                D: Tipo_Archivo_Datos;
               end;

 var
  Datos: Tipo_Datos;

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
 procedure Datos_Abrir(var ME: Tipo_Datos);
 procedure Datos_Cerrar(var ME: Tipo_Datos);
 procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
 procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
 procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
 procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
 procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);

implementation

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
  var
   sArchivoDatos: Tipo_cadena;
   bHayError: boolean;
```

```
begin
  sArchivoDatos:= sRuta+'\'+sNombre+'.dat';
  Assign(ME.D, sArchivoDatos);     //Asignamos archivo de Datos

  {$I-}
  Reset(Me.D);
  bHayError:= IoResult <> 0;
  if bHayError then Rewrite(Me.D);
  Close(Me.D);
  Datos_Crear:= bHayError;
  {$I+}
end;//DatosCrear

procedure Datos_Abrir(var ME: Tipo_Datos);
begin
  reset(Me.D);
end;//DatosAbrir

procedure Datos_Cerrar(var ME: Tipo_Datos);
begin
  close(Me.D);
end;//DatosCerrar

procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, FileSize(Me.D));
  write(Me.D, reg);
end;//DatosInsertar

procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
```

```
  Seek(Me.D, pos);
  write(Me.D, reg);
 end;//DatosModificar

 procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
 begin
  Seek(Me.D, pos);
  Read(Me.D, Reg);
 end;//DatosCapturar

 procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
 var
 RegistroDatos: Tipo_Registro_Datos;
 begin
  Seek(Me.D, pos);
  Read(Me.D, RegistroDatos);
  RegistroDatos.borrado:=true;
  Seek(Me.D, pos);
  Write(Me.D, RegistroDatos);
 end;//DatosCapturar

 procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);
 var
  cArchivoDatos: String;
 begin
  cArchivoDatos:= Ruta+'\'+Nombre+'.dat';
  Close(Me.D);
  Erase(Me.D);
 end; //DatosDestruir
end.
```

LO_DATOS_CLIENTES

```
unit LO_Datos_Clientes;
interface
 uses
   SysUtils;
 type
   Tipo_Clave = String [3];  //Entre A00 y Z99
   Tipo_Nombre = String [30]; //Nombre es String de hasta 30 caracteres
   Tipo_Dni = String [13];    //Dni formato 99-99999999-9
   Tipo_Cadena = String [255];
   Tipo_Posicion = LongInt;

   Tipo_Registro_Datos = record
                          Clave: Tipo_Clave;
                          Nombre: Tipo_Nombre;
                          Dni: Tipo_Dni;
                          Borrado: boolean;
                         end;

   Tipo_Archivo_Datos = file of Tipo_Registro_Datos;

   Tipo_Datos = record
                  D: Tipo_Archivo_Datos;
                end;

 var
   Datos: Tipo_Datos;

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
 procedure Datos_Abrir(var ME: Tipo_Datos);
 procedure Datos_Cerrar(var ME: Tipo_Datos);
 procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
 procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
 procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
```

```
 procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
 procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);

implementation

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
  var
   sArchivoDatos: Tipo_cadena;
   bHayError: boolean;
 begin
   sArchivoDatos:= sRuta+'\'+sNombre+'.dat';
   Assign(ME.D, sArchivoDatos);     //Asignamos archivo de Datos

   {$I-}
   Reset(Me.D);
   bHayError:= IoResult <> 0;
   if bHayError then Rewrite(Me.D);
   Close(Me.D);
   Datos_Crear:= bHayError;
   {$I+}

 end;//DatosCrear

 procedure Datos_Abrir(var ME: Tipo_Datos);
 begin
   reset(Me.D);
 end;//DatosAbrir

 procedure Datos_Cerrar(var ME: Tipo_Datos);
 begin
   close(Me.D);
 end;//DatosCerrar
```

```
procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, FileSize(Me.D));
  write(Me.D, reg);
end;//DatosInsertar

procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, pos);
  write(Me.D, reg);
end;//DatosModificar

procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
begin
  Seek(Me.D, pos);
  Read(Me.D, Reg);
end;//DatosCapturar

procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
var
RegistroDatos: Tipo_Registro_Datos;
begin
  Seek(Me.D, pos);
  Read(Me.D, RegistroDatos);
  RegistroDatos.borrado:=true;

  Seek(Me.D, pos);
  Write(Me.D, RegistroDatos);
end;//DatosCapturar

procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);
```

```
  var
    cArchivoDatos: String;
  begin
    cArchivoDatos:= Ruta+'\'+Nombre+'.dat';
    Close(Me.D);
    Erase(Me.D);
  end; //DatosDestruir
end.
```

LO_DATOS_COMPROBANTES

```
unit LO_Datos_Comprobantes;
interface
  uses
    SysUtils;
  type
    Tipo_Clave =  String [255];
    Tipo_Posicion = LongInt;
    Tipo_Cadena = String [255];
    Tipo_Precio = real;
    Tipo_Porcentaje = 0..100;
    Tipo_NroComprobante = Longint;
    Tipo_Registro_Datos = record
                          codFactura: Tipo_Clave;
                          codCliente: Tipo_Cadena;
                          fecha: Tipo_Cadena;
                          tipoComprobante: Tipo_Cadena;
                          nroComprobante: Tipo_NroComprobante;
                          neto : Tipo_Precio;
                          descuento : Tipo_Porcentaje;
                          gravado: Tipo_Precio;
                          IVA: Tipo_Precio;
                          total: Tipo_Precio;
```

```
                borrado: boolean;
              end;

  Tipo_Archivo_Datos = file of Tipo_Registro_Datos;

  Tipo_Datos = record
                D: Tipo_Archivo_Datos;
               end;

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
 procedure Datos_Abrir(var ME: Tipo_Datos);
 procedure Datos_Cerrar(var ME: Tipo_Datos);
 procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
 procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
 procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
 procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
 procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);

implementation

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
  var
   sArchivoDatos: Tipo_cadena;
   bHayError: boolean;
 begin
   sArchivoDatos:= sRuta+'\'+sNombre+'.dat';
   Assign(ME.D, sArchivoDatos);     //Asignamos archivo de Datos

   {$I-}
   Reset(Me.D);
   bHayError:= IoResult <> 0;
   if bHayError then Rewrite(Me.D);
   Close(Me.D);
```

```
  Datos_Crear:= bHayError;
  {$I+}

end;//DatosCrear

procedure Datos_Abrir(var ME: Tipo_Datos);
begin
  reset(Me.D);
end;//DatosAbrir

procedure Datos_Cerrar(var ME: Tipo_Datos);
begin
  close(Me.D);
end;//DatosCerrar

procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, FileSize(Me.D));
  write(Me.D, reg);
end;//DatosInsertar

procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, pos);
  write(Me.D, reg);
end;//DatosModificar

procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
begin
  Seek(Me.D, pos);
  Read(Me.D, Reg);
```

```
 end;//DatosCapturar

 procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
 var
 RegistroDatos: Tipo_Registro_Datos;
 begin
  Seek(Me.D, pos);
  Read(Me.D, RegistroDatos);
  RegistroDatos.borrado:=true;
  Seek(Me.D, pos);
  Write(Me.D, RegistroDatos);
 end;//DatosCapturar

 procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);
 var
  cArchivoDatos: String;
 begin
  cArchivoDatos:= Ruta+'\'+Nombre+'.dat';
  Close(Me.D);
  Erase(Me.D);
 end; //DatosDestruir
end.
```

LO_DATOS_DETALLE

```
unit LO_Datos_Detalles;
interface
 uses
  SysUtils;

 type
  Tipo_Clave =  String [255];
```

```
  Tipo_Posicion = LongInt;
  Tipo_Cadena = String [255];
  Tipo_Cantidad = longint;
  Tipo_Precio = real;
   Tipo_Registro_Datos = record
                    codFactura: Tipo_Clave;
                    codArticulo: Longint;
                    descripcion: Tipo_Cadena;
                    cant: Tipo_Cantidad;
                    precioUnitario: Tipo_Precio;
                    Borrado: boolean;
                   end;

  Tipo_Archivo_Datos = file of Tipo_Registro_Datos;

  Tipo_Datos = record
                D: Tipo_Archivo_Datos;
               end;

 var
  Datos: Tipo_Datos;

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
 procedure Datos_Abrir(var ME: Tipo_Datos);
 procedure Datos_Cerrar(var ME: Tipo_Datos);
 procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
 procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
 procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
 procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
 procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);

implementation
```

```
function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
 var
  sArchivoDatos: Tipo_cadena;
  bHayError: boolean;
begin
  sArchivoDatos:= sRuta+'\'+sNombre+'.dat';
  Assign(ME.D, sArchivoDatos);     //Asignamos archivo de Datos

  {$I-}
  Reset(Me.D);
  bHayError:= IoResult <> 0;
  if bHayError then Rewrite(Me.D);
  Close(Me.D);
  Datos_Crear:= bHayError;
  {$I+}

end;//DatosCrear

procedure Datos_Abrir(var ME: Tipo_Datos);
begin
  reset(Me.D);
end;//DatosAbrir

procedure Datos_Cerrar(var ME: Tipo_Datos);
begin
  close(Me.D);
end;//DatosCerrar

procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, FileSize(Me.D));
  write(Me.D, reg);
```

```
end;//DatosInsertar

procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, pos);
  write(Me.D, reg);
end;//DatosModificar

procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
begin
  Seek(Me.D, pos);
  Read(Me.D, Reg);
end;//DatosCapturar

procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
var
RegistroDatos: Tipo_Registro_Datos;
begin
  Seek(Me.D, pos);
  Read(Me.D, RegistroDatos);
  RegistroDatos.borrado:=true;
  Seek(Me.D, pos);
  Write(Me.D, RegistroDatos);
end;//DatosCapturar

procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);
var
  cArchivoDatos: String;
begin
  cArchivoDatos:= Ruta+'\'+Nombre+'.dat';
  Close(Me.D);
  Erase(Me.D);
```

```
 end; //DatosDestruir
end.
```

LO_DATOS_RUBROS

```
unit LO_Datos_Rubros;
interface
 uses
  SysUtils;

 type
  Tipo_Clave = String [3];  //Entre 100 y 999
  Tipo_Nombre = String [40]; //Nombre es String de hasta 40 caracteres
  Tipo_Cantidad= longint;
  Tipo_Cadena = String [255];
  Tipo_Posicion = LongInt;
  Tipo_Registro_Datos = record
                    Clave: Tipo_Clave;
                    Nombre: Tipo_Nombre;
                    minimoOperativo: Tipo_Cantidad;
                                maximoAReponer: Tipo_Cantidad;
                    Borrado: boolean;
                   end;
  Tipo_Archivo_Datos = file of Tipo_Registro_Datos;
  Tipo_Datos = record
                D: Tipo_Archivo_Datos;
               end;
 var
  Datos: Tipo_Datos;

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
 procedure Datos_Abrir(var ME: Tipo_Datos);
 procedure Datos_Cerrar(var ME: Tipo_Datos);
```

```
 procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
 procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
 procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
 procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
 procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);

implementation

 function Datos_Crear(var ME: Tipo_Datos; sRuta, sNombre: string): boolean;
  var
   sArchivoDatos: Tipo_cadena;
   bHayError: boolean;
  begin
   sArchivoDatos:= sRuta+'\'+sNombre+'.dat';
   Assign(ME.D, sArchivoDatos);     //Asignamos archivo de Datos

   {$I-}
   Reset(Me.D);
   bHayError:= IoResult <> 0;
   if bHayError then Rewrite(Me.D);
   Close(Me.D);
   Datos_Crear:= bHayError;
   {$I+}

 end;//DatosCrear

 procedure Datos_Abrir(var ME: Tipo_Datos);
 begin
   reset(Me.D);
 end;//DatosAbrir

 procedure Datos_Cerrar(var ME: Tipo_Datos);
 begin
```

```
  close(Me.D);
end;//DatosCerrar

procedure Datos_Insertar(var ME: Tipo_Datos; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, FileSize(Me.D));
  write(Me.D, reg);
end;//DatosInsertar

procedure Datos_Modificar(var Me: Tipo_Datos; pos: Tipo_Posicion; Reg: Tipo_Registro_Datos);
begin
  reg.Borrado:= false;
  Seek(Me.D, pos);
  write(Me.D, reg);
end;//DatosModificar

procedure Datos_Capturar(var Me: Tipo_Datos; pos: Tipo_Posicion; var Reg: Tipo_Registro_Datos);
begin
  Seek(Me.D, pos);
  Read(Me.D, Reg);
end;//DatosCapturar

procedure Datos_Eliminar(var Me: Tipo_Datos; pos: Tipo_Posicion);
var
RegistroDatos: Tipo_Registro_Datos;
begin
  Seek(Me.D, pos);
  Read(Me.D, RegistroDatos);
  RegistroDatos.borrado:=true;
  Seek(Me.D, pos);
  Write(Me.D, RegistroDatos);
end;//DatosCapturar
```

```
 procedure Datos_Destruir(var ME: Tipo_Datos; ruta, nombre: Tipo_Cadena);
 var
   cArchivoDatos: String;
 begin
   cArchivoDatos:= Ruta+'\'+Nombre+'.dat';
   Close(Me.D);
   Erase(Me.D);
 end; //DatosDestruir


end.
```